Formal Methods at Scale
2019 Workshops Report

Computing-Enabled Networked Physical Systems (CNPS)
Interagency Working Group (IWG)

Patrick Lincoln and William Scherlis, Co-Chairs

Report Co-Authors
Patrick Lincoln, William Martin, and William Scherlis

Workshops Organizers
William Martin, Sponsor
Katie Dey, Coordinator

May 2022

Contents

List of Tables

Executive Summary

Formal methods for systems assurance have a rich history spanning half a century. Even in the early days of computing, there were efforts directed at mathematical specifications and proof of properties of programs. Motivated by emerging uses of computing software and hardware in critical systems, several U.S. agencies invested in research in formal methods. For decades, however, formal methods tools and ecosystems could operate only on problems and systems of modest scale. Computer science students often had only limited exposure to formal methods techniques and tools, partly on the basis that the techniques were long considered to be a theoretical possibility but not a practical reality that could affordably provide real benefits to larger system and software engineering projects. Recently there have been revolutionary advances in tools, practices, training, and ecosystems that have facilitated the application of formal methods at larger scales, in a manner that is affordable and usable by professional software and hardware engineers.

Recognizing the opportunity afforded by these advances, two workshops were convened in fall 2019 on the topic of formal methods at scale. This report provides a summary of those workshops, including their principal conclusions and relevant reports on experience in practice. The workshops included participants from the U.S. Government, industry, and academia, gathering to discuss recent advances that address the challenges of both scalability and adoptability into practice, including evidence from early adopters, with a focus on understanding prospects for the future and how they might be better enabled. Following the workshop, select formal methods practitioners from the community offered abstracts communicating use cases and related discussions of formal methods at scale. This report concludes with a discussion of key findings and emerging capabilities that can speed the adoption of formal methods.

1.0 Introduction

Two workshops were convened in 2019 on the topic of formal methods at scale. The workshops took place on September 25, 2019 (East), and October 9, 2019 (West).[1] Participants from U.S. Government, industry, and academia gathered to discuss recent advances in the application of formal methods at scale and prospects for the future. Specific topics included improvements in tools, practices, and training and characteristics of existing and emerging applications with a focus on advances in formal methods technology, the scale of existing applications, and the potential for a new and broader scope for formal methods applications.

2.0 Formal Methods: History, Challenges, and Progress

Formal methods[2] for systems assurance have a rich history spanning half a century. Even in the early days of computing, there were efforts directed at mathematical specifications and proof of properties of programs. Motivated by emerging uses of computing software and hardware in critical systems (e.g., space or aircraft flight control, communication security, or medical devices), several U.S. agencies invested in research in formal methods. For decades, however, formal methods tools and ecosystems could operate only on problems and systems of modest scale. Computer science students often had only limited exposure to formal methods techniques and tools, partly on the basis that the techniques were long considered to be a theoretical possibility but not a practical reality that could affordably provide real benefits to larger system and software engineering projects. Recently there have been revolutionary advances in tools, practices, training, and ecosystems that have facilitated the application of formal methods at larger scales, in a manner that is affordable and usable by professional software and hardware engineers. The following subsections describe challenges that have been at least partially addressed.

2.1 Models and Composition

Early formal tools required a high degree of mathematical sophistication on the part of users, developers, and evaluators. In the modern context, an explicit focus has emerged on "invisible" formal methods techniques, with enough success to demonstrate that deep mathematical sophistication is not an intrinsic requirement for users of the formal methods approach. Programmers with no formal training already receive significant value from formal analysis even when they are unaware of the analyses employed behind the scenes. Principal examples include modern type systems and safe abstractions, which are now widely evident in programming languages and tooling, including safe variants for traditionally unsafe languages such as C/C++ and JavaScript. Modern tools manifest scalability through a combination of components with a focus on a diversity of models and scaling through composition. The growing diversity of models and methods allows an incremental adoption approach in which particularly salient system attributes can be addressed rigorously while other, less critical attributes may be addressed using more conventional methods including testing and inspection. This is a realization of the concept (from the late 1980s) of "small theorems about large programs."

[1] https://cps-vo.org/group/FMatScale

[2] See https://ep.jhu.edu/courses/605729-formal-methods/ for more information about formal methods.

2.2 Specifications

Early formal specification languages were often difficult to learn and use, creating challenges for both verification (consistency of system and specification) and validation (consistency of specification with actual requirements). Building specifications and sustaining their consistency with the evolution of systems and their operating context require the ability to develop specifications incrementally and rapidly iterate. This reduced boundaries in the engineering process among specification writing, model building, analysis and reasoning, and evolution of the system itself. This is analogous to established engineering practices for cloud-based software as a service (SaaS) such as development operations (DevOps) and development, security, and operations (DevSecOps). Modern specification languages have been developed with familiar syntax and semantics, and, importantly, domain-specific ties to existing languages and tools. Tools associated with these languages support not only explicit authoring by humans but also inference and learning of intent to be expressed, based on both forward-looking requirements analysis and backward-looking legacy system realities.

2.3 Tools

Early formal toolchains were often quirky and incompatible with other development environments. Indeed, many toolchains required the use of programming languages and tools that were unfamiliar to engineers and that had uncertain domain compatibility. The effect was an unacceptably high degree of risk because of the commitment required prior to any realization of benefit. Modern formal tools can be incrementally integrated with standard development tools and operate within virtualized environments. Formal methods tools are already integrated into many development environments to provide support for modeling and analysis. Indeed, there are several examples of hybrid specification and execution languages that incorporate a combination of code development, execution, modeling, and proving.

2.4 Evidence and Proofs

Early proofs and formal models of systems were mostly disconnected from the actual running code. Indeed, much of the formal reasoning identified in widely adopted security standards relates only to specifications and designs, not to the code that will actually be executed. Emerging formal toolchains support continuous integration and direct connections between models and executable artifacts. They not only support keeping proofs in sync with evolving systems, but also can integrate other kinds of engineering evidence including tests and regression suites. Integrating mechanisms such as argumentation structures and truth maintenance can assist in sustaining consistency within the body of evidence and system artifacts.

2.5 Performance

High-performance systems sometimes used optimizations that, with early *a posteriori* methods, were not readily formally modeled or proven, increasing the gap between the formally analyzed model and the actual running system; e.g., with regard to numerical stability as systems are optimized and tuned to particular high performance architectures. Modern formal tools can support refinement and other transformations for performance, thus maintaining proofs and high assurance as systems are tuned for performance.

In early systems, assumptions underlying key formal reasoning steps were sometimes implicit, lost during system evolution, or ignored. Modern methods are increasing integration with rigorous approaches to assumption tracking, safety/security cases, argumentation, and discharge of assumptions.

Early formal tools tended to assume an unrealistic top-down or waterfall model of system development, and often required development of full system-wide specifications before analysis could be undertaken. Modern formal methods tools can support incremental and iterative processes, along with analyses targeted to more specific critical attributes. These are a much better match for engineering practices including opportunistic design and implementation, continuous integration, continuous development, and continuous verification.

3.0 Summary of Results of the Workshops

At the workshops, progress in the beneficial use of formal methods at scale was highlighted. Discussion acknowledged that there are multiple dimensions of scale to be considered:

- Complexity and size of systems and their supply chains, including issues related to composability and the influence of structural and semantic architectural decisions.
- Range of specific system properties and qualities, both functional and quality focused, such as aspects of security, safety, performance, fault tolerance, hybrid systems, and real-time, that are modeled and reasoned.
- Effectiveness and efficiency of formal methods-related modeling and tooling to include integration into mainstream integration and practice. This includes a range of issues related to usability, training, and integration with legacy tooling and practices.
- Ability to rapidly co-evolve systems and associated evidence within continuous integration and continuous development frameworks. Evidence may include a mix of formal and informal elements such as partial proofs, strong type checking, and principled test plans and results. These elements may be included as a consequence of language design (e.g., type systems and memory safety) as well as explicit modeling and analysis (e.g., various safety and liveness properties). This can include programming language designs that are verification aware; for example, including annotation capabilities for assertion tagging as well as prover capabilities to support developers as they author code and assertions.
- Ease of use (particularly for non-expert developers and evaluators). One goal would be to make obvious to a casual technical observer any critical features or flaws in a formal argument. Another goal would be to support incremental assimilation of formal methods techniques into practices and toolchains. This area of focus includes issues related to validation, assisting engineers in developing specifications and models that are correctly aligned with requirements and system operating context.

Following the workshop, select formal methods practitioners from the community offered abstracts communicating use cases and related discussions of formal methods at scale.[3] Table 3-1 offers a presentation of these use cases and a mapping of associated dimensions of scale.

Table 3-1. Participant Use Cases

Point(s) of Contact	Use Cases	Dimensions of Scale
• Clark Barrett	Solvers for Boolean satisfiability and Satisfiability Modulo Theories (SMT) as industrial workhorses	• Complexity and the size of systems • Range of properties and qualities • Effectiveness and efficiency of formal methods-related modeling and tooling • Ease of use

[3] See https://cps-vo.org/group/FMatScale/report. Note: This site requires a user name and password

Point(s) of Contact	Use Cases	Dimensions of Scale
• Darren Cofer • Matt Wilding	Large aerospace and defense systems	• Effectiveness and efficiency of formal methods-related modeling and tooling
• Michael Collins • Kristin Giammarco	Modeling behavior of complex systems concepts supporting reasoning methods critical to national cyber and cryptologic missions	• Range of properties and qualities • Ease of use
• Takeaways from presentation by Byron Cook	Cloud Services foundational assurance (e.g. cryptography, virtualization, storage)	• Complexity and size of systems • Range of properties and qualities • Ability to rapidly co-evolve systems and associated evidence within continuous integration/deployment (CI/CD) • Ease of use
• Mike Dodds • John Launchbury • Stephen Magill	Cryptography & Formal Methods as a Service	• Range of properties and qualities • Effectiveness and efficiency of formal methods-related modeling and tooling • Ability to rapidly co-evolve systems and associated evidence within CI/CD • Ease of use
• Kathleen Fisher	Cyber-retrofit of DoD platforms	• Range of properties and qualities • Ability to co-evolve systems
• Warren A. Hunt, Jr. • J. Strother Moore	Creation, analysis, and maintenance of models of computational systems, assist with creating, analyzing, and maintaining models of computational systems developed by in use by companies including AMD, ARM, Centaur Technology, IBM, and Intel	• Effectiveness and efficiency of formal methods-related modeling and tooling • Range of properties and qualities • Complexity and the size of systems • Ability to rapidly co-evolve systems and associated evidence within CI/CD • Ease of use
• Peter O'Hearn	Scaling the impact of analysis of apps for Android and iOS, Facebook Messenger, Instagram, and other apps	• Complexity and size of systems • Range of properties and qualities • Effectiveness and efficiency of formal methods-related modeling and tooling • Ability to rapidly co-evolve systems and associated evidence within CI/CD • Ease of use
• Ray Richards	DoD military systems	• Complexity and size of systems • Effectiveness and efficiency of formal methods-related modeling and tooling • Ability to rapidly co-evolve systems and associated evidence within CI/CD • Ease of use

Point(s) of Contact	Use Cases	Dimensions of Scale
• Natarajan Shankar	Designing, analyzing, and creating computer systems	• Complexity and size of systems • Range of properties and qualities • Effectiveness and efficiency of formal methods-related modeling and tooling • Ability to rapidly co-evolve systems and associated evidence within CI/CD • Ease of use
• Nikhil Swamy	Components in HTTPS ecosystem, including transport layer security, the main protocol at the heart of HTTPS, as well as the main underlying cryptographic algorithms, such as AES, SHA2 or X25519	• Complexity and size of systems • Range of properties and qualities • Effectiveness and efficiency of formal methods-related modeling and tooling • Ability to rapidly co-evolve systems and associated evidence • Ease of use

Participants offered presentations regarding significant applications of formal methods, and on addressing various dimensions of scale, such as to major systems for Government and industrial use and to the proof of deep theorems in mathematics and theoretical computer science (including some new results). Participants also offered presentations regarding the status of the various formal methods ecosystems, the ensembles of proof formalisms, proof techniques, tools, libraries, training, and expertise.

There were significant advancements reported in areas of ecosystems and integration into application communities. Reports regarding several of the principal formal methods ecosystems highlighted progress in the development of formal toolchains, associated libraries of theories, and large sets of worked examples. Several of the most prominent ecosystems, which are very often open source, have been sustained and advanced over multiple decades by communities of researchers. Members of these communities are often funded to work on particular technical challenges or applications, and they sustain and enhance those ecosystems as part of that work.

Integration of formal approaches into broader applications communities was also discussed, including intensive use to increase the assurance of availability, security, privacy, and integrity of cloud services; embedded systems from the commercial domain; and military systems.

This intensifying use of formal methods is evidenced by the increasing demand for formal methods experts and practitioners. A key signal regarding potential demand for formal methods capabilities is the extent to which the presented application cases came about through "pull" from aspirational users, based on a business rationale, rather than "push" by researchers to adopt and advance a favored technology.

3.1 Payoff of Successful Application of Formal Methods at Scale

- Successful application of formal methods at scale can provide many benefits in both the short and the long term:
    - Early discovery of bugs. Formal approaches, uniquely, are able to verify an absence of bugs. Many of the popular heuristic tools identify large numbers of issues; however, because of often abundant false negatives, they give no basis for full confidence.

    - Reduction of the number of design iterations needed to achieve system quality goals, and thereby reduce time-to-market and reduce total cost of development. Formal methods can be applied to design models and architectural choices (structural and semantic) as well as code, enabling reasoning about design choices earlier in the process. This concept of "moving to the left" is an important contribution of emerging modern approaches to modeling and specification. The goal, in other words, is to understand the consequences of particular design and engineering commitments as soon as possible, ideally before they are made, and to minimize the extent and severity of uncertainties that linger beyond those commitments.
- Principled and efficient means to generate test plans. Testing strategies driven by modeling and analysis, such as fuzzing supported by symbolic execution, are now well established.
- Aid in integration efforts, providing stronger and less subject-to-change interface designs. When changes are needed, formal methods can support the comprehensive analysis required to ensure that all system elements are appropriately adapted in response to the needed change.
- Removal of certain runtime checks of values when such can be proven unnecessary. This shifting of checks, roughly speaking, from runtime to compile time, is complementary to the insertion of additional security checks to support zero trust architectural approaches. The effect of this reconfiguring of runtime checks is that, despite the added checking needed for strong cyber defense, system performance may actually be improved, since high-frequency dynamic checks (e.g., for data integrity) can more likely be removed. Ironically, this may be a signal that up-front assurances of safety and security may have the potential to reduce performance costs, counter to the myth that these benefits come at a cost of performance. This mythology has long been a justification for programming with unprotected abstractions, including lack of memory safety, as well as unsafe mutual trust within a system perimeter.
- Increase in the ability to reuse components or subsystems through assured evidence of the subsystems' quality, correctness, and performance. This ability can include reuse of models and proofs of critical common libraries and components. When strong assurances are associated with individual components, composition of those components can be safer.
- High assurance of key properties of requirements, system architecture, design commitments, implementation choices, configuration settings, test coverage, and maintenance operations. Automation of evidence and checking into builds and toolchains can create an "electric," or "live," configuration of documents, models, requirements, and other artifacts whose consistency (and coverage) is ensured on an ongoing basis. This capability is in contrast to early document-heavy systems engineering approaches in which design and engineering documents, once submitted, would immediately lose consistency with an evolving as-built system.
- Development of multiple models of complex systems. An important feature of modern formal methods is the use of multiple models covering different aspects of a system design and execution. This support for integration of models enables an incremental addition to existing systems of new modeling and analysis capabilities as they emerge. In a security setting, for example, side channels (characteristics not modeled or analyzed) can be reduced, or made more costly to attackers, as new models are developed to address those characteristics.
- Animation and partial testing of designs and architectures, even before full implementations are available. This capability is enabled through the use of provers, including the coupling of provers with modeling and execution. This approach is already evident in a number of experimental toolsets. This is part of the concept of moving to the left in test and evaluation.

- Support for, and alternative types of, system documentation. Formal models can provide a precise basis, for example, to generate explanations and other documents that may be more easily read by stakeholders than the underlying formal mathematical models and analyses. In particular, "formal" does not necessarily mean mathematical and impenetrable to non-expert engineers. Modern type systems illustrate that it is often possible to "hide the math" and create high levels of usability for formally powerful capabilities.

4.0 Key Findings

The discussions at the workshops led to a number of findings:

- There are several **major industrial use cases** of formal methods. These use cases go beyond traditional, narrowly targeted formal methods applications such as flight controls and cryptography and now include larger, more diverse, and more complex business-critical systems. Leaders of those projects assert a business case with high return-on-investment for the businesses involved. Indeed, this is accelerating demand for formal methods expertise in these sectors. In addition, use cases in Government are emerging from an increasing diversity of research and development projects. Cross-institution academic use cases have focused on building formally verified infrastructure stacks, as well as entire systems, systems-of-systems, operating systems, and hardware. These infrastructures are increasingly becoming widely adopted, as signaled, for example, by the alliance of the Linux Foundation with the verified seL4 operating system.
- The **ecosystems** around several formal toolchains **are maturing and steadily evolving**, with meaningful sustainment activities that have increasing robustness, ease of use, and opportunities for training new staff to become expert users of the toolchains. Several of the ecosystems have been evolving over long periods, in some cases three or four decades.
- There is evident **opportunity to broaden the scope** of applicability of formal tools through increased usability, adoptability, invisibility, and "integratability" of multiple toolchains. The concept of integratability is important, since it enables integration into existing toolchains, practices, languages, and other engineering affordances that have been widely adopted.
- There are **increasing opportunities** to link formal methods to existing systems engineering practices, including safety cases, security cases, hazard analysis, model-based approaches, and test-plan generation **for critical systems**. These include harmonization with tools, repository infrastructures, and process models. There are continuing challenges, however, in applying formal methods techniques to support existing architectural patterns for cyber-physical and embedded systems. An approach of mutual adaptation, in which engineering practices adapt to the necessity of production of evidence, may be appropriate in these mode challenging cases.

Emerging capabilities that can greatly speed adoption include the following:

- Model-based development methods, such as SCADE, can directly connect to some formal reasoning tools, enabling early verification and continuous value-add by formal reasoning tools. Formalized threat models could be developed across a community, enabling wide agreement of what is meant by, for example, private information leakage. This kind of model reuse can be widely useful, enabling reuse of formalism and model development for structural features of systems and key quality and potentially functional attributes of systems. Threat models are important in informing priority setting in addressing potential vulnerabilities and weaknesses in a system.

- "Invisible" formal methods can provide the value of formal analysis without requiring all users to learn a new specification language or requiring complete and correct property specifications before value is delivered. The canonical example is strong typing in many modern programming languages (e.g., Java, Ada, and Typescript). In this case, ordinary software developers gain significant benefit in type integrity in a manner that almost entirely hides the complexity of the underlying mathematical theory.
- Tools to facilitate "small theorems about large programs" can provide relatively easy on-ramps to a wider user base; i.e., an incremental approach to the use of formal methods, in which the models and reasoning are focused on a relatively narrow set of critical properties rather than on all aspects of functionality and quality.
- Machine learning methods (whether based on neural networks or other foundations) are increasingly ubiquitous in modern systems, and in certain cases they can be subjected to formal analysis. There are a number of candidate analyses through which models can be constructed from networks, with methods ranging from game theory to the use of SMT solvers.[4] In addition, machine learning methods could be used to automatically suggest methods of proof or correction of issues identified by formal tools, reducing the total human time needed and easing the path for new users of these techniques toward achieving assurance or other goals.
- The emerging cloud computing infrastructure can be used to facilitate larger scale computing resources to formal methods, particularly where there is strong reliance on the use of various kinds of solvers. It can itself be the subject of formal analysis.

5.0 Conclusion

Opportunities for additional actions with the potential for a large positive impact are as follows:

- **Integration with legacy systems**. Develop methods to provide for the maintenance of assurance cases for a system under change and for new integrations in existing platforms and systems-of-systems. This action enables enhancement to legacy systems through careful balancing of modeling of existing artifacts and incremental adaptation of those artifacts.
- **Security and privacy**. Expand existing and develop new methods to apply formal methods to problems in computer security and privacy. This action includes modeling related to a wide range of concerns such as threats, configuration integrity and root of trust, security attributes at the system level, security attributes at internal interfaces (such as for so-called zero trust designs), data flows, and other metadata.
- **Evidence and tool integration**. Develop practical methods to ensure that evidence including formal artifacts is brought along with systems as they are deployed, modified, and maintained. This is increasingly straightforward as tooling for proof management becomes increasingly common and capable of integration with more conventional software development toolchains. This also includes the use of domain-specific models and languages, as well as generation and synthesis tooling.

  User experience. Develop user experience concepts to support property specification, proof creation, and proof presentation, which can ensure that flaws in formal evidence are readily apparent to casual technical observers.

[4] This is an incremental step but not a solution to reliable machine learning, since challenges arise at scale and with different kinds of machine learning network architectures.

Appendix A. Abbreviations

AES	advanced encryption standard
CI/CD	continuous integration/continuous deployment
CNPS	Computing-Enabled Networked Physical Systems
DARPA	Defense Advanced Research Projects Agency
DevOps	development operations
DevSecOps	development, security, and operations
DoD	Department of Defense
I2O	Information Innovation Office
IEEE	Institute of Electrical and Electronics Engineers
IWG	Interagency Working Group
NITRD	Networking and Information Technology Research and Development
NSA	National Security Agency
R&D	research and development
SaaS	software as a service
SHA-2	secure hash algorithm 2
SMT	Satisfiability Modulo Theories

About the Authors

Dr. William Scherlis assumed the role of office director for DARPA's Information Innovation Office (I2O) in September 2019. In this role, he leads program managers in the development of programs, technologies, and capabilities to ensure information advantage for the United States and its allies, and coordinates this work across the Department of Defense and U.S. Government. He is a fellow of the IEEE and a Lifetime National Associate of the National Academy of Sciences. He currently serves as the DARPA NITRD Subcommittee representative.

Dr. Patrick Lincoln is Vice President of Information and Computing Sciences, and also director of the Computer Science Laboratory at SRI International. Dr. Lincoln leads research in the fields of formal methods, computer security and privacy, cyber-physical systems, computational biology, scalable distributed systems, and nanoelectronics. He serves on boards of directors and on boards of advisors of companies and government agencies.

Mr. William Martin joined DARPA as a program manager in the Information Innovation Office (I2O) in spring 2021. Martin joins DARPA from the National Security Agency (NSA), where he has served in a variety of roles, most recently as acting technical director and cybersecurity subject matter expert of the Laboratory for Advanced Cybersecurity Research. While at NSA, Martin focused on domain-specific languages, system analysis, and trustworthy AI. Mr. Martin serves as co-chair of the NITRD CNPS IWG.

Acknowledgments

Thanks to participants of the two Formal Methods at Scale workshops, to formal methods practitioners from across the community who offered abstracts communicating use cases and related discussions of formal methods at scale, and NITRD staff who supported the publication of this report. Special appreciation to Ms. Katie Dey, Vanderbilt University, for her administration, coordination, and execution efforts to make the workshops a reality.