AI based Formal Specification for Scientific Security

Noah Evans
Sandia National Laboratories

Sam Pollard
Sandia National Laboratories

Robert Armstrong
Sandia National Laboratories

Jacob Hobbs
Sandia National Laboratories

October 14, 2021

1 Introduction

The correctness and security of science codes and the national cyber infrastructure is increasingly in doubt. The considerable investment already made in verification and validation (V&V) of scientific simulations can be extended in new directions to help address these needs. Building on traditional V&V to formally "correct by construction" [1] techniques in code development and coupling new AI techniques to discover security and correctness issues in existing codes is an opportunity to address these problems.

To make scientific computing secure we must first be able to understand its behavior. **We envision a future where – given a problem, a set of constraints, and a target – we leverage Machine Learning (e.g., adversarial ML, model building, reinforcement learning, genetic programming) and formal methods to infer the models, heuristics, tests, and/or policies that govern and optimize the computational science workflows needed for DOE and the NNSA's mission needs. This understanding will provide the foundation for the rigorous cyber-security guarantees on mission codes.** This need for automated workflows is increasingly urgent as V&V becomes the largest expenditure in computational science –Software must be secure as well as correct. To make rigorous levels of trust possible and V&V expenditures more manageable, our proposed workflow would analyze the behavior of existing scientific applications and use learned models to derive formal specifications of their behavior. At the compiler level, for example, the workflow could be used to learn heuristics for better code portability, while at the system level, the workflow could suggest directed cyber-security tests or more resilient resource management policies. Formal methods-based derivations based on learned system behavior models would enable a full-stack "Model Based System Engineering" (MBSE) workflow that provides greater assurance of scientific correctness and cyber-security in current and future systems and scientific applications.

2 Necessary Research

This combination of Machine Learning and Formal Methods techniques make it possible to understand and verify cyber-security guarantees on scientific applications.

2.1 New types of AI research for scientific cybersecurity

Effective application of "black box" Model Learning, used when analysis of source code is impractical, will require tens of thousands of input/output traces of the application under analysis. In cases where production source code is available, "white box" Model Learning may be used and the source code itself (and analyses derived thereof) comprises another data set. These artifacts will provide the foundation for the Model Learning data set. This data set will form the "System Under Learning" upon which we will apply active automata learning techniques such as Angluin's Algorithm to derive formal specifications for existing artifacts.

This research would lead to AI architecture for Model Learning of scientific applications and hardware behaviors from application source code and hardware artifacts.

2.2 New types of formal methods

Using these generated models it would be possible to manage the complexity of providing cybersecurity and privacy guarantees for scientific software using advances in automated theorem proving and proof assistants.

HPC computing applications have grown in complexity to both utilize the exponential increases in computing power as well as achieve the most efficient use of available hardware. Both of these lead to code which is nearly impossible for one person to understand its full behavior, which in turn makes verification and validation out of reach for most scientific computing teams.

Proof assistants such as Coq and automated theorem proving tools such as Satisfiability Modulo Theories (SMT) potentially make this complexity tractable by providing powerful automation and proof engineering tools to verify software security properties and to create programs which are provably correct to their specification.

However, the greatest strength and most difficult challenge of advanced theorem proving is that it makes no assumptions about a given system, scientific or otherwise, and so all behavior must be completely specified before a theorem prover will guarantee a proof is correct.

The work to provide the underlying foundation to prove scientific computing systems secure and correct is ongoing. Two relevant examples are those for real analysis [2] and floating-point arithmetic [3], though there are many such libraries in areas such as cryptography or other high-level mathematics.

References

[1] Andrew W Appel, Lennart Beringer, Adam Chlipala, Benjamin C Pierce, Zhong Shao, Stephanie Weirich, and Steve Zdancewic. Position paper: the science of deep specification. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 375(2104):20160331, 2017.

[2] Sylvie Boldo, Catherine Lelay, and Guillaume Melquiond. Coquelicot: A User-Friendly Library of Real Analysis for Coq, March 2015.

[3] Sylvie Boldo and Guillaume Melquiond. Flocq: A unified library for proving floating-point algorithms in coq. In Elisardo Antelo, David Hough, and Paolo Ienne, editors, *Proceedings of the 20th IEEE Symposium on Computer Arithmetic*, ARITH '11, pages 243–252, Tübingen, Germany, July 2011. IEEE Computer Society.