Certified Control for Self-Driving Cars

Abstract

Certified control is a new architectural pattern for achieving high assurance of safety in autonomous cars. As with a traditional safety controller or interlock, a separate component oversees safety and intervenes to prevent safety violations. This component (along with sensors and actuators) comprises a trusted base that can ensure safety even if the main controller fails. But in certified control, the interlock does not use the sensors directly to determine when to intervene. Instead, the main controller is given the responsibility of presenting the interlock with a certificate that provides evidence that the proposed next action is safe. The interlock checks this certificate, and intervenes only if the check fails. Because generating such a certificate is usually much harder than checking one, the interlock can be smaller and simpler than the main controller, and thus assuring its correctness is more feasible.

Introduction

The growing complexity of software in autonomous cars makes it harder to ensure its reliability. Even if aggregate measures of safety improve, the risk of unexpected catastrophic failures will remain, and is exacerbated by the threat of malicious attacks. Ex post facto methods for obtaining assurance are unlikely to be feasible in the near term: statistical testing would require billions of miles traveled [7], and formal verification is not currently feasible for systems of this magnitude, especially those including machine learning components.

An effective (and traditional) solution for systems that control physical plant is to augment the main controller with a safety controller or *interlock* that monitors the world around and intervenes when an accident is imminent. By maintaining a safety envelope, the interlock can ensure that whenever it decides to act there will still be time and space to prevent disaster. Since the interlock has fewer and simpler responsibilities than the main controller, it can be much smaller, and thus more feasible to verify. The design of the system as a whole guarantees that the interlock is isolated from the main controller and has prioritized access to actuators, so that (along with the sensors and actuators) it comprises a trusted base. Failures and compromises of components outside the trusted base can be ignored in making an assurance case.

At least that's the theory. In practice, applying this idea to autonomous cars seems to be impractical. The key problem is that in order for the interlock to have sufficiently few false positives and negatives—that is, intervening when it should not and failing to when it should—it must be equipped with tools for perception and situational awareness that are no less sophisticated than those of the main controller. It must know, for example, how the road is divided into lanes, and it must be able to distinguish a pedestrian running across the street from a plastic bag blowing in the wind.

We propose a new approach (Figure 1) that provides the isolation and independence of a traditional interlock, and keeps the interlock small and simple, but at the same time accommodates the need for rich situational awareness. The interlock does *not* use the sensors directly to determine whether the car is in a safe situation. Instead, it acts as a checker for the actions taken by the main controller. In each control cycle, the main controller analyzes the situation and determines a suitable action (which may include continuing on its current course). It presents this action to the interlock, along with evidence that the action is safe. The interlock checks the evidence using a predefined safety argument, and if it is credible, passes the action on to the low-level controller. If not, the interlock replaces the action by a safety mitigation, such as braking.

The trusted base in this design is not limited solely to the interlock. As with a traditional interlock, it must include the sensors and actuators; in our design we also include the low-level controller that translates commanded actions into actuations of the brakes, throttle and steering. Nevertheless, the software components of the main controller—which typically comprise a planner, sensor fusion, and perceptual analyses and thus the bulk of the software complexity—are excluded from the trusted base.

A Certified Control Scenario

To illustrate the approach, consider a simple scenario of an autonomous car driving on a straight highway. Suppose the main controller determines, using LIDAR, that the only obstacle ahead of it is a car traveling at a distance of 100m away. The ego car is driving towards this car at 10m/s, and can decelerate at 5m/s^2. This

implies a stopping distance of 10m, which would rise to 40m if the speed were doubled. The controller therefore reasons that increasing the ego car's speed to 20m/s would maintain safe separation (even under the unlikely possibility that the lead car comes to a stop instantaneously). It therefore seeks to convince the interlock that this speed increase is justified.

To do so, it presents the interlock with evidence of the ego car's current speed, and evidence that there is no obstacle within 100m. For the former, it passes on a timestamped speed reading from a specialized speed computation unit (not shown in the figure) that relies only on timing wheel rotations, and is signed with that unit's private key. For the latter, it selects some small number of LIDAR points that demonstrate that along certain trajectories there is no obstacle within 100m. These points are selected from the contour of the lead car that the perception algorithm has extracted from LIDAR data, and are also timestamped and signed (in this case by the LIDAR unit).

Together, these pieces of evidence comprise a certificate passed to the interlock that contains: (a) the current speed; (b) some LIDAR points; (c) the claimed distance between the ego car and the lead car; (d) the proposed action (increasing speed to 20m/s). The interlock checks that the speed (a) and distance readings (b) are sound—that is, their timestamps are current and their signatures can be verified using the respective public keys; that the claimed distance (c) is no greater than the distances in the distance readings; and that the proposed action (d) is safe given these readings and assumptions about the ego car's maximum deceleration. If so, the interlock passes on the proposed action, and if not, it replaces the action by a braking action to bring the car to a stop.

Discussion

We now explain some of the ramifications and subtleties of this approach.

Tolerance for errors. You might wonder why the interlock's rejection of the certificate should lead to immediate braking. Perhaps the interlock could rely on previous (successfully verified) certificates that establish the distance between the ego and lead cars. Some latitude may be desirable here, but in general, a failed verification means that the main controller is broken and the car is driving blind.

Safety of intervention. The braking intervention itself may be dangerous (for example, if there is truck bearing down on the lead car from behind). For this reason, it may be desirable for the interlock to have some direct access to sensors solely to determine whether the invention itself is safe (shown as a dashed arrow in the figure).

Non-certifiable perception. LIDAR-based perceptions seems straightforwardly amenable to generating evidence. However complex the algorithm for analyzing the LIDAR point cloud, computing contours, etc, it should be possible to find a subset of readings to substantiate the claim that the detected obstacle is a certain distance away. But such a strategy would likely not work for camera-based perception, since there is no subset of the pixels that would convincingly corroborate the inferred situation. This does not mean that such perception mechanisms could not be used, just that they would have to be augmented with mechanisms that can provide evidence.

Lane following. A convincing safety case for the proposed action should also include evidence of road and lane layout; in the case of our scenario that the lane in which the ego car is traveling extends forward the requisite distance. To do so, the main controller might pass onto the interlock some map segments and a GPS reading signed by respective specialized units.

Misleading certificates. Suppose the obstacle is a trailer carrying a pile of pipes, and the certificate contains LIDAR points that correspond to trajectories that pass through the pipes, thus giving the misleading impression that the obstacle is further away than it really is. This problem reflects the limitations of perception and thus seems unavoidable, and could be countered by raising the bar for evidence (eg, by requiring more LIDAR points in the certificate). Certified control does admittedly exacerbate this problem in one respect: one could imagine (perhaps implausibly) a malicious compromise of the main controller that seeks opportunities to create such misleading certificates, although the effectiveness of such an attack seems limited.

Incomplete model. The credibility of the certificate depends on an underlying model of the world (shared by the main controller and interlock). This model must of necessity be incomplete, so the interlock will only be able to guard against accidents that are implicitly covered by the model. For example, the certificate might not include evidence that there are no boulders rolling down a mountain adjacent to the road; or that the weather conditions

have not made the road surface more slippery than usual; or that no portion of the road has collapsed ahead of the car due to an earthquake.

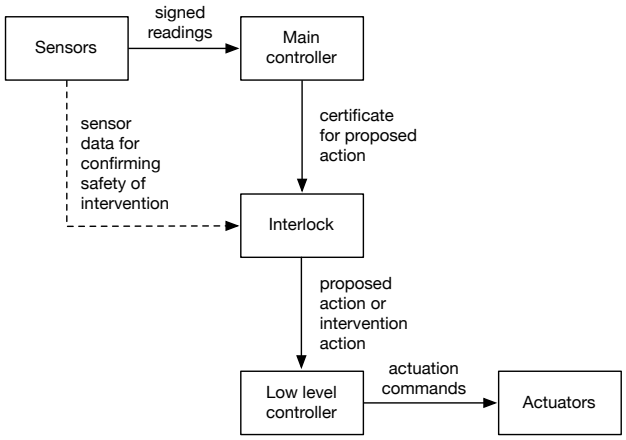

Figure 1: Certified control architecture

Simulation

We have prototyped this proposal in two forms: in a 1/10-scale mini racecar equipped with Velodyne LIDAR that runs ROS, and in a simulation framework in processing.js, both running the same core code. The action is encoded as a point in configuration space as the immediate goal. At the workshop (if invited) we will demonstrate the simulation and show video of the racecar operating.

Related Work

Certified control exploits the gap in complexity between generating a solution to a problem and checking it, an idea that has been used in many contexts before, including program checking [2], proof carrying code [9] and credible compilation [10]. The idea of the monitoring interlock (or safety controller), which unlike our interlock relies on its own interpretation of perceptual data, is well known, and has been described before, both in general [1] and in robotics [4], and continues to be a focus of new work, for example in synthesis [13]. The risks of unpredictable behavior in components (such as an autonomous car's perception) based on machine learning are widely recognized, and have led to various proposals, including applying formal verification [12] and using run-time assertions as sanity checks [8]. Like our scheme, RSS [11] uses stopping distances as a check on the main controller, but does not isolate the checker. Safety or dependability cases [6] that include reasoning about both software, hardware and environment [5] have been checked at runtime [3] but to our knowledge using the idea of certificates to reduce the trusted base in this context is new.

References

[1] Stanley Bak, Deepti K. Chivukula, Olugbemiga Adekunle, Mu Sun, Marco Caccamo and Lui Sha. The System-Level Simplex Architecture for Improved Real-Time Embedded System Safety. *15th IEEE Real-Time and Embedded Technology and Applications Symposium*, San Francisco, CA, 2009, pp. 99-107.

[2] Manuel Blum and Sampath Kanna. Designing programs that check their work. *21st Annual ACM symposium on Theory of Computing (STOC '89)*, D. S. Johnson (Ed.). ACM, New York, NY, USA, 1989, pp. 86-97.

[3] Ewen Denney, Ganesh Pai and Ibrahim Habli. Dynamic Safety Cases for Through-Life Safety Assurance. *International Conference on Software Engineering, Florence*, 2015, pp. 587-590.

[4] Ankush Desai, Indranil Saha, Jianqiao Yang, Shaz Qadeer and Sanjit A. Seshia. DRONA: A Framework for Safe Distributed Mobile Robotics. *8th International Conference on Cyber-Physical Systems* (ICCPS), Pittsburgh, PA, 2017, pp. 239-248.

[5] Carl A. Gunter, Elsa L. Gunter, Michael Jackson, and Pamela Zave. A Reference Model for Requirements and Specifications. *IEEE Software*. 17, 3 (May 2000), 37-43.

[6] Daniel Jackson, Martyn Thomas, and Lynette I. Millett, Editors. Committee on Certifiably Dependable Software Systems. *Software for dependable systems: Sufficient evidence?* National Academies Press, 2007.

[7] Nidhi Kalra and Susan M. Paddock. *Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?* RAND Corporation, 2016. https://www.rand.org/pubs/research_reports/RR1478.html.

[8] Daniel Kang, Deepti Raghavan, Peter Bailis, and Matei Zaharia. Model assertions for debugging machine learning. *NeurIPS MLSys Workshop*, 2018.

[9] George C. Necula. Proof-carrying code. *24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. ACM, New York, NY, USA, 1997, 106-119.

[10] Martin Rinard. Credible compilation. 2003.

[11] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. arXiv preprint arXiv:1708.06374 (2017)

[12] Cumhur Erkan Tuncali, James Kapinski, Hisahiro Ito, Jyotirmoy V. Deshmukh. Reasoning about safety of learning-enabled components in autonomous cyber-physical systems. *DAC 2018*: 30:1-30:6.

[13] He Zhu, Zikang Xiong, Stephen Magill and Suresh Jagannathan. An Inductive Synthesis Framework for Verifiable Reinforcement Learning. To appear, *40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2019.