CSE 595X: Independent Study Project Proposal

Implementation of a Type Inference System for a Lua-like Language

Student: Dominic Bertolo
Student ID: 007958672
Date: November 19, 2025

1 Motivation

The Lua programming language is a widely used scripting language, known for its simplicity and efficiency. Due to it being lightweight, Lua is often embedded in applications to provide scripting capabilities. Despite its lightweight, it offers powerful features such as first-class functions, coroutines, and a flexible metaprogramming system. However, these powerful features are dynamic in nature, causing difficulty for the analysis of the reliability and security of Lua code. A robust type system could help mitigate these issues by providing static guarantees about the behavior of Lua programs, thus enhancing their reliability and security.

There exist several attempts to implement a type system for Lua, such as Typed Lua, Teal, and Luau. However, these type systems often are implemented either as a separate language that compiles to Lua, a modified version of Lua that is not fully compatible with standard Lua, or a type checker that does not completely enforce type safety. This project aims to explore the design and implementation of a pure type system based upon an algorithmic type system, such as Hindley-Milner. This would allow for pure type inference and checking, without having to modify the Lua language itself or introduce new syntax.

Due to the dynamic nature of Lua, static analysis cannot effectively capture all possible runtime behaviors. In order to provide an effective static type inference and checking system, certain features of Lua may need to be restricted. This project will explore a subset of Lua that balances core features of the language with the ability to perform effective static type analysis. By designing the type system for a subset of Lua, it will be possible to incrementally expand the type system to cover more features of Lua in the future. Additionally, Lua is a multi-paradigm language, supporting both imperative and functional programming styles. This project will focus on the functional programming aspects of Lua, as they align more closely with the principles of type systems like Hindley-Milner.

2 Primary Objectives

The primary goal of the project is to design and implement a static type inference and checking system for a subset of the Lua programming language. The specific objectives include:

1. **Formalize the Subset of Lua Grammar:** Define a formal grammar for a subset of Lua that adheres to the core features of the language and functional programming paradigms.
2. **Develop the Language Frontend:** Implement a parser and abstract syntax tree (AST) representation for the defined subset of Lua.
3. **Implement Constraint-Based Type Inference:**
    - Develop a **Constraint Generator** that traverses the AST and emits type constraints for expressions, functions, and control flow.
    - Develop a **Unification Engine** (based on Algorithm W) to solve the generated type constraints detect type inconsistencies.
4. **Validation and Reporting:** Create a collection of test cases to validate the type inference system and have the system return descriptive error messages for type violations.

3 Timeline

The project is expected to be completed over a 12-week period, with the following milestones:

1. **Weeks 1-2:** Scaffolding and Frontend Design - Define the subset of Lua grammar, implement the lexer, test tokenization, and establish project structure.
2. **Weeks 3-4:** Parsing Logic - Implement the Recursive Descent Parser, handle precedence for expressions, build AST nodes for assignments, branches, loops, and functions. **Milestone:** Complete AST visualization for sample programs.
3. **Weeks 5-7:** Data Structures - Implement parsing for table literals and index expressions, implement symbol table and type environment structures, define scoping rules for variables and functions
4. **Weeks 8-9:** Static Analysis Core - Implement logic to generate constraints for arithmetic, comparisons, function calls, and returns, and map AST nodes to type variables.
5. **Weeks 10-11:** The Type System - Implement the Hindley-Milner unification algorithm, and implement error reporting for type mismatches. **Milestone:** Functional Type Inference System.
6. **Week 12+:** Testing and Documentation - Integration testing with a suite of programs covering various language features within the subset of Lua, evaluate performance and accuracy of type inference, and preparation for final reports and presentations.

4 Project Deliverables

At the conclusion of the Independent Study, the following deliverables will be available:

- **Source Code:** A fully functional repository containing the implementation of the language lexer, parser, and type inference system.
- **Documentation:** Comprehensive documentation detailing the design decisions, architecture, and usage instructions for the type inference system.
- **Test Suite:** A collection of test cases demonstrating both successful type inference and the correct flagging of type errors.
- **Final Report:** A detailed report summarizing the project objectives, methodology, results, and potential future work.
- **Presentation:** A presentation summarizing the project for academic review, highlighting key findings and contributions.