Architectural optimizations for scalable model-checking

Supervisor: Michalis Kokologiannakis

1 Overview

Stateless model checking (SMC) verifies a concurrent program by enumerating all of its interleavings, up to equivalence. SMC does this dynamically: while executing the program, it records alternative exploration options, and then repeatedly re-executes it until all such options have been explored.

Crucially, SMC's performance relies on an execution engine (e.g., an interpreter). The goal of this project is to exploit architectural optimizations in this execution engine in order to increase the scalability of SMC.

Our tool of choice is GenMC, a state-of-the-art stateless model checker for programs written in C, C++ and Rust. GenMC uses an interpreter over LLVM IR to execute programs written in any of the above languages.

In this project, we will explore two (orthogonal) directions. First, we will optimize the performance of GenMC's interpreter by leveraging techniques from (advanced) compiler construction (e.g., threaded code), and then benchmark against the existing implementation. Then, we will investigate different ways to restore the program state to the initial one (this is necessary to re-execute the program) by modeling the state using persistent data structures (e.g., RRB vectors).

2 Goals

- Get familiar with the current architecture of GenMC.
- Rewrite the LLVM interpreter in a threaded fashion.
- Replace existing data structures representing the memory state with persistent ones.
- Benchmark the resulting implementation against the previous one.

3 Extension goals

- Implement custom persistent data structures to further improve state-restoring performance.

4 Required skills

This project combines concepts from various parts of programming languages. A good grade in Parallel Programming, Formal Methods and Functional Programming and Compiler Design is highly encouraged.