A Systematic Framework to Identify Violations of Scenario-dependent Driving Rules in Autonomous Vehicle Software

Qingzhao Zhang
University of Michigan
qzzhang@umich.edu

David Ke Hong
University of Michigan
kehong@umich.edu

Ze Zhang
University of Michigan
zezhang@umich.edu

Qi Alfred Chen
University of California Irvine
alfchen@uci.edu

Scott Mahlke
University of Michigan
mahlke@umich.edu

Z. Morley Mao
University of Michigan
zmao@umich.edu

ABSTRACT

Safety compliance is paramount to the safe deployment of autonomous vehicle (AV) technologies in real-world transportation systems. As AVs will share road infrastructures with human drivers and pedestrians, it is an important requirement for AVs to obey standard driving rules. Existing AV software testing methods, including simulation and road testing, only check fundamental safety rules such as collision avoidance and safety distance. Scenario-dependent driving rules, including crosswalk and intersection rules, are more complicated because the expected driving behavior heavily depends on the surrounding circumstances. However, a testing framework is missing for checking scenario-dependent driving rules on various AV software.

In this paper, we design and implement a systematic framework *AVChecker* for identifying violations of scenario-dependent driving rules in AV software using formal methods. *AVChecker* represents both the code logic of AV software and driving rules in proposed formal specifications and leverages satisfiability modulo theory (SMT) solvers to identify driving rule violations. To improve the automation of systematic rule-based checking, *AVChecker* provides a powerful user interface for writing driving rule specifications and applies static code analysis to extract rule-related code logic from the AV software codebase. Evaluations on two open-source AV software platforms, *Baidu Apollo* and *Autoware*, uncover 19 true violations out of 28 real-world driving rules covering crosswalks, traffic lights, stop signs, and intersections. Seven of the violations can lead to severe risks of a collision with pedestrians or blocking traffic.

CCS CONCEPTS

• **Software and its engineering** → **Automated static analysis**; • **Computer systems organization** → Embedded and cyber-physical systems.

KEYWORDS

Autonomous vehicle; Software system; Formal methods

ACM Reference Format:
Qingzhao Zhang, David Ke Hong, Ze Zhang, Qi Alfred Chen, Scott Mahlke, and Z. Morley Mao. 2021. A Systematic Framework to Identify Violations of Scenario-dependent Driving Rules in Autonomous Vehicle Software. In *Abstract Proceedings of the 2021 ACM SIGMETRICS / International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '21 Abstracts), June 14–18, 2021, Virtual Event, China.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3410220.3460101

SIGMETRICS '21 Abstracts, June 14–18, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8072-0/21/06.
https://doi.org/10.1145/3410220.3460101

1 INTRODUCTION

Emerging autonomous vehicles (AVs) hold great promise in transforming today's transportation systems and mobility services while driving safety is the most important design requirement before their real-world deployment. The AV is a complicated cyber-physical system where a central AV software interacts with digital sensors and physical devices to automatically drive the vehicle. Unanimously agreed by AV software vendors and government authorities, the AV developers should validate the AV software's compliance with essential safety standards (e.g., traffic laws, voluntary safety standards from NTHSA [1], Responsibility-Sensitive Safety [9]).

Existing AV software safety testing approaches (e.g., simulation, track, or road testing) [2, 4, 6] mainly use the black-box dynamic testing. They collect driving traces from the real-world test drives or simulations and then verify their safety properties. However, these methods only target a limited set of driving rules related to basic safety requirements such as collision avoidance and safe distance on urban roads or highways. Such fundamental rules should be hold all time during the drive and only consider relation among vehicles. More complicated driving rules including on-road crosswalks, four-way stop signs, and intersections with traffic lights are ignored by general AV testing techniques. We name the above rules **scenario-dependent driving rules** because the expected behavior of the AV heavily depends on the vehicle's location as well as surrounding circumstance. To check these complicated rules, road/track testing or simulation requires test cases with high quantity and diversity to enumerate possible map layouts and different behaviors of all pedestrians and vehicles, which is difficult due to regulation barriers, high cost/risk of real-world testing, and the complexity of the driving scenarios even in simulators. In addition,

a general framework is still missing for testing driving rule compliance on AV software. Existing AV testing methods are mostly designed for one specific AV software or one specific driving rule. A general framework is in need to support various AV software and a wide range of driving rules simultaneously.

To address this limitation, we propose *AVChecker*, a framework for AV developers to systematically find violations of complex scenario-dependent driving rules in AV software using formal methods. Instead of enumerating test cases in dynamic testing, *AVChecker* proposes a formal representation to model complex driving scenarios considering both the map layouts and behaviors of moving objects. Then we can apply formal verification techniques to identify driving rule violations by analyzing the formal representations. To start with, *AVChecker* takes driving rule specifications and source code of AV motion planning (i.e., the module implementing driving rules) as inputs. The driving rule specifications are formatted in the proposed formal representation which is encoded in Satisfiability Modulo Theory (SMT), and provided by issuers of traffic rules. Meanwhile, a pipeline of static code analysis extracts AV's driving behaviors from the AV source code with the assistance of code annotations provided by AV developers. The driving behaviors are encoded into a finite state machine based specification, called behavior specification. Next, we apply formal methods (i.e., SMT theorem proving) to reveal the violation between the driving rule specification and AV's behavior specification. The violation means that at one specific moment, the AV's driving behavior is different from the expected behavior defined in the driving rule. As long as an violation exists between the two specifications, the SMT solver can generate a counterexample which helps in reproducing the identified flaw. *AVChecker* will post-process the generated violation case to construct a test case of simulation, which can help to rule out false positives or debug the software.

To identify rule violations through formal methods, our work addresses two key challenges. First, performing a direct comparison between driving rules expressed in natural language and AV software code is impractical because of the semantic gap. To bridge this gap, we propose a domain-specific formal representation modeling traffic scenes. Both rule specifications and AV's behavior specifications are based on the same formal representation so that they can be analyzed in the same domain. In addition, *AVChecker* provides user interfaces with various levels of abstractions to minimize manual effort for generating specifications. Second, characterizing continuous driving scenarios requires modeling real-world physical dynamics in continuous time and space domains, which is challenging to realize using the theory of SMT because of the complexity. Thus, the model of driving scenarios should abstract the real world as much as possible but maintain a necessary expressivity for rule compliance checking. *AVChecker* abstracts the continuous driving behaviors by splitting the time sequence into moments and checks AV's driving decisions on each moment. To achieve the abstraction, *AVChecker* constructs one moment of the driving scenario using SMT symbolic variables, called symbolic **traffic snapshot**. *AVChecker* detects violations when AV's behaviors break the driving rule at any possible traffic snapshot in this scenario without the need of considering over-complicated details of physical dynamics during a time sequence, making our SMT-based approach scalable on complicated driving scenarios.

We prototype *AVChecker* using LLVM [8] and *Z3* SMT solver [7], and evaluate it on two open-source AV software platforms, *Baidu Apollo* [3] and *Autoware* [5] which have 108K LOC and 42K LOC in their planning modules respectively. Our prototype is able to detect 19 true violations with driving rules for crosswalks, traffic lights, stop signs, and intersection scenes on both platforms. The simulation-based validation further confirms that the violations are all true positives and 7 of them may lead to severe safety consequences, including the risk of hitting a sprinting pedestrian on a crosswalk or blocking traffic at intersections. The violations are caused by the incomplete implementation, ignorance of corner cases, or bugs. The specification API of *AVChecker* is demonstrated to be effective for reducing AV developer's specification efforts, requiring less than 10 lines of code for specifying different complex scenes for the evaluated driving rules and reducing manual specification efforts by 15x. Moreover, the snapshot abstraction significantly reduces the state space so that *AVChecker* can complete the violation identification within 13 seconds for each targeted rule.

The contributions of this paper are as follows:

1) We propose an AV domain-specific abstraction, which is a formal representation amenable to SMT solving, to bridge the semantic gap between software code and safety specifications. This new abstraction simplifies the representation of infinite-state space in the physical world and addresses the scalability challenge in SMT-based driving rule compliance checking.

2) We design and implement *AVChecker*, to the best of our knowledge, the first general framework for checking scenario-dependent driving rules in AV software in a systematic manner. Using the user interface of *AVChecker*, AV developers can reveal rule violations in the code logic against predefined driving rule specifications with the minimum manual effort.

3) We evaluate *AVChecker* on *Baidu Apollo* [3] and *Autoware* [5] to check rule compliance with 28 DMV's driving rules. *AVChecker* uncovers 13 violations in *Apollo* and 6 violations in *Autoware*, which are all validated by simulation.

Our paper with full details can be found at [10].

REFERENCES

[1] [n.d.]. Automated Driving Systems 2.0: A Vision for Safety. https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/13069a-ads2.0_090617_v9a_tag.pdf.

[2] 2019. A Matter of Trust Ford's Approach to Developing Self-driving Vehicles. https://media.ford.com/content/dam/fordmedia/pdf/Ford_AV_LLC_FINAL_HR_2.pdf.

[3] 2019. ApolloAuto: An open autonomous driving platform. https://github.com/ApolloAuto/apollo.

[4] 2019. General Motors 2018 Self-Driving Safety Report. https://www.gm.com/content/dam/company/docs/us/en/gmcom/gmsafetyreport.pdf.

[5] 2020. Autoware: Open-source software for self-driving vehicles. https://gitlab.com/autowarefoundation/autoware.ai.

[6] 2020. Waymo Safety Report. https://waymo.com/safety.

[7] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*.

[8] Chris Lattner and Vikram Adve. 2004. LLVM: A compilation framework for lifelong program analysis & transformation. In *International Symposium on Code Generation and Optimization, 2004. CGO 2004.* IEEE, 75–86.

[9] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2017. On a Formal Model of Safe and Scalable Self-driving Cars. *CoRR* (2017).

[10] Qingzhao Zhang, Ke David Hong, Ze Zhang, Qi Alfred Chen, Scott Mahlke, and Z. Morley Mao. 2021. A Systematic Framework to Identify Violations of Scenario-dependent Driving Rules in Autonomous Vehicle Software. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* (2021).