Dynamic analysis of concurrency - algorithms vs AI

Short Description

LLMs can offer assistance in different areas of life. In engineering, LLMs are still observed with skepticism considering their issues with the reproducibility of the results and general imprecision. However, in software engineering, certain problems are undecidable. Static analysis in certain programming languages cannot statically evaluate program state and is hence forced to make over proximations. Dynamic analysis is more precise, but suffers from coverage. To do any analysis, dynamic approach considers execution of code, gathering of the execution trace, and analysis of the execution trace.

In concurrent software systems with multiple threads, there is a great need to reconstruct shared states and synchronization mechanisms (especially in languages such as C and C++). However, it is extremely hard to generate adequate tests that would produce appropriate traces. Furthermore, gathering huge execution traces is impractical for analysis. This project aims to employ several LLM tools and, through a case study, conclude what is the potential of these tools to complement dynamic analysis (analyze concurrent software, perform source code instrumentation, generate test cases, execute test cases to gather the trace, analyze the trace using standard algorithms). As part of the project, the candidate will also develop algorithms for analysis of execution traces (e.g., Eraser Lockset [1]) and use existing tools to compare their results with the LLM tools.

[1] Eraser: A dynamic data race detector for multi-threaded programs – Savage et al. 1997