Inductive Validity Cores for Formal Verification

Elaheh Ghassabani

Abstract

Increasing usage of computer systems in safety-critical applications demands the utmost care in their specification, design, and implementation. Formal verification is a useful method for mathematical/systematic examination of the requirements a system must meet. However, due to the complexity of modern systems, safety analysis is challenging. My research focuses on developing efficient techniques/ tools that facilitate the verification task. Contribution to this field may have significant impacts on any areas where computer systems play a critical role. To this end, I have been working on a novel idea, Inductive Validity Cores, that effectively helps with a lot of safety-analysis tasks.

1. Introduction

Computer systems have become an integral part of our daily life being used in various environments (such as homes, hospitals, factories) and application areas (such as medical devices, aircraft flight control, weapons, and nuclear systems). The increasing reliance of critical applications on information processing makes it vital to verify the soundness and safety of the computer systems because failure in these systems could be disastrous leading to financial loss, loss of life, or damage to the environment. To address this issue, *formal verification* comes into play. Formal verification is the process of mathematically proving or disproving the correctness of a system with respect to certain requirements or properties.

One of the most successful and powerful methods for formal verification is called *model checking*. Model checking tools are attractive both because they are automated, requiring little or no interaction with the user and, if the answer to a correctness query is negative, they provide an evidence that helps the system designer/analyst detect bugs. However, when it comes to the valid properties, oftentimes, tools do not provide any useful information as to why the system satisfies a given property. It is well known that issues such as vacuity and inconsistencies can cause verification to succeed despite errors in a property specification or in the system design. In such cases, a system or subsystem component will not exhibit the expected behavior in the operating environment although the analyst gains over-confidence or false assumption on the correctness of the system, which may bring about serious damage. Therefore, the level of feedback provided by the tool to the user matters.

On the other hand, with growing complexity of computer systems, any kind of formal verification task, including model checking, is becoming more difficult and challenging. Hence, there is an increased demand for devising/improving model checking tools and algorithms. In addition, every technique that makes the verification and system analysis tasks easier and more efficient will make a huge difference in industry and the quality of the products.

The goal of my research is to address the above challenges. My research contributions started with the novel idea of *Inductive Validity Core (IVC)*, which takes advantage of the state of the art of model checking techniques [1]. The IVC idea makes a lot of useful system analyses/engineering tasks possible and more efficient. Specifically, it is useful when the validity of a safety requirement has been stablished by the model checker. In this case, IVCs provide usable information both formal and human-understandable that explains why the requirement is satisfied. Such information is valuable in analyzing safety-critical systems.

2. Current Research & Contributions

We have recently published the IVC idea in FSE'16 [1] while introducing a set of useful verification analyses for it. We have integrated/implemented this idea into an industrial model checker, called `JKind` [2]. The efficiency and efficacy of our technique has been evaluated over a large corpus of benchmarks containing industrial cases from medical devices and microwave control software [3]. Recently we have been exploring the applications of this idea to different problem domains [4, 5]. Over the course of one year, this technique has been employed in several realistic systems (listed in Appendix I), which shows the impact of the IVCs on the formal methods community in terms of research and practice.

Generally speaking, an IVC is a set of design artifacts that are necessary for a particular property to be valid. The algorithm proposed in [1] aims to compute one single IVC for a given property; however, computing all IVCs of a given property is an interesting problem that provides several useful analyses. For this purpose, we have proposed a new algorithm proving its correctness and completeness [6]. The new algorithm has been also implemented in an experimental branch of `JKind` [7] and benchmarked [8].

2.1. State of the Art

Our work builds on top of a substantial foundation provided by special tools known as *constraint solvers*. Constraint solving is a powerful mathematical method that allows the computer to solve a problem formulated by the user. Many verification problems can be reduced to constraint satisfaction problems and solved with tools known as *SMT[1] solvers*. A lot of useful formal methods are built on top of SMT solvers, such as model checking algorithms, abstraction techniques, and proof-certificate generation. There is significant amount of valuable research on these topics in the literature. Although such reasoning techniques are helpful, they are not expressive enough to provide good insights into the quality of a system or specification. With the IVC idea, we are able to bridge the gap between verification techniques and the user insight into the results provided by the tools. The goal behind this idea is different from existing applications of constraint solving. The IVC idea shares many similarities with existing approaches for computing proof certificates, and in fact the IVC algorithm performs this computation as well. However, there is a substantive difference; to find a guaranteed minimal set of certificates, it is usually necessary to find new proofs involving new invariants not used in the original proof, which existing techniques do not deal with.

2.2. Significance

The IVC idea adds to the power of SMT-based model checkers by making useful engineering tasks efficiently possible or automatic. Computing a single IVC [1] provides applications for automatically performing proof-based traceability between requirements and system constraints, assessing model coverage, and explaining test obligations failure when using model checkers for test-case generation. Without IVCs, such tasks have been accomplished manually and based on human guesses without any formal proof. However, certification standards for safety-critical systems usually require some quality assurance arguments obtaining from such analyses. Offering automatic and proof-based correctness evidences, via IVCs, extremely decreases analysis effort/cost, while increases accuracy and soundness.

[1] Satisfiability Modulo Theory

Other important and additional analyses that can efficiently be performed by our new algorithm on finding *all* IVCs [6] are as follows:

- **Impact analysis**: Given all IVCs, it is possible to determine which requirements may be falsified by changes to the system. This analysis allows for selective regression verification of tests and proofs: if there are alternate proof paths that do not require the modified portions of the system, then the requirement does not need to be re-verified.
- **Robustness analysis**: As we proposed in [4], it is possible partition the system elements into MUST and MAY categories based on whether they are in every IVC or only some IVCs, respectively. This may allow insight into the relative importance of different model elements for property. For example, if the MUST set is empty, then the requirement has been implemented in multiple ways, such as would be expected in a fault-tolerant system.
- **Coverage Analysis [5]:** IVCs can be used to define coverage metrics for properties by examining the percentage of model elements required for a proof. However, since IVCs are not unique, there are multiple, equally legitimate coverage scores possible. Having all IVCs allows one to define *additional* coverage metrics.

Moreover, the Requirements Engineering community is keenly interested in approaches to manage requirements traceability. In most cases, it is assumed that there is a single "golden" set of trace links that describes how requirements are implemented in software. However, if there are multiple IVCs, then it is possible that there are several equally valid sets of trace links. Examining the diversity of all IVCs could lead to changes in how traceability is performed and managed in critical.

3. Future Research

We are planning to improve our implementation by devising/employing more efficient model checking algorithms. Another interesting direction is to parallelize our algorithms so to increase their scalability.

We also plan to investigate additional applications of the idea for software engineering activities. When performing compositional verification of a software architecture, IVC techniques may be able to determine minimal component sets within an architecture that can satisfy a given set of requirements, which may be helpful for design space exploration.

Finally, we are interested in adapting the notion of (all) validity cores for bounded model checking. In bounded analysis, the verification task terminates when the tool hits a certain time/depth boundary. Extending the notion of IVCs to bounded model checking can facilitate more low-level analyses, such as testing of complex programs.

3.1. Career Goals

Primarily, I want my work to have a positive impact on people's lives. I enjoy research and development with the purpose of solving large-scale realistic problems that software/hardware industry faces. I'm particularly interested in compositional verification, model-based development, and large-scale verification techniques for software/hardware systems. It is my career goal to be able to continue my research on these areas. I hope to attain employment within a national lab or corporate research facility, where I can pursue these passions and benefit the scientific community with my work.

References

[1] Ghassabani, E., Gacek, A. and Whalen, M. W., 2016, November. Efficient generation of inductive validity cores for safety properties. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (pp. 314-325). ACM.

[2] JKind Model Checker *[online]*. http://loonwerks.com/tools/jkind.html

[3] IVC, Experimental Evaluation *[online]*. https://github.com/elaghs/Working/tree/master/support/experiments

[4] Murugesan, A., Whalen, M.W., Ghassabani, E. and Heimdahl, M.P., 2016, September. Complete traceability for requirements in satisfaction arguments. In *Requirements Engineering Conference (RE), 2016 IEEE 24th International* (pp. 359-364). IEEE.

[5] Ghassabani, E., Gacek, A. and Whalen, M. W., and M. Heimdahl, "Proof-based coverage metrics for formal verification," in International Symposium on Software Testing and Analysis (ISSTA), 10 pages, 2017 [under submission].

[6] Ghassabani, E., Whalen, M. W. and Gacek, A. "Efficient generation of all IVCs," in Conference on Computer Aided Verification (CAV), 20 pages, 2017 [under submission].

[7] JKind Model Checker, forked branch *[online]*. https://github.com/elaghs/jkind/tree/IVCs

[8] Efficient Generation of All IVCs, Experimental Evaluation *[online]*. https://github.com/elaghs/Working/tree/master/all_ivcs/experiments

Appendix I

A list of industrial projects that have used the IVC technique so far:

- NASA CVFCS on the Quad-redundant flight controller
- Rockwell Collins, part of the AFRL SpEAR project
- Helicopter architecture proofs in the DARPA HACMS project
- NSF Medical device project on the GPCA model
- AGREE Symbolic Simulator, Part of DARPA SOSITE project
- AGREE/JKind Test-Case Generator