



COMPLUTENSE DE MADRID

PRÁCTICA PROCESADORES DE LENGUAJES.

Cuarta Fase.

Kyle Tan & Lukas Häring
Grupo 9

May 8, 2019

Tabla de contenidos

1	Diagrama sintáctico	2
2	Sintaxis abstracta mediante diagrama de clases	3
3	Gramática de atributos	4

1 Diagrama sintáctico

La descripción de la función constructora está basada en el lenguaje matemático:

$$\langle \text{funcion} \rangle : \langle \text{arg}_1 \rangle \times \cdots \times \langle \text{arg}_n \rangle \rightarrow \langle \text{Valor a devolver} \rangle$$

Para la función constructora de inicialización:

$$iInit \rightarrow Type \text{ ID} ; iInit \mid Type \text{ ID}$$

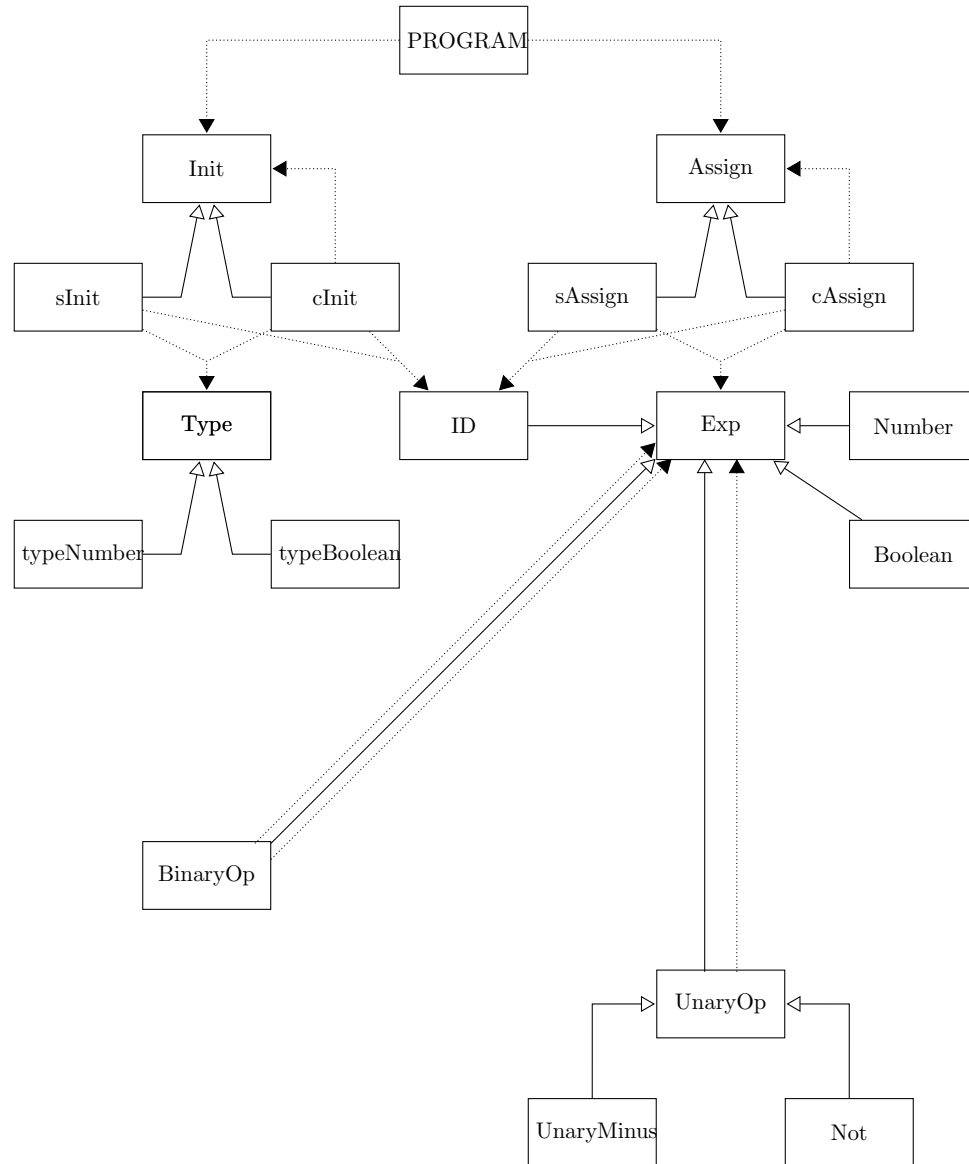
$$Type \rightarrow bool \mid num$$

Para la función constructora de asignación:

$$iAssign \rightarrow ID = exp ; iAssign \mid ID = exp$$

Expresión	Función Constructora Programa
$iInit \ \&\& \ iAssign \rightarrow Program$	PROGRAM: $iInit \times iAssign \rightarrow Program$
Expresión	Función Constructora Inicialización
$Type \text{ ID} ; iInit \rightarrow iInit$	$cInit: Type \times \mathbf{string} \times iInit \rightarrow iInit$
$Type \text{ ID} \rightarrow iInit$	$sInit: Type \times \mathbf{string} \rightarrow iInt$
Expresión	Función Constructora Asignación
$ID = Exp ; iAssign \rightarrow iAssign$	$cAssign: \mathbf{string} \times Exp \times iAssign \rightarrow iAssign$
$ID = Exp \rightarrow iAssign$	$sAssign: \mathbf{string} \times Exp \rightarrow iAssign$
Expresión	Función Constructora Operaciones
$Exp + Exp \rightarrow Exp$	Add: $Exp \times Exp \rightarrow Exp$
$Exp - Exp \rightarrow Exp$	Sub: $Exp \times Exp \rightarrow Exp$
$Exp * Exp \rightarrow Exp$	Mul: $Exp \times Exp \rightarrow Exp$
$Exp / Exp \rightarrow Exp$	Div: $Exp \times Exp \rightarrow Exp$
$Exp \text{ and } Exp \rightarrow Exp$	And: $Exp \times Exp \rightarrow Exp$
$Exp \text{ or } Exp \rightarrow Exp$	Or: $Exp \times Exp \rightarrow Exp$
Expresión	Función Constructora Comparaciones
$Exp == Exp \rightarrow Exp$	Eq: $Exp \times Exp \rightarrow Exp$
$Exp != Exp \rightarrow Exp$	nEq: $Exp \times Exp \rightarrow Exp$
$Exp < Exp \rightarrow Exp$	Less: $Exp \times Exp \rightarrow Exp$
$Exp <= Exp \rightarrow Exp$	LessThan: $Exp \times Exp \rightarrow Exp$
$Exp > Exp \rightarrow Exp$	Greater: $Exp \times Exp \rightarrow Exp$
$Exp >= Exp \rightarrow Exp$	GreaterThan: $Exp \times Exp \rightarrow Exp$
Expresión	Función Constructora Unarias
$\text{not } Exp \rightarrow Exp$	Not: $Exp \rightarrow Exp$
$-Exp \rightarrow Exp$	UnaryMinus: $Exp \rightarrow Exp$
Expresión	Función Constructora Tokens
$\mathbf{bool} \rightarrow Type$	typeBoolean: $\mathbf{num} \rightarrow Type$
$\mathbf{num} \rightarrow Type$	typeNumber: $\mathbf{bool} \rightarrow Type$
$ID \rightarrow Exp$	ID: $\mathbf{string} \rightarrow Exp$
$\mathbf{number} \rightarrow Exp$	Number: $\mathbf{string} \rightarrow Exp$
$\mathbf{boolean} \rightarrow Exp$	Boolean: $\mathbf{string} \rightarrow Exp$

2 Sintaxis abstracta mediante diagrama de clases



3 Gramática de atributos

Utilizando la implementación de la gramática incontextual de la segunda práctica.

Gramática	Gramática Programa
$\text{Program} \rightarrow \text{iInit}\&\&\text{iAssign}$	$\text{Program.a} = \text{PROGRAM}(\text{iInit.a}, \text{iAssign.a})$
Gramática	Gramática Inicialización
$\text{iInit} \rightarrow \text{Init}; \text{iInit}$	$\text{iInit.a} = \text{sInit}(\text{Init.type}, \text{Init.id})$
$\text{iInit} \rightarrow \text{Init}$	$\text{iInit.a} = \text{cInit}(\text{iInit.a}, \text{Init.type}, \text{Init.id})$
$\text{Init} \rightarrow \text{Type ID}$	$\text{Init.type} = \text{Type} \wedge \text{Init.id} = \text{ID.lex}$
Gramática	Gramática Asignación
$\text{iAssign} \rightarrow \text{Asgn}; \text{iAssign}$	$\text{iAssign.a} = \text{sAssign}(\text{Asgn.id}, \text{Asgn.exp})$
$\text{iAssign} \rightarrow \text{Asgn}$	$\text{iAssign.a} = \text{cAssign}(\text{iAssign.a}, \text{Asn.id}, \text{Asn.exp})$
$\text{Asn} \rightarrow \text{ID} = \text{Exp}$	$\text{Asn.id} = \text{ID.lex} \wedge \text{Asn.exp} = \text{Exp.a}$
Gramática	Gramática Expresiones
$\text{Exp} \rightarrow \text{Exp} + \text{E1}$	$\text{Exp.a} = \text{Add}(\text{Exp.a}, \text{ELv1.a})$
$\text{Exp} \rightarrow \text{Exp} - \text{E1}$	$\text{Exp.a} = \text{Sub}(\text{Exp.a}, \text{ELv1.a})$
$\text{Exp} \rightarrow \text{ELv1}$	$\text{Exp.a} = \text{ELv1.a}$
$\text{ELv1} \rightarrow \text{Exp} + \text{E1}$	$\text{ELv1.a} = \text{And}(\text{ELv2.a}, \text{ELv1.a})$
$\text{ELv1} \rightarrow \text{Exp} - \text{E1}$	$\text{ELv1.a} = \text{Or}(\text{ELv2.a}, \text{ELv2.a})$
$\text{ELv1} \rightarrow \text{ELv2}$	$\text{ELv1.a} = \text{ELv2.a}$
$\text{ELv2} \rightarrow \text{ELv3 Ob ELv3}$	$\text{ELv1.a} = \text{opRel}(\text{Ob.op}, \text{ELv3.a}, \text{ELv3.a})$
$\text{Ob} \rightarrow \text{op}$	$\text{Ob.op} = \text{op}$
$\forall \text{op} \in \{\text{or}, \text{not}, \text{==}, \text{!=}, <, <=, >, >=\}$	
$\text{ELv2} \rightarrow \text{ELv3}$	$\text{ELv2.a} = \text{ELv3}$
$\text{ELv3} \rightarrow \text{ELv3} * \text{ELv4}$	$\text{ELv3.a} = \text{Mul}(\text{ELv3.a}, \text{ELv4.a})$
$\text{ELv3} \rightarrow \text{ELv3} / \text{ELv4}$	$\text{ELv3.a} = \text{Div}(\text{ELv3.a}, \text{ELv4.a})$
$\text{ELv3} \rightarrow \text{ELv4}$	$\text{ELv3.a} = \text{ELv4.a}$
$\text{ELv4} \rightarrow -\text{ELv4}$	$\text{ELv4.a} = \text{UnaryMinus}(\text{ELv4.a})$
$\text{ELv4} \rightarrow \text{not ELv5}$	$\text{ELv4.a} = \text{Not}(\text{ELv5.a})$
$\text{ELv4} \rightarrow \text{ELv5}$	$\text{ELv4.a} = \text{ELv5.a}$
$\text{ELv5} \rightarrow (\text{Exp})$	$\text{ELv5.a} = \text{Exp.a}$
$\text{ELv5} \rightarrow \text{num}$	$\text{ELv5.a} = \text{Number}(\text{num.lex})$
$\text{ELv5} \rightarrow \text{bool}$	$\text{ELv5.a} = \text{Boolean}(\text{bool.lex})$
$\text{ELv5} \rightarrow \text{ID}$	$\text{ELv5.a} = \text{ID}(\text{ID.lex})$

Nota: *Asn* es equivalente a *Assign*.

$$\text{opRel}(\text{op}, \text{left}, \text{right}) = \begin{cases} \text{op} = "=" & \Rightarrow \text{Eq}(\text{left}, \text{right}) \\ \text{op} = "!=" & \Rightarrow \text{nEq}(\text{left}, \text{right}) \\ \text{op} = "<" & \Rightarrow \text{Less}(\text{left}, \text{right}) \\ \text{op} = "<=" & \Rightarrow \text{LessThan}(\text{left}, \text{right}) \\ \text{op} = ">" & \Rightarrow \text{Greater}(\text{left}, \text{right}) \\ \text{op} = ">=" & \Rightarrow \text{GreaterThan}(\text{left}, \text{right}) \end{cases}$$