Reusable and Understandable Formal Verification for Cyber-Physical Systems

Taylor T. Johnson[1]

Vanderbilt University, Nashville, TN 37215, USA,
taylor.johnson@vanderbilt.edu,
WWW home page: http://www.TaylorTJohnson.com

Abstract. Cyber-physical systems (CPS) encompass networked embedded computing systems coordinating to meet objectives in the physical world through sensor and actuator interfaces. Modern CPS are complex engineered systems composed through networking and coordination among numerous subsystems. In CPS industries ranging from automotive to aerospace, subsystems are frequently reused in numerous systems, such as electronic control modules in motor vehicles and avionics control systems in airplanes. This abstract describes our ongoing research to develop novel algorithms implemented within our HyST software framework to leverage prior verification results to ease the verification and validation process, through three primary methods. First, we have developed incremental model checking algorithms that essentially cache prior verification results and leverage these when design changes are made, and are a standard approach in usable industry-grade tools, but that have not previously been explored in the context of CPS. Second, as a part of reusing prior verification results, we are exploring how designers can understand sets of changes in their designs and the resulting modification of property satisfaction or violation. Third, we have developed a runtime middleware for CPS that interprets the behavior of controllers modeled as networks of hybrid automata at runtime, and we have preliminarily implemented this method within the MathWorks' Simulink/Stateflow CPS design tool to preserve properties verified at design time through their runtime implementation. Finally, we have evaluated these methods on several CPS studies, such as electrical distribution networks in CPS (microgrids) and in groups of unmanned aerial vehicles (UAVs).

Keywords: cyber-physical systems, formal methods, hybrid systems

1 From Cyber-Physical Design Reuse to Understandable Verification Reuse

Complex CPS, such as manned and unmanned aerial systems (UAS) and satellite constellations, rely on the correct operation of numerous subcomponents over many versions during possibly decades-long lifespans. In some domains, such as aerospace, CPS may be used over decades-long lifespans, are subject to frequent cyber-physical design reuse, and have the same potential to reuse

verification results over iterations of the same design to improve design and verification efficiency and cost-effectiveness. Owing in part to the long operational lifespans in some CPS domains (e.g., in aerospace), and the systems engineering process for other CPS (e.g., in automotive), CPS are frequently composed of legacy subcomponents and recent state-of-the-art subcomponents. Additionally, as CPS incorporate increased autonomy with little human supervision, reusing and understanding verification results for existing legacy systems incorporated with new systems incorporating autonomy, reducing the design and verification effort is a crucial concern. Reusing verification results and artifacts across design iterations and abstraction layers in CPS is a primarily unaddressed challenge that our research addresses by building on previous fundamental results for verification reuse in digital logic design and in purely software systems.

Design-reuse problems have famously occurred in aviation, such as the Ariane 5 flight 501 disaster was a result of reusing Ariane 4's software without appropriate updates for the increased thrust of the new rocket [1]. In Ariane 5, software made an assumption about the physical dynamics of the rocket, but the software was reused from Ariane 4, while Ariane 5 had greater thrust, so this assumption was invalid, and we call such assumptions *cyber-physical specification mismatches* [19].

While it is clear there is significant reuse of designs in CPS due to budget constraints and a desire to use commercial off-the-shelf (COTS) components, there are fewer efforts to leverage reuse of verification effort in CPS. This is surprising since verification dominates the cost (in every metric, from dollars to manpower hours) of system development. For CPS in domains subject to certification by external auditors, such as aerospace, medical devices, and nuclear, there do exist processes for certification reuse, where subsequent versions of systems attempt to leverage regulatory approval of future versions of the system on past versions. While certification reuse is a process-based technique to build improved confidence and trust in future systems based on reuse of existing V&V evidence and results, it is different from verification reuse, where verification methods are algorithmically designed to build on existing verification results from prior analysis efforts of past versions of systems.

Our primary efforts toward understanding and reusing verification results are as follows and are implemented within our HyST software framework for verifying hybrid automaton models of CPS (Figure 1). The Hybrid Source Transformation and Translation tool (HyST) is a design framework for hybrid automata networks modeling CPS [4]. HyST supports several state-of-the-art hybrid systems model checkers, including Flow* [12], SpaceEx [18], HyComp (built on top of nuXmv) [14], and also integrates with the MathWorks' Simulink/Stateflow [2]. HyST's primary technical purpose is as an evaluation framework for sound abstractions to address the state-space explosion problem for hybrid automata [16, 26], and supports numerous automated abstractions implemented as source-to-source model transformations, including hybridization [3], continuization [5], decoupling [24], and order-reduction [31]. HyST has been used to evaluate numerous CPS case studies, including power electronics [7–10], autonomous systems

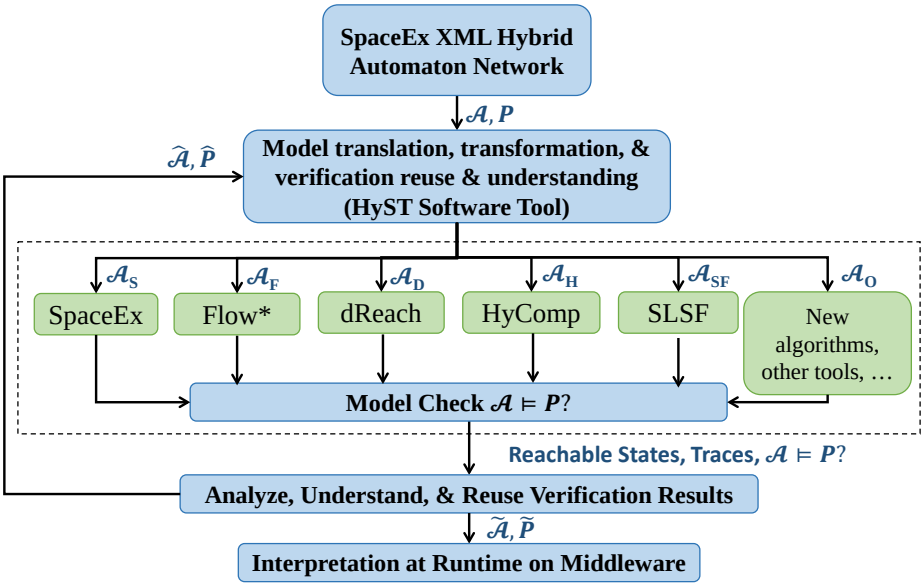

Fig. 1. Overview of our HyST formal verification software framework including verification reuse and understanding of a portfolio of verification tools for hybrid automaton models of CPS.

and swarm robotics [30], invariant generation for nonlinear ODEs [25], large-scale systems [29], automotive systems [23], and general nonlinear systems [29].

Within HyST, we have created the first incremental model checking algorithms for hybrid automata, where subsequent executions of model checking algorithms reuse existing verification results, avoiding on average having to explore the entire state-space again and drastically saving computation time, particularly over design iterations. To scale model checking methods, we have developed formal abstractions for CPS by fundamentally transforming the approximations of order-reduction methods from control theory to have provable error guarantees, which has drastically alleviated the state-space explosion problem in model checking CPS, allowing verification of systems with thousands of real state-variables [31]. We have designed the first runtime environment middleware to interpret hybrid automata at runtime, where the middleware has been designed with provable guarantees that will preserve any properties verified using the hybrid automaton model in its actual execution in CPS [2, 10]. We are evaluating the reusable and understandable formal verification methods in challenging CPS evaluations, such as distributed electric power distribution networks (microgrids) [7, 10] and in distributed swarm robotics systems using groups of UAVs [11].

2 Verification Reuse and Understanding

The most basic examples of verification reuse arise in mathematics through both the application of theorems and reuse of proofs or subparts of proofs. Suppose a method to establish a certain type of theorem is known. For instance, a standard method to show two sets A and B are equal is to show A is a subset of B, $A \subseteq B$, and to show B is a subset of A, $B \subseteq S$, so that one can conclude A and B contain the same elements and thus $A = B$. This statement is a theorem, more formally stated as $A \subseteq B \wedge B \subseteq A$ if and only if (iff) $A = B$, and it can be reused and applied to check whether arbitrary sets A and B are equal or not. Such a theorem application (or theorem reuse) is an example of verification reuse, and this type of reasoning is the basis for most mathematics building on existing results.

In addition to theorem reuse, a possibly more important instance of verification reuse is in proof reuse, where the logical, structural, and creative arguments used to establish a theorem may be reused to establish other proofs. Mathematicians commonly use proof reuse through tools like proof templates. Many modern computer-based automated/semi-automated reasoning tools, such as interactive theorem provers (ITPs), automated theorem provers (ATPs), satisfiability solvers (SAT-solvers), and satisfiability modulo theories (SMT-solvers) have in their bases proof and theorem reuse. This is done in part for computational benefits, as well as for usability and understandability benefits. The computational benefits include being able to check a given proof of a theorem, which is of course a much simpler problem (in the computational complexity sense) than finding a proof of a theorem. The usability benefits include being able to apply a theorem to establish a result once its hypotheses are known.

The computational benefits of proof and theorem reuse underlie the modern algorithms in SAT-solvers and SMT-solvers, where the Davis-Putnam-Logemann-Loveland (DPLL) and DPPL(T) search algorithms learn clauses and backtrack when conflicts arise. The subsequent Conflict-Driven Clause Learning (CDCL) algorithms used in most modern SAT-solvers have this reuse at their heart [6]. Theorem and proof reuse have inspired similar attempts to reuse verification results in systems and software. While there are efforts to develop verification reuse methods in purely discrete systems [22, 28], such as incremental model checking [27] and component reuse [15], this problem is unaddressed for CPS that involve both physical and computational processes modeled respectively by differential equations and state machines.

3 Outlook: CPS Verification Understanding and Reuse

Through our research toward reusing prior formal verification results in CPS, we hope to improve the engineering process of realistic CPS composed of legacy and novel subsystems. A key component of this effort is how to understand prior verification results, from both a high-level conceptual standpoint as well as a low-level detailed standpoint on actual tool outputs, such as sets of reachable states. In the context of recent efforts such as the DARPA High-Assurance

Cyber Military Systems (HACMS) program that aimed in part to develop verified components [17], the BEDROCK project [13], and verified microkernels like seL4 [20] and verified optimizing compilers like CompCert [21], enabling reuse of verification results in CPS will likely rely on both assume-guarantee and compositional verification approaches, which may allow the verification results of these clean-slate subsystems to be reused in later systems. We are not yet to the state where CPS may be fully verified (e.g., at the system level, or even at every subcomponent), although progress is being made in many ongoing efforts (in e.g., verification of cryptographic protocols, key exchanges, control systems, compilers, operating systems, etc.). In this presentation, we will discuss our past efforts in this area as well our ongoing research, challenges, and directions for future progress toward reusing and understanding verification results for CPS.

References

1. Ariane 5 flight 501 failure, report by the inquiry board. Technical report, ESA Inquiry Board, Paris, France, July 1996.
2. Stanley Bak, Omar Ali Beg, Sergiy Bogomolov, Taylor T. Johnson, Luan Viet Nguyen, and Christian Schilling. Hybrid automata: from verification to implementation. *Software Tools for Technology Transfer (STTT)*, August 2017.
3. Stanley Bak, Sergiy Bogomolov, Thomas A. Henzinger, Taylor T. Johnson, and Pradyot Prakash. Scalable static hybridization methods for analysis of nonlinear systems. In *Proc. of the 19th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC)*. ACM, April 2016.
4. Stanley Bak, Sergiy Bogomolov, and Taylor T. Johnson. HyST: A source transformation and translation tool for hybrid automaton models. In *Proc. of the 18th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC)*. ACM, 2015.
5. Stanley Bak and Taylor T. Johnson. Periodically-scheduled controller analysis using hybrid systems reachability and continuization. In *36th IEEE Real-Time Systems Symposium (RTSS)*, San Antonio, Texas, December 2015. IEEE Computer Society.
6. Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In Ed Clarke, Thomas Henzinger, and Helmut Veith, editors, *Handbook of Model Checking*. 2014.
7. Omar Ali Beg, Houssam Abbas, Taylor T. Johnson, and Ali Davoudi. Model validation of pwm dc-dc converters. *IEEE Transactions on Industrial Electronics. (TIE)*, June 2017.
8. Omar Ali Beg, Ali Davoudi, and Taylor T. Johnson. Charge pump phase-locked loops and full wave rectifiers for reachability analysis (benchmark proposal). In *3rd Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Vienna, Austria, April 2016.
9. Omar Ali Beg, Ali Davoudi, and Taylor T. Johnson. Reachability analysis of transformer-isolated dc-dc converters (benchmark proposal). In *4th Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Pittsburgh, PA, April 2017.
10. Omar Ali Beg, Taylor T. Johnson, and Ali Davoudi. Detection of false-data injection attacks in cyber-physical dc microgrids. *IEEE Transactions on Industrial Informatics*, 2017.

11. Leonardo Bobadilla, Taylor T. Johnson, and Amy LaViers. Verified planar formation control algorithms by composition of primitives. In *AIAA SciTech*, Kissimmee, Florida, January 2015. AIAA.
12. X. Chen, Erika Abraham, , and Sriam Sankaranarayanan. Talyor model flowpipe construction for non-linear hybrid systems. In *IEEE Real-Time Systems Symposium*, pages 183–192, 2012.
13. Adam Chlipala. Mostly-automated verification of low-level programs in computational separation logic. In *Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '11, pages 234–245, New York, NY, USA, 2011. ACM.
14. Alessandro Cimatti, Alberto Griggio, Sergio Mover, and Stefano Tonetta. HyComp: An SMT-based model checker for hybrid systems. In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 9035 of *Lecture Notes in Computer Science*, pages 52–67. Springer Berlin Heidelberg, 2015.
15. Laurent Doyen, Thomas A. Henzinger, Barbara Jobstmann, and Tatjana Petrov. Interface theories with component reuse. In *Proceedings of the 8th ACM International Conference on Embedded Software*, EMSOFT '08, pages 79–88, New York, NY, USA, 2008. ACM.
16. Parasara Sridhar Duggirala, Chuchu Fan, Matthew Potok, Bolun Qi, Sayan Mitra, Mahesh Viswanathan, Stanley Bak, Sergiy Bogomolov, Taylor T. Johnson, Luan Viet Nguyen, Christian Schilling, Andrew Sogokon, Hoang-Dung Tran, and Weiming Xiang. Tutorial: Software tools for hybrid systems verification, transformation, and synthesis: C2e2, hyst, and tulip. In *Proceedings of the IEEE Multi-Conference on Systems and Control (¡a href="http://www.msc2016.org/"¿MSC 2016¡/a¿)*, Las Vegas, NV, USA, September 2016.
17. Kathleen Fisher. Using formal methods to enable more secure vehicles: Darpa's hacms program. In *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming*, ICFP '14, pages 1–1, New York, NY, USA, 2014. ACM.
18. Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In *Computer Aided Verification (CAV)*, LNCS. Springer, 2011.
19. Taylor T. Johnson, Stanley Bak, and Steven Drager. Cyber-physical specification mismatch identification with dynamic analysis. In *International Conference on Cyber-Physical Systems (ICCPS)*, 2015.
20. Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. sel4: Formal verification of an os kernel. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pages 207–220, New York, NY, USA, 2009. ACM.
21. Xavier Leroy. Formal verification of a realistic compiler. *Commun. ACM*, 52(7):107–115, July 2009.
22. Magnus O. Myreen. Reusable verification of a copyingcollector. In Gary T. Leavens, Peter O'Hearn, and Sriram K. Rajamani, editors, *Verified Software: Theories, Tools, Experiments*, volume 6217 of *Lecture Notes in Computer Science*, pages 142–156. Springer Berlin Heidelberg, 2010.

23. Luan Viet Nguyen, James Kapinski, Xiaoqing Jin, Jyotirmoy Deshmukh, Ken Butts, and Taylor T. Johnson. Abnormal data classification using time-frequency temporal logic. In *20th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC 2017)*. ACM, April 2017.
24. Andrew Sogokon, Khalil Ghorbal, and Taylor T. Johnson. Decoupled simulating abstractions of non-linear ordinary differential equations. In *Proceedings of the 21st International Symposium on Formal Methods (¡a href="http://fm2016.cs.ucy.ac.cy"¿FM 2016¡/a¿)*. Limassol, Cyprus, December 2016.
25. Andrew Sogokon, Khalil Ghorbal, and Taylor T. Johnson. Non-linear continuous systems for safety verification (benchmark proposal). In *3rd Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Vienna, Austria, April 2016.
26. Andrew Sogokon, Paul Jackson, and Taylor T. Johnson. Verifying safety and persistence properties of hybrid systems using flowpipes and continuous invariants. In *9th NASA Formal Methods Symposium (NFM 2017)*, May 2017.
27. Oleg V. Sokolsky and Scott A. Smolka. Incremental model checking in the modal mu-calculus. In DavidL. Dill, editor, *Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 351–363. Springer Berlin Heidelberg, 1994.
28. N. Soundarajan and S. Fridella. Inheritance: from code reuse to reasoning reuse. In *Software Reuse, 1998. Proceedings. Fifth International Conference on*, pages 206–215, 1998.
29. Hoang-Dung Tran, Luan Viet Nguyen, and Taylor T. Johnson. Large-scale linear systems from order-reduction (benchmark proposal). In *3rd Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Vienna, Austria, April 2016.
30. Hoang-Dung Tran, Luan Viet Nguyen, Weiming Xiang, and Taylor T. Johnson. Distributed autonomous systems (benchmark proposal). In *4th Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Pittsburgh, PA, April 2017.
31. Hoang-Dung Tran, Luan Viet Nguyen, Weiming Xiang, and Taylor T. Johnson. Order-reduction abstractions for safety verification of high-dimensional linear systems. *Discrete Event Dynamic Systems (DEDS)*, 2017.