UGP-I CS395A

INTEGRATING SADHAK WITH UCLID5

Rishabh Arijeet – BT/CSE/210842

Supervisor: Dr. Subhajit Roy

Project Mentor: Sujit Muduli

April 25 ,2023

Acknowledgments

I am deeply indebted to Professor Subhajit Roy for allowing me to work with him.I am thankful to him for having regular meetings despite his busy schedule and helping me acquire the necessary perseverance for solving a research problem. I would also like to thank Sujit Muduli, who helped me a great deal in this project and for giving me his time.

Abstract

UCLID5 is an integrated tool for modeling, verification, and synthesis that relies on Delphi, a solver that can handle SMT theories modulo closed-box functions. However, Sadhak is another solver that also handles closed-box constraints, but with more speed and benchmark coverage than Delphi. The central idea behind Sadhak is to use a fuzzer to reason about closed-box functions and an SMT engine to solve constraints related to SMT theories. By using a synergistic combination of these two components, Sadhak can handle complex problems involving closed-box functions.

In this work, the aim is to replace Delphi with Sadhak in UCLID5 to improve the tool's results. This replacement will enable UCLID5 to solve more benchmarks and provide better performance overall. With the incorporation of Sadhak, UCLID5 will be able to offer a more comprehensive and efficient approach to modeling, verification, and synthesis of computational systems.

Introduction

Formal verification, synthesis, and modeling have become increasingly important in the design and development of complex computational systems. Formal methods offer rigorous techniques for ensuring the correctness and reliability of such systems, which are crucial in safety-critical and mission-critical applications.

In this report, we will focus on two innovative tools that contribute significantly to the field of formal methods: UCLID5 and Sadhak. UCLID5 is an integrated tool for modeling, verification, and synthesis, while Sadhak is a solver designed to handle SMT theories modulo closed-box functions. We will provide an overview of both tools, their key features, and their advantages over existing methods. We will also be comparing Sadhak and Delphi, the default solver used in UCLID5. Finally, we will explore the potential of integrating Sadhak into UCLID5 to improve its performance and broaden its capabilities.

UCLID5

UCLID5 is a tool for the multi-modal formal modeling, verification, and synthesis of systems. UCLID5 supports both synthesis and verification. The Synth-Lib interface constructs either a verification or a synthesis problem from the assertions generated by the symbolic simulator. The verification problems are passed to the SMT-LIB interface, which converts each assertion in UCLID5’s intermediate representation to an assertion in SMT-LIB. Similarly, the synthesis problems are passed to the SyGuS-IF interface, which converts each assertion to an assertion in SyGuS-IF. The verification and synthesis problems are then passed to the appropriate provided

external solver and the result is reported back to the user. UCLID5 supports oracle function symbols in verification by interfacing with a solver that supports Satisfiability Modulo Theories and Oracles (SMTO). Oracle function symbols are declared like functions, with the keyword oracle, and an annotation pointing to the binary implementation. For instance oracle function [isprime] Prime (x: integer) : boolean would indicate to the solver that the binary isprime takes an integer as input and returns a boolean. By default, UCLID5 uses Delphi , another SMT solver, which uses closed box functions(oracles).

SADHAK

Sadhak's Conflict-Driven Fuzz Loop (CDFL) is a unique approach that combines an SMT solver with a fuzzer in a synergistic combination to solve closed-box functions. The SMT solver loads the constraints for the respective SMT theories, while the fuzzer focuses solely on the closed-box functions. The fuzzer starts by creating a partial model, and if this model conflicts with the constraints within the SMT solver, the fuzzer analyzes the conflict and borrows only the relevant constraints that led to the conflict. It then proposes a new model to the SMT solver, and this process continues until the fuzzer constructs a model that is consistent with the constraints loaded in the SMT solver.

The CDFL strategy is particularly effective because it allows the fuzzer to iteratively propose new models guided by conflicts from the SMT solver. This iterative approach reduces the search space and allows for faster and more accurate solutions to problems involving closed-box functions. The CDFL strategy is one of the key features that sets Sadhak apart from other solvers, making it a valuable tool for researchers and practitioners in the field of formal methods.

Comparison of Sadhak and Delphi shows that Sadhak is a highly effective tool for solving problems involving closed-box functions. Sadhak is able to solve 36.45% more benchmarks than the best-performing mode of Delphi and has a 5.72× higher PAR-2 score. When the two tools solved the same benchmarks, Sadhak was 14.62× faster on average than the bit-blast mode of Delphi.

INTEGRATION

For our implementation, we wrote a shell script to provide UCLID5 with the external solver SADHAK. We also included binary files for the oracle functions, which are functions without an implementation but associated with a user-provided external binary that can be queried by the solver.

To ensure compatibility between UCLID5 and Sadhak, we made some modifications to the SMTLIB-Interface so that it could dump a correct SMT file for Sadhak to run. This allowed us to use Sadhak as the default solver for UCLID5, instead of Delphi.

With the inclusion of the oracle functions and the use of Sadhak as the default solver, UCLID5 can now handle closed-box constraints more efficiently and accurately.

Bibliography

1. SADHAK:https://dl.acm.org/doi/pdf/10.1145/3563332
2. UCLID5: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8556946&tag=1
3. UCLID5 github repo: https://github.com/uclid-org/uclid
4. Delphi Tool: https://github.com/polgreen/delphi
5. Oracle repository: https://github.com/polgreen/oracles
6. Satisfiability Modulo Oracles: https://people.eecs.berkeley.edu/~sseshia/pubdir/symo-vmcai22.pdf