CSCI 5535 Fundamentals of Programming Languages

Potential Course Projects

CU Programming Languages
& Verification

Project 1: Rust Core Calculus

1. Formalize the core features of Rust language, including its type and borrow checker.
2. Prove Type and Borrow Safety theorem.
3. Extend it to concurrency and prove the absence of data races.
4. Reference: Pearce, A Lightweight Formalism for Reference Lifetimes and Borrowing in Rust, TOPLAS'21.

Project 2: Liquid Types

1. Embed “Liquid Types” (Rondon et al, PLDI’08) in Coq.
2. Demonstrate deep and shallow embeddings.
3. Write LTac-based proof automation scripts that automate (most of) liquid type checking.
4. Bonus: extend the type system to handle “structural relations” (Kaki et al, ICFP’14).

Project 3: Distributed Protocol Verification

1. Formalize models of well-known distributed protocols in Coq:
    1. One-shot leader election,
    2. Leader election in a ring, and
    3. Paxos.
2. In each case, verify that the safety property:
    1. Leader, if any, is unique.
    2. No two nodes make different decisions.
3. Write LTac-based proof automation scripts that partially automate verification.

Project 4: C-lite to JVM Byte Code

1. Formalize lightweight C language (with variables, pointers, assignments, While loop, and function calls) in Coq. Define its operational semantics.
2. Formalize an equivalent subset of JVM instructions and define its operational semantics.
3. Write a “compiler” from C-lite to JVM. Prove its correctness by showing the observational equivalence (bi-simulation) of source (C-lite) and target (JVM) programs.

More Projects

1. Formalize PageRank algorithm in Coq and prove its convergence.
2. Implement Software Foundations Vol1 in Lean.