ScenaVRo: Ensuring Variety of Scenario Set for Testing Autonomous Driving Systems

Seongdeok Seo
Graduate School of Computer Science and Engineering
Ulsan National Institute of Science & Technology
Ulsan, Republic of Korea
deokk1112@unist.ac.kr

***Index Terms*—Autonomous Driving Systems, Fuzzing, Scenario Abstraction, Test Suite Optimization**

I. Context

In recent years, testing Autonomous Driving Systems (ADS) has gained increasing attention due to the importance of ensuring safety and reliability, with Software-in-the-Loop Simulation (SILS) emerging as a central tool, offering efficient testing capabilities in a virtual environment. SILS enables virtual software testing without the need for physical hardware, providing an effective means of emulating driving scenarios. It offers a virtual environment for evaluating ADS, allowing various scenarios to be simulated under repeatable conditions.

In Simulation-based ADS testing, where a variety of driving environments are feasible, ensuring variety within the set of driving scenarios is highly crucial. Bagschik et al. [3] introduced a scenario layer model that organizes scenario parameters into five layers based on associated elements. In our approach, we classified the parameters comprising these five layers into Low-Layer (Layer 1 to 3) and High-Layer (Layer 4 to 5). The criterion for this classification is based on whether the parameters relate to the geographical scope of the HD-Map where driving scenarios are executed (Low-Layer) or pertain to objects generated within the simulator and the simulated environment (High-Layer).

Recent research has employed two primary approaches to determine the seed driving scenario set: Manual Crafting and Generation-based Fuzzing. Manual Crafting is primarily used to configure seed driving scenarios, especially in mutation-based fuzzing. While Manual Crafting allows researchers to focus on specific scenarios of interest, it may overlook some corner-case scenarios and can be labor-intensive. In contrast, Generation-based Fuzzing has contributed to introducing diversity and generating various scenarios for autonomous driving scenario creation. However, these methods mainly concentrate on the Low-Layer aspects of driving scenarios and do not consider the potential diversity that can arise from the High-Layer of scenario execution. For example, they do not take into account how the driving behavior of the ego-vehicle can change due to interactions between the ego-vehicle and Non-Player Characters (NPCs). Consequently, considering diversity in actual execution results becomes challenging.

In this paper, we introduce a method to diversify both the Low-Layer and High-Layer aspects of driving scenarios simultaneously, along with an approach to evaluate the diversity in scenario execution outcomes. Our proposed approach ensures variety across all layers of driving scenarios by managing diversity in the Low-Layer while considering the influence from the High-Layer. As a result, through various driving scenario executions, we can detect corner cases and optimize the test suite through evaluation methods. Our approach aims to pave the way for more robust and effective testing methodologies, ultimately improving the safety and reliability of ADS.

II. Gap

Optimizing the test suite is generally a crucial process, but it becomes even more critical in ADS testing due to the significantly longer time required to execute a driving scenario compared to typical software programs. Therefore, the importance of methods that maintain variety while minimizing the driving scenario set is highly emphasized. Deng et al. [4] introduced a reduction method to reduce the size of individual driving scenarios. In contrast, our approach validates through fuzzing cycles while keeping the driving scenario set various.

Mutation-based Fuzzing. Numerous work focusing on testing ADS through SILS aim to validate safety violations [5]–[18]. Typically, these studies employ methodologies that automatically generate driving scenarios violating safety by mutating manually crafted seed driving scenarios. While these efforts have greatly improved the identification of challenging ADS scenarios, they still face the inherent limitation of not sufficiently covering diverse driving scenarios. To maximize variety within the given seed scenarios, CRT [16] and BehAVExplor [17] consider covering the widest possible range of the ego-vehicle's driving behaviors alongside safety violation verification. Meanwhile, MOSAT [18] standardizes NPC vehicle driving behaviors and explores methods to incorporate a variety of NPC driving actions. However, given that the variety of driving scenarios still doesn't deviate significantly from the provided seed scenario, there's a potential for ADS to not comprehensively test the various challenges they face on the road. These limitations are particularly noteworthy in real-world, as autonomous vehicles navigate through a multitude of unpredictable situations in complex driving environments.

Generation-based Fuzzing. Research aimed at generating various driving scenarios to address the diversity issue in driving scenarios is also being conducted. ScenoRITA [19] and DoppelTest [20] employ random-based fuzzing, randomly configuring the ego-vehicle's driving path across the entire map area before finding safety violations. ComOpT [21] and Atlas [22] utilize model-based fuzzing, automating the generation of driving scenarios by categorizing Low-Layer components. These approaches, however, do not consider the diversity in the High-Layer and its associated impacts.

In contrast to prior work, we evaluate the uniqueness of scenarios within a driving scenario set without being constrained by the types of seed scenarios for driving scenarios. This is achieved by analyzing the results of scenario execution. In summary, this paper makes the following contributions:

- **Low-Layer Variation.** We propose a method for analyzing HD-Maps to extract drivable routes and constructing an input corpus by analyzing the characteristics of these routes to ensure variety. (Section III-A)
- **Variety Evaluation.** We introduce a novel approach to abstract the results of scenario execution into essential sequences of driving actions, allowing us to verify the uniqueness of driving scenarios. (Section III-C1)
- **Test Suite Minimization.** Our method uses scenario abstraction to minimize the test suite by identifying and removing similar scenarios, optimizing resource utilization and testing time. (Section III-C3)

III. Innovation

In this paper, we introduce ScenaVRo, a tool designed to ensure Scenario Variety in Execution Routes, as illustrated in Fig. 1, which shows its abstracted workflow. In the input generation phase, we construct an systematical input corpus to ensure Low-Layer variety, while in the evaluation phase, we evaluate the uniqueness of driving behaviors influenced by other dynamic objects to ensure High-Layer variety.

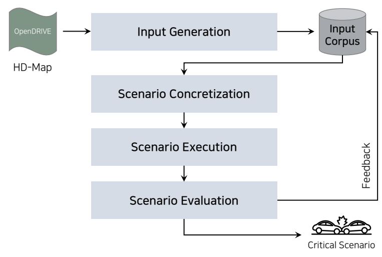

Fig. 1. Abstracted Overview of ScenaVRo

A. Input Generation

In the input generation phase of ScenaVRo, we automatically extract all minimal unit driving routes from the HD-Map, creating keys to represent their characteristics and storing them in a dictionary. By employing this route dictionary, we ensure Low-Layer diversity, defining scenarios without the High-Layer as Low-Level Scenarios. Fig. 2 illustrates the progression of the input generation phase of ScenaVRo.

1) Unit Driving Route: We introduce the `RoadGraph`, an efficient storage and management system for the necessary information from HD-Maps. The `RoadGraph` stores road-to-road connectivity information from ASAM OpenDRIVE format map data in the form of a directed graph. The `RoadGraph` enables the automatic generation of minimal driving scenario scopes by selecting a target junction road and retrieving information about connected roads from it. This enables us to automatically generate unit driving route that start from the predecessor road, pass through the target junction road, and reach the successor road.

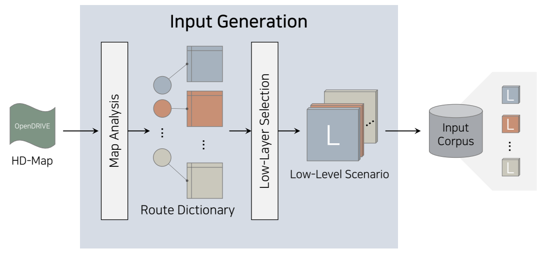

Fig. 2. Input Generation Phase of ScenaVRo

2) Route Type Classification: Each route extracted through `RoadGraph` is assigned a route key that can represent the features of the composed roads and junctions. Fig. 3 provides a detailed illustration of what specific road features are represented. The `RouteDictionary` is structured as a dictionary structure containing lists of routes for each route key present in the `RoadGraph`. We generate driving routes for scenarios from the `RouteDictionary`. This enables us to cluster Low-Level Scenarios and ensures variety in the Low-Layer.

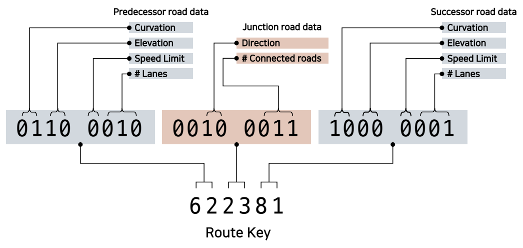

Fig. 3. Generating Dictionary Key for One Unit Driving Route

B. Scenario Concretization

The scenario concretization phase of ScenaVRo is a step in which we select one Low-Level Scenario from the input corpus and add High-Layer components. This phase, depicted in the upper part of Fig. 4, is a step in ScenaVRo's fuzzing cycle. It involves High-Layer components like dynamic objects and weather parameters, where weather parameters are randomly chosen to match simulator specifications, and dynamic objects use data from the `RoadGraph`. Since we have knowledge of the vehicle's driving area, we automatically generate information such as the positions and driving directions of dynamic objects that can influence the given driving path.

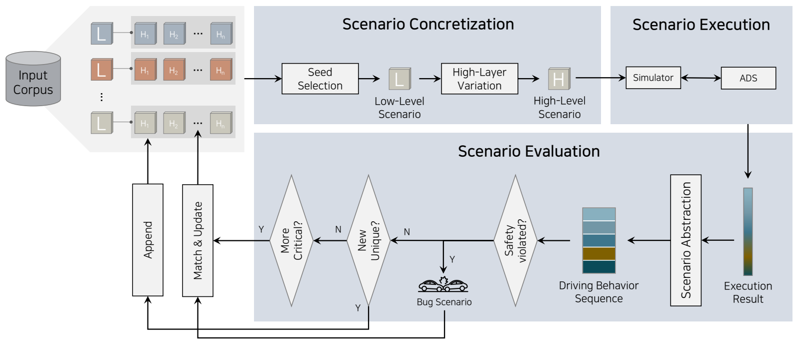

Fig. 4. Fuzzing Cycle of ScenaVRo

C. Scenario Evaluation

In this section, we describe how ScenaVRo evaluates driving scenario results and provides feedback to the input corpus. To evaluate driving scenario results, they are summarized as sequences of driving behaviors. Through the driving behavior sequence resulting from abstraction, we can ultimately evaluate whether a driving scenario includes safety violation behaviors and is unique. In Section III-C3, we explain how these evaluation results are used to optimize the input corpus.

1) Scenario Abstraction: Given that seemingly identical driving behaviors can manifest in various forms, it is crucial to classify these behaviors in detail. To translate simulation results into more granular driving actions, we implemented an Abstract State Machine (ASM) that predefines six driving behavior states (Lane-Follow, Lane-Change, Junction-Cross, Slow-Down, Stop, Collide) and allows for computable state transitions. These six defined driving behavior states are further refined based on factors such as road or intersection types, turning directions, and object types affecting control. Three analysis are required to calculate these factors and define state transitions. Control analysis and collision analysis virtualize the bounding boxes representing sensor perception ranges and vehicle drivable areas to identify causes of driving behaviors. Location analysis differentiates the types of roads and intersections being navigated and incorporates steer information at specific positions, providing a clearer understanding of driving behaviors. As a result, we can obtain a specific driving behavior sequence through ASM, which allows us to evaluate the uniqueness of each driving scenario.

2) Safety Evaluation: We classify highly dangerous driving actions such as collision and hitting illegal lines as safety violation behaviors. These classifications can be confirmed through the results of collision analysis and location analysis.

3) Feedback in Fuzzing Cycle: Repetitive execution of similar driving scenarios is inefficient in terms of testing time utilization. To address this, optimization of the driving input corpus is necessary. We propose a method for optimizing the input corpus by identifying scenarios with identical sequences of driving behaviors and eliminating duplicates. If a safety violation occurred during a driving scenario, a constraint is conveyed to the input corpus to prevent the selection of the same driving behavior sequence again. If no safety violation occurred but a new sequence of driving behaviors is discovered, the corresponding driving scenario is added to the input corpus. Finally, if no safety violation occurred, and an existing sequence of driving behaviors is identified, the input corpus is updated to replace the existing scenario with a more hazardous driving scenario among the two. This process contributes to maintaining continuous High-Layer diversity in driving scenarios and optimizing the input corpus.

IV. Evaluation Plan

We will apply ScenaVRo to generate bug scenarios for testing Autoware [23], a full-fledged (Level 4) Autonomous Driving System (ADS) integrated with the CARLA simulator [1]. In order to assess the efficiency and effectiveness of ScenaVRo, we will address the following research questions:

- *RQ1: How effective is ScenaVRo in finding safety violations of ADS compared to the selected baselines?*
- *RQ2: How effective is ScenaVRo in generating various driving scenarios compared to the selected baselines?*

To compare ScenaVRo, we selected two baseline approaches: BehAVExplor(one of mutation-based fuzzing tools), and ComOpT(one of generation-based fuzzing tools).

The effectiveness of ScenaVRo is expected to be maximized as the time spent on fuzzing increases since ScenaVRo utilizes time resources for both diversifying seed executions and identifying safety violations. In the experiments related to *RQ1*, which focus on the efficiency of detecting safety violations, ScenaVRo is anticipated to outperform ComOpT, which primarily emphasizes diversity. In comparison with BehAVExplor, it is expected that ScenaVRo may initially find fewer safety violation scenarios during the early stages of fuzzing. However, as fuzzing progresses, ScenaVRo is anticipated to surpass BehAVExplor due to differences in seed diversity. The experiments related to *RQ2*, which assess the efficiency of discovering various driving scenarios, are expected to show superior performance for ScenaVRo over BehAVExplor, which is geared toward finding safety violation behaviors within the same seeds. In comparison to ComOpT, ScenaVRo is expected to follow a similar pattern initially, but ultimately generate a more diverse range of driving scenarios due to its consideration of High-Layer diversity.

References

[1] Dosovitskiy, Alexey, et al. ”CARLA: An open urban driving simulator.” Conference on robot learning. PMLR, 2017.

[2] Rong, Guodong, et al. ”Lgsvl simulator: A high fidelity simulator for autonomous driving.” 2020 IEEE 23rd International conference on intelligent transportation systems (ITSC). IEEE, 2020.

[3] Bagschik, Gerrit, Till Menzel, and Markus Maurer. ”Ontology based scene creation for the development of automated vehicles.” 2018 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2018.

[4] Deng, Yao, et al. ”Scenario-based test reduction and prioritization for multi-module autonomous driving systems.” Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2022.

[5] Ben Abdessalem, Raja, et al. ”Testing advanced driver assistance systems using multi-objective search and neural networks.” Proceedings of the 31st IEEE/ACM international conference on automated software engineering. 2016.

[6] Abdessalem, Raja Ben, et al. ”Testing autonomous cars for feature interaction failures using many-objective search.” Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. 2018.

[7] Abdessalem, Raja Ben, et al. ”Testing vision-based control systems using learnable evolutionary algorithms.” Proceedings of the 40th International Conference on Software Engineering. 2018.

[8] Zhou, Jinwei, and Luigi del Re. ”Safety verification of adas by collision-free boundary searching of a parameterized catalog.” 2018 Annual American Control Conference (ACC). IEEE, 2018.

[9] Li, Guanpeng, et al. ”Av-fuzzer: Finding safety violations in autonomous driving systems.” 2020 IEEE 31st international symposium on software reliability engineering (ISSRE). IEEE, 2020.

[10] Calò, Alessandro, et al. ”Generating avoidable collision scenarios for testing autonomous driving systems.” 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST). IEEE, 2020.

[11] Luo, Yixing, et al. ”Targeting requirements violations of autonomous driving systems by dynamic evolutionary search.” 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2021.

[12] Ghodsi, Zahra, et al. ”Generating and characterizing scenarios for safety testing of autonomous vehicles.” 2021 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2021.

[13] Kim, Seulbae, et al. ”Drivefuzz: Discovering autonomous driving bugs through driving quality-guided fuzzing.” Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. 2022.

[14] Zhong, Ziyuan, Gail Kaiser, and Baishakhi Ray. ”Neural network guided evolutionary fuzzing for finding traffic violations of autonomous vehicles.” IEEE Transactions on Software Engineering (2022).

[15] Wan, Ziwen, et al. ”Too afraid to drive: systematic discovery of semantic DoS vulnerability in autonomous driving planning under physical-world attacks.” arXiv preprint arXiv:2201.04610 (2022).

[16] Tang, Yun, et al. ”Route coverage testing for autonomous vehicles via map modeling.” 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021.

[17] Cheng, Mingfei, Yuan Zhou, and Xiaofei Xie. ”BehAVExplor: Behavior Diversity Guided Testing for Autonomous Driving Systems.” Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. 2023.

[18] Tian, Haoxiang, et al. ”MOSAT: finding safety violations of autonomous driving systems using multi-objective genetic algorithm.” Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2022.

[19] Huai, Yuqi, et al. ”sceno RITA: Generating Diverse, Fully-Mutable, Test Scenarios for Autonomous Vehicle Planning.” IEEE Transactions on Software Engineering (2023).

[20] Huai, Yuqi, et al. ”Doppelgänger Test Generation for Revealing Bugs in Autonomous Driving Software.” 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). IEEE, 2023.

[21] Li, Changwen, et al. ”Comopt: Combination and optimization for testing autonomous driving systems.” 2022 International Conference on Robotics and Automation (ICRA). IEEE, 2022.

[22] Tang, Yun, et al. ”Systematic testing of autonomous driving systems using map topology-based scenario classification.” 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2021.

[23] Kato, Shinpei, et al. ”Autoware on board: Enabling autonomous vehicles with embedded systems.” 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS). IEEE, 2018.