Gramática de Atributos para la Traducción (2º Cuatrimestre)

Funciones semánticas adicionales:

- *apilaDirRetorno: (r: Entero)**:*** Almacena la dirección de retorno r.

- *prologo: (nivel: Entero, tamVarLocales: Entero)**:*** Guarda el valor anterior que tenía el display asociado con el procedimiento, fija el valor del display para la activación actual y reserva espacio para los datos locales.

- *epilogo: (nivel: Entero)**:*** Libera el espacio para variables locales y parámetros, recupera el antiguo display y apila la dirección de retorno.

- *direccionPFormal: (pFormal: Parametro)**:*** Suma la dirección del parámetro introducido a la cima de la pila.

- *pasoParametro: (modol: Modo, param: Parametro)**:*** Si el parámetro es variable se copia el valor en el registro de activación. Si es un valor se desapila ese valor en el registro de activación.

- *accesoVariable: (id: Identificador):* Desplazamiento relativo al valor del display del bloque en el que reside la variable.

- *longAccesoVariable: (id: Identificador):* Devuelve la longitud de la función *accesoVariable.*

- *inicio: (numNiveles: Entero, tamDatos: Entero):* Se fijan el display a 0 y el contador de programa

Constantes utilizadas:

- ***longInicio*** *= 4*
- ***longApilaRetorno*** *= 5*
- ***longPrologo*** *= 13*
- ***longEpilogo*** *= 13*
- ***longPasoParametro*** *= 1*
- ***longDireccionPFormal*** *= 2*
- ***inicioPaso*** *= apila-dir(0) || apila (3) || suma*
- ***longInicioPaso*** *= 3*
- ***finPaso*** *= desapila*
- ***longFinPaso*** *= 1*

Gramática de atributos:

Prog ::= Cabecera Decs Bloque
Prog.cod = inicio(Decs.nivel, Decs.dir) || ir_a (Decs.etq) || Decs.cod || Bloque.cod || stop
Decs.etqh = longInicio +1;
Bloque.etqh = Decs.etq

Cabecera ::= PROGRAM id PYCOMA

Sección de declaraciones

Decs ::= DTipos Vars Procs
Procs.etqh = Decs.etqh
Decs.etq = Procs.etq
Decs.cod = Procs.cod

Decs ::= DTipos Vars
Decs.etq = Decs.etqh
Decs.cod = λ

Decs ::= DTipos Procs
Procs.etqh = Decs.etqh
Decs.etq = Procs.etq
Decs.cod = Procs.cod

Decs ::= Vars Procs
Procs.etqh = Decs.etqh
Decs.etq = Procs.etq
Decs.cod = Procs.cod

Decs ::= Vars
Decs.etq = Decs.etqh
Decs.cod = λ

Decs ::= Procs
Procs.etqh = Decs.etqh
Decs.etq = Procs.etq
Decs.cod = Procs.cod

Decs ::= λ
Decs.etq = Decs.etqh
Decs.cod = λ

Declaración de procedimientos

Procs ::= TDProc Procs
TDProc.etqh = $Procs_0$.etqh
$Procs_1$.etqh = TDProc.etq
$Procs_0$.etq = $Procs_1$.etq
$Procs_0$.cod = TDProc.cod || $Procs_1$.cod

Procs ::= TDProc
TDProc.etqh = Procs.etqh
Procs.etq = TDProc.etq
Procs.cod = TDProc.cod

TDProc ::= PROC id Params PYCOMA BloqProc
BloqProc.etqh = TDProc.etqh
TDProc.etq = BloqProc.etq
TDProc.cod = BloqProc.cod

BloqProc ::= Decs2 Bloque
BloqProc.inicio = Bloque.etq
Bloque.etqh= BloqProc.etqh + longPrologo
BloqProc.etq = Bloque.etq + longEpilogo + 1
BloqProc.cod = prologo(BloqProc.nivelh, Decs2.dir) || Bloque.cod
epilogo(BloqProc.nivelh) || ir-ind

Cuerpo del programa

Bloque ::= INICIO TBloque2 FIN
Bloque.cod = TBloque2.cod
TBloque2.etqh = Bloque.etqh
Bloque.etq = TBloque2.etq

TBloque2 ::= λ
Tbloque2.cod = λ
Tbloque2.etq = Cuerpo.etqh

TBloque2 ::= TSentencia TBloque2
$TBloque2_0$.cod= TSentencia.cod || $TBloque2_1$.cod
TSentencia.etqh = $TBloque2_0$.etqh
$TBloque2_1$.etqh = TSentencia.etq
$TBloque2_0$.etq = $TBloque2_1$.etq

TSentencia ::= TAsig
TSentencia.cod = TAsig.cod
TAsig.etqh = TSentencia.etqh
TSentencia.etq = TAsig.etq

TSentencia ::= TRead
TSentencia.cod = TRead.cod
TRead.etqh = TSentencia.etqh
TSentencia.etq = TRead.etq

TSentencia ::= TWrite
TSentencia.cod = TWrite.cod
TWrite.etqh = TSentencia.etqh
TSentencia.etq = TWrite.etq

TSentencia ::= TNPunt
TSentencia.cod = TNPunt.cod
TNPunt.etqh = TSentencia.etqh
TSentencia.etq = TNPunt.etq

TSentencia ::= TLiberar
TSentencia.cod = TLiberar.cod
TLiberar.etqh = TSentencia.etqh
TSentencia.etq = TLiberar.etq

TSentencia ::= TLlamadaProc
TSentencia.cod = TLlamadaProc.cod
TLlamadProc.etqh = TSentencia.etqh
TSentencia.etq = TLlamadaProc.etq

TSentencia ::= TIf
TSentencia.cod = TIf.cod
TIf.etqh = TSentencia.etqh
TSentencia.etq = TIf.etq

TSentencia ::= TWhile
TSentencia.cod = TWhile.cod
TWhile.etqh = TSentencia.etqh
TSentencia.etq = TWhile.etq

TIf ::= SI PA Exp PC ENTONCES INICIO TBloque2 FIN SINO INICIO TBloque2 FIN
TIf.cod= Exp.cod || ir_f(TBloque2$_0$.etq+1)|| TBloque2$_0$.cod|| ir_a(TBloque2$_1$.etq)|| TBloque2$_1$.cod
Exp.etqh= TIf.etqh
TBloque2$_0$.etqh= Exp.etq+1
TBloque2$_1$.etqh= TBloque2$_0$.etq +1
TIf.etq= TBloque2$_1$.etq
Exp.parh= false

TIf ::= SI PA Exp PC ENTONCES INICIO TBloque2 FIN
TIf.cod= Exp.cod || ir_f(TBloque2.etq+1)||TBloque2.cod
Exp.etqh= TIf.etqh
TBloque2.etqh= Exp.etq+1
TIf.etq= TBloque2.etq
Exp.parh= false

TWhile ::= MIENTRAS PA Exp PC HACER INICIO TBloque2 FIN
TWhile.cod= Exp.cod ||ir_f(TBloque2.etq+1)|| TBloque2.cod||ir_a(TWhile.etqh)
Exp.etqh= TWhile.etqh
TBloque2.etqh= Exp.etq +1
TWhile.etq= TBloque2.etq +1
Exp.parh= false

TLlamadaProc ::= id PA Params3 PC PYCOMA
TLlamadaProc.cod= apilaDirRetorno(TLlamadaProc.etq)|| Params3.cod||
ir_a(TLlamadaProc.tsph[id.lex].inicio)
Params3.etqh= TLlamadaProc.etqh+ longApilaRetorno
TLlamadaProc.etq= Params3.etq +1

Params3 ::= ListaParams3
Params3.cod = inicioPaso || ListaParams3.cod || finPaso
ListaParams3.etqh = Params3.etqh +longInicioPaso
Params3.etq = ListaParams3.etq + longFinPaso

Params3 ::= λ
Params3.cod = λ
Params3.etq = Params3.etqh

ListaParams3 ::= Exp
ListaParams3.cod = copia ||Exp.cod||
pasoParametro(Exp.modo,ListaParams3.paramsh[1])
Exp.etqh = ListaParams3.etq +1
ListaParams3.etq = Exp.etq + longPasoParametro
Exp.parh = ListaParams3.paramsh[1].modo = variable

ListaParams3 ::= Exp COMA ListaParams3
ListaParams3$_0$.cod = ListaParams3$_1$.cod || copia ||
direccionPFormal (ListaParams3$_0$.paramsh[ListaParams3$_0$.nparams]) ||
Exp.cod || pasoParametro(Exp.modo,
ListaParams3$_0$.paramsh[ListaParams3$_0$.nparams])
ListaParams3$_1$.etqh = ListaParams3$_0$.etqh
Exp.etqh = ListaParams3$_1$.etq +1
ListaParams3$_0$.etq = Exp.etq + longDireccionPFormal + longPasoParametro
Exp.parh=ListaParams3.paramsh[ListaParams3$_0$.nparams].modo==variable

TRead ::= LEER PA id PC PYCOMA
TRead.cod=lecturaPantalla(TRead.tsph[id.lex].dir)
TRead.etq=TRead.etqh

TWrite ::= ESCRIBIR PA id PC PYCOMA
TWrite.cod=escrituraPantalla(cimaPila())
TWrite.etq=TWrite.etqh

TNPunt ::= NUEVO PA id PC PYCOMA
TNPunt.cod= reservar(TNPunt.tsph[id.lex].tam) || desapila_ind
TNPunt.etq= TNPunt.etqh

TLiberar ::= LIBERAR PA id PC PYCOMA
TLiberar.cod= liberar(TLiberart.tsph[id.lex].tam)
TLiberar.etq= TLiberar.etqh

TAsig ::= Descriptor ASIG Exp
TAsig.cod = if compatible(Descriptor.tipo, <t: Entero>, Exp.tsph) v
compatible(Descriptor.tipo, <t: Boolean>, Exp.tsph) then Descriptor.cod || Exp.cod ||
desapilaIndice() else Descriptor.cod || Exp.cod || mueve(Descriptor.tipo.tam)
Descriptor.etqh = TAsig.etqh
Exp.etqh = Descriptor.etq
TAsig.etq = Exp.etq + 1
Exp.parh = false

Descriptor ::= Descriptor2
Descriptor.cod = Descriptor2.cod
Descriptor2.etqh = Descriptor.etqh
Descriptor.etq = Descriptor2.etq

Descriptor2 ::= id
Descriptor2.cod = accesoVar(Descriptor2.tsph[id.lex])
Descriptor2.etq = Descriptor2.etqh + longAccesoVar (Descriptor2.tsph[id.lex])

Descriptor2 ::= Descriptor2[Exp]
$Descriptor2_0.cod = Descriptor2_1.cod$ || Exp.cod ||apila($Descriptor2_1.tipo.tBase.tam$)
|| multiplica || suma
$Descriptor2_1.etqh = Descriptor2_0.etqh$
Exp.etqh = $Descriptor2_1.etq$
$Descriptor2_0.etq$ = Exp.etq + 3

Descriptor2 ::= ^Descriptor2
$Descriptor2_0.cod = Descriptor2_1.cod$ || apilaInd
$Descriptor2_1.etqh = Descriptor2_0.etqh$
$Descriptor2_0.etq = Descriptor2_1.etq$ +1

Exp ::= Exp OpRel ExpSum
$Exp_0.cod$= $Exp_1.cod$||ExpSum.cod||OpRel.op
$Exp_1.etqh$= $Exp_0.etqh$
ExpSum.etqh= $Exp_1.etq$
$Exp_0.etq$= ExpSum.etq+1
ExpSum.parh= false
$Exp_1.parh$= false

Exp ::= ExpSum
Exp.cod= ExpSum.cod
ExpSum.etqh= Exp.etqh
Exp.etq= ExpSum.etq
ExpSum.parh= Exp.parh

ExpSum ::= ExpSum OpAd ExpProd
$ExpSum_0.cod$= $ExpSum_1.cod$||ExpProd.cod||OpAd.op
$ExpSum_1.etqh$= $ExpSum_0.etqh$
ExpProd.etqh= $ExpSum_1.etq$
$ExpSum_0.etq$= ExpProd.etq+1
ExpProd.parh= false
$ExpSum_1.parh$= false

ExpSum ::= ExpSum OR ExpProd
$ExpSum_0.cod$= $ExpSum_1.cod$||ExpProd.cod||or
$ExpSum_1.etqh$= $ExpSum_0.etqh$
ExpProd.etqh= $ExpSum_1.etq$
$ExpSum_0.etq$= ExpProd.etq+1
ExpProd.parh= false
$ExpSum_1.parh$= false

ExpSum ::= ExpProd
ExpSum.cod= ExpProd.cod
ExpProd.etqh= ExpSum.etqh
ExpSum.etq= ExpProd.etq
ExpProd.parh= ExpSum.parh

ExpProd ::= ExpProd OpProd ExpFact
$ExpProd_0$.cod= $ExpProd_1$.cod||ExpFact.cod||OpProd.op
$ExpProd_1$.etqh= $ExpProd_0$.etqh
ExpFact.etqh= $ExpProd_1$.etq
$ExpProd_0$.etq= ExpFact.etq+1
ExpFact.parh= false
$ExpProd_1$.parh= false

ExpProd ::= ExpProd AND ExpFact
$ExpProd_0$.cod= $ExpProd_1$.cod||ExpFact.cod|| and
$ExpProd_1$.etqh= $ExpProd_0$.etqh
ExpFact.etqh= $ExpProd_1$.etq
$ExpProd_0$.etq= ExpFact.etq+1
ExpFact.parh= false
$ExpProd_1$.parh= false

ExpProd ::= ExpFact
ExpProd.cod= ExpFact.cod
ExpFact.etqh=ExpProd.etqh
ExpProd.etq= ExpFact.etq
ExpFact.parh= ExpProd.parh

ExpFact ::= (Exp)
ExpFact.cod= Exp.cod
Exp.etqh= ExpFact.etqh
ExpFact.etq= Exp.etq
Exp.parh= ExpFact.parh

ExpFact ::= OpAd ExpFact
$ExpFact_0$.cod = if (OpAd.op==suma) then $ExpFact_1$.cod
else $ExpFact_1$.cod) || resta
ExpFact.etq = ExpFact.etqh +1

ExpFact ::= numero
ExpFact.cod = apila(valorDe(numero.lex))
ExpFact.etq = ExpFact.etqh +1

ExpFact ::= True
ExpFact.cod= apila(True)
ExpFact.etq= ExpFact.etqh +1

ExpFact ::= False
ExpFact.cod= apila(False)
ExpFact.etq= ExpFact.etqh +1

ExpFact ::= Not ExpFact
$ExpFact_0$.cod= $ExpFact_1$.cod || negacion
$ExpFact_1$.etqh= $ExpFact_0$.etqh
$ExpFact_0$.etq= $ExpFact_1$.etq+1
$ExpFact_1$.parh= false

ExpFact ::= Descriptor
ExpFact.cod= if (compatible(Descriptor.tipo,<t:Integer>, ExpFact.tsph) v compatible(Descriptor.tipo,<t:Bool>, ExpFact.tsph)) ^ ¬ExpFact.parh then Descriptor.cod||apila_ind else Descriptor.cod
Descriptor.etqh= ExpFact.etqh
ExpFact.etq= Descriptor.etq + (if (compatible(Descriptor.tipo,<t:Integer>, ExpFact.tsph) v compatible(Descriptor.tipo,<t:Bool>, ExpFact.tsph)) ^ ¬ExpFact.parh then 1 else 0)