Challenges of Applying Formal Methods to Automotive Control Systems

Xiaoqing Jin, Jyotirmoy V. Deshmukh, James Kapinski, Koichi Ueda, Ken Butts
Powertrain Control – Model-based Development,
Toyota Technical Center
{xiaoqing.jin, jyotirmoy.deshmukh, jim.kapinski, koichi.ueda, ken.butts}@tema.toyota.com

Abstract: The automotive industry has significantly improved safety, driving experience, and fuel economy over the past few decades. These advances have come at the expense of an increase in the number of networked electronic control units (ECUs) and the complexity of the embedded software. To manage this complexity, the automotive industry has embraced model-based development (MBD). MBD provides better integration of hardware and software design and early verification and validation (V&V) capability by high-fidelity simulation. Simulation techniques, while effective at improving design confidence, are, unfortunately, not rigorous enough to guarantee bug-free designs. In this position paper, we call for reliable and efficient approaches to leverage formal methods to enhance the extensive, but incomprehensive, nature of simulation-based validation.

I. AUTOMOTIVE CONTROL SYSTEMS

Modern automobiles have more than 70 networked electronic control units (ECUs) [13]. Wireless technologies enabling the use of keyless entry, GPS navigation, and Bluetooth are being routinely deployed to improve the driving experience, while anti-lock braking, stability control, radar-based adaptive cruise control and collision warning systems provide improved safety. Two automotive industry strategic elements include a) improved efficiency and alternative fuels to reduce environmental impact, and, b) connected vehicles that communicate with each other or with the surrounding infrastructure to improve safety [12]. Automobiles are thus becoming more green and intelligent, but at the expense of a dramatic increase in the complexity of the underlying control and software systems. To meet the increased computational needs, parallel computing technology, such as multi-core ECUs on high speed networks, is being adopted. This makes the problem of designing and verifying reliable automotive software even more challenging.

In the automotive and avionics industries, the model-based development (MBD) paradigm is being promoted to reduce development time and improve design quality. In this paradigm, designers build models of the physical processes or components (plant models) and models of the control algorithm in a block-diagram-oriented visual language (such as Simulink®[1]). Early verification and validation (V&V) of the system typically involves design validation by extensive open-loop (only the controller) and closed-loop (plant + controller model) testing. Such testing usually leverages the technologies of (automatic) code generation, (automatic) test generation and simulation. While such a testing-based approach can be made as extensive as permitted by the available resources and time-to-market constraints, it cannot be made exhaustive due to fundamental limitations.

II. V&V FOR AUTOMOTIVE CONTROL SYSTEMS

A key challenge for the V&V process for industrial MBD designs is the scale and complexity of the designs. Automotive control systems are cyber-physical: discrete-time controllers on embedded hardware (cyber) interacting with continuous-time plants through sensors and actuators (physical). As the verification problem for standard software systems is undecidable or otherwise intractable, it is to be expected that the verification problem for cyber-physical systems (CPS) is intractable as well. The physical aspect of CPS makes the verification problem even more formidable as it disallows existing verification methods from being directly applicable. Traditionally, the CPS verification community has focused on sound but incomplete methods to certify system correctness; for example, techniques to overapproximate the set of reachable states [1, 2]. The accuracy of these approaches relies heavily on the quality of abstractions used for analysis. For verification, this can lead to sound but often inaccurate results such as spurious counterexamples. In industrial practice, we often deal with highly complex systems, and obtaining high fidelity abstractions while maintaining soundness is nearly impossible. Further, the format of industrial models is often not amenable to formal analysis (as in the case of Simulink models).

On the other hand, in the MBD paradigm, engineers often validate their design models using simulations. Simulations

[1] Simulink is a registered trademark of The MathWorks, Inc., Natick, Ma.

or other concrete executions are also often obtained as a natural part of the test and certification process (e.g., model in the loop system (MILs), software in the loop (SILs), and hardware in the loop systems (HILs) testing). If we can leverage formal methods to assist us in reasoning about concrete executions (simulations), we could improve the reliability and correctness in design.

We propose investigation into such new technologies that would scale to complex, industrial-scale MBD designs. We posit that techniques that utilize simulation data and have a foundation in formal analysis can add significant value to the MBD process, while simultaneously bridging the gap between formal methods that have been heretofore successful in academic settings and MBD designs of industrial scale and complexity. Such techniques that require only the ability to simulate the embedded control system could be applied to any representation of the system created as a natural artifact of the MBD design process. In the following section, we present a number of challenge problems for these techniques and highlight some of the existing early research in this area.

III. CHALLENGE PROBLEMS FOR SIMULATION-GUIDED FORMAL ANALYSIS (SiGFA)

A. *Requirement Extraction.*

A pervasive issue for many industrial MBD designs is that requirements are rarely expressed in a formalism that can be digested by verification tools or techniques. These are often expressed only in natural language in the form of requirement documents, which informally (albeit carefully) provide specifications for system components. Another common scenario is that requirements are usually provided at a very high-level (e.g. reduce exhaust gas emissions), but their translation to concrete design goals or precise after-design characterizations is often unavailable. Furthermore, correlations between high-level design requirements and decomposed specifications for specific design components are also typically absent. Our thesis is that a large amount of design knowledge can be gleaned from simulations of the design and can be used to extract *de facto* logical specifications of the design. These logical specifications can then be refined by an MBD engineer to a set of formal requirements to obtain a *requirement model* as a substitute for the existing requirement documents. This can then be leveraged when a prototype design is refined and mapped onto an actual implementation, or in subsequent revisions to the design. In previous work [3], we explored the possibility of mining temporal logic requirements from closed-loop control models. Our algorithm assumes the designer provides a requirement template and uses simulation traces to generate a candidate requirement. It then leverages a simulation-guided *falsification* tool to check if there is any simulation trace of the closed-loop model that does not satisfy the candidate requirement, and if it exists, the algorithm uses the trace to refine the candidate requirement. The approach is promising as it is able to extract formal requirements for a fairly complicated industrial MBD design.

An open question is whether requirement extraction can be done directly from simulation traces with the help of learning algorithms without the use of template requirements. This reduces the burden on control designers to understand and write appropriate templates. Another open question is to identify the right requirement formalism that combines the elegance of temporal logic with the traditional control-oriented specifications such as stability, robustness, optimality, and disturbance rejection.

B. *Stability Analysis and Computing Performance Bounds.*

Proving Lyapunov stability of systems has been an important problem in the control theory literature. A common approach involves identifying a *Lyapunov function*, which has specific properties that serve as sufficient conditions to prove stability. Identifying such functions is a challenging problem that has had continuous and sustained interest in the CPS community. A related problem is that of computing the performance bounds of a given system within a region of interest. Typically, it is assumed that a closed-form analytic representation of the system dynamics exists and is known. This is in stark contrast with industrial models which may contain black-box components with unknown dynamics, legacy code, or components with models in proprietary formats (such as Simulink models) where the formal semantics are not always available.

A challenge problem is to obtain candidate Lyapunov functions and performance bounds, or to perform any other Lyapunov-like analysis using the data available from simulation traces of a system. In [4, 5] the authors assume that precise system dynamics are known, but use simulations to assist estimation of a region of attraction using Sum-of-Squares (SoS) optimization and semi-definite programming. In [6], we propose a simulation-guided framework to obtain candidate SoS polynomial Lyapunov functions directly from system traces, along with an optimization-guided *falsification* tool to help refine the candidate. Many open questions remain: (1) extending SiGFA techniques to obtain piecewise

Lyapunov functions for hybrid systems, which would broaden the applicability of the technique to a wider range of systems, (2) using alternative forms for Lyapunov functions such as non-polynomial, purely piecewise-linear, *etc.,* and (3) obtaining soundness guarantees for black-box systems.

C. *State Space Coverage*

Current design validation methods for open-loop testing focus on modified condition/decision coverage (MC/DC) of controller models. This coverage metric inspired from software verification checks if branches in the program control flow of the controller model are adequately exercised by a given set of test-cases. Most research effort involves automatically generating test-cases so as to maximize the MC/DC metric. In the CPS setting, it is unclear whether MC/DC is enough. For example, consider the case where the plant model is continuous and the MC/DC criterion is inapplicable. Previous work on measuring the coverage of continuous state-spaces proposes using metrics inspired from statistics such as the *Star Discrepancy* metric [8]. An open problem is to extend such notions of coverage to hybrid systems. Another open question is to define a metric that is able to quantify whether the given set of test cases is *diverse* enough, *i.e.,* whether it explores sufficiently distinct regions of the hybrid state-space over time.

D. *Simulation-based Falsification*

Traditional verification approaches focus on over-approximating the set of reachable states of a hybrid system over a bounded time horizon, and checking if this set intersects with a known unsafe state set. For systems with piecewise linear or affine dynamics, such approaches have slowly matured, and tools such as SpaceEx [2] can now perform reachability analysis for systems with a small amount of switching with reasonable accuracy. However, as industrial models have highly nonlinear dynamics, features such as look-up tables, variable transport delays, and copious amount of switching, applying conservative approaches is challenging. *Falsification* tools such as S-TaLiRo [9] and Breach [10], on the other hand, optimize over the large number of model inputs and initial conditions with the help of a nonlinear or stochastic global optimizer on the backend, in order to find behaviors that do not satisfy a given temporal logic property. These approaches typify the SiGFA paradigm, and have already seen success in the automotive domain [3, 11]. An open challenge is to improve the kind of formal guarantees that these tools provide, for example by combining falsification tools with statistical model checking [7] to give probabilistic guarantees of correctness. Another open challenge is to improve the performance of these approaches on models that contain a large number of discrete components (which makes the underlying optimization problem significantly harder).

References

[1] Chutinan, A. and Krogh, B. H., *Verification of infinite-state dynamic systems using approximate quotient transition systems*. IEEE Transactions on Automatic Control, *46*(9), pp. 1401-1410, 2001.

[2] Frehse, G., Le Guernic C., Donzé A., Cotton S., Ray R., Lebeltel O., Ripado R., Girard A., Dang T. and Maler O., *SpaceEx: Scalable verification of hybrid systems,* In Proc. of Computer Aided Verification, pp. 379-395, 2011.

[3] Jin, X., Donzé, A., Deshmukh, J. V. and Seshia, S. A. *Mining requirements from closed-loop control models*. In Proc. of the 16th international conference on Hybrid systems: computation and control, pp. 43-52, 2013.

[4] Topcu, U. and Packard, A.: *Stability region analysis for uncertain nonlinear systems*. IEEE Transactions on Automatic Control, *54*, pp. 1042-1047, 2009.

[5] Topcu, U., Seiler, P. and Packard, A.: *Local stability analysis using simulations and sum-of-squares programming*. Automatica, *44*, pp.2669-2675, 2008.

[6] Kapinski J., Deshmukh J., Sankaranarayanan S., and Aréchiga N., *Simulation-guided Lyapunov analysis for hybrid dynamic systems*, Pending publication.

[7] Zuliani, P., Platzer, A., and Clarke, E. M., *Bayesian statistical model checking with application to Simulink/Stateflow verification*, In Proc. of Hybrid systems: computation and control, pp. 243-252, 2010.

[8] Nahhal, T. and Dang, T., *Test coverage for continuous and hybrid systems*. In Proc. of Computer Aided Verification, pp. 449-462, 2007.

[9] Annpureddy, Y., Liu, C., Fainekos, G., and Sankaranarayanan, S., *S-TaLiRo: A tool for temporal logic falsification for hybrid systems*, In Proc. of Tools and Algorithms for the Construction and Analysis of Systems, pp. 254–257, 2011.

[10] Donzé, A., *Breach, a toolbox for verification and parameter synthesis of hybrid systems*. In Proc. of Computer Aided Verification, pp. 167-170, 2010.

[11] Fainekos, G. E., Sankaranarayanan, S., Ueda, K., and Yazarel, H., *Verification of automotive control applications using S-TaLiRo*. In Proc. of American Control Conference, pp. 3567-3572, 2012.

[12] *Toyota 2013 North Environmental Report,* URL: http://www.toyota.com/about/environmentreport2013/index.html

[13] Christof Ebert and Capers Jones. 2009. Embedded Software: Facts, Figures, and Future. *Computer* 42, 4 (April 2009), 42-52.