Regla	Producción	Acción semántica
Programa	Lista_sentencias PUNTO	
Lista_sentencia	Lista_sentencias Sentencia PTOCOMA	
	Sentencia PTOCOMA	
Sentencia	VAL ID = Exp_Rel	
	Exp_Rel	
	FUN MatchingFunc	
MatchingFunc	ID ID CurryingList	
	ID TuplePattern IGUAL Exp_Rel	
CurryingList	ID CurryingList	
	IGUAL Exp_Rel	
TuplePattern	PARABRE TuplePattern2 PARCIERRA	
TuplePattern2	TuplePattern2 COMA ID	

Exp_Rel	Exp_Rel:e1 OPREL Exp_cons:e2	Attrs t = tipoMasEspecifico(e1,e2);
		if (t!=null){
		if (esTipoTextOrNum(t)){
		RESULT = new Attrs("boolean");
		}else{
		throw new Exception("String or num expected.");
		}
		}else{
		throw new Exception("String or num expected.");
		}

	Exp_Rel:e1 DISTINTO Exp_cons:e2	Attrs t = tipoMasEspecifico(e1,e2);
		if (t!=null){
		t = tipoMasEspecifico(t,crearPolitipo("politipo con igualdad"));
		if (t!=null && !t.getType().equals("fun")){
		RESULT = new Attrs("boolean");
		}else{
		throw new Exception("Function does not admit equality.");
		}
		}else{
		throw new Exception("Types are not compatibles.");
		}

	Exp_Rel:e1 IGUAL Exp_cons:e2	Attrs t = tipoMasEspecifico(e1,e2);
		if (t!=null){
		t = tipoMasEspecifico(t,crearPolitipo("politipo con igualdad"));
		if (t!=null && !t.getType().equals("fun")){
		RESULT= new Attrs("boolean");
		}
		else{
		throw new Exception("Function does not admit equality.");
		}
		}else{
		throw new Exception("Types are not compatibles.");
		}

	Exp_cons	RESULT = e;

Regla	Producción	Acción semántica

Exp_cons	Exp_add:e1 CONS Exp_cons:e2	Attrs t = tipoMasEspecifico(new Attrs("list",e1),e2);
		if (t!=null){
		RESULT = t;
		}
		else{
		throw new Exception("The list has elements of different types.");
		}

	Exp_add:e	RESULT = e;

Exp_add	Exp_add:e1 OPADD Exp_mul:e2	Attrs t = tipoMasEspecifico(e1,e2);
		if (t!=null){
		t = tipoMasEspecifico(t,crearPolitipo("politipo numerico"));
		if (t!=null){
		RESULT = t;
		}else{
		throw new Exception("Integer or real expected.");
		}
		}else{
		throw new Exception("Integer or real expected.");
		}

	Exp_add:e1 ORELSE Exp_mul:e2	Attrs t = tipoMasEspecifico(e1,e2);
		if (t!=null){
		t = tipoMasEspecifico(t,new Attrs ("boolean"));
		if (t!=null){
		RESULT = t;
		}else{
		throw new Exception("Boolean expected.");
		}
		}else{
		throw new Exception("Boolean expected.");
		}

	Exp_add:e1 CONCAT Exp_mul:e2	Attrs t = tipoMasEspecifico(e1,e2);
		if (t!=null){
		t = tipoMasEspecifico(t,new Attrs ("string"));
		if (t!=null){
		RESULT = t;
		}else{
		throw new Exception("String expected.");
		}
		}else{
		throw new Exception("String expected.");
		}

	Exp_mul:e	RESULT = e;

Exp_mul	Exp_mul:e1 MUL Exp_un:e2	Attrs t = tipoMasEspecifico(e1,e2);
		if (t!=null){
		t = tipoMasEspecifico(t,crearPolitipo("politipo numerico"));
		if (t!=null){
		RESULT = t;
		}else{
		throw new Exception("Integer or real expected.");
		}
		}else{
		throw new Exception("Integer or real expected.");
		}

Regla	Producción	Acción semántica

	Exp_mul:e1 DIVREAL Exp_un:e2	Attrs t = tipoMasEspecifico(e1,e2);
		if (t!=null){
		t = tipoMasEspecifico(t,new Attrs("real"));
		if (t!=null){
		RESULT = t;
		}else{
		throw new Exception("Real expected.");
		}
		}else{
		throw new Exception("Real expected.");
		}

	Exp_mul:e1 DIVINT Exp_un:e2	Attrs t = tipoMasEspecifico(e1,e2);
		if (t!=null){
		t = tipoMasEspecifico(t,new Attrs("int"));
		if (t!=null){
		RESULT = t;
		}else{
		throw new Exception("Integer expected.");
		}
		}else{
		throw new Exception("Integer expected.");
		}

	Exp_mul:e1 MOD Exp_un: e2	ttrs t = tipoMasEspecifico(e1,e2);
		if (t!=null){
		t = tipoMasEspecifico(t,new Attrs("int"));
		if (t!=null){
		RESULT = t;
		}else{
		throw new Exception("Integer expected.");
		}
		}else{
		throw new Exception("Integer expected.");
		}

	Exp_mul:e1 ANDALSO Exp_un:e2	Attrs t = tipoMasEspecifico(e1,e2);
		if (t!=null){
		t = tipoMasEspecifico(t,new Attrs("boolean"));
		if (t!=null){
		RESULT = t;
		}else{
		throw new Exception("Boolean expected.");
		}
		}else{
		throw new Exception("Boolean expected.");
		}

	Exp_un:e	Exp_mul.tipo = "boolean"
Exp_un	NOT Exp_fun:fun	}
	MINUS Exp_fun:fun	else
	Exp_fun:e	Error

Exp_fun	Exp_fun:f Exp_atom:atom	Attrs aux = tipoMasEspecifico(f,new Attrs("fun",
		atom,crearPolitipo("politipo")));
		if (aux!=null){
		RESULT = aux.getRange();
		}
		else{
		throw new Exception("Not a valid function.");
		}

	Exp_atom:atom	RESULT = atom;

Regla	Producción	Acción semántica
Exp_atom	INT |	RESULT = new Attrs("int");
	REAL |	RESULT = new Attrs("real")
	STRING |	RESULT = new Attrs("string") ;

	ID |	Attrs aux = obtenerTipo(id.getLexeme());
		if (aux!=null){
		if (!esTipoSimple(aux)){ //agrego a la lista de ids involucrados cualquier cosa que no tenga su tipo definido
		aux.agregarID(id.getLexeme());
		}
		RESULT = aux;
		}
		else
		throw new Exception("Undeclared identifier (" + id.getLexeme() +")");

	NIL |	RESULT = new Attrs("list",crearPolitipo("politipo")) ;
	PARABRE Exp_Rel PARCIERRA |	RESULT = e;
	Tuple |	RESULT = t;
	List	RESULT = l;

Tuple	PARABRE Exp_Rel:e COMA Tuple2:t2 PARCIERRA	t2.addTupleTypeBegin(e);
		corregirTupla(t2);
		RESULT = t2;

Tuple2	uple2:t COMA Exp_Rel:e	t.addTupleTypeFinal(e); RESULT = t;
	Exp_Rel:e	Attrs t = new Attrs("tuple", new ArrayList()); t.addTupleTypeFinal(e); RESULT = t;
List	CORABRE List2:l CORCIERRA	RESULT = l;

List2	List2:list COMA Exp_Rel: exp	Attrs t = tipoMasEspecifico(list.getListType(),exp);
		if (t!=null)
		RESULT = tipoMasEspecifico(list,new Attrs("list", t));
		else
		throw new Exception("The list has elements of different types.");

	Exp_Rel:exp	RESULT = new Attrs("list",exp);