Compilers Final Project

Simple Static Type Checker

Jordan Lam 100751857

CSCI 4020U Winter 2024

April 14, 2024

PL.g4:

- This file contains the grammar definition of the programming language
- It defines the syntax rules for constructing programs of the language, including statements, expressions, literals, identifiers, and whitespace
- It's used by ANTLR to generate a parser that can parse source code written in the language

data.kt:

- This file defines data classes representing different types of data of the language
- It contains definitions for Data, None, Integer, StringData, and BooleanData classes
- Each class represents a specific type of data (e.g., integers, strings, booleans) and provides methods for working with that data

expr.kt:

- This file defines classes representing expressions in the language
- It contains definitions for classes like Expr, NoneExpr, IntegerExpr, StringExpr, BooleanExpr, VariableExpr, AssignmentExpr, PrintExpr, ConcatExpr, MultiplyExpr, IfElseExpr, FunctionDefExpr, and FunctionCallExpr
- Each class represents a specific type of expression (e.g., variable access, assignment, print statement) and provides methods for evaluating or processing that expression

runtime.kt:

- This file defines the Runtime class, which represents the runtime environment for executing programs of the language
- It contains methods for managing the symbol table (mapping variable names to their values) and creating subscopes
- The Runtime class is used during the evaluation of expressions to access variables and perform other runtime operations.

TypeChecker.kt:

- This file contains the TypeChecker class, which is responsible for performing static type checking on expressions in the language.
- The TypeChecker class has a single public method, check, which takes an Expr as input and initiates the type checking process.
- The core of the type checking logic is implemented in the private checkExpr method, which recursively traverses the AST and checks the type of each expression.
- Within the checkExpr method, each type of expression is handled differently:
    - For literals (IntegerExpr, StringExpr, BooleanExpr), default values of their corresponding data types are returned
    - For variables (VariableExpr), no type checking is performed, and None is returned
    - For assignment (AssignmentExpr), the type of the expression being assigned is checked recursively, but no specific type checking is done on the assignment itself.
    - For print statements (PrintExpr), the type of the expression being printed is checked recursively, but no specific type checking is done on the print statement itself.
    - For binary operations (ConcatExpr, MultiplyExpr), the types of the operands are checked recursively, and appropriate type errors are thrown if the operands are of incompatible types.
    - For conditional expressions (IfElseExpr), the condition is checked to be boolean, and the types of the if and else branches are recursively checked to ensure they have compatible types.
    - Function definitions (FunctionDefExpr) are ignored, as they typically don't require type checking in this context.
    - For function calls (FunctionCallExpr), the types of the arguments are recursively checked, and appropriate type errors are thrown if the number of arguments doesn't match or if the argument types don't match the expected types defined by the function signature.