Future Reseach Plan

Pritom Rajkhowa

October 8, 2019

The results of my research provide enough evidence of the power of our quantified translation of programs to first-order logic. The application of this approach to address many open questions is one of the motivations for me to continue working in this field. In my future research agenda, I want to investigate how the integration of the verification theories and data-driven methods help in designing formal verifiers.

Data Structure Verification : To reason about data-structure is a challenging problem in formal verification. In future work, I aim to improve the ability and scalability of my automatic verification tool VIAP to solve important verification tasks of sophisticated data structures. I want to extend my translation algorithm such that it can translate programs with loops and sophisticated data structures to first-order theories with quantifiers over first-order logic sort and natural numbers, then tries to reason about specifications of the programs using inductive proof over quantified variables on SMT solver as the base theorem prover. Our approach solves a particularly challenging exercise of discovering sound and complex invariants of sophisticated data structure for analyses, i.e., reachability. I have already demonstrated the effectiveness of this approach for the program with array data structure [1]. The central challenge to effectively proving the specifications of such programs with complex control-flow behavior using induction proof is automatically discovering suitable lemma in some cases. I am planning to design the learning algorithm in the selective sampling framework to address that issue. The integration of our translation and data-driven approach will increase the capability of the designed verification approach.

Quantitative Properties of Programs : Static analysis of programs derives the information about their quantitative behavior statically which plays an important role in code optimization and verification. This information can be used to automatically prove certain program properties such as their time complexity. In future work, I aim to design a tool with the ability to derive quantitative properties of the loop. I want to highlight the major weakness of existing approaches with the help of the following examples of C code snippet.

```
P1  :    int x , C;  while(x < C) { x = x + 1; }

P2  :    int x ,y,C;  while(x + y < C) { x = x + 1;  y = y + 1; }
```

The potential-based techniques and amortized analysis base tool C^4B [2][1] can easily find the bound for the program P_1 where C^4B return the bound $1.00|[0, C]|$ for the program P_1. But all the tools including C^4B failed to find the bound for the program P_2. Whereas my approach can find the bound of both programs $|[0, C - x]|$ and $|[0, (C - x - y)/2]|$ for P_1 and P_2 respectively. Compared to more classical approaches based on ranking functions or amortized reasoning, my approach inherits the benefits of not explicitly generating any invariant. However, state-of-the-art classical approaches are often unable to or inefficient in derive suitable non-linear invariants to find suitable resource bounds. For example, inferring non-linear invariants is a long-standing challenge for conventional safety property verifiers as well [3, 4]. The crux of my technique is to derive non-linear resource bounds by constructing a proof.

Timed-automata :The extension of finite-state automata with real-valued clock variables to capture the time information, timed-automata, is extensively used to model embedded software, timed protocols, cyber-physical

[1]http://www.cs.yale.edu/homes/qcar/aaa/

systems, etc. Existing approaches suffer from scalability issues. I am planning to address such issues using my approach. The integration of my translation and classification-based learning can enhance the capabilities of my approach.

References

[1] P. Rajkhowa and F. Lin, "Extending viap to handle array programs," in *10th Working Conference on Verified Software: Theories, Tools, and Experiments, VSTTE 2018, Oxford, UK, July 18-19*, 2018. [Online]. Available: https://github.com/VerifierIntegerAssignment/sv-comp/blob/master/extending-viap-array.pdf

[2] Q. Carbonneaux, J. Hoffmann, and Z. Shao, "Compositional certified resource bounds," in *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '15. New York, NY, USA: ACM, 2015, pp. 467–478. [Online]. Available: http://doi.acm.org/10.1145/2737924.2737955

[3] T. Ball, R. Majumdar, T. Millstein, and S. K. Rajamani, "Automatic predicate abstraction of c programs," in *ACM SIGPLAN Notices*, vol. 36, no. 5. ACM, 2001, pp. 203–213.

[4] K. L. McMillan, "Lazy abstraction with interpolants," in *International Conference on Computer Aided Verification.* Springer, 2006, pp. 123–136.