Comments on "Adaptive Cruise Control: Hybrid, Distributed, and Now Formally Verified"

Theodore P. Pavlic, Paolo A.G. Sivilotti, Alan D. Weide, and Bruce W. Weide
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210
{pavlic.3, sivilotti.1, weide.3, weide.1}@osu.edu

Technical Report OSU-CISRC-7/11-TR22

July 2011

This material is based upon work supported by the National Science Foundation under Grant No. ECCS-0931669. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the views of the National Science Foundation.

This report is available electronically at:

ftp://ftp.cse.ohio-state.edu/pub/tech-report/2011/TR22.pdf

Background

A recent paper [1] that claims to have "formally verified" adaptive cruise control has rightly generated considerable interest, but also has led to overstatements by some commentators. For example, the authors' institution issued a press release [2] touting that the methods developed "Keep Bugs Out of Software for Self-Driving Cars". This slight exaggeration was amplified in an *IEEE Spectrum* blog [3] describing the result as being "Autonomous Car Software That's Provably Safe". The latter asserts, among other things: "In other words, the software itself **provably** cannot cause an accident." One of the reader comments in reply is: "total balderdash & propaganda these are NOT provably safe".

While we certainly did not write the blog comment quoted above, we agree with the sentiment that it is important not to oversell the goals or achievements of "formal verification". Our research involves formal verification of software. We are, therefore, quite aware of the long-term negative impact *on the field* of claiming more than is appropriate. To that end, this short note explains our reaction to the above developments by making two points:

- No software has been verified in [1].
- The hypothetical adaptive cruise control systems admissible under the conservative constraints imposed in [1] are not very realistic.

For brevity in this report, we assume the reader has read all three sources cited above, and understands the basic intended operation of adaptive cruise control as well as the standard simple physical model of moving vehicles.

Has Adaptive Cruise Control Software Been Verified?

Figure 1 shows how one formally reasons about a ***cyber-physical system (CPS)***, where some behaviors are governed by physical laws of nature (e.g., the physics of vehicles on highways) and others by control software (e.g., an adaptive cruise control system that seeks to avoid colliding with the vehicle ahead). In each of these two aspects on the left side of Figure 1, one abstracts away from certain details and develops a mathematical model of the behavior of interest on the right side, ignoring everything else on the grounds that it is either irrelevant or of minor import. The assumptions involved in modeling the physical behavior of the real system are generally made explicit (e.g., the assumption in [1] that highway lanes are straight) because these may vary from one treatment of a CPS to another. The result of this abstraction and modeling is that in the top box on the right we have a purely symbolic description of physical behavior that "abstracts away" many of the details.

For the control software portion of the real system, the situation is notably different. The mathematical model of behavior described in the bottom box on the right *is* the control software: the code itself is a precise symbolic description of its own behavior when interpreted under the semantics of the programming language. Questions about whether the real physical computer executes this software as it is supposed to, whether someone might interrupt the power supply of the computer, whether real sensors might provide noisy readings, etc., are all features of physical behavior and not of software behavior: they either need to be accounted for in the description in the top box on the right, or assumed not to matter for the context and analysis at hand.

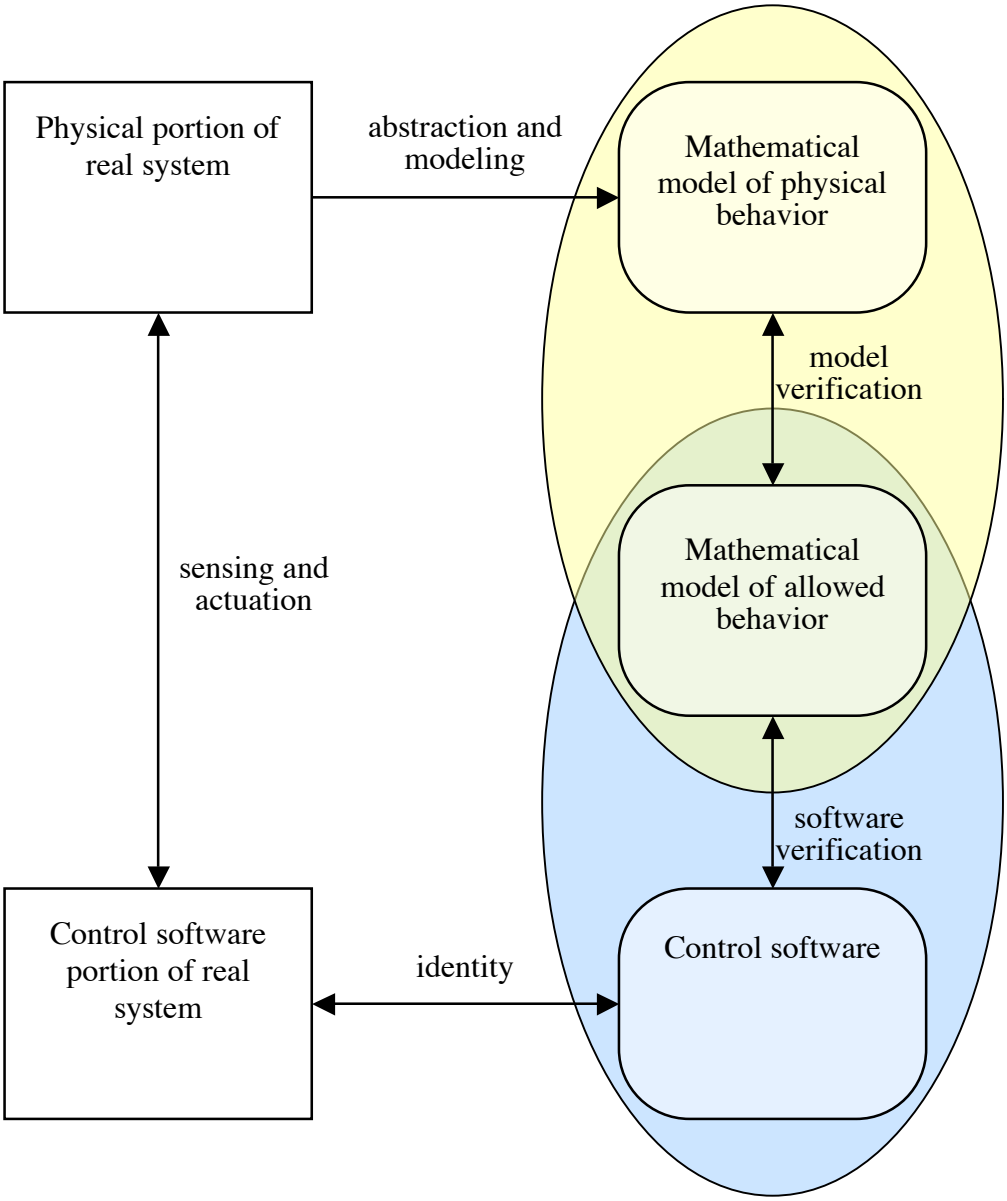

Figure 1: Modeling and Reasoning About Cyber-Physical Systems

This leaves the middle box on the right of Figure 1. A critical part of how cyber-physical systems are analyzed, it serves as an intermediary between the two mathematical models described above. Its first role relates to the top box on the right. It summarizes—using the same kinds of terms and concepts as appear in the description in the top box—the envelope of allowable behaviors of the control software in order that the mathematical model of physical behavior should satisfy some desired properties (e.g., the follower never crashes into the leader). In its second role, the middle box serves as or induces a ***specification*** of permissible behavior for the control software (e.g., the actual code that makes periodic decisions about how much the follower should accelerate or brake).

In [1], the authors focus on the top two boxes on the right side of Figure 1, i.e., the light-yellow shaded oval. This is where their formal verification tool, KeYmaera, shines. They carry out ***model verification***: a proof that *if* the physical system of the vehicles behaves according to the model described in the top box and *if* the control software for adaptive cruise control behaves within the envelope of allowable behaviors described in the middle box, *then* the desired properties will hold (e.g., the follower will not crash into the leader). They do not carry out ***software verification***: a proof that a particular piece of control software in an adaptive cruise control system operates within the envelope of allowable behaviors described in the middle box. As we are currently in the process of working on the latter question ourselves to include the light-blue shaded oval as well, we find the especially clear and precise specification of allowable software behavior developed in [1] to be a valuable contribution. But no such software has been verified in [1], so we find commentators' claims to that effect to be a serious overstatement.

What About More Realistic Adaptive Cruise Control?

The analysis in [1] makes a braking-distance argument about safety: if the leader's braking distance plus the gap between the two vehicles exceeds the distance traveled by the follower during the reaction time of the follower plus the braking distance of the follower at that point, then there will be no collision. In general, this argument is invalid. Safety in the physical world requires the follower to remain behind the leader *at all times* during which the above braking scenario is playing out, not just at the end-point when both are stopped. The braking-distance

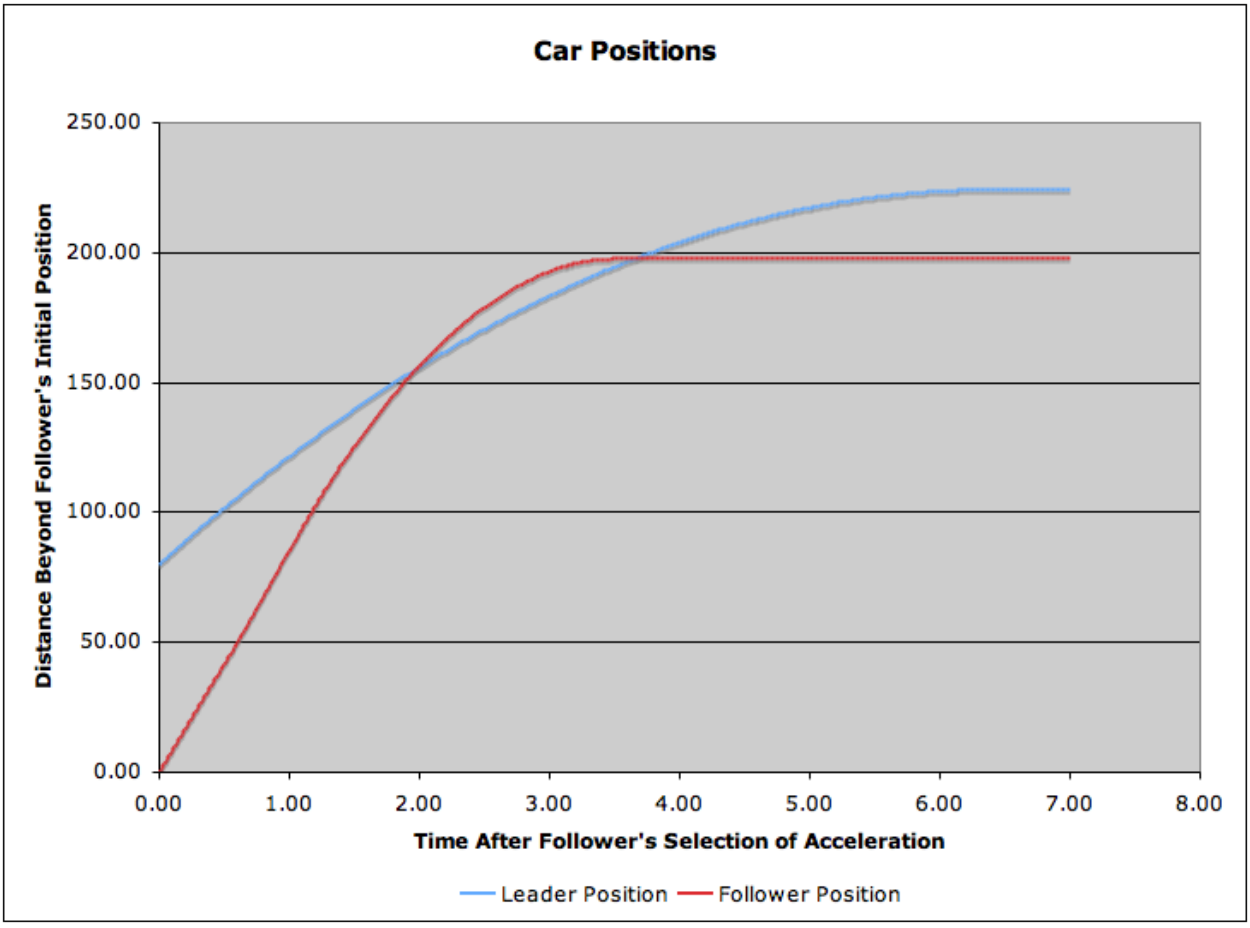

Figure 2: How Realistic Adaptive Cruise Control Can Be Tricky

argument is too weak to establish the property that the gap between vehicles remains positive *throughout the interval* in question—except under certain conditions. Figure 2 illustrates what might happen given arbitrary braking of two vehicles: the leader brakes softly, the follower brakes hard, and though the braking-distance bound is satisfied there is a collision at about $t = 2$ seconds while both cars are still moving.

The analysis in [1] relies on maintaining the gap between the two vehicles large enough to allow the follower to avoid a collision even when the follower brakes with the minimum hard-braking force of any car on the highway (and hence brakes no harder than the leader, which is sufficient but not necessary to make the braking-distance argument valid). With the parameter values that lead to the situation in Figure 2, the safety condition in [1] requires the follower to be at least 269 ft behind the leader at time $t = 0$, when in fact the actual minimum following distance to avoid a collision in this situation is only about 1/3 of that: 91 ft. In other words, if adaptive cruise control software had to operate within the overly conservative envelope of behaviors allowed and (model-)verified in [1], then vehicles would be vastly more spread out on the highway than actually would be required to ensure safety.

We have developed (but have not yet published) our own model for the physical behavior in the top box on the right in Figure 1 that is similar to that presented in [1], along with a description of allowable software behavior for adaptive cruise control software in the middle box that is quite different. The latter is more complex because we seek a superset of desired properties that are representative of a realistic adaptive cruise control system. For example, still subject to the constraint of safety of course, suppose one seeks to optimize fuel efficiency. One might wish to incorporate into the control software an algorithm that tries to reduce fuel consumption at highway speeds by keeping vehicles as close together as possible, as in platoons or convoys. This leads to a more difficult model-verification task (which KeYmaera might or might not still be able to complete) as well as a more intricate control algorithm. The added functionality and complexity leaves model verification as important as ever, but it increases the importance of software verification: the resulting control software is no longer simple straight-line code and it would be easy to get it wrong.

In short, the formal verification of adaptive cruise control systems is not a solved problem—despite what one might gather from some observers' initial reactions to [1].

References

[1] Loos, S.M., Platzer, A., and Nistor, L., "Adaptive Cruise Control: Hybrid, Distributed, and Now Formally Verified", *Proceedings 17th International Symposium on Formal Methods, LNCS 6664*, Springer, 2011, 42-56.

[2] Spice, B., "Carnegie Mellon Methods Keep Bugs Out Of Software For Self-Driving Cars", http://www.cmu.edu/news/archive/2011/June/june21_selfdrivingcars.shtml viewed 1 July 2011.

[3] Ackerman, E., "CMU Develops Autonomous Car Software That's Provably Safe", http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/cmu-develops-autonomous-car-software-that-is-provably-safe, viewed 1 July 2011.