Model Checking Transportation Software

Sagar Chaki, Carnegie Mellon Software Engineering Institute

Abstract: The safe and secure operation of transportation systems is increasingly becoming predicated on software correctness. The complexity of such software, driven by its tight coupling with the physical environment, and increasing concurrency and distributedness, is limiting the effectiveness of traditional validation methods in assuring its correctness. Software Model Checking (SMC) provides a promising way to address some of the key problems in this domain. SMC has been applied successfully, and at scale, in other domains. However, transportation software remains a largely unchartered territory for it. Successful application of model checking to transportation systems will require a more open mindset, cross-community collaboration, and technical breakthroughs in specific areas.

Introduction. The transportation sector is undergoing a historical paradigm shift. Two particular trends are increasingly evident. First, vehicles are becoming software intensive, and cost pressures are forcing software of different criticality levels to share physical resources. In many cases, software is added after the vehicle is purchased, as an "update" [Wired12]. Second, interconnectedness between vehicles is being introduced gradually, and will become the norm in years to come. For example, Europe is moving toward connected cars [GSMA13], and the US will likely follow soon. Both trends lead to complexity, and expose limitations of traditional validation methodologies, based on testing, in providing sufficient assurance of safe and secure operation of vehicles. However, the marriage of formal verification techniques with domain-specific design and development constraints provides a way forward.

Model checking [CES09] is an algorithmic and exhaustive verification technique to check whether a finite state model M satisfies a temporal logic specification φ. In addition to being automated, model checking provides a counterexample as diagnostic feedback if M does not satisfy φ. While model checking was proposed originally to verify correctness of synchronization skeletons of concurrent programs [CE81], it has since been applied successfully to verify a wide range of systems, including both hardware [BCMH92] and software [JM09]. Moreover, it is now known how to make model checkers "certifying", i.e., augment them to generate machine-checkable proofs of correctness, in addition to counterexamples. Due to its exhaustive and formal nature, model checking is a good basis for verifying safety-critical software, which is widely prevalent in the transportation domain. Still a number of challenges must be overcome in order to make the best use of model checking technology for verifying transportation software.

Physicality. Traditionally, model checking has been developed from a purely logical perspective. However, transportation is a classical cyber-physical domain. For example, vehicular control algorithms are tightly coupled with the physical environment, and are deigned to produce the right results only in a world governed by known physical laws. State-of-the-art source code analysis tools are based on logics that are either discrete (e.g., SLAM [BLR11]) or linear (e.g., ASTREE [CC05]). Physical laws, on the other hand, involve continuous non-linear dynamics (e.g., differential equations). Hybrid automata provide a rich enough formalism to express such laws, but the results of verifying hybrid automata must be translated over to the verification of source code. One approach is to develop efficient SMT solvers (such as dReal [GKC13]) for non-linear theories, together with new abstract domains [RJGF12] and abstraction-refinement techniques that leverage such theories. The control and real-time communities have addressed issues like stability and schedulability, but it is necessary to incorporate their theories into program semantics in order to reproduce their results for software.

Concurrency. Software on any vehicle is inherently concurrent. Typically, a vehicle consists of several processors connected by a communication bus. Each processor has several threads executing in parallel, and communicating with each other via a message-passing or shared memory abstraction offered by the operating system and middleware. Concurrency is known to make model checking difficult since it exacerbates the statespace explosion problem. The distributed nature of connected vehicles makes this even more challenging. In general, a vehicle could be connected not only to other vehicles, but to non-vehicular nodes (e.g., GPS satellites and cell towers).

Compositional reasoning, such as assume-guarantee [P84], is widely recognized to be the most promising avenue for dealing with statespace explosion. However, traditional notions of concurrency are inadequate for the transportation domain. For example, shared memory abstraction is not appropriate for communication between vehicles since it is impossible to ensure sequential consistency. An asynchronous network is also unrealistic since it prevents us from leveraging constraints, such as known message passing delays, bounded clock jitters.

The distributed algorithms community has made rapid strides by developing algorithms and proving their correctness for a variety of communication models. However, this has been done at the pseudo-code level using manual approaches [Lynch96], and such algorithms have largely avoided the physical environment. More recently, new physics-aware distributed algorithms have been proposed [ABRM13], but they have been validated via simulations. These algorithms must be proven correct at the source code level. Compositional model checking is a promising approach, but in order to apply it soundly, we must first provide a well-defined semantics of the concurrency. Next, compositional proof rules must be defined, and model checking algorithms developed that can discharge the proof obligations. All these remain open areas of research.

Uncertainty. Vehicles operate in inherently uncertain environments. Traditionally, the job of dealing with this uncertainty has been delegated to human controllers. As the push toward more autonomy intensifies, the software in the vehicle must take on more of this responsibility. In recent years, two flavors of formal probabilistic verification have emerged, and been applied to a number of different domains – probabilistic model checking [KNP04] and statistical model checking [LDB10]. Both are promising approaches for achieving probabilistic assurance in transportation systems. However, they have not yet been applied to large distributed transportation systems, and scaling them compositionally [FHKP11] to such instances remains an open problem.

Evolution and Certification. Software is always undergoing changes. During development, it changes from high-level pseudo-code to source code to binaries. If a model-driven development approach is performed, then source code is first generated, and then hand-tweaked for performance. Each of these steps rely of large trusted computing bases (e.g., compilers), that are often larger and more complex than the software being transformed. Even after software is released, it is continuously updated for maintenance and feature enhancement. Re-verification of the software after each step is infeasible, and wasteful. An important problem is whether the evidence generated by model checking at one level can be reused at a different level. For example, can the inductive invariants computed at the source code level during reachability analysis be transferred to the binary level in a provably sound manner during compilation [CILWZ07]. When a certain component is updated (or a new component inserted) to add a new feature, can assumptions made it about other components be reused to make re-verification more efficient [CCSS08]. A related question is how to incorporate such evidence into the certification process for transportation software to strengthen the overall process.

Conclusion. Transportation is one of the most prominent cyber-physical domains that is rapidly becoming software reliant. At the same time, assuring this software is critical for ensuring safety of passengers, and limiting liabilities in case of accidents. Due to tight coupling with the physical world, and massive degree of concurrency, transportation software can only be assured to a high level via exhaustive and automated techniques. The

assurance must be obtained at a level that is as close to hardware as possible, and must be repeatable in an incremental manner once the software is modified. Based on its success in other domains, such as device drivers, software model checking provides a promising approach to implement such an assurance technique. However, specific technical breakthroughs and cross-disciplinary research is needed for this vision to be achieved[1].

References

[Wired12] http://www.wired.com/autopia/2012/09/tesla-over-the-air

[GSMA13] http://www.gsma.com/newsroom/gsma-every-new-car

[CES09] Edmund M. Clarke, E. Allen Emerson, Joseph Sifakis: Model checking: algorithmic verification and debugging. Commun. ACM 52(11): 74-84 (2009)

[CE81] Edmund M. Clarke, E. Allen Emerson: Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. Logic of Programs 1981: 52-71

[BCMDH92] Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill, L. J. Hwang: Symbolic Model Checking: 10^20 States and Beyond. Inf. Comput. 98(2): 142-170 (1992)

[JM09] Ranjit Jhala, Rupak Majumdar: Software model checking. ACM Comput. Surv. 41(4) (2009)

[BLR11] Thomas Ball, Vladimir Levin, Sriram K. Rajamani: A decade of software model checking with SLAM. Commun. ACM 54(7): 68-76 (2011)

[CC05] Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, Xavier Rival: The ASTREÉ Analyzer. ESOP 2005: 21-30

[GKC13] Sicun Gao, Soonho Kong, Edmund M. Clarke: dReal: An SMT Solver for Nonlinear Theories over the Reals. CADE 2013: 208-214

[RJGF12] Pierre Roux, Romain Jobredeaux, Pierre-Loïc Garoche, Eric Feron: A generic ellipsoid abstract domain for linear time invariant systems. HSCC 2012: 105-114

[P84] Amir Pnueli: In transition from global to modular temporal reasoning about programs. In Logics and Models of Concurrent Systems, K. R. Apt, Ed. NATO ASI, vol. 13. 1984. Springer-Verlag, 123–144.

[Lynch96] Nancy A. Lynch: Distributed Algorithms. Morgan Kaufmann 1996, ISBN 1-55860-348-4

[ABRM13] Seyed (Reza) Azimi, Gaurav Bhatia, Ragunathan Rajkumar, Priyantha Mudalige: Reliable intersection protocols using vehicular networks. ICCPS 2013: 1-10

[KNP04] Marta Z. Kwiatkowska, Gethin Norman, David Parker: PRISM 2.0: A Tool for Probabilistic Model Checking. QEST 2004: 322-323

[LDB10] Axel Legay, Benoît Delahaye, Saddek Bensalem: Statistical Model Checking: An Overview. RV 2010: 122-135

[FHKP11] Lu Feng, Tingting Han, Marta Z. Kwiatkowska, David Parker: Learning-Based Compositional Verification for Synchronous Probabilistic Systems. ATVA 2011: 511-521

[CILWZ07] Sagar Chaki, James Ivers, Peter Lee, Kurt C. Wallnau, Noam Zeilberger: Model-Driven Construction of Certified Binaries. MoDELS 2007: 666-681

[CCSS08] Sagar Chaki, Edmund M. Clarke, Natasha Sharygina, Nishant Sinha: Verification of evolving software via component substitutability analysis. Formal Methods in System Design 32(3): 235-266 (2008)

[1] Copyright 2013 Carnegie Mellon University. This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense. NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT. This material has been approved for public release and unlimited distribution.. DM-0000829