Attribute Grammar

Attributes

Symbol	Attribute Name	Java Type	Inherited/Synthesized	Description
expression	tipo	Tipo	Synthetized	Tipo of the expression
expression	Lvalue	Boolean	Synthetized	True if the expression can appear to the left of an assigment
funcionSentencia	tipo	Tipo	Synthetized	Tipo of the funcionSentencia
declaracionfuncion	tipo	Tipo	Synthetized	Tipo of the declaracionfuncion
Sentencia	padre	declaracionfuncion	Inherited	Se usara tanto para los returns

Rules

Node	Predicates	Semantic Functions
program → declaraciones*		
declaracionstructs:declaraciones → nombre:string definicion*		
declaracionglobales:declaraciones → definicion	Declaracionglobales.tipo == definicion.tipo;	

declaracionfuncion:declaraciones → nombre:string argumento:definicion* tipo? variablesLocales:definicion* sentencia*	Definicion*.getTipo().tipoSimple())
	
	If(Tipo.exist())
	Tipo.tipoSimple()
		If(tipo.exist){
		Declaracionfuncion.tipo = tipo;
		}else{
		Declaracionfuncion.tipo = VoidTipo
		}

definicion → IDENT:string tipo		
printSentencia:sentencia → expression*	expression*.getTipo().tipoSimple() expression*.lvalue == true	
readSentencia:sentencia → expression*	expression*.getTipo().tipoSimple() expression*.lvalue == true	
printspSentencia:sentencia → expression*	expression*.getTipo() .tipoSimple() expression*.lvalue == true	

printlnSentencia:sentencia → expression*	expression*.getTipo() .tipoSimple() expression*.lvalue == true	

funcionSentencia:sentencia → nombre:string argumento:expression*	Expression.lenght == funcionSentencia.getDeclaracionfuncion().getTipo().get().params.length
	
	For(var a : expression){
	a.tipo() == funcionSentencia.getDeclaracionfuncion().getTipo().get().params[indice].Tipo();
	}
		FuncionSentencia.tipo = funcionSentencia.getDeclaracionfuncion().getTipo().get()

asignacionSentencia:sentencia → left:expression expression	sameTipo(left.tipo, expression.tipo) left.tipo.tipoSimple() left.lvalue == true	

returnSentencia:sentencia → expression?	If(returnSentecia.declaracionfucnion().getTipo().isEmpty()){
	expression.isEmpty()
	}
	
	If(returnSentencia.declaracionFuncion.getTipo().isPresent()){
	Expression.isPresent()
	}
	
	ReturnSentecia.getDeclaFuncion().getTipo == expression.get().getTipo();	

ifSentencia:sentencia → condicion:expression entonces:sentencia* otro:sentencia*	Condicion.getTipo() == int;	
whileSentencia:sentencia → condicion:expression entonces:sentencia*	Condicion.getTipo() == int;	
intExpresion:expression → intValue:int		IntExpresion.tipo = int; IntExpresion.lvalue = false;
realExpresion:expression → doubleValue:double		RealExpresion.tipo = float; RealExpresion.lvalue = false;
identificadorExpresion:expression → name:string		IdentificadorExpresion.tipo = identificadorExpresion.getDefinicion().getTipo() IdentificadorExpresion.lvalue = false
charExpresion:expression → charValue:char		charExpresion.tipo = char CharExpresion.lvalue = false

accessoArrayExpresion:expression → acceso:expression indice:expression	Acceso.getTipo() == Arraytipo Indice.getTipo() == IntTipo	ArrayTipo tipoArray = (ArrayTipo) indice.getTipoExpresion(); AccessoArrayExpresion.setTipoExpresion(tipoArray.getTipo()); AcessoArayExpresion.lvalue = True
parentesisExpresion: expression → expression		ParetesisExpresion.tipo = expression.tipo ParentisisExpresion.lvalue = false
castExpresion:expression → tipo expression	Tipo.tipoSimple() Expression.tipo != tipo	CastExpresion.tipo = tipo CastExpresion.lvalue = false
negacionExpresion:expression → expression	expression.tipo == int	NegacionExpresion.tipo = expression.tipo Arithmetic.lvalue = false
arithmeticExpresion: expression → left:expression operator:string right:expression	NullOVoid(Left.getTipoexpresion()) NullOVoid(Right.getTipoExpresion()) sameTipo(left.tipo, right.tipo) Left.getTipoExpresion() == IntTipo | FloatTipo Right.getTipoExpresion() == IntTipo|FloatTipo	arithmeticExpresion.tipo = left.tipo arithmetic.lvalue = false
logicExpression:expression → left:expression operator:string right:expression	SameTipo(left.tipo, right.tipo) Left.getTipoExpresion() == IntTipo | FloatTipo Right.getTipoExpresion() == IntTipo|FloatTipo	LogicExpression.tipo = IntTipo LogicExpression.lvalue = false
boolExpression:expression → left:expression operator:string right:expression	SameTipo(left.tipo, right.tipo) Left.getTipoExpresion() == IntTipo Right.getTipoExpresion() == IntTipo	boolExpression.tipo = IntTipo boolExpression.lvalue = false
acederCap:expression → left:expression right:string	Left.tipo == StringTipo ((StringTipo) Left.type).getdefinitions(). foreach(def -> def.getIDENT().equals(right)	AccederCap.tipo = def.tipo AccederCap.lvalue = true

funcionExpresion:expression → nombre:string argumentos:expression *	Expression.lenght == funcionSentencia.getDeclaracionfuncion().getTipo().get().params.length
	
	For(var a : expression){
	a.tipo() == funcionSentencia.getDeclaracionfuncion().getTipo().get().params[indice].Tipo();
	}
		FuncionExpresion.tipo = funcionExpresion.getDeclaracionfuncion.getTipo().get()
		
		FuncionExpresion.lvalue = false

intTipo:tipo → name:string		
floatTipo:tipo → name:string		

charTipo:tipo → name:string		
arrayTipo:tipo → tamArray:int tipo		
StringTipo:tipo → name:string		

Operators samples (cut & paste if needed):
⇒ ⇔ ≠ ∅ ∈ ∉ ∪ ⋂ ⊂ ⊄ ∑ ∃ ∀

Auxiliary Functions

Name	Description
TipoSimple()	Comprueba que el tipo sea simple
NullOVoid()	Comprueba que el tipo pueda ser null o V