Gramática de atributos final:

Partiendo de nuestra gramática de atributos final el primer paso en la construcción será hacer que esta sea apropiada para el análisis predictivo recursivo y l-atribuida (no tenga atributos heredados por la derecha). La gramática debe acondicionarse mediante la eliminación de recursión a izquierdas directa, y la factorización.

Agrupación de las siguientes gramáticas de atributos:

- Gramática de atributos de construcción de la tabla de símbolos
- Gramática de atributos de las restricciones contextuales
- Gramática de atributos de la traducción

Prog ::= program id PYCOMA Bloque PUNTO
Prog.err = Bloque.err
***Prog*.cod = *Bloque*.cod || stop**

Bloque ::= TBloque
TBloque.tsh = creaTS()
Bloque.err = TBloque.err
***Bloque*.cod = *TBloque*.cod**

Bloque ::= Tvar TBloque
TBloque.tsh = Tvar.ts
Bloque.err = TBloque.err v Tvar.err
***Bloque*.cod = *Tvar*.cod || *TBloque*.cod**

Tvar ::= var Tvar2
Tvar.ts = Tvar2.ts
Tvar.err = Tvar2.err
***Tvar*.cod = *Tvar2*.cod**

Tvar2 ::= id 2PUNTOS Tipo PYCOMA RTvar2
Tvar2.tsh = añadeID (RTvar2.ts, id.lex, Tipo.tipo, RTvar2.dir)
Tvar2.dirh = RTvar2.dir
Tvar2.errh = RTvar2.err v existeID(Tvar2.ts, id.lex)

RTvar2 ::= λ
RTvar2.dir = 0
RTvar2.ts = creaTS()
RTvar2.err = false

RTvar2 ::= Tvar2
RTvar2.tsh = Tvar2.ts
RTvar2.dirh = Tvar2.dir + 1
RTvar2.err = Tvar2.err

Tipo ::= integer

Tipo ::= boolean

TBloque ::= begin TBloque2 end

TBloque2.tsh = TBloque.tsh
TBloque.err = Tbloque2.err
***TBloque*.cod = *TBloque2*.cod**

TBloque2 ::= λ

TBloque2 ::= TSentencia TBloque2

TBloque21.tsh = TBloque20.tsh
TSentencia.tsh = TBloque20.tsh
TBloque20.err = TBloque21.err v TSentencia.err
TBloque20.cod = TSentencia.cod || TBloque21.cod

TSentencia ::= TAsig

TAsig.tsh = TSentencia.tsh
TSentencia.cod = TAsig.cod
TSentencia.err = TAsig.err

TSentencia ::= TRead

TRead.tsh = TSentencia.tsh
TSentencia.cod = TRead.cod
TSentencia.err = TRead.err

TSentencia ::= TWrite

TWrite.tsh = TSentencia.tsh
TSentencia.cod = TWrite.cod
TSentencia.err = TWrite.err

TRead ::= read TA id TC PYCOMA

TRead.err = ¬ existeID(TRead.tsh, id.lex)
***TRead*.cod = lee || desapila_dir(dameDir(*TRead.*tsh,id.lex))**

TWrite ::= write TA Text TC PYCOMA

Text.tsh = TWrite.tsh
TWrite.err = Text.err
***TWrite*.cod = *Text*.cod**

Text ::= texto

Text.err = false
***Text*.cod = apila(valorDe(texto.lex)) || escribe**

Text ::= id

Text.err = ¬ existeID(Text.tsh, id.lex)
***Text*.cod = apila_dir(dameDir(*Text*.tsh,id.lex)) || escribe**

TAsig ::= id ASIG Exp

Exp.tsh = TAsig.tsh
TAsig.err = ¬ existeID(TAsig.tsh, id.lex) v Exp.err v
(dameTipo(TAsig.tsh, id.lex) != Exp.tipo)
***TAsig*.cod = *Exp*.cod || desapila_dir(dameDir(*TAsig.*tsh,id.lex))**

```
Exp ::= ExpSimple RestoExpSimple
```
ExpSimple.tsh = Exp.tsh
RestoExpSimple.tsh = Exp.tsh
Exp.err = ExpSimple.err v RestoExpSimple.err
***Exp*.cod = *ExpSimple*.cod || RestoExpSimple.cod**

```
RestoExpSimple ::= λ
```
RestoExpSimple.err = false

```
RestoExpSimple ::= Comp ExpSimple
```
ExpSimple.tsh = RestoExpSimple.tsh
RestoExpSimple.err = ExpSimple.err v
(if (Comp.id == '=') v (Comp.id == '!=') then
RestoExpSimple.tipo != ExpSimple.tipo
else
(RestoExpSimple.tipo != integer) v (ExpSimple.tipo != integer))
***RestoExpSimple*.cod = *ExpSimple*.cod || *Comp*.op**

```
ExpSimple ::= RestoTerm Term
```
RestoTerm.tsh = ExpSimple.tsh
Term.tsh = ExpSimple.tsh
ExpSimple.err = RestoTerm.err v Term.err
ExpSimple.tipo = RestoTerm.tipo
ExpSimple.cod = RestoTerm.cod || Term.cod

```
RestoTerm ::= λ
```
RestoTerm.err = false

```
RestoTerm ::= OpAd Term
```
Term.tsh = RestoTerm.tsh
RestoTerm.tipo = OpAd.tipo
RestoTerm.err = Term.err v (Term.tipo != OpAd.tipo)
RestoTerm.cod = Term.cod || OpAd.cod

```
Term ::= RestoFact Fact
```
RestoFact.tsh = Term.tsh
Fact.tsh = Term.tsh
Term.err = RestoFact.err v Fact.err
Term.cod = RestoFact.cod || Fact.cod

```
RestoFact ::= λ
```
RestoFact.err = false

```
RestoFact ::= OpMul Term
```
Term.tsh = RestoFact.tsh
RestoFact.tipo = OpMul.tipo
RestoFact.err = Term.err v (Term.tipo != OpMul.tipo)
RestoFact.cod = Term.cod || OpMul.cod

Fact ::= numero
Fact.err = false
Fact.cod = apila(valorDe(numero))

Fact ::= true
Fact.err = false
Fact.cod = apila(true)

Fact ::= false
Fact.err = false
Fact.cod = apila(false)

Fact ::= id
Fact.err = ¬ existeID(Fact.tsh, id.lex)
Fact.cod = apila_dir(dameDir(Fact.tsh,id.lex))

Fact ::= *OpUn Fact*
Fact1.tsh = Fact0.tsh
Fact0.err = Fact1.err v Fact1.tipo != OpUn.tipo
Fact0.cod = Fact1.cod || OpUn.op

Fact ::= (*Exp*)
Exp.tsh = Fact.tsh
Fact.err = Exp.err
Fact.cod = Exp.cod

OpAd ::= +
OpAd.tipo = integer
OpAd.op = suma

OpAd ::= -
OpAd.tipo = integer
OpAd.op = resta

OpAd ::= or
OpAd.tipo = boolean
OpAd.op = or

OpMul ::= *
OpMul.tipo = integer
OpMul.op = multiplica

OpMul ::= /
OpMul.tipo = integer
OpMul.op = divide

OpMul ::= and
OpMul.tipo = Boolean
OpMul.op = and

OpUn ::= +
OpMul.tipo = integer
OpUn.op = positivo

OpUn ::= -

OpMul.tipo = integer

OpUn.op = negativo

OpUn ::= not

OpMul.tipo = Boolean

OpUn.op = not

Comp ::= <=

Comp.id = <=.lex

Comp.op = menor_igual

Comp ::= >=

Comp.id = >=.lex

Comp.op = mayor_igual

Comp ::= <

Comp.id = <.lex

Comp.op = menor

Comp ::= >

Comp.id = >.lex

Comp.op = mayor

Comp ::= =

Comp.id = =.lex

Comp.op = igual

Comp ::= !=

Comp.op = distinto

Comp.id = !=.lex