Position Paper: Automotive CPS Workshop Oct. 2010

Fault-Tolerant Discrete Control Logic in Automotive Applications

Richard Hill
Department of Mechanical Engineering, University of Detroit Mercy
Stéphane Lafortune
Department of Electrical Engineering and Computer Science, University of Michigan

Motivation and Background

Modern automobiles include an extensive amount of discrete logic, implemented in software, that is needed to control and safeguard various aspects of a vehicle's operation: braking systems, steering systems, occupant safety systems, etc. As the complexity and coupling of these automotive control systems increase, traditional industrial practices have become insufficient to produce the necessary performance and reliability. Currently, most discrete logic design is performed heuristically in a decentralized manner. Problems that arise due to system interactions are generally identified during vehicle-level simulation, or worse, during tests on actual vehicles. At this point, debugging an underlying flaw is much more difficult and the resulting fix is generally more complicated and expensive to implement. It is even possible that a fault due to system interaction could fail to be identified during the development phase altogether. No amount of testing or simulation can guarantee that all possible permutations of software execution will be evaluated.

In recent years, considerable advances have taken place in computer science and in control engineering for the development of techniques for the *verification* and *synthesis* of given discrete-logic systems. In computer science, research on formal verification and model checking of software[1] has matured and has begun to allow increasingly complex software programs to be automatically verified with respect to a given specification. These techniques can be employed to verify the correctness of discrete control logic, but manual intervention is still needed to debug and modify the software when it fails the correctness test. In parallel, in control engineering, theory has been developed that not only verifies the correctness of discrete control logic, but is also able to automatically synthesize additional control logic which, when coupled with the original logic, leads to behavior that is provably correct with respect to a given specification. This body of work is often referred to as *Discrete-event Control Theory*, abbreviated as DCT hereafter. In the context of this type of discrete-event control, *correctness* means that the resulting controlled system is *safe* and *nonblocking*. Safety means that the controlled system does not execute any behavior not allowed by a given specification (e.g., avoidance of illegal states), and nonblockingness means that the system can always reach a "goal" state. For instance, a safety property could be: "do not deploy an airbag if the output of a weight sensor is within some range that indicates a child is in the seat." The property of nonblockingness is especially important when studying a set of interacting

[1] E. Clarke, O. Grumberg, and D. Peled, *Model Checking*, The MIT Press, 2001.

systems; each system may perform satisfactorily in isolation, but unexpected interactions may occur when the systems are coupled: the two classical problems are *deadlock* (i.e., no further progress is made) and *livelock* (i.e., the tasks at hand cannot be completed). As an example of how interaction could lead to deadlock, consider a central door lock system. This system interacts with many other systems and their interaction could lead a window to be stuck open with the system deadlocked in an unanticipated state with no control logic existing that would allow the window to reach the goal state of being closed.

DCT is a relatively new area of control engineering[2] that is finding its way into applications. It has been used for instance in automated manufacturing applications, and recently for failure avoidance in concurrent software applications.[3] *A promising research agenda is to build on this recent work on general-purpose concurrent software and use DCT to tackle the verification and synthesis of control software in automotive applications.*

Proposed Research Agenda

Given a formal model of the set of systems and of the specifications, DCT techniques can be employed to verify if the interacting systems behave according to the specifications and, if not, to synthesize algorithmically a "controller" (i.e., additional control logic), so that the "closed-loop" controlled system is *provably correct* with respect to the specifications. Solid theoretical results and associated algorithmic techniques are available in this regard. At Michigan, we have developed a software tool called DESUMA that implements many of these algorithms.

The two primary obstacles to application of this theory are the lack of existing models and possibly the complexity of the control logic. Like other theories in control engineering, DCT is model-based. The algorithmic techniques for controller synthesis in DCT require formal models of the set of interacting systems (aka the "plant" in control engineering terms) and of the specifications to be enforced (e.g., illegal states). In the automotive industry it is generally the case that models of the specifications do exist, just not in the form commonly employed by existing DCT techniques. Specifically, automotive specifications are given in the form of if-then-else statements, tables, and natural language descriptions, while most academic theory requires models in the form of automata (finite-state machines), Petri nets, or process algebras. Therefore, algorithms must be developed to convert those specification models used in the automotive industry into the modeling formalisms used in DCT. On the other hand, discrete models of the *plant* (i.e., the set of interacting systems) often do not exist at all. Such modeling is a challenging task; deep designer understanding of the whole system is necessary to develop the required discrete models for the constituent systems. Based on our experience on control of software for deadlock avoidance, we envisage an approach where system models are constructed from the existing, possibly unverified and incomplete, discrete control logic modules themselves. In other words, the "plant" would be the unverified and/or

[2] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd. Ed., Spinger, 2008.

[3] T. Kelly, Y. Wang, S. Lafortune, and S. Mahlke, "Eliminating Concurrency Bugs with Control Engineering," *IEEE Computer*, Cover feature, December 2009.

incomplete discrete control logic, and the goal of controller design would be to synthesize additional control logic that will guarantee that the complete set of interacting systems satisfies the given specification. A key issue here is the level of abstraction to be used in the construction of these models; this has to be attuned to the specification under consideration. The desired formal models must capture the relevant interactions of the systems so that the global safety and nonblocking properties are satisfied by control.

The second challenge that arises is the large amount of computation that may be required for the verification and synthesis of the control logic for a system as complex as a modern automobile. This complexity arises from the concurrent operation of so many interacting subsystems. For example, an automotive braking system could require mediation of commands from a range of different safety systems, like adaptive cruise control, stability control, and rollover prevention, as well as from the energy management system in the case of a system with regenerative braking. This complexity motivates the need for formal techniques, but also demonstrates the computational challenge. Techniques that mitigate this complexity by designing the control logic in a modular or hierarchical manner have been developed[4] and could be advanced for the purposes of application to automotive systems.

Bio Sketches

Richard Hill: ME Dept., University of Detroit Mercy; hillrc@udmercy.edu; 313-578-0428; fax: 313-578-0428. He received the B.S. degree in Mechanical Engineering, summa cum laude, from the University of Southern California in 1998, and the M.S. degree in Mechanical Engineering from the University of California, Berkeley in 2000. The subject of his master's research was braking control for autonomous vehicles as part of the larger Partners for Advanced Transit and Highways (PATH) program. In 2008 he received the Ph.D. degree in Mechanical Engineering and the M.S. degree in Applied Mathematics from the University of Michigan, Ann Arbor. In 2008 he joined the faculty of the Mechanical Engineering Department at the University of Detroit Mercy. Since joining the faculty he has spent two visiting stints at Ford Motor Company first working on diagnostics for hybrid electric vehicles and then to develop a course on the modeling and control of advanced electric vehicles.
Stéphane Lafortune: EECS Dept., University of Michigan; stephane@eecs.umich.edu; 734-763-0591; fax: 734-763-8041. He received the B. Eng degree from Ecole Polytechnique de Montréal in 1980, the M. Eng. degree from McGill University in 1982, and the Ph.D. degree from the University of California at Berkeley in 1986, all in electrical engineering. He joined the EECS Department at Michigan in 1986 as an Assistant Professor and was promoted to Professor in 1998. He was elected Fellow of the IEEE in 1999. He received the Presidential Young Investigator Award from the National Science Foundation in 1990 and the George S. Axelby Outstanding Paper Award from the Control Systems Society of the IEEE in 1994 (for a paper co-authored with S. L. Chung and F. Lin) and in 2001 (for a paper co-authored with G. Barrett).

[4]See, e.g., the recent results in: R. Hill, D. Tilbury, and S. Lafortune, "Modular Supervisory Control with Equivalence-Based Abstraction and Covering-Based Conflict Resolution." *Discrete Event Dynamic Systems: Theory and Applications*, Vol. 20, No. 1, March 2010, pp. 138-185.