
PROCESADORES DE LENGUAJES

MEMORIA DE PROYECTO - HITO 2: ANALIZADOR SINTÁCTICO

GRUPO 10

SERGIO COLET GARCÍA
LAURA MARTÍNEZ TOMÁS
RODRIGO SOUTO SANTOS
LI JIE CHEN CHEN

*GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID*



2023-2024

Índice general

1 | Especificación de la sintaxis abstracta

1.1. Eliminación de terminales sin carga semántica

programa \rightarrow *bloque*
bloque \rightarrow *declaraciones instrucciones*
declaraciones \rightarrow *lista-declaraciones*
declaraciones $\rightarrow \epsilon$
lista-declaraciones \rightarrow *lista-declaraciones declaracion*
lista-declaraciones \rightarrow *declaracion*
lista-variables \rightarrow *lista-variables declaracion*
lista-variables \rightarrow *declaracion*
declaracion \rightarrow *tipo identificador*
declaracion \rightarrow **type** *tipo identificador*
declaracion \rightarrow **identificador** *par-formales bloque*
tipo \rightarrow *tipo literalEntero*
tipo \rightarrow *tipo2*
tipo2 \rightarrow \wedge *tipo2*
tipo2 \rightarrow *tipo3*
tipo3 \rightarrow **bool**
tipo3 \rightarrow **int**
tipo3 \rightarrow **real**
tipo3 \rightarrow **string**
tipo3 \rightarrow **identificador**
tipo3 \rightarrow **struct** *lista-variables*
instrucciones \rightarrow *lista-instrucciones*
instrucciones $\rightarrow \epsilon$
lista-instrucciones \rightarrow *lista-instrucciones instruccion*
lista-instrucciones \rightarrow *instruccion*
par-formales \rightarrow *lista-par-formal*
par-formales $\rightarrow \epsilon$
lista-par-formal \rightarrow *par-formal lista-par-formal*
lista-par-formal \rightarrow *par-formal*
par-formal \rightarrow *tipo & identificador*
par-formal \rightarrow *tipo identificador*
par-reales \rightarrow *lista-par-real*
par-reales $\rightarrow \epsilon$
lista-par-real \rightarrow *E lista-par-real*
lista-par-real \rightarrow *E*
instruccion \rightarrow *eval*
instruccion \rightarrow **if** *E bloque*
instruccion \rightarrow **if** *E bloque bloque*
instruccion \rightarrow **while** *E bloque*
instruccion \rightarrow **new** *E*
instruccion \rightarrow **delete** *E*
instruccion \rightarrow **read** *E*
instruccion \rightarrow **write** *E*
instruccion \rightarrow **call** *identificador par-reales*
instruccion \rightarrow **nl**
instruccion \rightarrow *bloque*
eval \rightarrow *E*
E \rightarrow *E1 = E*
E \rightarrow *E1*

$E1 \rightarrow E1 \text{ OP1 } E2$
 $E1 \rightarrow E2$
 $E2 \rightarrow E2 + E3$
 $E2 \rightarrow E3 - E3$
 $E2 \rightarrow E3$
 $E3 \rightarrow E4 \text{ and } E3$
 $E3 \rightarrow E4 \text{ or } E4$
 $E3 \rightarrow E4$
 $E4 \rightarrow E4 \text{ OP4 } E5$
 $E4 \rightarrow E5$
 $E5 \rightarrow OP5 \ E6$
 $E5 \rightarrow E6$
 $E6 \rightarrow E6 \text{ OP6}$
 $E6 \rightarrow E7$
 $E7 \rightarrow \text{true}$
 $E7 \rightarrow \text{false}$
 $E7 \rightarrow \text{literalEntero}$
 $E7 \rightarrow \text{literalReal}$
 $E7 \rightarrow \text{literalCadena}$
 $E7 \rightarrow \text{identificador}$
 $E7 \rightarrow \text{null}$
 $E7 \rightarrow E$
 $OP1 \rightarrow <$
 $OP1 \rightarrow <=$
 $OP1 \rightarrow >$
 $OP1 \rightarrow >=$
 $OP1 \rightarrow ==$
 $OP1 \rightarrow !=$
 $OP4 \rightarrow *$
 $OP4 \rightarrow /$
 $OP4 \rightarrow \%$
 $OP5 \rightarrow +$
 $OP5 \rightarrow -$
 $OP5 \rightarrow \text{not}$
 $OP6 \rightarrow E$
 $OP6 \rightarrow \text{identificador}$
 $OP6 \rightarrow ^$

1.2. Simplificación de la sintaxis

$\text{bloque} \rightarrow \text{declaraciones instrucciones}$
 $\text{declaraciones} \rightarrow \text{lista-declaraciones}$
 $\text{declaraciones} \rightarrow \epsilon$
 $\text{lista-declaraciones} \rightarrow \text{lista-declaraciones declaracion}$
 $\text{lista-declaraciones} \rightarrow \text{declaracion}$
 $\text{lista-variables} \rightarrow \text{lista-variables declaracion}$
 $\text{lista-variables} \rightarrow \text{declaracion}$
 $\text{declaracion} \rightarrow \text{tipo identificador}$
 $\text{declaracion} \rightarrow \text{type tipo identificador}$
 $\text{declaracion} \rightarrow \text{identificador par-formales bloque}$
 $\text{tipo} \rightarrow \text{tipo literalEntero}$
 $\text{tipo} \rightarrow ^ \text{tipo}$
 $\text{tipo} \rightarrow \text{bool}$
 $\text{tipo} \rightarrow \text{int}$
 $\text{tipo} \rightarrow \text{real}$
 $\text{tipo} \rightarrow \text{string}$
 $\text{tipo} \rightarrow \text{identificador}$
 $\text{tipo} \rightarrow \text{struct lista-variables}$

$instrucciones \rightarrow lista-instrucciones$
 $instrucciones \rightarrow \epsilon$
 $lista-instrucciones \rightarrow lista-instrucciones\ instrucción$
 $lista-instrucciones \rightarrow instrucción$
 $par-formales \rightarrow lista-par-formal$
 $par-formales \rightarrow \epsilon$
 $lista-par-formal \rightarrow par-formal\ lista-par-formal$
 $lista-par-formal \rightarrow par-formal$
 $par-formal \rightarrow tipo\ \&identificador$
 $par-formal \rightarrow tipo\ \mathbf{identificador}$
 $par-reales \rightarrow lista-par-real$
 $par-reales \rightarrow \epsilon$
 $lista-par-real \rightarrow E\ lista-par-real$
 $lista-par-real \rightarrow E$
 $instrucción \rightarrow E$
 $instrucción \rightarrow \mathbf{if}\ E\ bloque$
 $instrucción \rightarrow \mathbf{if}\ E\ bloque\ \mathbf{else}\ bloque$
 $instrucción \rightarrow \mathbf{while}\ E\ bloque$
 $instrucción \rightarrow \mathbf{new}\ E$
 $instrucción \rightarrow \mathbf{delete}\ E$
 $instrucción \rightarrow \mathbf{read}\ E$
 $instrucción \rightarrow \mathbf{write}\ E$
 $instrucción \rightarrow \mathbf{call}\ identificador\ par-reales$
 $instrucción \rightarrow \mathbf{nl}$
 $instrucción \rightarrow bloque$
 $E \rightarrow E = E$
 $E \rightarrow E < E$
 $E \rightarrow E <= E$
 $E \rightarrow E > E$
 $E \rightarrow E >= E$
 $E \rightarrow E == E$
 $E \rightarrow E != E$
 $E \rightarrow E + E$
 $E \rightarrow E - E$
 $E \rightarrow E * E$
 $E \rightarrow E / E$
 $E \rightarrow E \% E$
 $E \rightarrow E\ \mathbf{and}\ E$
 $E \rightarrow E\ \mathbf{or}\ E$
 $E \rightarrow -E$
 $E \rightarrow \mathbf{not}\ E$
 $E \rightarrow E\ E$
 $E \rightarrow E.\mathbf{identificador}$
 $E \rightarrow E^{\wedge}$
 $E \rightarrow \mathbf{true}$
 $E \rightarrow \mathbf{false}$
 $E \rightarrow \mathbf{literalEntero}$
 $E \rightarrow \mathbf{literalReal}$
 $E \rightarrow \mathbf{literalCadena}$
 $E \rightarrow \mathbf{identificador}$
 $E \rightarrow \mathbf{null}$

1.3. Asignación de géneros a cada no-terminal

Cuadro 1.3.1: Géneros de los no-terminales

No terminal	Género
bloque	Blo
declaraciones	Decs
lista-declaraciones	LDecs
lista-variables	LVar
declaracion	Dec
tipo	Tipo
instrucciones	Insts
lista-instrucciones	LInst
par-formales	PFmls
lista-par-formal	LPFml
par-formal	PFml
par-reales	PReales
lista-par-reas	LPReal
instruccion	Inst
E	Exp

1.4. Constructores de las reglas

Cuadro 1.4.1: Constructores de las diferentes reglas

Regla	Constructor
$\text{bloque} \rightarrow \text{declaraciones instrucciones}$	$\text{bloq}: \text{Decs} \times \text{Insts} \rightarrow \text{Blo}$
$\text{declaraciones} \rightarrow \text{lista-declaraciones}$	$\text{si_decs}: \text{LDecs} \rightarrow \text{Decs}$
$\text{declaraciones} \rightarrow \epsilon$	$\text{no_decs}: \rightarrow \text{Decs}$
$\text{declaraciones} \rightarrow$	LVar
$\text{lista-declaraciones} \rightarrow \text{lista-declaraciones declaracion}$	$\text{muchas_decs}: \text{LDecs} \times \text{Dec} \rightarrow \text{LDecs}$
$\text{lista-declaraciones} \rightarrow \text{declaracion}$	$\text{una_dec}: \text{Dec} \rightarrow \text{LDecs}$
$\text{lista-variables} \rightarrow \text{lista-variables declaracion}$	$\text{muchas_var}: \text{LVar} \times \text{Dec} \rightarrow \text{LVar}$
$\text{lista-variables} \rightarrow \text{declaracion}$	$\text{una_var}: \text{Dec} \rightarrow \text{LVar}$
$\text{declaracion} \rightarrow \text{tipo identificador}$	$\text{dec_simple}: \text{Tipo} \times \text{string} \rightarrow \text{Dec}$
$\text{declaracion} \rightarrow \text{type tipo identificador}$	$\text{dec_type}: \text{Tipo} \times \text{string} \rightarrow \text{Dec}$
$\text{declaracion} \rightarrow \text{identificador par-formales bloque}$	$\text{dec_proc}: \text{string} \times \text{PFmls} \times \text{Blo} \rightarrow \text{Dec}$
$\text{tipo} \rightarrow \text{tipo literalEntero}$	$\text{tipo_array}: \text{Tipo} \times \text{string} \rightarrow \text{Tipo}$
$\text{tipo} \rightarrow \text{^tipo}$	$\text{tipo_punt}: \text{Tipo} \rightarrow \text{Tipo}$
$\text{tipo} \rightarrow \text{bool}$	$\text{tipo_bool}: \rightarrow \text{Tipo}$
$\text{tipo} \rightarrow \text{int}$	$\text{tipo_int}: \rightarrow \text{Tipo}$
$\text{tipo} \rightarrow \text{real}$	$\text{tipo_real}: \rightarrow \text{Tipo}$
$\text{tipo} \rightarrow \text{string}$	$\text{tipo_string}: \rightarrow \text{Tipo}$
$\text{tipo} \rightarrow \text{identificador}$	$\text{tipo_ident}: \text{string} \rightarrow \text{Tipo}$

Continúa en la siguiente página

Cuadro 1.4.1: Constructores de las diferentes reglas (Continuación)

Regla	Constructor
$tipo \rightarrow \mathbf{struct} \text{ lista-variables}$	$tipo_struct: LVar \rightarrow Tipo$
$instrucciones \rightarrow \text{lista-instrucciones}$	$si_inst: LInst \rightarrow Insts$
$instrucciones \rightarrow \epsilon$	$no_inst: \rightarrow Insts$
$\text{lista-instrucciones} \rightarrow \text{lista-instrucciones instrucion}$	$muchas_inst: LInst \times Inst \rightarrow LInst$
$\text{lista-instrucciones} \rightarrow \text{instrucion}$	$una_inst: Inst \rightarrow LInst$
$\text{par-formales} \rightarrow \text{lista-par-formal}$	$si_pformal: LPFml \rightarrow PFmls$
$\text{par-formales} \rightarrow \epsilon$	$no_pformal: \rightarrow PFmls$
$\text{lista-par-formal} \rightarrow \text{par-formal lista-par-formal}$	$muchos_pformal: PFml \times LPFml \rightarrow LPFml$
$\text{lista-par-formal} \rightarrow \text{par-formal}$	$un_pformal: PFml \rightarrow LPFml$
$\text{par-formal} \rightarrow \text{tipo} \ \&\mathbf{identificador}$	$pformal_ref: Tipo \times \mathbf{string} \rightarrow PFml$
$\text{par-formal} \rightarrow \text{tipo} \ \mathbf{identificador}$	$pformal_noref: Tipo \times \mathbf{string} \rightarrow PFml$
$\text{par-reales} \rightarrow \text{lista-par-real}$	$si_preales: LPReal \rightarrow PReales$
$\text{par-reales} \rightarrow \epsilon$	$no_preales: \rightarrow PReales$
$\text{lista-par-real} \rightarrow E \text{ lista-par-real}$	$muchas_exp: Exp \times LPReal \rightarrow LPReal$
$\text{lista-par-real} \rightarrow E$	$una_exp: Exp \rightarrow LPReal$
$\text{instrucion} \rightarrow E$	$inst_eval: Exp \rightarrow Inst$
$\text{instrucion} \rightarrow \mathbf{if} \ E \ \text{bloque}$	$inst_if: Exp \times Blo \rightarrow Inst$
$\text{instrucion} \rightarrow \mathbf{if} \ E \ \text{bloque} \ \mathbf{else} \ \text{bloque}$	$inst_else: Exp \times Blo \times Blo \rightarrow Inst$
$\text{instrucion} \rightarrow \mathbf{while} \ E \ \text{bloque}$	$inst_while: Exp \times Blo \rightarrow Inst$
$\text{instrucion} \rightarrow \mathbf{new} \ E$	$inst_new: Exp \rightarrow Inst$
$\text{instrucion} \rightarrow \mathbf{delete} \ E$	$inst_delete: Exp \rightarrow Inst$
$\text{instrucion} \rightarrow \mathbf{read} \ E$	$inst_read: Exp \rightarrow Inst$
$\text{instrucion} \rightarrow \mathbf{write} \ E$	$inst_write: Exp \rightarrow Inst$
$\text{instrucion} \rightarrow \mathbf{call} \ \mathbf{identificador} \ \text{par-reales}$	$inst_call: \mathbf{string} \times PReales \rightarrow Inst$
$\text{instrucion} \rightarrow \mathbf{nl}$	$inst_nl: \rightarrow Inst$
$\text{instrucion} \rightarrow \text{bloque}$	$inst_blo: Blo \rightarrow Inst$
$E \rightarrow E = E$	$exp_asig: Exp \times Exp \rightarrow Exp$
$E \rightarrow E < E$	$exp_menor: Exp \times Exp \rightarrow Exp$
$E \rightarrow E \leq E$	$exp_menIgual: Exp \times Exp \rightarrow Exp$
$E \rightarrow E > E$	$exp_mayor: Exp \times Exp \rightarrow Exp$
$E \rightarrow E \geq E$	$exp_mayIgual: Exp \times Exp \rightarrow Exp$
$E \rightarrow E == E$	$exp_igual: Exp \times Exp \rightarrow Exp$
$E \rightarrow E \neq E$	$exp_dist: Exp \times Exp \rightarrow Exp$
$E \rightarrow E + E$	$exp_sum: Exp \times Exp \rightarrow Exp$
$E \rightarrow E - E$	$exp_resta: Exp \times Exp \rightarrow Exp$
$E \rightarrow E * E$	$exp_mult: Exp \times Exp \rightarrow Exp$
$E \rightarrow E / E$	$exp_div: Exp \times Exp \rightarrow Exp$
$E \rightarrow E \% E$	$exp_mod: Exp \times Exp \rightarrow Exp$
$E \rightarrow E \ \mathbf{and} \ E$	$exp_and: Exp \times Exp \rightarrow Exp$
$E \rightarrow E \ \mathbf{or} \ E$	$exp_or: Exp \times Exp \rightarrow Exp$

Continúa en la siguiente página

Cuadro 1.4.1: Constructores de las diferentes reglas (Continuación)

Regla	Constructor
$E \rightarrow -E$	exp_menos: $\text{Exp} \rightarrow \text{Exp}$
$E \rightarrow \text{not } E$	exp_not: $\text{Exp} \rightarrow \text{Exp}$
$E \rightarrow E E$	exp_index: $\text{Exp} \times \text{Exp} \rightarrow \text{Exp}$
$E \rightarrow E .\text{identificador}$	exp_reg: $\text{Exp} \times \text{string} \rightarrow \text{Exp}$
$E \rightarrow E^{\wedge}$	exp_ind: $\text{Exp} \rightarrow \text{Exp}$
$E \rightarrow \text{true}$	exp_true: $\rightarrow \text{Exp}$
$E \rightarrow \text{false}$	exp_false: $\rightarrow \text{Exp}$
$E \rightarrow \text{literalEntero}$	exp_litEnt: $\text{string} \rightarrow \text{Exp}$
$E \rightarrow \text{literalReal}$	exp_litReal: $\text{string} \rightarrow \text{Exp}$
$E \rightarrow \text{literalCadena}$	exp_litCad: $\text{string} \rightarrow \text{Exp}$
$E \rightarrow \text{identificador}$	exp_iden: $\text{string} \rightarrow \text{Exp}$
$E \rightarrow \text{null}$	exp_null: $\rightarrow \text{Exp}$

2 | Especificación del constructor ATS's mediante una gramática s-atribuida

2.1. Especificación del Constructor de ASTs

```

programa  $\rightarrow$  bloque
|
|   programa.a = bloque.a
bloque  $\rightarrow$  { declaraciones instrucciones }
|
|   bloque.a = bloq(declaraciones.a, instrucciones.a)
declaraciones  $\rightarrow$  lista-declaraciones &&
|
|   declaraciones.a = si_decs(lista-declaraciones.a)
declaraciones  $\rightarrow$   $\epsilon$ 
|
|   declaraciones.a = no_decs()
lista-declaraciones  $\rightarrow$  lista-declaraciones ; declaracion
|
|   lista-declaraciones0.a = muchas_decs(lista-declaraciones1.a, declaraciones.a)
lista-declaraciones  $\rightarrow$  declaracion
|
|   lista-declaraciones.a = una_dec(declaraciones.a)
lista-variables  $\rightarrow$  lista-variables , declaracion
|
|   lista-variables0.a = muchas_var(lista-variables1.a, declaraciones.a)
lista-variables  $\rightarrow$  declaracion
|
|   lista-variables.a = una_var(declaraciones.a)
declaracion  $\rightarrow$  tipo identificador
|
|   declaracion.a = dec_simple(tipo.a, identificador.lex)
declaracion  $\rightarrow$  type tipo identificador
|
|   declaracion.a = dec_type(tipo.a, identificador.lex)
declaracion  $\rightarrow$  identificador par-formales bloque
|
|   declaracion.a = dec_proc(identificador.lex, par-formales.a, bloque.a)
tipo  $\rightarrow$  tipo[literalEntero]
|
|   tipo.a = tipo_array(tipo.a, literalEntero.lex)
tipo  $\rightarrow$  tipo2
|
|   tipo.a = tipo2.a
tipo2  $\rightarrow$  ^ tipo2
|
|   tipo2.a = tipo_punt(tipo2.a)
tipo2  $\rightarrow$  tipo3
|
|   tipo2.a = tipo3.a
tipo3  $\rightarrow$  bool
|
|   tipo3.a = tipo_bool
tipo3  $\rightarrow$  int
|
|   tipo3.a = tipo_int
tipo3  $\rightarrow$  real
|
|   tipo3.a = tipo_real
tipo3  $\rightarrow$  string
|
|   tipo3.a = tipo_string
tipo3  $\rightarrow$  identificador
|
|   tipo3.a = tipo_ident(identificador.lex)
tipo3  $\rightarrow$  struct { lista-variables }
|
|   tipo3.a = tipo_struct(lista-variables.a)
instrucciones  $\rightarrow$  lista-instrucciones
|
|   instrucciones.a = si_inst(lista-instrucciones.a)
instrucciones  $\rightarrow$   $\epsilon$ 
|
|   instrucciones.a = no_inst()
lista-instrucciones  $\rightarrow$  lista-instrucciones ; instruccion

```

```

|      lista-instrucciones0.a = muchas_inst(lista-instrucciones1.a, instruccion.a)
lista-instrucciones → instruccion
|      lista-instrucciones.a = una_inst(instruccion.a)
par-formales → (lista-par-formal)
|      par-formales.a = si_pformal(lista-par-formal.a)
par-formales → ()
|      par-formales.a = no_pformal()
lista-par-formal → par-formal, lista-par-formal
|      lista-par-formal0.a = muchos_pformal(par-formal.a, lista-par-formal1.a)
lista-par-formal → par-formal
|      lista-par-formal.a = un_pformal(par-formal.a)
par-formal → tipo &identificador
|      par-formal.a = pformal_ref(tipo.a, indentificador.lex)
par-formal → tipo identificador
|      par-formal.a = pformal_noref(tipo.a, indentificador.lex)
par-reales → (lista-par-real)
|      par-reales.a = si_preales(lista-par-real.a)
par-reales → ()
|      par-reales.a = no_preales()
lista-par-real → E0, lista-par-real
|      lista-par-real0.a = muchas_exp(E0.a, lista-par-real1.a)
lista-par-real → E0
|      lista-par-real.a = una_exp(E0.a)
instruccion → eval
|      instruccion.a = eval.a
instruccion → E0
|      instruccion.a = inst_eval(E0.a)
instruccion → if E0 bloque
|      instruccion.a = inst_if(E0.a, bloque.a)
instruccion → if E0 bloque else bloque
|      instruccion.a = inst_else(E0.a, bloque.a, bloque.a)
instruccion → while E0 bloque
|      instruccion.a = inst_while(E0.a, bloque.a)
instruccion → new E0
|      instruccion.a = inst_new(E0.a)
instruccion → delete E0
|      instruccion.a = inst_delete(E0.a)
instruccion → read E0
|      instruccion.a = inst_read(E0.a)
instruccion → write E0
|      instruccion.a = inst_write(E0.a)
instruccion → call identificador par-reales
|      instruccion.a = inst_call(identificador.a, par-reales.a)
instruccion → nl
instruccion → bloque
|      instruccion.a = inst_blo(bloque.a)
eval → @ E0
|      eval.a = E0.a
E0 → E1 = E0
|      E00.a = exp_asig(E1.a, E01.a)
E0 → E1
|      E.a = E1.a
E1 → E1 OP1 E2
|      E10.a = mkop(OP1.op, E11.a, E2.a)
E1 → E2
|      E1.a = E2.a
E2 → E2 + E3
|      E20.a = exp_suma(E21.a, E3.a)
E2 → E3 - E3
|      E2.a = exp_resta(E30.a, E31.a)

```

```

E2 → E3
|   E2.a = E3.a
E3 → E4 and E3
|   E30.a = exp_and(E4.a, E31.a)
E3 → E4 or E4
|   E3.a = exp_or(E40.a, E41.a)
E3 → E4
|   E3.a = E4.a
E4 → E4 OP4 E5
|   E40.a = mkop(OP4.op, E41.a, E5.a)
E4 → E5
|   E4.a = E5.a
E5 → OP5 E6
|   E5.a = mkop(OP5.op, E6.a)
E5 → E6
|   E5.a = E6.a
E6 → E6 OP6
|   E6.a = mkop(E6.a, OP6.op)
E6 → E7
|   E6.a = E7.a
E7 → true
|   E7.a = exp_true(true.lex)
E7 → false
|   E7.a = exp_false(false.lex)
E7 → literalEntero
|   E7.a = exp_litEnt(literalEntero.lex)
E7 → literalReal
|   E7.a = exp_litReal(literalReal.lex)
E7 → literalCadena
|   E7.a = exp_litCad(literalCadena.lex)
E7 → identificador
|   E7.a = exp_iden(identificador.lex)
E7 → null
|   E7.a = exp_null(null.lex)
E7 → ( E0 )
|   E7.a = E.a
OP1 → <
|   OP1.op = “ < ”
OP1 → <=
|   OP1.op = “ <= ”
OP1 → >
|   OP1.op = “ > ”
OP1 → >=
|   OP1.op = “ >= ”
OP1 → ==
|   OP1.op = “ == ”
OP1 → !=
|   OP1.op = “ != ”
OP4 → *
|   OP4.op = “ * ”
OP4 → /
|   OP4.op = “ / ”
OP4 → %
|   OP4.op = “ % ”
OP5 → -
|   OP5.op = “ - ”
OP5 → not
|   OP5.op = “ not ”
OP6 → [ E0 ]
|   OP6.op = E0.a

```

```

OP6  $\rightarrow$  identificador
|      OP6.op = identificador.lex
OP6  $\rightarrow$  ^
|      OP6.op = “^”

```

```

fun mkop(op, opnd1, opnd2) :
|      op = "<"  $\rightarrow$  return exp_menor(opnd1, opnd2)
|      op = "<="  $\rightarrow$  return exp_menIgual(opnd1, opnd2)
|      op = ">"  $\rightarrow$  return exp_mayor(opnd1, opnd2)
|      op = ">="  $\rightarrow$  return exp_mayIgual(opnd1, opnd2)
|      op = "=="  $\rightarrow$  return exp_igual(opnd1, opnd2)
|      op = "="  $\rightarrow$  return exp_dist(opnd1, opnd2)
|      op = "*"  $\rightarrow$  return exp_mult(opnd1, opnd2)
|      op = "/"  $\rightarrow$  return exp_div(opnd1, opnd2)
|      op = "%"  $\rightarrow$  return exp_mod(opnd1, opnd2)

```

```

fun mkop(op, opnd1) :
|      op = "-"  $\rightarrow$  return exp_menos(opnd1)
|      op = "not"  $\rightarrow$  return exp_not(opnd1)
|      op = E0.a  $\rightarrow$  return exp_index(opnd1)
|      op = identificador.lex  $\rightarrow$  return exp_reg(opnd1)
|      op = “^”  $\rightarrow$  return exp_ind(opnd1)

```

3 | Acondicionamiento de la especificación del constructor ATS's

```

programa → bloque
|
|   programa.a = bloque.a
bloque → { declaraciones instrucciones }
|
|   bloque.a = bloq(declaraciones.a, instrucciones.a)
declaraciones → lista-declaraciones &&
|
|   declaraciones.a = si_decs(lista-declaraciones.a)
declaraciones → ε
|
|   declaraciones.a = no_decs()
lista-declaraciones → declaracion rlista-decs
|
|   rlista-decs.ah = una_dec(declaracion.a)
|   lista-declaraciones.a = rlista-decs.a
rlista-decs → ; declaracion rlista-decs
|
|   rlista-decs1.ah = muchas_decs(rlista-decs0.ah, declaraciones.a)
|   rlista-decs0.a = rlista-decs1.a
rlista-decs → ε
|
|   rlista-decs.a = rlista-decs.ah
lista-variables → declaracion rlista-var
|
|   rlista-var.ah = una_var(declaracion.a)
|   lista-variables.a = rlista-var.a
rlista-var → , declaracion rlista-var
|
|   rlista-var1.ah = muchas_var(rlista-var0.ah, declaraciones.a)
|   rlista-var0.a = rlista-var1.a
rlista-var → ε
|
|   rlista-var.a = rlista-var.ah
declaracion → tipo identificador
|
|   declaracion.a = dec_simple(tipo.a, identificador.lex)
declaracion → type tipo identificador
|
|   declaracion.a = dec_type(tipo.a, identificador.lex)
declaracion → identificador par-formales bloque
|
|   declaracion.a = dec_proc(identificador.lex, par-formales.a, bloque.a)
tipo → tipo2 rtipo
|
|   rtipo.ah = tipo2.a
|   tipo.a = rtipo.a
rtipo → [literalEntero] rtipo
|
|   rtipo1.ah = tipo_array(rtipo0.ah, literalEntero.lex)
|   rtipo0.a = rtipo1.a
tipo2 → ^ tipo2
|
|   tipo2.a = tipo_punt(tipo2.a)
tipo2 → tipo3
|
|   tipo2.a = tipo3.a
tipo3 → bool
|
|   tipo3.a = tipo_bool
tipo3 → int
|
|   tipo3.a = tipo_int
tipo3 → real
|
|   tipo3.a = tipo_real
tipo3 → string
|
|   tipo3.a = tipo_string
tipo3 → identificador
|
|   tipo3.a = tipo_ident(identificador.lex)
tipo3 → struct { lista-variables }
|
|   tipo3.a = tipo_struct(lista-variables.a)
instrucciones → lista-instrucciones
|
|   instrucciones.a = si_inst(lista-instrucciones.a)

```

```

instrucciones  $\rightarrow \epsilon$ 
|   instrucciones.a = no_inst()
lista-instrucciones  $\rightarrow$  instruccion rlista-inst
|   rlista-inst.ah = una_inst(instruccion.a)
|   lista-instrucciones.a = rlista-inst.a
rlista-inst  $\rightarrow$ ; instruccion rlista-inst
|   rlista-inst1.ah = muchas_inst(rlista-inst0.ah, instrucciones.a)
|   rlista-inst0.a = rlista-inst1.a
rlista-inst  $\rightarrow \epsilon$ 
|   rlista-inst.a = rlista-inst.ah
par-formales  $\rightarrow$  (lista-par-formal)
|   par-formales.a = si_pformal(lista-par-formal.a)
par-formales  $\rightarrow$  ()
|   par-formales.a = no_pformal()
lista-par-formal  $\rightarrow$  par-formal, lista-par-formal
|   lista-par-formal0.a = muchos_pformal(par-formal.a, lista-par-formal1.a)
lista-par-formal  $\rightarrow$  par-formal
|   lista-par-formal.a = un_pformal(par-formal.a)
par-formal  $\rightarrow$  tipo &identificador
|   par-formal.a = pformal_ref(tipo.a, identificador.lex)
par-formal  $\rightarrow$  tipo identificador
|   par-formal.a = pformal_noref(tipo.a, identificador.lex)
par-reales  $\rightarrow$  (lista-par-real)
|   par-reales.a = si_preales(lista-par-real.a)
par-reales  $\rightarrow$  ()
|   par-reales.a = no_preales()
lista-par-real  $\rightarrow$  E0, lista-par-real
|   lista-par-real0.a = muchas_exp(E0.a, lista-par-real1.a)
lista-par-real  $\rightarrow$  E0
|   lista-par-real.a = una_exp(E0.a)
instruccion  $\rightarrow$  eval
|   instruccion.a = eval.a
instruccion  $\rightarrow$  E0
|   instruccion.a = inst_eval(E0.a)
instruccion  $\rightarrow$  if E0 bloque
|   instruccion.a = inst_if(E0.a, bloque.a)
instruccion  $\rightarrow$  if E0 bloque else bloque
|   instruccion.a = inst_else(E0.a, bloque.a, bloque.a)
instruccion  $\rightarrow$  while E0 bloque
|   instruccion.a = inst_while(E0.a, bloque.a)
instruccion  $\rightarrow$  new E0
|   instruccion.a = inst_new(E0.a)
instruccion  $\rightarrow$  delete E0
|   instruccion.a = inst_delete(E0.a)
instruccion  $\rightarrow$  read E0
|   instruccion.a = inst_read(E0.a)
instruccion  $\rightarrow$  write E0
|   instruccion.a = inst_write(E0.a)
instruccion  $\rightarrow$  call identificador par-reales
|   instruccion.a = inst_call(identificador.a, par-reales.a)
instruccion  $\rightarrow$  nl
instruccion  $\rightarrow$  bloque
|   instruccion.a = inst_blo(bloque.a)
eval  $\rightarrow$  @ E0
|   eval.a = E0.a
E0  $\rightarrow$  E1 = E0
|   E00.a = exp_asig(E1.a, E01.a)
E0  $\rightarrow$  E1
|   E.a = E1.a
E1  $\rightarrow$  E2 rE1

```

```

|       $rE1.\mathbf{ah} = E2.a$ 
|       $E1.\mathbf{a} = rE1.\mathbf{a}$ 
 $rE1 \rightarrow OP1\ E2\ rE1$ 
|       $rE1_1.\mathbf{ah} = \mathbf{mkop}(OP1.\mathbf{op}, rE1_0.\mathbf{ah}, E2.\mathbf{a})$ 
|       $rE1_0.\mathbf{a} = rE1_1.\mathbf{a}$ 
 $rE1 \rightarrow \epsilon$ 
|       $rE1.\mathbf{a} = rE1.\mathbf{ah}$ 
 $E2 \rightarrow E3 - E3\ rE2$ 
|       $rE1.\mathbf{ah} = \mathbf{exp\_resta}(E3_0.\mathbf{a}, E3_1.\mathbf{a})$ 
|       $E2.\mathbf{a} = rE2.\mathbf{a}$ 
 $E2 \rightarrow E3\ rE2$ 
|       $rE2.\mathbf{ah} = E3.\mathbf{a}$ 
|       $E2.\mathbf{a} = rE2.\mathbf{a}$ 
 $rE2 \rightarrow +\ E3\ rE2$ 
|       $rE2_1.\mathbf{ah} = \mathbf{exp\_suma}(rE2_0.\mathbf{ah}, E3.\mathbf{a})$ 
|       $rE2_0.\mathbf{a} = rE2_1.\mathbf{a}$ 
 $rE2 \rightarrow \epsilon$ 
|       $rE2.\mathbf{a} = rE2.\mathbf{ah}$ 
 $E3 \rightarrow E4\ \text{and}\ E3$ 
|       $E3_0.\mathbf{a} = \mathbf{exp\_and}(E4.\mathbf{a}, E3_1.\mathbf{a})$ 
 $E3 \rightarrow E4\ \text{or}\ E4$ 
|       $E3.\mathbf{a} = \mathbf{exp\_or}(E4_0.\mathbf{a}, E4_1.\mathbf{a})$ 
 $E3 \rightarrow E4$ 
|       $E3.\mathbf{a} = E4.\mathbf{a}$ 
 $E4 \rightarrow E5\ rE4$ 
|       $rE4.\mathbf{ah} = E5.a$ 
|       $E4.\mathbf{a} = rE4.\mathbf{a}$ 
 $rE4 \rightarrow OP4\ E5\ rE4$ 
|       $rE4_1.\mathbf{ah} = \mathbf{mkop}(OP4.\mathbf{op}, rE4_0.\mathbf{ah}, E5.\mathbf{a})$ 
|       $rE4_0.\mathbf{a} = rE4_1.\mathbf{a}$ 
 $rE4 \rightarrow \epsilon$ 
|       $rE4.\mathbf{a} = rE4.\mathbf{ah}$ 
 $E5 \rightarrow OP5\ E6$ 
|       $E5.\mathbf{a} = \mathbf{mkop}(OP5.\mathbf{op}, E6.\mathbf{a})$ 
 $E5 \rightarrow E6$ 
|       $E5.\mathbf{a} = E6.\mathbf{a}$ 
 $E6 \rightarrow E7\ rE6$ 
|       $rE6.\mathbf{ah} = E7.a$ 
|       $E6.\mathbf{a} = rE6.\mathbf{a}$ 
 $rE6 \rightarrow OP6\ rE6$ 
|       $rE6_1.\mathbf{ah} = \mathbf{mkop}(rE6_0.\mathbf{ah}, OP6.\mathbf{op})$ 
|       $rE6_0.\mathbf{a} = rE6_1.\mathbf{a}$ 
 $rE6 \rightarrow \epsilon$ 
|       $rE6.\mathbf{a} = rE6.\mathbf{ah}$ 
 $E7 \rightarrow \mathbf{true}$ 
|       $E7.\mathbf{a} = \mathbf{exp\_true}(\mathbf{true.lex})$ 
 $E7 \rightarrow \mathbf{false}$ 
|       $E7.\mathbf{a} = \mathbf{exp\_false}(\mathbf{false.lex})$ 
 $E7 \rightarrow \mathbf{literalEntero}$ 
|       $E7.\mathbf{a} = \mathbf{exp\_litEnt}(\mathbf{literalEntero.lex})$ 
 $E7 \rightarrow \mathbf{literalReal}$ 
|       $E7.\mathbf{a} = \mathbf{exp\_litReal}(\mathbf{literalReal.lex})$ 
 $E7 \rightarrow \mathbf{literalCadena}$ 
|       $E7.\mathbf{a} = \mathbf{exp\_litCad}(\mathbf{literalCadena.lex})$ 
 $E7 \rightarrow \mathbf{identificador}$ 
|       $E7.\mathbf{a} = \mathbf{exp\_iden}(\mathbf{identificador.lex})$ 
 $E7 \rightarrow \mathbf{null}$ 
|       $E7.\mathbf{a} = \mathbf{exp\_null}(\mathbf{null.lex})$ 
 $E7 \rightarrow ( E0 )$ 
|       $E7.\mathbf{a} = E.\mathbf{a}$ 

```

```
OP1 → <
|      OP1.op = “ < ”
OP1 → <=
|      OP1.op = “ <= ”
OP1 → >
|      OP1.op = “ > ”
OP1 → >=
|      OP1.op = “ >= ”
OP1 → ==
|      OP1.op = “ == ”
OP1 → !=
|      OP1.op = “ != ”
OP4 → *
|      OP4.op = “ * ”
OP4 → /
|      OP4.op = “ / ”
OP4 → %
|      OP4.op = “ % ”
OP5 → -
|      OP5.op = “ - ”
OP5 → not
|      OP5.op = “ not ”
OP6 → [ E0 ]
|      OP6.op = E0.a
OP6 → identificador
|      OP6.op = identificador.lex
OP6 → ^
|      OP6.op = “ ^ ”
```


4 | Especificación del procesamiento de impresión

```
imprime ( prog ( Bloque ) :  
    imprime ( Bloque )  
  
imprime ( Bloque ( Decs , Instr ) ) :  
    print "{"  
    nl  
    imprime ( Decs )  
    nl  
    imprime ( Instr )  
    nl  
    print "}"  
    nl  
  
imprime ( si_decs ( LDecs ) :  
    imprime ( Ldecs )  
    nl  
  
imprime ( no_decs ( LDecs ) : noop  
imprime ( muchas_decs ( Ldecs , Dec ) :  
    imprime ( ldecs )  
    nl  
    print " , "  
    nl  
    imprime ( Dec )  
    nl  
imprime ( una_dec ( Dec ) :  
    imprime ( Dec )  
  
imprime ( muchas_var ( lvar , Dec ) :  
    imprime ( lvar )  
    print " , "  
    nl  
    imprime ( Dec )  
    nl  
imprime ( una_var ( Dec ) ) :  
    imprime ( Dec )  
    nl  
  
imprime ( si_inst ( lInstr ) :  
    imprime ( linstr )  
imprime ( no_inst ( ) ) : noop  
imprime ( muchas_inst ( linstr , instr ) :  
    imprime ( linstr )  
    nl  
    print " , "  
    nl  
    imprime ( instr )  
    nl  
  
imprime ( una_inst ( intr ) :  
    imprime ( instr )  
  
imprime ( dec_simple ( Tipo , Id ) :
```

```
    imprime(Tipo)
    nl
    print " "
    nl
    print Id
    nl

imprime(dec_proc(Id, Par-formal, Bloque)):
    print Id
    nl
    imprime(Par-formal)
    imprime(Bloque)

imprime(Tipo_array(Tipo, N):
    imprime(Tipo)
    nl
    print ", "
    nl
    imprime(litEnt)

imprime(Tipo2):

imprime(Tipo_punt(Tipo2):
    imprime(Tipo2)

imprime(Tipo3):

imprime(Tipo_bool(bool)):
    print "<"
    print Bool
    print ">"
    nl

imprime(Tipo_int(int)):
    print "<"
    print int
    print ">"
    nl

imprime(Tipo_real(real)):
    print "<"
    print real
    print ">"
    nl
imprime(Tipo_string(string)):
    print "<"
    print string
    print ">"
    nl

imprime(Tipo_ident(Id)):
    print Id
    nl

imprime(Tipo_struct(lvar)):
    print "<"
    print struct
    print ">"
    nl
```

```
    print "{"
    nl
    imprime(lvar)
    nl
    print "}"
    nl

imprime(si_pformal(l-par-formal):
    imprime(l-par-formal)

imprime(no_pformal()):noop

imprime(muchos_pformal(par-formal,l-par-formal):
    imprime(par-formal)
    nl
    print ","
    nl
    imprime(l-par-formal)

imprime(un_pformal(par-formal):
    imprime(par-formal)

imprime(pformal-ref(Tipo,Id):
    imprime(Tipo)
    nl
    print Id
    nl

imprime(pformal_noref(Tipo, Id):
    imprime(Tipo)
    nl
    print Id

imprime(si_preales(l-par-real):
    imprime(l-par-real)

imprime(no_preales()):noop

imprime(muchas_exp(Exp, l-par-real):
    imprime(Exp)
    nl
    print ","
    nl
    imprime(l-par-real)
    nl

imprime(una_exp(Exp):
    imprime(Exp)
    nl

imprime(Eval)

imprime(inst_eval(Exp)):
    imprime(Exp)

imprime(inst_if(Exp, Bloque):
    print "<"
    print if
    print ">"
    nl
```

```
imprime(Exp)
nl
imprime(Bloque)
nl
```

```
imprime(inst_else(Exp, Bloque, Bloque):
    print "<"
    print else
    print ">"
    nl
    imprime(Exp)
    nl
    imprime(Bloque)
    nl
```

```
imprime(inst_while(Exp, Bloque):
    print "<"
    print while
    print ">"
    nl
    imprime(Exp)
    nl
    imprime(Bloque)
    nl
```

```
imprime(inst_new(Exp)):
    print "<"
    print new
    print ">"
    nl
    imprime(Exp)
    nl
```

```
imprime(inst_delete(Exp)):
    print "<"
    print delete
    print ">"
    nl
    imprime(Exp)
    nl
```

```
imprime(inst_read(read)):
    print "<"
    print read
    print ">"
    nl
    imprime(Exp)
    nl
```

```
imprime(inst_write(write)):
    print "<"
    print write
    print ">"
    nl
    imprime(Exp)
    nl
```

```
imprime(inst_call(Id, par-real):
```

```
    print "<"
    print call
    print ">"
    nl
    print Id
    imprime(par-real)
    nl

imprime(inst_blo(Bloque):
    imprime(Bloque)
    nl

imprime(dec_proc(Id, par-formales, Bloque)):
    print "<"
    print proc
    print ">"
    nl
    print Id
    nl
    imprime(par-formales)
    imprime(Bloque)
    nl

imprime(dec_type(tipo, Id)):
    print "<"
    print type
    print ">"
    nl
    imprime(tipo)
    print Id

imprime(nl):
    print "<"
    print nl
    print ">"
    nl

imprime(exp_true(true):
    print "<"
    print true
    print ">"
    nl

    imprime(exp_false(false):
    print "<"
    print false
    print ">"
    nl

imprime(exp_and(Exp4, Exp3)):
    imprime(Exp4)
    nl
    print "<"
    print and
    print ">"
    nl
    imprime(Exp3)
    nl
```

```

imprime(exp_or(Exp4, exp4)):
    imprime(Exp4)
    nl
    print "<"
    print or
    print ">"
    nl
    imprime(Exp4)
    nl

imprime(" not "(not )):
    print "<"
    print not
    print ">"

imprime(exp_null(null)):
    print "<"
    print null
    print ">"

imprime(lit_Ent(N)):
    print N
imprime(lit_Real(R)):
    print R
imprime(lit_Cad(C)):
    print C
imprime(EOF(eof)):
    print "<"
    print EOF
    print ">"

imprime(exp_asig(Opnd0, Opnd1)):
    imprimeExpBin(Opnd0, "=", Opnd1, 1, 0)
imprime(exp_suma(Opnd0, Opnd1)):
    imprimeExpBin(Opnd0, "+", Opnd1, 2, 3)
imprime(exp_resta(Opnd0, Opnd1)):
    imprimeExpBin(Opnd0, "-", Opnd1, 3, 3)
imprime(exp_and(Opnd0, Opnd1)):
    imprimeExpBin(Opnd0, "and", Opnd1, 4, 3)
imprime(exp_or(Opnd0, Opnd1)):
    imprimeExpBin(Opnd0, "or", Opnd1, 4, 4)
imprime(exp_mult(Opnd0, Opnd1)):
    imprimeExpBin(Opnd0, "*", Opnd1, 4, 5)
imprime(exp_div(Opnd0, Opnd1)):
    imprimeExpBin(Opnd0, "/", Opnd1, 4, 5)
imprime(exp_mod(Opnd0, Opnd1)):
    imprimeExpBin(Opnd0, "%", Opnd1, 4, 5)

imprime(exp_men(Opnd0, Opnd1)):
    imprimeExpBin(Opnd0, "<", Opnd1, 1, 2)
imprime(exp_menIgual(Opnd0, Opnd1)):
    imprimeExpBin(Opnd0, "<=", Opnd1, 1, 2)
imprime(exp_mayor(Opnd0, Opnd1)):
    imprimeExpBin(Opnd0, ">", Opnd1, 1, 2)
imprime(exp_mayIgual(Opnd0, Opnd1)):
    imprimeExpBin(Opnd0, ">=", Opnd1, 1, 2)
imprime(exp_igual(Opnd0, Opnd1)):
    imprimeExpBin(Opnd0, "=", Opnd1, 1, 2)
imprime(exp_dist(Opnd0, Opnd1)):
    imprimeExpBin(Opnd0, "!", Opnd1, 1, 2)

```

```

imprime(exp_menos(Opnd0)):
    imprimeExpBin(" -", Opnd0, 6)
imprime(exp_not(Opnd0)):
    imprimeExpBin(" not ", Opnd0, 6)
imprime(exp_index(Opnd0)):
    imprimeExpBin(" Exp ", Opnd0, 6)
imprime(exp_reg(Opnd0)):
    imprimeExpBin(" Id ", Opnd0, 6)
imprime(exp_ind(Opnd0)):
    imprimeExpBin(" ^", Opnd0, 6)

imprimeExpUnario(Opnd0, Op, np0):
    print " "++Op++" "
    imprimeOpnd(Opnd0, np0)

imprimeExpBin(Opnd0, Op, Opnd1, np0, np1):
    imprimeOpnd(Opnd0, np0)
    print " "++Op++" "
    imprimeOpnd(Opnd1, np1)

imprimeOpnd(Opnd, MinPrior):
    if prioridad(Opnd) < MinPrior
        print (
            nl
        end if
    imprime(Opnd)
    ml
    if prioridad(Opnd) < MinPrior
        print (
            l
        end if

prioridad(exp_suma(_, _)): return 2
prioridad(exp Resta(_, _)): return 2
prioridad(exp_and(_, _)): return 3
prioridad(exp_or(_, _)): return 3
prioridad(exp_mul(_, _)): return 4
prioridad(exp_div(_, _)): return 4
prioridad(exp_mod(_, _)): return 4

prioridad(exp_asig(_, _)): return 0
prioridad(exp_men(_, _)): return 1
prioridad(exp_menorIgual(_, _)): return 1
prioridad(exp_mayor(_, _)): return 1
prioridad(exp_mayIgual(_, _)): return 1
prioridad(exp_igual(_, _)): return 1
prioridad(exp_dist(_, _)): return 1

prioridad(exp_menos(_)): return 5
prioridad(exp_not(_)): return 5
prioridad(exp_index(_)): return 6
prioridad(exp_reg(_)): return 6
prioridad(exp_ind(_)): return 6

```

Índice de figuras

Temporary page!

L^AT_EX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.

If you rerun the document (without altering it) this surplus page will go away, because L^AT_EX now knows how many pages to expect for this document.