Luis Alejandro Urbina
Ma. Isabel Ortiz Naranjo

# Análisis Semántico

## Diseño del compilador

El compilador funciona utilizando ANTLR4 y Python3. Se parte de la gramática de YAPL definida en el archivo YAPL.g4, a partir de ella se construyen las clases de YAPLLexer.py, YAPLParser.py y YAPLVisitor.py. A partir de esta última, se construyen dos visitors que recorren el árbol de análisis sintáctico generado por YAPLParser añadiendo cada símbolo a la tabla de símbolos y a su vez verificando las reglas semánticas (definidas por el propio lenguaje e implementadas en el compilador) y de tipos (definidas más adelante en este documento). El output de esta fase es una impresión de la tabla de símbolos así como la lista de errores semánticos encontrados en un programa de entrada con extensión .cl provisto al correr el programa main.py.

## Diseño de la tabla de símbolos

La tabla de símbolos presentada guarda 3 tipos de símbolos: **Class, Attribute, y Function.** Estos están represntados por clases (con el mismo nombre) que se instancian y se agregan a la tabla de símbolos a medida que se van encontrando en la recorrida del árbol de análisis sintáctico. A continuación se presenta una descripción de las clases y sus atributos.

Clase: **Class**

Atributos:

**id (int):** identificador único de la clase
**name (str):** nombre de la clase
**type (str):** tipo de la clase
**scope (int):** alcance o ámbito de la clase
**isFrom (str):** superclase a la que pertenece

Clase: **Function**

Atributos:

**id (int):** identificador único de la función
**name (str):** nombre de la función
**type (str):** tipo de la función
**scope (int):** alcance o ámbito de la función
**isFrom (str):** clase a la que pertenece la función

Clase: **Attribute**

Atributos:

**name (str):** nombre del atributo
**type (str):** tipo del atributo
**scope (int):** alcance o ámbito del atributo
**insideClass (str):** clase a la que pertenece el atributo
**insideMethod (int):** id del método/función al que pertenece el atributo
**isParameterOfFunction (boolean):** indica si el atributo es un parámetro de una función

# Reglas de tipos

YAPL cuenta con tres tipos básicos, los cuales son **Int, String, y Bool.** A continuación se definen las reglas de tipos para las operaciones implementadas en el presente proyecto.

**Add**

Γ ⊢ a: Int
Γ ⊢ b: Int
---
a + b: Int

Γ ⊢ a: String
Γ ⊢ b: String
---
a + b: String

Cualquier otro tipo de resta devuelve un error.

**Substract**

Γ ⊢ a: Int
Γ ⊢ b: Int
---
a - b: Int

Cualquier otro tipo de resta devuelve un error.

**Divide**

Γ ⊢ a: Int
Γ ⊢ b: Int
---
a / b: Int

Cualquier otro tipo de divisón devuelve un error.

**Multiply**

Γ ⊢ a: Int
Γ ⊢ b: Int
---
a * b: Int

Cualquier otro tipo de multiplicación devuelve un error.

**NOT**

Γ ⊢ a: Int
---
not a: Int

Γ ⊢ a: Bool
---
not a: Bool

Cualquier otro tipo de negación devuelve un error.

**LESS THAN**

Γ ⊢ a: Int
Γ ⊢ b: Int
---
a < b: Bool

Cualquier otro tipo de negación devuelve un error.

**LESS EQUAL**

Γ ⊢ a: Int
Γ ⊢ b: Int
---
a <= b: Bool

Cualquier otro tipo de negación devuelve un error.

**EQUAL**

Γ ⊢ a: Int
Γ ⊢ b: Int
---
a = b: Bool

Γ ⊢ a: String
Γ ⊢ b: String
---
a = b: Bool

Γ ⊢ a: Bool
Γ ⊢ b: Bool
---
a = b: Bool

Γ ⊢ a: Int
Γ ⊢ b: String
---
a = b: Bool

Γ ⊢ a: Bool
Γ ⊢ b: String
---
a = b: Bool

Γ ⊢ a: Int
Γ ⊢ b: Bool
---
a = b: Bool

Cualquier otro tipo de negación devuelve un error.