Prasita Mukherjee Cover Letter

My research focuses on broadening the scope of intractable problems in the domain of formal methods and programming languages by providing learning-based solutions. A second major focus of my research is to develop automated solutions for problems that can be mapped to first/higher order logic- with little to no user help.

Research Background In my academic career, I have worked on several research projects in the domains of formal methods, programming languages and AI. In this section, I describe the two key research areas that I have worked on and that interest me the most.

Learning based solutions in the domain of computationally intractable problems like, LTL model checking and program synthesis, primarily seek to provide low-cost solutions with generalization capabilities.

Traditional model checkers prove costly as they suffer from the state space explosion problem that prevents model checking from being applied to a wider domain. To bridge this gap and make model checking more accessible, we designed a Graph Representation Learning framework **OCTAL** [1] that takes advantage of the natural graph structure of the automaton and expression tree structure of LTL to build a unified graph structure, and learn the structure and semantics of the system and specification through message passing. We performed rigorous experiments on OCTAL, and obtained up to $\sim$ 95% accuracy and $\sim 2\times$ speedups over traditional model checkers, hence broadened the scope of model checking by providing a low cost learning based solution, compromising some accuracy.

The second body of work I am investigating in this domain called **SynVer** [2], seeks to advance synthesis and verification efforts for mainstream languages like C, by deploying a Large Language Model (LLM) as a black-box synthesizer, with full functional specifications. We identify syntactic and semantic biases in the search space and specifications, that are amenable to automated verification, and design prompts accordingly. We also propose a specification-verification tool on top of Verified Software Toolchain (VST) [3] to automate the verification process. Our results demonstrate that the biases reflected in the prompt design and specifications, synthesizes and verifies most of the programs in the benchmarks, at the first try.

Full automated reasoning seeks to provide correct solutions to questions framed in logic with no user assistance. This body of research is very challenging and appealing as questions in first, or higher order logic are undecidable. Hence, providing provably correct solutions and identifying decidable fragments of the problem at hand, would make it easier, especially for end users. My research in this domain has been on three projects: First, analyzing the equivalence of snippets of a programming language. Second, analyzing the feasibility of shared memory data structures in a distributed system setting [4], and third, relational verification of programs using E-Graphs [5]. The first and second projects rely on SOTA SMT solvers like Z3 [6] and CVC5 [7] to analyze and provide solution models for counter example cases. My research in these two projects was to encode the execution semantics of the programming language and data structures into an SMTLIB [8] format, query an SMT solver for analysis, identify the decidable fragment of the language and perform experimental analysis to draw conclusions for the respective problems. The third project, aims to verify if two programs are observationally equivalent, by constructing a product program alignment using e-graphs. We demonstrate that the product programs are relatively easier to verify, requiring simpler invariants. My contribution was to come up with manual loop invariants where automated inference failed, and prove them in VST, along with the demonstration that the invariants

although complex enough to require interactive theorem proving, were still simpler as compared to other alignments.

References

[1] Prasita Mukherjee, Haoteng Yin, Susheel Suresh, and Tiark Rompf. "OCTAL: Graph Representation Learning for LTL Model Checking". *CoRR* (2022). DOI: 10.48550/arXiv.2207.11649. arXiv: 2207.11649. URL: https://doi.org/10.48550/arXiv.2207.11649.

[2] Prasita Mukherjee and Benjamin Delaware. *Towards Automated Verification of LLM-Synthesized C Programs*. 2024. arXiv: 2410.14835 [cs.PL]. URL: https://arxiv.org/abs/2410.14835.

[3] Andrew W Appel. "Verified Software Toolchain: (Invited Talk)". *European Symposium on Programming*. 2011.

[4] Kartik Nagar, Prasita Mukherjee, and Suresh Jagannathan. "Semantics, Specification, and Bounded Verification of Concurrent Libraries in Replicated Systems". *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part I*. Ed. by Shuvendu K. Lahiri and Chao Wang. Lecture Notes in Computer Science. Springer, 2020. DOI: 10.1007/978-3-030-53288-8_13. URL: https://doi.org/10.1007/978-3-030-53288-8_13.

[5] Robert Dickerson, Prasita Mukherjee, and Benjamin Delaware. "KestRel: Relational Verification Using E-Graphs for Program Alignment". *CoRR* (2024). DOI: 10.48550/ARXIV.2404.08106. arXiv: 2404.08106. URL: https://doi.org/10.48550/arXiv.2404.08106.

[6] Leonardo De Moura and Nikolaj Bjørner. "Z3: An efficient SMT solver". *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. 2008.

[7] Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, et al. "cvc5: A versatile and industrial-strength SMT solver". *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. 2022.

[8] Clark Barrett, Aaron Stump, Cesare Tinelli, et al. "The smt-lib standard: Version 2.0". *Proceedings of the 8th international workshop on satisfiability modulo theories (Edinburgh, UK)*. 2010.