Human-In-The-Loop Oracle Learning for Simulation-Based Testing

Ben-Hau Chia
Carnegie Mellon University
Pittsburgh, USA
bhchia@cmu.edu

Eunsuk Kang
Carnegie Mellon University
Pittsburgh, USA
eskang@cmu.edu

Christopher S. Timperley
Carnegie Mellon University
Pittsburgh, USA
ctimperley@cmu.edu

***Abstract*—Ensuring safety and providing rigorous behavioral guarantees are critical for robotic systems operating in high-stakes environments such as autonomous driving. Field testing is common, but costly and risky. Simulation-based testing offers a safer and lower-cost alternative for automatically generating traces for analysis and performance assessment. An *oracle* is essential for evaluating each trace, assessing whether a robot behavior fulfills key criteria such as task completion, safety, efficiency, and reliability. Supervised learning for oracle learning is accurate but costly and time-consuming due to manual labeling, whereas unsupervised learning requires no labels but often sacrifices accuracy. To overcome these limitations, we propose *human-in-the-loop oracle learning* as a new approach to develop and refine oracles that are capable of distinguishing good from bad behaviors with reduced manual effort. We illustrate this approach through a conceptual framework for integrating human-in-the-loop learning into robotic system evaluation.**

***Index Terms*—robotics, testing, simulation, oracle learning**

I. Introduction

Robotic systems are increasingly being deployed across diverse industries, offering transformative capabilities in both automation and autonomy. Robotic systems such as drones, autonomous delivery robots and industrial robotic arms have been applied to precision agriculture [1], hospitality services [2], and manufacturing [3], respectively. It is essential to verify that the robots behave as intended to ensure the safety of the entire system. *Field testing* [4] is a common approach to evaluate robot behaviors, allowing collection of observations of interactions with the environment, uncovering bugs, and generating execution traces for later analysis. However, it can also be costly, as unexpected actions or malfunctions can damage equipment. Safety is also a concern as unexpected robot behaviors can cause injuries when humans share the experimental space. Even with expected robot behavior, building the testing robots, setting up the testing environment, and coordinating a full testing team can still be costly. Alternatively, developers may wish to validate the software on a robotic system before the actual physical robot is built.

To minimize costs and risks, *simulation-based testing* provides an appealing alternative and is widely adopted across domains, including quadrotors [5], self-driving cars [6], and marine robotics [7], helping to detect bugs before real-world field testing. Compared to physical testing, simulations are typically more cost-effective, safer, scalable, and easier to control. Simulations also enable task execution to be faster than field tests, allowing efficient and automated performance evaluation across scenarios. In addition, simulators provide ground-truth data on the robot and environment, including the positions of surrounding objects, which is difficult to obtain in the field due to limited robot perception and the impracticality of equipping all objects with sensors like GPS. While this data may not perfectly reflect real-world dynamics, it enables accurate evaluation and testing in controlled conditions.

A key challenge in simulation-based testing is evaluating robot behavior, which is essential for diagnosing robot performance and identifying flaws. An *oracle* serves as a reference specification to judge whether a robot behaves correctly in a given simulation trace. Oracles can take various forms, such as human domain experts, predefined algorithms, or a learned set of constraints. Human experts provide accurate judgments but are costly and labor-intensive, whereas oracles based on predefined algorithms offer efficiency and consistency but struggle to adapt to unexpected scenarios or corner cases. To this end, *learning* a set of constraints, specified in languages such as Signal Temporal Logic (STL) [8], to serve as an oracle for assessing robot performance offers a promising balance, reducing human supervision while adapting to the diversity and unpredictability of real-world conditions. Temporal logic properties can be learned by assuming formula structures such as parametric STL (PSTL) [9]–[12], by decision trees [13], [14], and by neural networks [15].

To derive such an oracle, *supervised learning* [16], [17] provides a solid foundation for training oracles to make accurate predictions using labeled traces, enabling them to learn patterns and relationships aligned with human-defined criteria. However, manually labeling the data can be expensive and time-consuming, as it requires human inspection of each trace. On the other hand, *unsupervised learning* [18]–[20] infers patterns directly from unlabeled data. Unsupervised learning algorithms usually rely on the assumption that similar data points have similar characteristics; therefore, these algorithms can learn the implicit grouping relationships among the data without knowing the ground truth. However, unsupervised learning can compromise accuracy because no labeled data is used to guide the learning process, making it difficult to reliably classify acceptable from unacceptable robot behaviors.

Semi-supervised learning [21], [22] provides an attractive

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00341

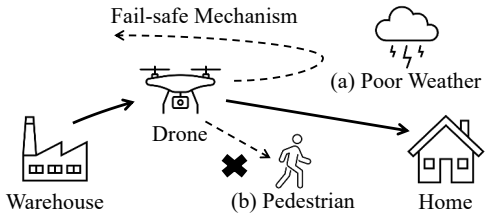

Fig. 1. Two challenges a drone may encounter when delivering goods to a customer's home: (a) environmental factors and (b) obstacle avoidance.

characteristic in this case. It enables the oracle to understand good and bad patterns using a smaller portion of the labeled data, so labeling the entire training dataset is unnecessary. This learning method can strike a balance between reducing the cost and time of human labeling and achieving acceptable accuracy. Active learning [23], [24], namely "human-in-the-loop" oracle learning, is a semi-supervised learning approach where the system obtains human feedback on selected unlabeled traces whenever the inferred oracle is uncertain about its judgment.

This paper aims to highlight the role of humans in the oracle learning loop for robotic systems and discuss the potential challenges of how to leverage human knowledge efficiently. We first introduce a motivating example to demonstrate a real-world case of oracle learning in robotic systems. We then propose a framework for integrating humans into the oracle learning loop. Finally, we conclude by enumerating research challenges and future work to realize the proposed framework.

II. Motivating Example

To illustrate the core idea of this paper, we use drone-based last-mile delivery as a motivating example (Figure 1). Rapid growth in e-commerce has increased the demand for faster and more efficient last-mile delivery [25]–[27]. Delivery drones can avoid road traffic for more reliable and predictable arrival times, and access narrow areas where traditional vehicles may struggle. In the remainder of this section, we focus on drone systems as a representative case of robotic last-mile delivery.

The implementation of drone technology in delivery services presents several challenges related to perception, decision making, and safety. Drones rely on integrated sensing, planning, and control components to perceive their environment, make decisions, and execute flight maneuvers. However, external factors such as adverse weather can hinder flight stability and safety (Figure 1(a)). In addition, sensor errors can lead to inaccurate distance estimates, leading to miscalculations and misjudgments. These issues can create dynamic situations where drone decision-making processes may not always align with human expectations or safety standards. Although timely delivery is the main objective, the manner in which drones navigate to achieve this goal is equally critical to public safety. For example, drones should avoid flying too close to pedestrians to prevent collisions (Figure 1(b)).

To operate safely and effectively, drones must plan and follow optimal routes while considering travel time, safety, and environmental constraints. They must avoid obstacles, such as buildings and pedestrians, and trigger fail-safe mechanisms, such as return-to-base and emergency landing, when encountering emergencies. Throughout navigation, drones must comply with safety requirements, including avoiding collisions with birds or other drones and steering clear of restricted areas (e.g., airports or military bases). Ideally, drones should adapt their behavior based on surrounding conditions; for example, they should navigate more conservatively in poor weather.

Given the diversity and unpredictability of real-world conditions, hard-coded rules and fixed safety margins are insufficient. For instance, the minimum safe distance from pedestrians may depend on visibility, wind, and pedestrian density; enumerating or hand-coding all possible conditions would be infeasible.

III. Approach Overview

As discussed in Section I, using a human expert as an oracle can be too costly and labor-intensive, while a predefined algorithm can fail to respond to rare cases. Therefore, we consider the oracle to be a set of learned constraints, such as temporal logic formulas [28], [29], and the oracle learning framework evaluates traces based on it. For simplicity, we refer to the oracle learning framework as the *framework* for the remainder of the paper. The oracle takes traces from the simulator as input and outputs scores that indicate how well or poorly the robot performs during the simulations. To make the oracle adaptable to different situations and to enable it to learn from data, the framework queries about the performance qualities of a trace. "Qualities" refer to a standard that balances safety, efficiency, reliability, and other important factors, although the exact interpretation may vary among individuals. Because manually labeling all traces is impractical, the framework selectively queries humans for feedback on specific traces. However, querying humans too frequently is still expensive. To minimize this cost, we propose a conceptual framework for the human-in-the-loop oracle learning process, which efficiently integrates human feedback into the learning process, allowing the oracle to refine its understanding and generalize to similar traces by updating or adding constraints.

Framework: Figure 2 illustrates the proposed human-in-the-loop oracle learning framework. The framework starts from *Scenario Generation*, which simulates the environments in which the robot is to be tested. Various scenarios can be generated by adjusting the setup parameters to imitate both dynamic environments and diverse behaviors the drone may exhibit in the real world. For example, in the drone delivery system, the setup parameters may include the drone's initial position, maximum velocity, the absolute time during flight, and so on. *Trace Collection* runs simulations to generate an initial dataset D_0, recording features such as drone position, velocity, and environmental context (e.g., time, temperature). With this dataset, *Oracle Inference* infers the initial oracle O_0 with unsupervised learning algorithms, providing the framework with a coarse evaluation of the structure and trace quality of the dataset. We use the subscript to indicate the current

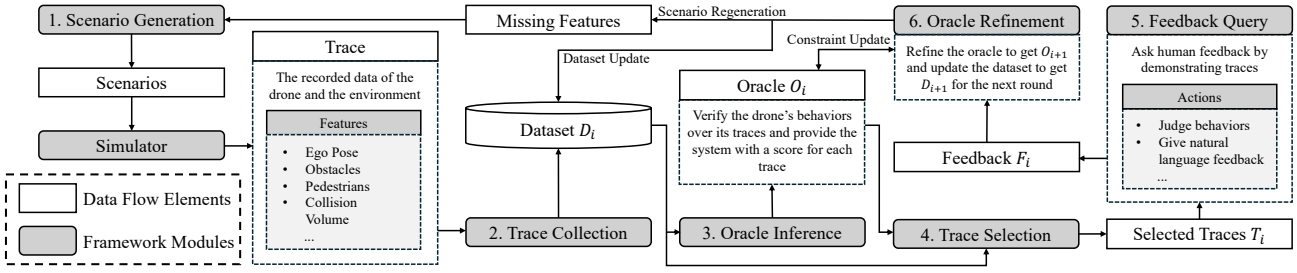

Fig. 2. The proposed framework for introducing humans into the oracle learning loop.

round, where a round starts with *Oracle Inference* and follows by *Trace selection*, *Feedback Query*, and *Oracle Refinement*.

After inferring the oracle O_i in the i-th round, *Trace Selection* selects a set of traces T_i from the dataset that are expected to yield the highest improvement if human feedback is provided. This step is formalized as a selection function $T_i = select(D_i, O_i)$. The selection may focus on traces that lie near classification boundaries or represent corner cases with high uncertainty. For example, a speed that is safe for a drone on sunny days may be dangerous on rainy nights because of reduced visibility. To address the uncertain speed limit on rainy nights, *Trace Selection* would choose a trace that shows borderline behavior to clarify the correct threshold.

Based on the human feedback set F_i associated with the traces in T_i, *Oracle Refinement* updates both the oracle and the dataset using three mechanisms.

1) *Constraint Update*: The oracle is refined by adding or modifying constraints based on feedback. This step is modeled as a *learn function* $O_{i+1} = learn(O_i, F_i)$. For example, if humans specify that drone speed should never exceed 15 kilometers per hour (kph), *Oracle Refinement* adds a new corresponding constraint to check drone speed in each trace.
2) *Dataset Update*: Feedback is recorded and added to the dataset to support future oracle inference. This is modeled as an *update function*: $D_{i+1} = update(D_i, F_i)$.
3) *Scenario Regeneration*: If the feedback references some specific rare or absent scenario in the dataset, *Oracle Refinement* can trigger *Scenario Generation* to generate those scenarios to collect more representative traces for oracle inference. For example, if humans provide feedback discussing the safety distance between drone and pedestrians, *Scenario Generation* can be triggered to generate more scenarios that demonstrate different distances for the oracle to learn a better distance threshold.

With the updated trace dataset and the oracle, the framework can trigger *Oracle Inference* for the next round.

We will evaluate the proposed approach using three metrics: (1) oracle accuracy, (2) sample efficiency, measured by the number of traces required to reach an accuracy threshold, and (3) inference efficiency, measured by the computation time per inference round. We will compare these metrics with and without human feedback to assess its contribution.

IV. Research Challenges

In this section, we propose research challenges to support the human-in-the-loop oracle learning process in Figure 2.

Scenario Generation: To support oracle inference, the framework must identify simulated scenarios that accurately replicate real-world conditions. There are simulators that generate simulation scenarios, such as SUMO [30], Gazebo [31], CARLA [32], and AirSim [33]. A scenario includes elements that define both the robot (e.g., its model, tasks, goals, and initial conditions like position and orientation) and its environment (e.g., obstacles and pedestrians) as well as how they change over time. Scenarios can be created manually, but setting parameters requires significant human effort and can result in limited diversity, failing to cover all environmental conditions. To address this, recent work has explored automated scenario generation from formal requirements or textual descriptions. Domain-specific probabilistic languages, such as Scenic [34]–[36], have focused on automatically generating scenes for perception simulation.

One major research challenge is to identify the relevant components that should be included in simulation scenarios. For example, in the drone delivery system, the testing environment should be outdoors and obstacles (e.g., buildings and pedestrians) with which the drone interacts. However, for domestic robots, the testing environment would be indoor, with objects such as furniture and family members with which the robot interacts. One possible approach is to leverage large language models (LLMs) to interpret scenario documentation (e.g., crash report for autonomous vehicles [37]) and translate them into a scenario description language. *Scenario Generation* can generate the corresponding scenarios in the simulator. For example, if the documentation highlights an unsafe interaction between drones and pedestrians, the LLMs should suggest including pedestrians in the scenario to collect interactions between drones and pedestrians.

Trace Collection: Once the scenarios are defined, the next step is to run simulations to collect traces for oracle inference. During simulation, the simulator can capture the state of the overall simulation by recording features related to both the robot (e.g., position) and the environment (e.g., time); the feature can be captured in either continuous (e.g., time of day) or categorical (e.g., day and night) form [38]. In most existing work, the set of recorded features is predetermined

before simulations begin, which is often based on domain knowledge. However, this approach assumes prior knowledge of the data that will be useful for oracle learning.

A key research challenge is automatically identifying the related features to capture from natural language documentation and finding memory-efficient methods to collect these features during simulations. Specifically, how can we store a representation of the environment that is adequate for oracle inference, performance evaluation, and human demonstration while using the least amount of memory? Collecting too few features can compromise accuracy as it leaves insufficient data, while capturing too many features can increase memory usage and slow down the oracle inference process. In the drone delivery system example, omitting the time of day during simulations prevents defining time-based constraints, such as "The drone should fly slower at night than during the day."

Oracle Inference: The system can infer an oracle to evaluate the robot performance of each trace in the dataset, and oracles can be of various types. One common type is based on temporal logic formulas, such as STL [8]. The oracle evaluates the behavior of a trace based on its learned temporal logic formulas: a trace represents good behavior if the formulas are satisfied; otherwise, it is considered bad. However, most existing works targeting STL inference rely on predetermined formula templates [13], [14], which could narrow down the search space for the oracle. Another type of oracle can be learned from clustering methods [38]–[40], where the oracle evaluates a trace by matching it to a cluster and seeing whether the cluster represents good or bad behaviors. The challenge lies in ensuring that clusters capture the subtle difference between good and bad behaviors without under- or over-fitting.

Making oracles human-interpretable is also a key challenge in oracle inference. For example, contrary to previous assumptions, STL is less interpretable than human thought [41]. To improve usability, formulas should prioritize clarity and interpretability that help capture key behavioral constraints, helping developers better understand the oracle's learned constraints, and increasing trust in its decisions. Addressing these challenges is vital for making the learned oracles reliable, efficient, and adaptable for robust simulation-based testing.

Trace Selection: Asking humans for feedback is effective but expensive. To minimize cost, the framework should prioritize the most informative traces. Randomly selecting traces is intuitive but can waste effort on obvious cases, such as drone crashes. To this end, we advocate a targeted approach in which the oracle queries humans about ambiguous traces, those for which it is uncertain whether they are good or bad.

One possible approach to identify ambiguous traces is through *STL robustness scores* [42]. The magnitude of a trace's robustness score quantifies how well it satisfies or violates an STL formula, with positive indicating satisfaction and negative values indicating violation. A trace with a robustness score around zero could be an ideal candidate for feedback, as it implies that the behavior is close to the decision boundary. We refer interested readers to [42] for further details.

Another possible approach is to use clustering methods [38]–[40]. Traces that do not belong to any major clusters can be considered corner cases, while those far from the centroid of a cluster may be problematic cases, both of which can be candidates to ask for human feedback [38].

Oracle Refinement: Once the framework receives human feedback, the next step is to refine the oracle based on the feedback. The first research challenge lies in modeling the *learn function* as mentioned in Section III. If feedback is explicit, the framework can update the oracle directly by adding additional constraints or adjusting existing constraints in the oracle. Methods that transform natural languages into formal specifications (e.g., [43], [44]) may be one promising approach to dealing with feedback interpretation; there is also work that converts traffic rules into metric temporal logic (MTL) [45]. However, ambiguous or informal feedback may require additional interpretation. In such cases, LLM-based approaches could help the framework understand the feedback.

The second research challenge is how to handle conflict and design the *oracle update function*, described in Section III, when multiple pieces of feedback are inconsistent. Human feedback can introduce ambiguity and be open to multiple interpretations, making it difficult for the framework to discern the correct action or understanding. For example, two humans might disagree on whether a trace is safe, or a single user might revise their judgment over time. Determining which feedback to trust and when to revise or override existing rules is crucial for robust oracle refinement.

V. Conclusion

As robotics and autonomous technologies continue to advance, behavior testing becomes crucial for ensuring the safety and correctness of robots' actions. Simulating-based testing is an attractive approach because it reduces costs in terms of time and money. In this paper, we focus on the human-in-the-loop oracle learning framework, which helps the oracle assess the accuracy of traces obtained from the simulation. By incorporating human insights, the oracle can resolve ambiguities in robot behavior, allowing developers to find potential bugs through simulation. We have demonstrated how the learning framework can leverage human feedback to refine the oracle iteratively, ultimately enhancing the overall reliability of the robotic system. In addition, we have outlined key challenges and future directions for expanding this framework.

References

[1] U. R. Mogili and B. Deepak, "Review on application of drone systems in precision agriculture," *Procedia computer science*, vol. 133, pp. 502–509, 2018.

[2] S. Srinivas, S. Ramachandiran, and S. Rajendran, "Autonomous robot-driven deliveries: A review of recent developments and future directions," *Transportation research part E: logistics and transportation review*, vol. 165, p. 102834, 2022.

[3] Y. Chen, L. Chen, J. Ding, and Y. Liu, "Research on real-time obstacle avoidance motion planning of industrial robotic arm based on artificial potential field method in joint space," *Applied Sciences*, vol. 13, no. 12, p. 6973, 2023.

[4] A. Afzal, C. Le Goues, M. Hilton, and C. S. Timperley, "A study on challenges of testing robotic systems," in *2020 IEEE 13th international conference on software testing, validation and verification (ICST)*, pp. 96–107, IEEE, 2020.

[5] K. Darshan and M. Rakesh, "Simulation of an autonomous navigation system using quadrotor model in robot operating system," *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NCRACES*, vol. 7, no. 10, 2019.

[6] S. Chen, Y. Chen, S. Zhang, and N. Zheng, "A novel integrated simulation and testing platform for self-driving cars with hardware in the loop," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 3, pp. 425–436, 2019.

[7] P. Cieślak, "Stonefish: An advanced open-source simulation tool designed for marine robotics, with a ros interface," in *OCEANS 2019-Marseille*, pp. 1–6, IEEE, 2019.

[8] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pp. 152–166, Springer, 2004.

[9] E. Asarin, A. Donzé, O. Maler, and D. Nickovic, "Parametric identification of temporal properties," in *Runtime Verification: Second International Conference, RV 2011, San Francisco, CA, USA, September 27-30, 2011, Revised Selected Papers 2*, pp. 147–160, Springer, 2012.

[10] S. Jha, A. Tiwari, S. A. Seshia, T. Sahai, and N. Shankar, "Telex: Passive stl learning using only positive examples," in *International Conference on Runtime Verification*, pp. 208–224, Springer, 2017.

[11] B. Hoxha, A. Dokhanchi, and G. Fainekos, "Mining parametric temporal logic properties in model-based design for cyber-physical systems," *International Journal on Software Tools for Technology Transfer*, vol. 20, pp. 79–93, 2018.

[12] S. Mohammadinejad, J. V. Deshmukh, A. G. Puranic, M. Vazquez-Chanlatte, and A. Donzé, "Interpretable classification of time-series data using efficient enumerative techniques," in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, pp. 1–10, 2020.

[13] G. Bombara, C.-I. Vasile, F. Penedo, H. Yasuoka, and C. Belta, "A decision tree approach to data classification using signal temporal logic," in *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pp. 1–10, 2016.

[14] E. Aasi, C. I. Vasile, M. Bahreinian, and C. Belta, "Classification of time-series data using boosted decision trees," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1263–1268, IEEE, 2022.

[15] D. Li, M. Cai, C.-I. Vasile, and R. Tron, "Learning signal temporal logic through neural network for interpretable classification," in *2023 American Control Conference (ACC)*, pp. 1907–1914, IEEE, 2023.

[16] O. M. Mozos, C. Stachniss, and W. Burgard, "Supervised learning of places from range data using adaboost," in *Proceedings of the 2005 IEEE international conference on robotics and automation*, pp. 1730–1735, IEEE, 2005.

[17] M. Menner, L. Neuner, L. Lünenburger, and M. N. Zeilinger, "Using human ratings for feedback control: A supervised learning approach with application to rehabilitation robotics," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 789–801, 2020.

[18] S. Koenig and R. G. Simmons, "Unsupervised learning of probabilistic models for robot navigation," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2301–2308, IEEE, 1996.

[19] N. Hirose, A. Sadeghian, P. Goebel, and S. Savarese, "To go or not to go? a near unsupervised learning approach for robot navigation," *arXiv preprint arXiv:1709.05439*, 2017.

[20] M. Zaric, J. Hollenstein, J. Piater, and E. Renaudo, "Unsupervised learning of effective actions in robotics," *arXiv preprint arXiv:2404.02728*, 2024.

[21] A. K. Tanwani, P. Sermanet, A. Yan, R. Anand, M. Phielipp, and K. Goldberg, "Motion2vec: Semi-supervised representation learning from surgical videos," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2174–2181, IEEE, 2020.

[22] J. Rückin, F. Magistri, C. Stachniss, and M. Popović, "Semi-supervised active learning for semantic segmentation in unknown environments using informative path planning," *IEEE Robotics and Automation Letters*, 2024.

[23] C. Chao, M. Cakmak, and A. L. Thomaz, "Transparent active learning for robots," in *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 317–324, IEEE, 2010.

[24] A. T. Taylor, T. A. Berrueta, and T. D. Murphey, "Active learning in robotics: A review of control principles," *Mechatronics*, vol. 77, p. 102576, 2021.

[25] Flytrex, "Flytrex," 2024. Accessed: 2024-12-08.

[26] Zipline, "Fly zipline," n.d. Accessed: 2024-12-08.

[27] Wing, "Wing," n.d. Accessed: 2024-12-08.

[28] A. Camacho and S. A. McIlraith, "Learning interpretable models expressed in linear temporal logic," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 29, pp. 621–630, 2019.

[29] A. Linard and J. Tumova, "Active learning of signal temporal logic specifications," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pp. 779–785, IEEE, 2020.

[30] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, "Sumo (simulation of urban mobility)-an open-source traffic simulation," in *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*, pp. 183–187, 2002.

[31] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3, pp. 2149–2154, Ieee, 2004.

[32] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*, pp. 1–16, PMLR, 2017.

[33] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics: Results of the 11th International Conference*, pp. 621–635, Springer, 2018.

[34] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: a language for scenario specification and scene generation," in *Proceedings of the 40th ACM SIGPLAN conference on programming language design and implementation*, pp. 63–78, 2019.

[35] D. J. Fremont, E. Kim, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: A language for scenario specification and data generation," *Machine Learning*, vol. 112, no. 10, pp. 3805–3849, 2023.

[36] E. Vin, S. Kashiwa, M. Rhea, D. J. Fremont, E. Kim, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "3d environment modeling for falsification and beyond with scenic 3.0," in *International Conference on Computer Aided Verification*, pp. 253–265, Springer, 2023.

[37] K. Elmaaroufi, D. Shanker, A. Cismaru, M. Vazquez-Chanlatte, A. Sangiovanni-Vincentelli, M. Zaharia, and S. A. Seshia, "Scenicnl: generating probabilistic scenario programs from natural language," in *First Conference on Language Modeling*, 2024.

[38] A. Afzal, C. Le Goues, and C. S. Timperley, "Mithra: Anomaly detection as an oracle for cyberphysical systems," *IEEE Transactions on Software Engineering*, vol. 48, no. 11, pp. 4535–4552, 2021.

[39] M. T. Guerreiro, E. M. A. Guerreiro, T. M. Barchi, J. Biluca, T. A. Alves, Y. de Souza Tadano, F. Trojan, and H. V. Siqueira, "Anomaly detection in automotive industry using clustering methods—a case study," *Applied Sciences*, vol. 11, no. 21, p. 9868, 2021.

[40] E. Vanem and A. Brandsæter, "Unsupervised anomaly detection based on clustering methods and sensor data on a marine diesel engine," *Journal of Marine Engineering & Technology*, vol. 20, no. 4, pp. 217–234, 2021.

[41] H. C. Siu, K. Leahy, and M. Mann, "Stl: Surprisingly tricky logic (for system validation)," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8613–8620, IEEE, 2023.

[42] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[43] J. He, E. Bartocci, D. Ničković, H. Isakovic, and R. Grosu, "Deepstl: from english requirements to signal temporal logic," in *Proceedings of the 44th International Conference on Software Engineering*, pp. 610–622, 2022.

[44] M. Cosler, C. Hahn, D. Mendoza, F. Schmitt, and C. Trippel, "nl2spec: interactively translating unstructured natural language to temporal logics with large language models," in *International Conference on Computer Aided Verification*, pp. 383–396, Springer, 2023.

[45] K. Manas, S. Zwicklbauer, and A. Paschke, "Tr2mtl: Llm based framework for metric temporal logic formalization of traffic rules," *arXiv preprint arXiv:2406.05709*, 2024.