Centre for Digital Trust and Security

Seedcorn Final Report 23/24

Project Title:
Formal Software Verification with Large Language Models in the Loop

Project Investigators:
Mustafa Mustafa (PI), Meropi Tzanetakis (Co-I), Lucas Cordeiro (Co-I), Youcheng Sun (Co-I), Laura McCulloch (RA), Yiannis Charalambous (Collaborator), Edoardo Manino (Collaborator)

Project overview:
This project addresses the widespread deployment of software, which often comes with security vulnerabilities. The aim is to integrate Large Language Models (LLMs) with formal verification methodologies to improve the generation and verification of secure code. Formal verification will be employed to identify security vulnerabilities and produce counterexamples when the code does not work correctly or violates security standards. These counterexamples, along with the problematic code, will then be fed into the LLM using a specialised prompt language designed for debugging and code generation. This approach will help diagnose and rectify security issues. The corrected code generated by the LLM will undergo formal verification to ensure its security. A prototype connecting a verifier with ChatGPT will be used, and publicly available code will be utilised to evaluate the method. Additionally, social science methods will be deployed to assess software developers' perceptions of LLM-enhanced formal verification tools.

Key findings:
Technical perspective

We have successfully integrated Large Language Models (LLMs) into one of our formal verification tools - ESBMC-AI. Our results show that LLMs can successfully be used to debug code when fed with counterexamples produced by a verification tool. However, more research is needed to fine-tune LLMs into improving their success rate.

More specifically, we used GPT-3.5-Turbo and various prompt engineering techniques to explore how well an LLM could repair a mutated code. In the process, we discovered that a long persona prompt with the role Automated Code Repair Tool is the most optimised at repairing AI Code. We then proposed methods for extracting the faulty source code from the large volume of AI code to circumvent the issues that arise due to LLMs' relatively small context window. Lastly, we used ESBMC-AI to test how the iterative Automated Programme Repair (APR) process improves repair performance. Some take-away messages are given below:

- Formal verification with LLM in the loop increases the automated program repair performance when LLM is given multiple attempts. The best number of attempts is 3 for prompts 9 and 11, as a successful repair becomes unlikely after the 2nd retry.
- Showing the history of patches to the LLM improve the performance of iterative APR. When comparing the number of successful repairs between the Latest State Only (LSO) experiments and the Forward History experiments, the latter has a much higher number of successful repairs.
- The best format to show the history is to display the oldest messages first and the last message being the latest. This has been observed when comparing Forward History and Reverse History.
- The optimal temperature to achieve the highest number of repaired samples is 0.0 (i.e., deterministic output) for prompts 9 and 11 for Forward History. A higher temperature is necessary for less conventional prompts to allow the LLM to parse it correctly.

In summary, we have showed that the iterative repair process provides a substantial increase in repair performance (from 18% APR success w/o iteration to 23% success with iteration). We also found that after attempt 3 the chances of successfully repairing a sample decrease significantly.

In the future, we plan to conduct more experiments using a diverse set of LLMs to discover whether our findings generalize beyond GPT-3.5-Turbo. In this respect, open-source LLMs would benefit our research, as they can be fine-tuned for program repair.

Social perspective
With the successful integration of LLMs into our formal verification tools, the open question that needs to be addressed is how the improved verification tools are perceived by practitioners working in the industry. This requires a qualitative research approach covering complex social phenomena.

We have successfully conducted 26 semi-structured interviews with software developers and software engineers. These qualitative in-depth interviews aim to explore how practitioners perceive and use formal verification tools when identifying bugs in large software systems.

- Software developers and engineers generally find formal verification tools accurate, with few false positives. However, tolerance for false positives varies depending on company objectives. While some developers rely solely on these tools, others use them alongside testing to ensure bug-free software. Verification tools are employed at different workflow stages, complementing other techniques rather than replacing them.
- Interview results indicate that scalability influences tool usage, particularly under tight deadlines. Developers prioritise fixes based on risk, especially in complex systems, balancing time and labour costs. A disconnect exists between developers and management, with managers sometimes viewing these tools as an extra cost, driven by cost-benefit analyses rather than technical necessity.
- Software developers and engineers might avoid these tools due to unfamiliarity with the programming languages used and the steep learning curve involved. High labour costs and lack of time during work hours further discourage usage, often requiring personal time investment. Additionally, the instructions on how to use these tools is perceived as overly complex and not user-friendly, often written by specialists for specialists, making it difficult for those without specific training to understand.

Future research focussing on the social aspects should focus on the perceptions and practices of developers and engineers across various industry contexts who use verification tools to address security vulnerabilities. This approach can help improve the application of LLM-enhanced formal verification tools.

Outputs to date:

- A prototype implementation of a formal verification tool with an LLM integrated for generating possible fixes of the identified bugs.
- We have successfully conducted 26 semi-structured interviews with software developers and software engineers, yielding valuable insights into our study's focal points. Transcripts from these interviews have been thoroughly compiled, serving as foundation for our ongoing data analysis.
- Dissemination through conference paper presentation: 36th Annual Meeting of the Society for the Advancement of Socio-Economics (SASE), 27-29 June 2024, University of Limerick
- Technical report “Automated Repair of AI Code with Large Language Models and Formal Verification“ has been written: https://arxiv.org/pdf/2405.08848.
- A technical report “Tasks People Prompt: A Taxonomy of LLM Downstream Tasks in Software Verification and Falsification Approaches” has been written: https://arxiv.org/pdf/2404.09384.

Were all planned outcomes achieved?
If not, how did you mitigate non-achievement?

All key outcomes have either been achieved or are on track to be delivered as follow-up activities.

Planned activities post-project:

- Knowledge exchange activities in the form of a findings briefing event at the University of Manchester with our research participants on 2nd October 2024, which is supported by Business Engagement Manager Rachel Kenyon
- Collaborative paper (with a tentative title *"It's made from researchers to researchers"- The Challenges of using Automated Software Verification Tools*) to be submitted to a top tier venue (IEEE S&P) in November 2024.
- Both technical reports, "Automated Repair of AI Code with Large Language Models and Formal Verification" and "Tasks People Prompt: A Taxonomy of LLM Downstream Tasks in Software Verification and Falsification Approaches", to be submitted to top venues.
- A large EPSRC grant application to be submitted in AY 2024/25.
- Dagstuhl Seminar application to be submitted in AY 2024/25.
- A workshop on the topic to be organised jointly by The University of Manchester and University of Liverpool in summer 2025.