Gramática Restricciones Contextuales 2º Cuatrimestre

Gramática de atributos que formaliza la comprobación de las restricciones contextuales.

Lista de ***atributos semánticos***:

- ***tipo****:* Es un atributo sintetizado. Puede ser Integer, Boolean, o tipoErr
- ***tipoErr:*** Es el valor que empleamos cuando es necesario para representar el error en el sistema de tipos.
- ***tipoh****:* Es un atributo heredado. Tiene la información heredada de tipo
- ***tsp:*** Es un atributo sintetizado. En esta tabla se guardan las variables creadas en el programa así como sus tipos y dirección de memoria.
- ***tsph:*** Es un atributo heredado y se emplea para difundir la tabla de símbolos por toda la gramática.
- ***err:*** Es un atributo sintetizado. True si hay algún error en el programa.
- ***errh:*** Es un atributo heredado. Tiene la información heredada de err

A continuación una lista de las ***funciones semánticas*** adicionales utilizadas en la especificación.

tipoInteger(tipo1,tipo2)::= if (tipo1=tip2) AND (tipo1=Integer) then
Integer
else
tipoErr

tipoBool(tipo1,tipo2)::= if (tip1=tip2) AND (tip1=Boolean) then
Boolean
else
tipoErr

tipoIgual(tipo1,tipo2)::= if (tip1=tip2) AND (tip1<>err) then
Boolean
else
tipoErr

tipoRelacion (tipo1,tipo2, op) *::=*
if (op Є {'=','<>'}) then
(tipo1 = tipo2) AND ((tipo1 = Entero) OR (tipo1=Boolean))
else
(tipo1 = Entero) AND (tipo2 = Entero)

errorEnAsignacion(tabla,id,tipo)::= ¬existeID(tabla, id) OR tipoID (tabla, id) <> tipo

compatibles(tipo1,tipo2,tabla)*:* comprueba la compatibilidad de los tipos introducidos utilizando la tabla para comprobar las referencias. Devuelve un booleano.

Referencia(tipo1,tabla): Devuelve el tipo al que referencia tipo1. En caso de que no exista o se produzca algun error, devuelve tipoError.

Longitud(array[]): entero Devuelve el tamaño de un array.

Estructura general

Prog ::= Cabecera Decs Bloque
Cabecera.err = Cabecera.err v Decs.err v Bloque.err

Cabecera ::= PROGRAM id PYCOMA

Decs ::= DTipos Vars Procs
Decs.err = DTipos.err v Vars.err v Decs.err v Procs.err

Decs ::= DTipos Vars
Decs.err = DTipos.err v Vars.err

Decs ::= DTipos Procs
Decs.err = DTipos.err v Procs.err

Decs ::= Vars Procs
Decs.err = Vars.err v Procs.err

Decs ::= DTipos
Decs.err = DTipos.err

Decs ::= Vars
Decs.err = Vars.err

Decs ::= Procs
Decs.err = Procs.err

Declaración de tipos

DTipos ::= SECTIPOS RTipos
DTipos.err = RTipos.err

RTipos ::= RTipos2 RTipos
$RTipos_0.err = RTipos2.err \vee RTipos_1.err$

RTipos ::= λ
RTipos.err = false

RTipos2 ::= id IGUAL Tipo PYCOMA
RTipos2.err = (existeID(RTipos2.tsp, id.lex) ^ (RTipos2.tsp[id.lex].nivel == RTipos2.nivelh)) v Tipo.err

Tipo ::= TIPENT
Tipo.err = false

Tipo ::= TIPBOOL
Tipo.err = false

Tipo::= id
Tipo.err = if existeID(Tipo.tsp, id.lex) then
(Tipo.tsp[id.lex].clase < > Tipo)
else
True

Tipo::= Array[0..numero] of Tipo
$Tipo_0$.err =$Tipo_1$.err v Referencia($Tipo_1$.tipo,$Tipo_0$.tsph)

Tipo::= TPUNTER Tipo
$Tipo_0$.err =$Tipo_1$.err

Declaración de variables

Vars ::= VAR Tvar2
Vars.err= Tvar2.err

Tvar2 ::= RTvar2 Tvar2
$Tvar2_0$.err = RTvar2.err v $Tvar2_1$.err

Tvar2 ::= λ
Tvar2.err = false

RTvar2 ::= id COMA RTvar2
$RTvar2_0$.err = (existeID($RTvar2_0$.tsph, id.lex) ^ $RTvar2_0$.tsp[id.lex].nivel == $RTvar2_0$.nivelh) v (existeID($RTvar2_1$.tsp,id.lex) v $RTvar2_1$.err

RTvar2 ::= id 2PUNTOS Tipos PYCOMA
RTvar2.err = (existeID(RTvar2.tsp, id.lex) ^ RTvar2.tsp[id.lex].nivel == RTvar2.nivelh) v Tipos.err

Tipos ::= TIPENT
Tipos.err = false

Tipos ::= TIPBOOL
Tipos.err= false

Tipos ::= id
Tipo.err = if existeID(Tipo.tsp, id.lex) then
(Tipo.tsp[id.lex].clase < > Tipo)
else
True

Declaración de procedimientos

Procs ::= TDProc Procs
Procs$_0$.err = TDProc.err v Procs$_1$.err

TDProc ::= PROC id Params PYCOMA BloqProc
TDProc.err= Params.err v BloqProc.err v (existeID(TDProc.tsp, id.lex) ^
(TDProc.tsp[id.lex].nivel == TDProc.nivelh))

Params ::= PA ListaParams PC
Params.err = ListaParams.err

Params ::= λ
Params.err = false

ListaParams ::= Params2
ListaParams.err = Params2.err

ListaParams ::= VAR Params2
ListaParams.err = Params2.err

ListaParams ::= Params2 PYCOMA Listaparams
ListaParams$_0$.err = Params2.err v Listaparams$_1$.err

ListaParams ::= VAR Params2 PYCOMA ListaParams
ListaParams$_0$.err = Params2.err v Listaparams$_1$.err

Params2 ::= id COMA Params2
Params2$_0$.err = (existeID(Params2$_0$.tsph, id.lex) ^ (Params2$_0$.tsp[id.lex].nivel ==
Params2$_0$.nivelh)) v existeID(Params2$_1$.tsp, id.lex) v Params2$_1$.err

Params2 ::= id 2PUNTOS Tipos
Params2.err = (existeID(Params2.tsph, id.lex) ^ (Params2.tsp[id.lex].nivel ==
Params2.nivelh)) v Tipos.err

BloqProc ::= Decs2 Bloque
BloqProc.err = Decs2.err v Bloque.err

Decs2 ::= Vars
Decs2.err = Vars.err

Decs2 ::= λ
Decs2.err = false

Cuerpo del programa principal

Bloque ::= INICIO TBloque2 FIN
Bloque.err = TBloque2.err

TBloque2 ::= TSentencia TBloque2
TBloque2$_0$.err = TSentencia.err v TBloque2$_1$.err

TBloque2 ::= λ
TBloque2.err = false

TSentencia ::= TAsig
TSentencia.err = TAsig.err

TSentencia ::= TRead
TSentencia.err = TRead.err

TSentencia ::= TWrite
TSentencia.err = TWrite.err

TSentencia ::= TNPunt
TSentencia.err = TNPunt.err

TSentencia ::= TLiberar
TSentencia.err = TLiberar.err

TSentencia ::= TLlamadaProc
TSentencia.err = TLlamadaProc.err

TSentencia ::= TIf
TSentencia.err = TIf.err

TSentencia ::= TWhile
TSentencia.err = TWhile.err

TIf ::= SI PA Exp PC ENTONCES INICIO TBloque2 FIN SINO INICIO
TBloque2 FIN
TIf.err = Expl.err v TBloque2$_0$.err v TBloque2$_1$.err v (Exp.tipo.t < > Boolean)

TIf ::= SI PA Exp PC ENTONCES INICIO TBloque2 FIN
TIf.err = Exp.err v TBloque2.err v (Exp.tipo.t < > Boolean)

TWhile ::= MIENTRAS PA Exp PC HACER INICIO TBloque2 FIN
TWhile.err = Exp.err v TBloque2.err v (Exp.tipo.t < > Boolean)

TLlamadaProc ::= id PA Params3 PC PYCOMA
Params3.paramsh = TLlamadaProc.tpsh[id.lex].params
TLlamadaProc.err = ¬existeID(TLlamadaProc.tsph, id.lex) v
(TLlamadaProc.tsph[id.lex].clase < > proc) v Params3.err v

(longitud(TLlamadaProc.tsph[id.lex].params) != Params3.nparams) v
(TLlamadaProc.tsph[id.lex].nivelh < > TLlamadaProc.nivelh)

Params3 ::= ListaParams3
ListaParams3.paramsh = Params3.paramsh
Params3.err = ListaParams3.err

Params3 ::= λ
ListaParams3.err = false

ListaParams3 ::= Exp
ListaParams3.err = (ListaParams3.nparams < > 1) v
(ListaParams3.paramsh[1].modo == variable ^ Exp.modo == valor) v
¬compatibles(ListaParams3.paramsh[1].tipo, Exp.tipo, ListaParams3.tsph) v Exp.err

ListaParams3 ::= Exp COMA ListaParams3
ListaParams3$_0$.err = (ListaParams3$_0$.nparams < > ListaParams3$_1$.nparams + 1) v
(ListaParams3$_0$.paramsh[1].modo == variable ^ Exp.modo= valor) v
¬compatibles(ListaParams3$_0$.paramsh[1].tipo, Exp.tipo, ListaParams3$_0$.tsph) v
Exp.err v ListaParams3$_1$.err

TRead ::= LEER PA id PC PYCOMA
TRead.err = ¬existeID(TRead.tsph, id.lex) v (TRead.tsph[id.lex].clase < > variable)

TWrite ::= ESCRIBIR PA id PC PYCOMA
TWrite.err = ¬existeID(TWrite.tsph, id.lex) v ((TWrite.tsph[id.lex].clase < > variable) ^ ((TWrite.tsph[id.lex].tipo.t == Entero) v (TWrite.tsph[id.lex].tipo.t == boolean))) v (TWrite.tsph[id.lex].nivel != TWrite.nivelh)

TNPunt ::= NUEVO PA id PC PYCOMA
TPunt.err = ¬existeID(TPunt.tsph, id.lex) v (TPunt.tsph[id.lex].tipo.t < > puntero) v (TPunt.tsph[id.lex].nivel != TPunt.nivelh)

TLiberar ::= LIBERAR PA id PC PYCOMA
TLiberar.err = ¬existeID(TLiberar.tsph, id.lex) v (TLiberar.tsph[id.lex].tipo.t < > puntero) v (TLiberar.tsph[id.lex].nivel != TLiberar.nivelh)

TAsig ::= Descriptor ASIG Exp
TAsig.err = ¬compatibles(Descriptor.tipo, Exp.tipo, TAsig.tsph) v Descriptor.err v Exp.err

Descriptor ::= Descriptor2
Descriptor.err = Descriptor2.err
Descriptor.tipo = Descriptor2.tipo

Descriptor ::= Descriptor2 CA Exp CC
Descriptor.err = Descriptor2.err v Exp.err
Descriptor.tipo = if (Descriptor2.tipo.t = Array ^ Exp.tipo.t = Entero) then
referencia(Descriptor2.tipo.tBase, Descriptor.tsph)

else
<t: tipoError>

Descriptor2 ::= id
Descriptor2.error = ¬existeID(Descriptor2.tsph,id.lex) v
(Descriptor2.tsph[id.lex].nivel != Descriptor2.nivelh)
Descriptor2.tipo =
if (existeID(Descriptor2.tsph, id.lex) ^ Descriptor2.tsph[id.lex].clase = variable) then
referencia(Descriptor2.tsph[id.lex].tipo, Descriptor2.tsph)
else
<t: tipoError>

Descriptor2 ::= ^Descriptor2
Descriptor$_0$.err = (Descriptor$_1$.tipo.t != puntero)
Descriptor$_0$.tipo = if (Descriptor$_1$.tipo.t = puntero) then
referencia(Descriptor2$_1$.tipo.tBase, Descriptor$_0$.tsph)
else
<t: tipoError>

Exp ::= Exp OpRelacion ExpSum
Exp$_0$.err = Exp$_1$.err v ExpSum.err
Exp$_0$.tipo = if tipoRelacion(Exp$_1$.tipo.t, ExpSum.tipo.t, OpRelacion) then
<t: Boolean>
else
<t: tipoError>

Exp ::= ExpSum
Exp.tipo = ExpSum.tipo
Exp.err = ExpSum.err

ExpSum ::= ExpSum OpAd ExpProd
ExpSum$_0$.tipo = tipoNum(ExpSum$_1$.tipo.t, ExpProd.tipo.t)
ExpSum$_0$.err = ExpSum$_1$.err v ExpProd.err v
(¬compatibles(ExpSum$_1$.tipo,ExpProd.tipo, ExpSum$_0$.tsph)

ExpSum ::= ExpSum OR ExpProd
ExpSum$_0$.err = ExpSum$_1$.err v ExpProd.err v
(¬compatibles(ExpSum$_1$.tipo,ExpProd.tipo, ExpSum$_0$.tsph)
ExpSum$_0$.tipo = tipoBoolean(ExpSum$_1$.tipo.t, ExpSum$_0$.tsph)

ExpSum ::= ExpProd
ExpSum.err = ExpProd.err
ExpSum.tipo = ExpProd.tipo

ExpProd ::= ExpProd OpProd ExpFact
ExpProd$_0$.err = ExpProd$_1$.err v ExpFact.err v
(¬compatibles(ExpProd$_1$.tipo,ExpFact.tipo, ExpProd$_0$.tsph)
ExpProd$_0$.tipo = tipoNum(ExpProd$_1$.tipo.t, ExpFact.tipo.t)

ExpProd ::= ExpProd AND ExpFact

ExpProd$_0$.err = ExpProd$_1$.err v ExpFact.err v
(¬compatibles(ExpProd$_1$.tipo,ExpFact.tipo, ExpProd$_0$.tsph)
ExpProd$_0$.tipo = tipoBoolean(ExpProd$_1$.tipo.t, ExpFact.tipo.t)

ExpProd ::= ExpFact
ExpProd.err = ExpFact.err
ExpProd.tipo = ExpFact.tipo

ExpFact ::= PA Exp PC
ExpFact.err = Exp.err
ExpFact.tipo = Exp.tipo

ExpFact:= OpAd ExpFact
ExpFact$_0$.err = ExpFact$_1$.err
ExpFact$_0$.tipo = if (ExpFact$_1$.tipo.t = Entero) then
ExpFact$_0$.tipo.t
else
<t: tipoError>

ExpFact := numero
ExpFact.err = false
ExpFact.tipo = Entero

ExpFact := True
ExpFact.err = false
ExpFact.tipo = Boolean

ExpFact := False
ExpFact.err = false
ExpFact.tipo = Bolean

ExpFact ::= Not ExpFact
ExpFact$_0$.err = false
ExpFact$_0$.tipo = if (ExpFact$_1$.tipo.t = Boolean) then
ExpFact$_0$.tipo.t
else
<t: tipoError>

ExpFact ::= Descriptor
ExpFact.err = Descriptor.err
ExpFact.tipo = if (Descriptor.tipo.t < > tipoError) then
tipoDeID(ExpFact.tsph, Descriptor.tipo)
else
<t: tipoError>