Research Statement

Pritom Rajkhowa

October 8, 2019

My research has been in the field of artificial Intelligence, and in particular, the principles of knowledge representation, reasoning, and learning, their applications in formal verification.

I have completed my Ph.D. in the Department of Computer Science and Engineering in The Hong Kong University of Science and Technology under the supervision of Professor Fangzhen Lin[1] on 27 Aug 2019. The title of my thesis is "VIAP - An Automated System for Verifying Integer Assignment Programs with Loops".

Current Research and Research Work During PhD

The automatic program verification has been a major research area since its beginning. Despite significant progress in automatic program verification, proving the correctness of programs with loops and arrays is still considered as a complex problem. The main difficulty in proving the correctness of programs with loops and array in traditional approaches is that it is not always possible to find the number iterations statically. In such cases, they tried to derive a logical property called the loop invariant which describes the properties of loops in each iteration. It provides a solution to this problem so that reasoning about loop becomes independent of the number of iterations. Although the automatic generation of loop invariants itself is considered as a long-standing complex problem. In the case of programs that use non-linear arithmetic, unbounded data structures like array, complex control flow (such as nested loops), it is difficult to automatically find suitable candidate invariants. That's why the automatic generation of suitable loop invariants is one of the major practical obstacles to full automation of program verification.

My research aims to design and develop a fully automated program verification tool to automatically verify non concurrent program written in C. The tool is based on the new approach where given a program and a requirement to verify, it first translates the given program into a set of first-order axioms independent of the requirement. The requirement is then verified as a query of the translated axioms. We call our system Verifier for Integer Assignment Programs (VIAP). What distinguishes VIAP is its handling of loops. The iterations are systematically encoded as inductive definitions, and the termination is axiomatized by constraints on specially introduced constants. VIAP can prove the correctness of programs without generating loop invariants. The efficiency of VIAP comes from many simplifying techniques, including a dedicated recurrence solver to compute the closed-form solutions of inductive definitions whenever possible. RS uses SymPy's rsolve() function as a base solver, and extends it to handle conditional recurrences. It also includes a solution for a simple case of mutual recurrences.

As a result, VIAP is able to automatically prove non-trivial properties of many benchmark programs that previously required either manually encoded loop invariants or powerful loop invariant generators. We also validated the capability of VIAP by successfully proving non-linear benchmark programs such as those for computing integer square roots, integer product, integer power, factorial and the sums of series of integers. The system is fully automatic and points to a new way of proving properties of programs. VIAP won first rank in the ReachSafety-Arrays and ReachSafety-Recursive sub-categories of the ReachSafety category in the SV-COMP 2019 competition.

Program equivalence checker : Program equivalence checking has significantly practical applications such as compiler optimization validation, information flow analysis, and regression verification. Program equivalence checking also presents the same challenges as program verification when dealing with loops, the inductive invariant

[1] http://www.cs.ust.hk/faculty/flin/

inference is still very challenging, especially in the presence of multi-path, nested loops and polynomial assignments. We observe that state-of-the-art automatic verifiers are often unable or inefficient to derive suitable invariants for the product program to complete equivalence proofs. We have extended our approach to design fully automated system called Eq-VIAP primarily for checking the equivalence of programs.

Verification of Probabilistic Program : Automatically proving some property φ associated with a probabilistic program is also considered as infamously more difficult compared to their deterministic counterparts. In our latest work, we present a new, fully automated system for reasoning about probabilistic programs. It is based on translation which translates the probabilistic program to a set of first-order logic axioms quantification over the domain of natural numbers. We tried to address issue associated with probabilistic program arises from introducing randomness into it that variables cannot be viewed as single values; they must be viewed as distributions. Whereas existing approaches such as [1, 2, 3] usually take into consideration only upper and lower bounds over program variables or expected values. Such information may be insufficient to characterize the distributions of variables as highlighted in [4]. Similarly, the (co-)variances and other higher-order moments of variables are also needed. Recently published work [5], tried to address these issues. However, they do not address loop conditions, nested loops, and multipath loops, while our work in this paper addresses these as well. We successfully evaluated our work on benchmark programs for the three analyses to demonstrate the practicality of the presented approach.

Verification of Smart Contract : Ethereum is a blockchain platform which is based on distributed ledgers is rising rapidly. The main reason for their fast growth is their ability to maintain a distributed public ledger, provide reliability, integrity, and auditability without trusted entities. Ethereum serves as an ecosystem for self-enforcing smart contracts. Smart contracts are responsible to manage assets of considerable valuation in the form of crypto-currency. Hence, it is crucial to ensure the correctness of the developed smart contracts. Because correct smart contracts are paramount to ensuring trust in blockchain-based systems. We present a framework for analyzing and verifying the safety, by ensuring the functional correctness and security of Ethereum smart contracts by translation to a set of first-order logic axioms. Then translated a set of axioms is used to reason about smart contract properties.

Research Work During MS

I have completed my MS in Software Systems 2012 from BITS, Pilani, India. During the course of my Master's degree, I have carried out a project on An application of Defeasible Logic Programming for Firewall Verification and Reconfiguration under the supervision of Professor Shyamanta M. Hazarika[2]. It involves the Defeasible Logic Programming (DeLP) in the network security field .

My Master's thesis work involved development of an automated model; translating a high level security policy to configuration rule set or filter of firewall, followed by a model for verification and reconfiguration of rule set without human interventions. It finds its application in the area of multi-agent systems, network security and artificial intelligence by following the works as listed below:

- The proposed policy translator model translates a high level security policy to low level firewall policy addressing the problem of inconsistency between security policy and firewall policy. Unlike all existing models, we use Natural Language Processing to translate security policy, which provides user flexibility without any constraints for defining security policy. The model has the capability to translate security policy to either stateless or state-full firewall policy.

- We developed the framework for intra firewall anomaly detection and removal using defeasible argumentation. In anomaly removal mechanism, intelligent agent uses results of anomaly detection and firewall policy for removing anomaly as well as reconfiguration of the firewall rule set.

In addition to the work done in the dissertation, I have developed a model to verify and reconfigure rule sets for distributed firewalls using multi agent system. In the model, each firewall is associated with an intelligent agent, which will be responsible for verifying and reconfiguring intra-firewall anomalies and also communicate with each other to verify and reconfigure inter-firewall anomaly.

[2]http://www.iitg.ac.in/s.m.hazarika/index.html

References

[1] J.-P. Katoen, A. K. McIver, L. A. Meinicke, and C. C. Morgan, "Linear-invariant generation for probabilistic programs:," in *Static Analysis*, R. Cousot and M. Martel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 390–406.

[2] F. Gretz, J.-P. Katoen, and A. McIver, "Prinsys—on a quest for probabilistic loop invariants," in *Quantitative Evaluation of Systems*, K. Joshi, M. Siegle, M. Stoelinga, and P. R. D'Argenio, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 193–208.

[3] A. Chakarov and S. Sankaranarayanan, "Expectation invariants for probabilistic program loops as fixed points," in *Static Analysis*, M. Müller-Olm and H. Seidl, Eds. Cham: Springer International Publishing, 2014, pp. 85–100.

[4] P. L. Novi Inverardi and A. Tagliani, "Discrete distributions from moment generating function," *Appl. Math. Comput.*, vol. 182, no. 1, pp. 200–209, Nov. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.amc.2006.03.048

[5] E. Bartocci, L. Kovács, and M. Stankovic, "Automatic generation of moment-based invariants for prob-solvable loops," *CoRR*, vol. abs/1905.02835, 2019. [Online]. Available: http://arxiv.org/abs/1905.02835