WHITEPAPER

Formal Scenario-Based Testing of Autonomous Vehicles: From Simulation to the Real World

Daniel J. Fremont,[1:4] Edward Kim,[1] Yash Vardhan Pant,[1] Sanjit A. Seshia,[1]
Atul Acharya,[2] Xantha Bruso,[2] Paul Wells,[2] Steve Lemke,[3] Qiang Lu,[3] Shalin Mehta[3]

ABSTRACT

We present a new approach to automated scenario-based testing of the safety of autonomous vehicles, especially those using advanced artificial intelligence-based components, spanning both simulation-based evaluation as well as testing in the real world. Our approach is based on formal methods, combining formal specification of scenarios and safety properties, algorithmic test case generation using formal simulation, test case selection for track testing, executing test cases on the track, and analyzing the resulting data. Experiments with a real autonomous vehicle at an industrial test track support our hypotheses that (i) formal simulation can be effective at identifying test cases to run on the track, and (ii) the gap between testing in simulated and real worlds can be systematically evaluated and bridged.

CONTEXT

A defining characteristic of the growth in autonomous vehicles (AVs) and automated driving systems (ADS) is the expanding use of machine learning (ML) and other artificial intelligence (AI) based components in them. ML components, such as deep neural networks (DNNs), have proved to be fairly effective at perceptual tasks, such as object detection, classification, and image segmentation, as well as for prediction of agent behaviors. However, it is known that ML components can be easily fooled by so-called adversarial examples, and there have been well-documented failures of AVs in the real world for which the evidence points to a failure (in part) of ML-based perception. Therefore, there is a pressing need for better techniques for the testing and verification of ML/AI-based ADS and AVs.

Simulation is regarded as an important tool in the design and testing of AVs with ML components. Several photorealistic 3D simulators are now available for AVs, providing designers with the ability to simulate "billions of miles" so as to test their AV components, cover corner-case scenarios that are hard to test in the real world, and diagnose issues that occur in real-world testing, such as disengagements. However, some key questions remain. How well does simulation match the real world? What is the value of simulation versus testing on a public road?

[1] University of California, Berkeley
[2] American Automobile Association (AAA) of Northern California, Nevada & Utah (NCNU)
[3] LG Electronics America R&D Lab
[4] University of California, Santa Cruz

This whitepaper summarizes a manuscript that has been accepted for publication in the proceedings of the 2020 IEEE Intelligent Transportation Systems Conference (ITSC): ieee-itsc2020.org

A pre-publication version is available on ArXiv: arxiv.org/abs/2003.07739

This work is supported in part by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1752814, NSF grants CNS-1545126 (VeHICaL) and CNS-1646208, the DARPA Assured Autonomy program, the Collaborative Sciences Center for Road Safety (CSCRS) under Grant No. 69A3551747113, Berkeley Deep Drive, and Toyota through the iCyPhy center.

A complement to simulation and public road testing is closed course or track testing. This form of testing involves driving the AV on roads at a test facility with a reasonable degree of control over the other agents around the AV, including, for example, pedestrian dummies and inflatable cars to use for crash testing. Track testing allows an AV to be run using its real hardware and software systems in environments that can be designed to mimic certain challenging driving conditions. With controllable agents, track testing scenarios can also be repeated in ways public road testing cannot. However, track testing can be very expensive, labor-intensive, and time-consuming to set up. Given these challenges, which tests should be run? For testing AVs with complex ML-based components, it is crucial to be able to run the tests that will prove most effective at increasing assurance in the safety of the AV and ADS.

APPROACH

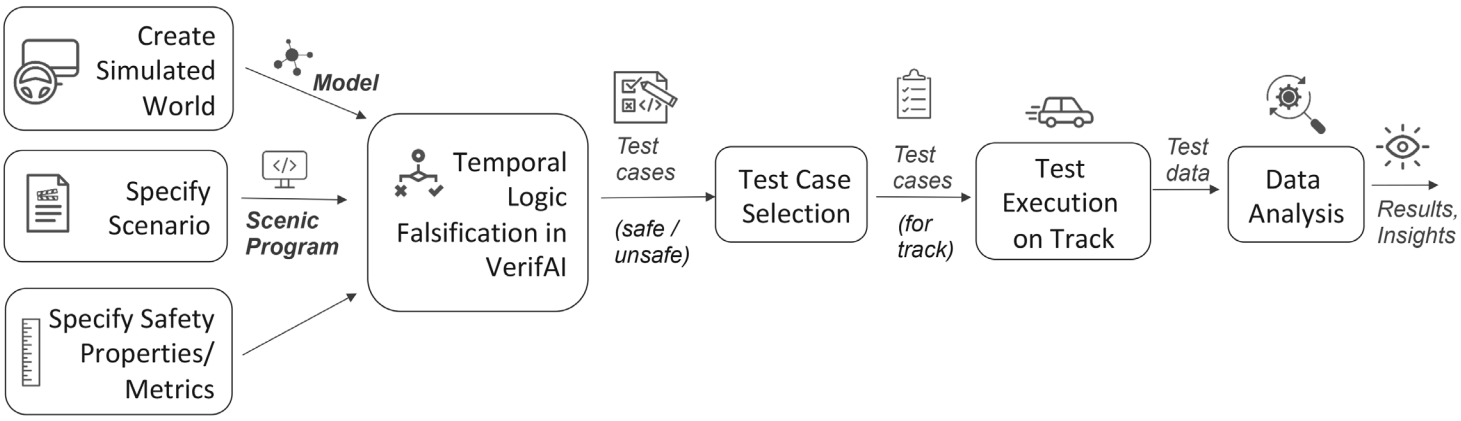
This paper presents a synopsis of our work on addressing these problems. In particular, we investigated the following two questions:

1. Can formal simulation aid in designing effective road tests for AVs? By formal simulation we mean simulation-based testing that is guided by the use of formal models of test scenarios and formal specification of safety properties and metrics. More specifically, do unsafe (safe) runs in simulation produce unsafe (safe) runs on the track? How should one select tests from simulation to run on the track?
2. How well can simulation match track testing of AVs? We aim to quantitatively and qualitatively compare simulation and track testing data for a test scenario that has been formally specified and implemented in both simulation and track testing.

Our approach is rooted in formal methods, a field centered on the use of mathematical models of systems and their requirements backed by computational techniques for their design and verification. In particular, we use a formal probabilistic programming language, Scenic,[1] to specify a test scenario, encapsulating key behaviors and parameters of the AV and its environment. Additionally, we use formal specification languages, such as Metric Temporal Logic,[2] to specify safety properties for AVs. We combine formally-specified scenario descriptions and requirements with algorithms for simulation-based verification of AVs, also known as falsification, implemented in an open-source toolkit called VerifAI.[3] These methods, when combined with an advanced photorealistic full-stack simulator for AVs, the LGSVL Simulator,[4] allow us to identify safe and unsafe behaviors of the AV in simulation. We seek to answer the above questions by incorporating into the simulator a "digital twin" of an industrial-scale test track, the GoMentum Station test facility in Concord, California.[5] We have developed and deployed a simulation-to-test-track flow where formal simulation is used to identify test cases to execute on the track, and these test cases are systematically mapped onto hardware that is used to control agents on the track in the AV's environment. We present the results of executing this flow in a scenario involving a pedestrian crossing in front of an AV, providing evidence for the effectiveness of formal simulation for identifying track tests, as well as a quantitative mechanism for comparing simulation results with those obtained on the track.

To the best of our knowledge, this paper is the first to apply a formal methods-based approach to evaluating the safety of machine learning-based autonomous vehicles employing formal specification of scenarios and safety properties, formal simulation-based test generation and selection for track testing, as well as evaluation of the methodology in both simulation and the real world, including systematically measuring the gap between simulation and track testing.

Figure 1: Formal Scenario-Based Testing Methodology



METHODOLOGY

Our overall methodology (see Figure 1) involved seven steps:

1. Create Simulation Model: We create a photorealistic simulation environment including dynamical models for a range of agents implementable on the test track (the LGSVL Simulator).[4]
2. Formalize Test Scenario: We then formalize the test scenario(s) to be executed on the track. We take a formal methods approach, specifying test scenarios in the Scenic probabilistic programming language.[1]
3. Formalize Safety Property/Metric: We specify one or more safety properties that capture the conditions under which the AV is deemed to be operating safely. In formal methods, safety properties over traces are usually specified in a logical notation such as temporal logics.
4. Identify Safe and Unsafe Test Cases: The simulation model, test scenario, and safety properties are then fed into the VerifAI tool[3] to perform falsification. The resulting test cases are fed to the next step.
5. Select Test Cases for Track Testing: VerifAI provides several techniques to automatically analyze the safe and error tables and extract patterns. We use it and other methods to identify different modes of behavior, and select representative test cases to execute on the track.
6. Implement Selected Test Cases on Track: We then plan track testing. For this, dynamic agents (e.g., pedestrian dummies) must be controllable using parameters, such as starting location, velocities, etc. specified in the Scenic program and synthesized into the test case.
7. Record Results and Perform Data Analysis: Data recorded during track testing including videos, all sensor data and log data from the AV software stack, as well as data from the test track hardware, are analyzed to evaluate the effectiveness of test case selection through formal simulation, the correspondence between simulation traces and the traces from track experiments, and potential reasons for unsafe or interesting behavior of the AV.

KEY FINDINGS

In this project, we demonstrated that a formal simulation approach can be effective at identifying relevant tests for track testing with a real AV. We also compared time-series data recorded in simulation and on the track for the same synthesized test cases, both quantitatively and qualitatively. Our results indicate that:

1. Our formal simulation-based approach is effective at synthesizing test cases that transfer well to the track: 62.5% of unsafe test cases indeed resulted in unsafe behavior on the track, including a collision, while 93.3% of safe test cases resulted in safe behavior on the track (and no collisions). Our results also shed light on potential causes for AV failure in perception, prediction, and planning.

2. While AV and pedestrian trajectories obtained in simulation and real-world testing for the same test were qualitatively similar, we also noted significant differences as quantified using time-series metrics,[6] and with metrics such as minimum distance between the AV and pedestrian.[7] Variations exist even amongst simulations of the same test case due to non-deterministic behavior of the AV stack, although these are smaller.

Regarding whether formal simulation was effective at designing track tests that revealed unsafe behaviors of the AV, our data indicates that this was in fact the case: out of 8 runs of the two failure tests that are identified in simulation to violate the safe distance between the AV and the pedestrian dummy, 5 violated this safe distance in reality, including one actual collision.

More noticeably, in 93.3% of all (robustly or marginally) safe runs, the AV satisfied the safe distance in reality. As expected, a violation case occurred in a marginally safe test, but none in safe tests. While the number of runs is small due to the limited time and resource budget, they clearly demonstrate how simulation can be efficiently used to identify real-world failures.

On the other hand, our data also shows that the results of a track test can deviate significantly from results in simulation. The track test results also have significant variability. When we look at these discrepancies in more detail, our data suggests that while our simulation environment was faithful enough to reality to produce qualitatively similar runs, it is not yet accurate enough to enable deriving guarantees on the real system purely through simulation.

Interestingly, although the simulations are clustered together much more closely than to the original track test, they still show substantial variation. As the LGSVL Simulator is deterministic, this demonstrates that either the asynchronous interface to Apollo or nondeterminism within Apollo itself can produce significantly different behavior on identical test cases. Our methodology could likely be improved by taking this nondeterminism into account: tests with lower variance are more likely to be reproducible on the track, yielding more robust transfer of safe/unsafe AV behaviors from simulation to reality.

There are several directions for future work, including evaluating our methodology on more complex higher dimensional scenarios, performing more detailed automated analysis of failures in perception, planning, or prediction, bridging the sim-to-real gap with further improvements to simulation technology, and the development and use of more sophisticated track test equipment that can better match simulation.

LINK TO RELEVANT WEBSITES:

Scenic: A Language for Scenario Specification and Scene Generation, github.com/BerkeleyLearnVerify/Scenic

LGSVL Simulator, lgsvlsimulator.com

GoMentum Station, gomentumstation.net

REFERENCES:

[1] D. J. Fremont et al., "Scenic: A language for scenario specification and scene generation," in Programming Language Design and Implementation (PLDI), 2019, pp. 63-78.

[2] R. Koymans, "Specifying real-time properties with metric temporal logic," Real-Time Systems, vol. 2, no. 4, pp. 255.299, 1990.

[3] T. Dreossi et al., "VerifAI: A toolkit for the formal design and analysis of artificial intelligence-based systems," in Computer Aided Verification (CAV), 2019, pp. 432-442.

[4] Advanced Platform Team, LG Electronics America R&D Lab, "LGSVL Simulator," lgsvlsimulator.com.

[5] GoMentum Station, gomentumstation.net.

[6] T. Giorgino, "Computing and visualizing dynamic time warping alignments in R: The dtw package," Journal of Statistical Software, vol. 31, no. 7, pp. 1-24, 2009.

[7] J. V. Deshmukh et al., "Quantifying conformance using the Skorokhod metric," Formal Methods in System Design, vol. 50, no. 2-3, pp. 168-206, 2017.