Parallel and Multi-Objective Falsification with SCENIC and VERIFAI

Kesav Viswanadha[1], Edward Kim[1], Francis Indaheng[1],
Daniel J. Fremont[2], and Sanjit A. Seshia[1]

[1] University of California, Berkeley
[2] University of California, Santa Cruz

Abstract. Falsification has emerged as an important tool for simulation-based verification of autonomous systems. In this paper, we present extensions to the SCENIC scenario specification language and VERIFAI toolkit that improve the scalability of sampling-based falsification methods by using parallelism and extend falsification to multi-objective specifications. We first present a parallelized framework that is interfaced with both the simulation and sampling capabilities of SCENIC and the falsification capabilities of VERIFAI, reducing the execution time bottleneck inherently present in simulation-based testing. We then present an extension of VERIFAI's falsification algorithms to support multi-objective optimization during sampling, using the concept of rulebooks to specify a preference ordering over multiple metrics that can be used to guide the counterexample search process. Lastly, we evaluate the benefits of these extensions with a comprehensive set of benchmarks written in the SCENIC language.

Keywords: Runtime Verification · Formal Methods · Falsification · Cyber-Physical Systems · Autonomous Systems · Parallelization

1 Introduction

The growing adoption of autonomous and semi-autonomous cyber-physical systems (CPS) such as self-driving vehicles brings with it pressing questions about ensuring their safety and reliability. In particular, the increasing use of artificial intelligence (AI) and machine learning (ML) components requires significant advances in formal methods, of which simulation-based formal analysis is a key ingredient [26].

Even with notable development in simulators and methods for simulation-based verification, there are four practical issues which require further advances in tools. First, simulation time can be a huge bottleneck, as falsification is typically done with high-quality, realistic simulators such as CARLA [13], which can be computation-intensive. Second, modeling interactive, multi-agent behaviors using general programming languages like Python can be very time-consuming. Third, autonomous systems usually need to satisfy multiple properties and metrics, with differing priorities, and convenient notation is needed to formally specify these. Fourth, we need to develop specification and sampling methods for falsification that can support multiple objectives. These issues have all been addressed in this paper with a series of features aimed at improving the scalability of falsification methods, both in terms of execution time and the richness of objectives that can be specified and falsified.

There has been prior work that addresses these four issues separately. There have been several ideas for falsification or, conversely, optimization of CPS subject to multiple objectives [10,5,6,31,23]. There are other tools that address simulation-based testing of CPS, including in a parallel context [4,22,3]. There has also been some prior work on exploration methods for constrained falsification [30]. However, these methods tend to either focus on testing specific CPS components (as opposed to full closed-loop CPS) or require complex code to use in a practical setting. More importantly, to our knowledge, no prior work has *jointly* addressed all of these issues and demonstrated these in a single tool. In this paper, we do so by extending the open-source VERIFAI toolkit [14].[3] VERIFAI is reasonably mature, having been demonstrated in multiple industrial case studies [19,15]. Our contributions to the toolkit support:

1. *Parallelized falsification*, running multiple simulations in parallel;
2. Falsification using the latest version of the SCENIC *formal scenario specification language*, extending support to the “dynamic” features of SCENIC for modeling interactive behaviors [17];
3. The ability to specify for falsification *multiple objectives with priority orderings*;
4. A *multi-armed bandit* algorithm that supports multi-objective falsification, and
5. Evaluation of these extensions with a comprehensive set of self-driving scenarios.

These contributions have had a profound impact on the capabilities of VERIFAI. With parallel falsification, we were able to cut down drastically on execution time, achieving up to 5x speedup over the current falsification methods in VERIFAI using 5 parallel simulation processes. Using the multi-objective multi-armed bandit sampler, we were able to find scenarios which falsify five objectives at the same time.

2 Background

SCENIC is a probabilistic programming language [16,17,7] that allows users to intuitively model *probabilistic scenarios* for multi-agent systems. A *concrete scenario* is a set of objects and agents, together with values for their static attributes, initial state, and parameters of dynamic behavioral models describing how their attributes evolve over time. In other words, a concrete scenario defines a specific trace. The state of each object or agent, such as a car, includes its semantic properties such as its position, orientation, velocity, color, model, etc. We refer to the vector of such semantic properties as a *semantic feature vector*; the concatenation of the semantic feature vectors of all objects and agents at a given time instant defines the overall semantic feature vector at that time. Agents also have behaviors defining a (possibly stochastic) sequence of actions for them to take as a function of the state of the simulation at each time step. A SCENIC program defines a *distribution over concrete scenarios*: by sampling an initial state and then executing the behaviors in a simulator, many different simulations can be obtained from a single SCENIC program. SCENIC provides a general formalism to express probabilistic scenarios for multiple domains, including traffic and other scenarios for autonomous vehicles, which can then be executed in a number of simulators

[3] Documentation of the extensions covered in this paper is available at: https://verifai.readthedocs.io/en/kesav-v-multi-objective/.

including CARLA [13]. In previous work on VERIFAI [14], the tool supported an earlier version of SCENIC without interactive, behavioral specifications. In this paper, we provide full support for SCENIC's newer dynamic features.

VERIFAI is a Python toolkit that provides capabilities for verification of AI-based systems [8]. A primary capability is *falsification*, the systematic search for inputs to a system that falsify a specification given in temporal logic or as a cost function. VERIFAI can use SCENIC as an environment modeling language, sampling from the distribution over semantic feature vectors defined by a SCENIC program to generate test cases. It then simulates these cases according to the dynamics specified in the SCENIC program, obtaining trajectories for each object. For a more detailed description of VERIFAI's falsification capabilities and interface with dynamic scenarios specified in the SCENIC language, please see [18].

After simulating a test case, VERIFAI evaluates the system's specification over the obtained trajectory, saving the results for offline analysis. These results are also used to guide further falsification, specifically by VERIFAI's *active samplers*, such as the cross-entropy sampler [25]. These samplers use the history of previously generated samples and their outcomes in simulation to drive the search process to find more counterexamples.

3 Parallel Falsification

In the typical pipeline used by a VERIFAI falsifier driven by a SCENIC program, semantic feature vectors (parameters) are generated using samplers in either SCENIC or VERIFAI These parameter values are then sent by the VERIFAI server to the client simulator to configure a simulation and generate a corresponding trajectory. This trajectory is then evaluated by the monitor, deemed either a safe example or a counterexample, and added to the corresponding table in the falsifier. Naturally, a bottleneck of this process is the generation of the trajectory in the simulator, as this is a rather compute-intensive task that can take a minute or more per sample, depending on the scenario description.

We present an improvement on this pipeline by parallelizing it using the Python library Ray [20], which encapsulates process-level parallelism optimized for distributed execution of computation-intensive tasks. Fig. 1 illustrates the new setup: we instantiate multiple instances of the simulator and open multiple SCENIC server connections from VERIFAI to the simulator instances for performing simulations (the connections now being bidirectional so that the behavior models in the SCENIC program can respond to the current state of the simulation). We then aggregate the results of these simulations into a single error table documenting all the counterexamples found during falsification.

4 Multi-Objective Falsification

There are typically many different metrics of interest for evaluating autonomous systems. For example, there are many well-known metrics used in the autonomous driving community to measure safety: no collisions, obeying traffic laws, and maintaining a minimum safe distance from other objects, among others [29]. It is also natural to assert, for example, that it is more important to avoid collisions than to follow traffic laws. We now discuss how to specify these metrics and their relative *priorities*.

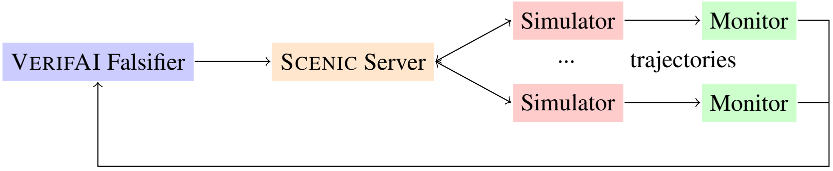

Fig. 1. Parallelized pipeline for falsification using VERIFAI.

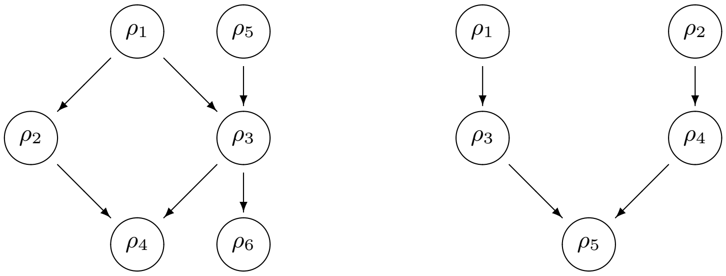

Fig. 2. Left: example rulebook over functions $\rho_1 \dots \rho_6$ [11]. Right: graph G used in experiments.

4.1 Specification of Multiple Objectives Using Rulebooks

Let $\rho(x)$ be a function mapping a simulation trajectory generated by SCENIC or VERIFAI to a vector-valued objective, where $\rho_j(x)$ is defined as the value of the j-th metric. Censi et al. [11] have developed a way to specify preferences over these metrics using a *rulebook* denoted by $\mathcal{R}$ – a directed acyclic graph (DAG) where the nodes are the metrics and a directed edge from node i to node j means $\rho_i(x)$ is more important than $\rho_j(x)$. We denote this using the $>_R$ operator, e.g. $\rho_i >_R \rho_j$.

Fig. 2 shows an example of a rulebook over six metrics $\rho_1, \dots, \rho_6$. In this example, we can make several inferences, such as ρ_1 is more important than ρ_3, ρ_3 is more important than ρ_4, and ρ_5 is more important than ρ_3. However, there are also many pairs of objective components that cannot be compared; for example ρ_1 and ρ_5. We would like to have a way to order objective vectors to know which values are maximally violating of the specification during active sampling. Because of these indeterminate incomparisons, the rulebook $\mathcal{R}$ only allows for a *partial ordering* $\succ$ over the objective vectors. Intuitively, we can think of this partial ordering as preferring examples that have lower values of higher priority objectives since we are trying to minimize the values of each objective for falsification. However, if there is any other indeterminate or higher priority objective that has a higher value, the $\succ$ relation does not hold. To satisfy these properties, we define our $\succ$ operator as follows:

$$\rho(x_1) \succ \rho(x_2) \triangleq \forall i \left(\rho_i(x_2) < \rho_i(x_1) \implies \exists j \neq i \left(\rho_j >_R \rho_i \land \rho_j(x_1) < \rho_j(x_2)\right)\right)$$

As an example, consider our rulebook from Fig. 2. Let $\rho(x_1) = \begin{bmatrix}1 \ 1 \ 1 \ 1 \ 1 \ 1\end{bmatrix}^T$, and $\rho(x_2) = \begin{bmatrix}1 \ 1 \ 2 \ 1 \ 0 \ 1\end{bmatrix}^T$. In this case we have $\rho(x_2) \succ \rho(x_1)$ because $\rho_5(x_2) < \rho_5(x_1)$, and even though $\rho_3(x_2) > \rho_3(x_1)$, $\rho_5 >_R \rho_3$ according to the rulebook, so the comparison of ρ_5 for the trajectories takes precedence. Since the rulebook defines a partial ordering over values of ρ, it is possible to have two trajectories x_1 and x_2 such that $\rho(x_1) \not\succ \rho(x_2)$ and $\rho(x_2) \not\succ \rho(x_1)$. In such cases, both values of ρ are maintained in the sampling algorithm; see below for more details.

4.2 Multi-Objective Active Sampling

When performing active sampling to search for unsafe test inputs, we need a specialized sampler to support having multiple objectives to guide the search process. Most of the samplers previously available in VERIFAI focused either entirely on exploration of the search space or entirely on exploitation to find unsafe inputs; we present a sampler that balances these and builds up increasingly-violating counterexamples in the multi-objective case.

The Multi-Armed Bandit Sampler. We present a more robust version of VERIFAI's cross-entropy sampler called the *multi-armed bandit sampler*; the idea of this sampler is to balance the trade-off between exploitation and exploration. To understand the motivation for the sampler, we first look at the formulation of the multi-armed bandit problem. Consider a bandit which has multiple lotteries, or "arms", to choose from, each being a random variable offering a probabilistic reward. The bandit does not know ahead of time which arm gives the highest expected reward, and must learn this information by efficiently sampling various arms, while also maximizing average earned reward during the sampling process.

Carpentier et al. [9] present the Upper Confidence Bound (UCB) Algorithm that effectively balances both of these goals, subject to a confidence parameter δ, by sampling the arm j that minimizes a quantity Q_j dependent on the number of timesteps t, the number of times the arm j was sampled $T_j(t-1)$, the observed reward of arm j given by $\hat{\mu}_j$, and the confidence parameter δ:

$$Q_j = \hat{\mu}_j + \sqrt{\frac{2}{T_j(t-1)} \ln\left(\frac{1}{\delta}\right)}$$

Qualitatively, this works as a balance between exploitation of the reward distribution learned so far (the first term), and exploration of seldom-sampled arms (the second term). We can easily see that this can be readily adapted to our cross-entropy sampler in VERIFAI, which splits the range of each sampled variable into N equally spaced *buckets*, which can be considered the "arms". We take $\hat{\mu}_j$ to be the proportion of counterexamples found in bucket j.

To compute μ_j for a vector-valued objective, we present the following incremental algorithm which builds up counterexamples that falsify more and more objectives (according to the priority order) over time. The steps of this algorithm are as follows. This assumes that the sampler is responsible for generating a d-dimensional feature vector.

Setup

1. Split the range of each component of the feature vector into N buckets, as in the cross-entropy sampler.
2. Initialize matrix T of size $d \times N$ where T_{ij} will keep track of the number of times that bucket j was visited for variable x_i.

3. Initialize a dictionary c mapping each maximal counterexample found so far to a matrix c_b of size $d \times N$ where $c_{b,ij}$ counts how many times sampling bucket j for variable x_i resulted in the specific counterexample b.
4. Sample from each bucket once initially, updating c and T according to the update algorithm described below. The purpose of this is to avoid division by zero when computing Q, as $T_j(t-1) = 0$ at initialization [2].

Sampling

1. Compute a matrix $\hat{\mu}$ where $\hat{\mu}_{ij}$ represents the observed reward from sampling bucket j for variable i by taking $\sum_b c_{b,ij}$.
2. Compute a matrix Q based on the upper confidence bound formula above. For the confidence parameter, we use a time-dependent value of $\frac{1}{\delta} = t$.
3. To sample x_i, take the bucket $j^* = \arg\max_j Q_{ij}$. *Break ties uniformly at random.* This is a key step in the sampling process as it is frequently the case initially that several buckets will have the exact same Q_j value, so we need to avoid bias towards any specific bucket. Sample uniform randomly within the range represented by bucket j^*.

Updating Internal State

1. Given the objective vector value ρ, we compute our vector of booleans b as described above.
2. If b does not exist in the dictionary c and is among the set of maximal counterexamples found so far, i.e. $\forall b' \in c, b' \not\succ b$ as defined by the rulebook $\mathcal{R}$, add b as a key to the dictionary c and initialize its value as $0^{d \times N}$.
3. For any $b' \in c$ such that $b \succ b'$, remove b' from c.
4. Increment the count c_b at each position $c_{b,ij}$ for the bucket j sampled from x_i.

5 Evaluation

We present a set of experiments designed to evaluate (i) the speedup in simulation time that we expect to see from parallelization; (ii) the benefits of the multi-armed bandit sampler in balancing exploration and exploitation; and (iii) the improved capabilities of falsification to support multiple objectives. We have developed a library of SCENIC scripts[4] based on the list of pre-crash scenarios described by the National Highway Traffic Safety Administration (NHTSA) [21]. For a list of the scenarios, see [28]. These scripts cover a wide variety of common driving situations, such as driving through intersections, bypassing vehicles, and accounting for pedestrians.

We selected 7 of these scenarios, running the VERIFAI falsifier on each one in CARLA [13] for 30 minutes, with individual simulations limited to 300 timesteps (∼30 seconds). For all of these scenarios, the monitor specifies that the centers of the ego vehicle and other vehicles must stay at least 5 meters apart at all times. This specification means that counterexamples approximately correspond to collisions or near-collisions. All parallelized experiments were run using 5 worker processes to perform simulation.

Fig. 3 shows the results of running these scenarios with a variety of configurations. First, across the scenarios, we observed a 3-5x speedup in the number of simulations

[4] Full listing and source code of these SCENIC scripts is available at: https://github.com/BerkeleyLearnVerify/Scenic/tree/kesav-v/multi-objective/examples/carla/Behavior_Prediction.

using 5 parallel simulation processes. The variation in the number of samples generated can be attributed to *termination conditions* set in SCENIC, which terminate simulations early if specific conditions are met. For some of these scenarios, termination occurred much sooner on average than other scenarios, leading to more simulations finishing in 30 minutes. These values also serve as partial evidence of the effectiveness of the multi-armed bandit sampler compared to cross-entropy, as the proportion of counterexamples found is comparable for the two samplers despite the increased exploration component in the multi-armed bandit sampler.

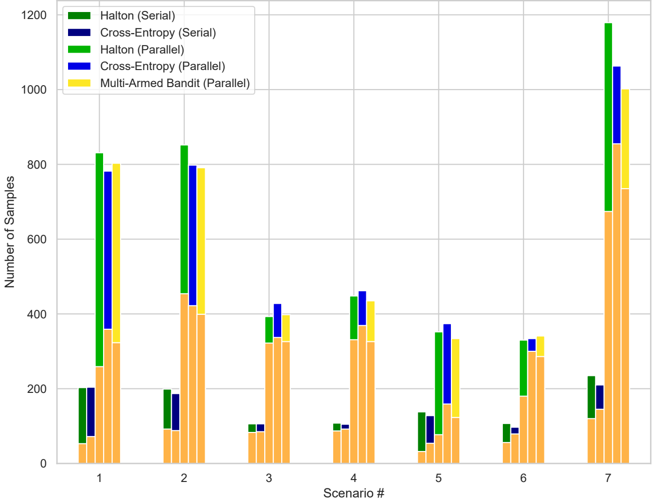

Fig. 3. Comparison of (i) the serial and parallel versions of the falsifier for cross-entropy and Halton sampling and (ii) the multi-armed bandit sampler with the cross-entropy and Halton samplers all in parallel. The orange part of the bars represent the number of counterexamples found out of the total number of samples.

To validate the scalability and explorative aspect of parallelized falsification, we present two metrics in Table 1. The first metric is the *speedup factor*, which is the ratio of the number of sampled scenarios in parallel versus serial falsification, averaged across the Halton and cross-entropy samplers. We are also interested in a metric of coverage of the scenario search space, as this ensures that a wide range of scenarios are tested by falsification. To this end, we present the *confidence interval width ratio* metric. This metric is computed by generating a 95% confidence interval [12] which provides a lower and upper bound on the probability that a randomly generated scenario results in unsafe behavior. Since confidence intervals are generated with the assumption of uniform random sampling, we only compute them for the serial and parallel Halton samplers since they are an approximation of random sampling. We take the ratio of the widths of the intervals in the parallel versus serial case to compare how tight we are able to make the bound in each case with the same level of confidence. The width of the interval in the parallel case is significantly smaller - up to half the width of the serial case. Since the width of the interval is proportional to $1/\sqrt{n}$ for n samples, this makes intuitive sense and can be viewed as having double the coverage of the search space.

Table 1. The speedup factor and confidence interval width ratio metrics for the 7 scenarios.

Scenario #	1	2	3	4	5	6	7
Speedup Factor	3.96	4.27	3.87	4.27	2.73	3.26	5.04
CI Width Ratio	0.51	0.48	0.48	0.53	0.61	0.56	0.44

Figs. 3 and 4 show the qualitative benefits of the multi-armed bandit sampler. The number of counterexamples generated by the multi-armed bandit sampler is higher than for the Halton sampler, but only slightly lower than cross-entropy. However, we can clearly see that multi-armed bandit sampling achieves a balance between number of counterexamples and their diversity that cross-entropy and Halton do not.

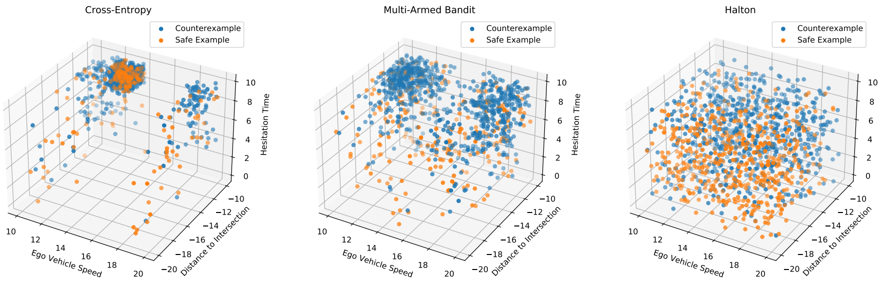

Fig. 4. Comparison of points sampled for cross-entropy, MAB, and Halton samplers.

To demonstrate the effectiveness of the multi-objective multi-armed bandit sampler in falsifying multiple objectives, we used a Scenic program that instantiates the ego vehicle, along with m adversarial vehicles at random positions with respect to a 4-way intersection and has all of them drive towards the intersection and either go straight or make a turn. The monitor, similarly to before, specifies metric components ρ_j which say the ego vehicle must stay at least 5 meters away from vehicle j. We use the following three rulebooks: a completely disconnected graph representing no preference ordering, a linked list structure $L \triangleq \rho_1 >_R \rho_2 >_R \ldots >_R \rho_5$ representing a total ordering, and the graph G on the right in Fig. 2. We found that when using L or G, we were able to falsify 4 of the 5 objectives with serial falsification, and all 5 objectives in the parallel case. When having no preference ordering, we were able to falsify 3 of the 5 objectives with serial falsification and 4 of the 5 objectives in the parallel case. By contrast, when we combined all of these objectives in disjunction as one single objective (such that only falsifying all 5 objectives is considered unsafe), the cross-entropy sampler was unable to find any counterexamples.

We have also tested these methods in experiments with the LGSVL simulator [24]. Using a multi-objective specification with a variety of common driving situations, we were able to generate a wide range of test cases that cover much of the space of possible scenarios. These experiments were run with Apollo, an open-source autonomous driving software stack [1]. We discovered a number of bugs in Apollo using these new capabilities of VerifAI and Scenic, such as issues with stopping for pedestrians and properly avoiding encroaching vehicles [27].

6 Conclusion and Future Work

The extensions to SCENIC and VERIFAI we report in this paper address important problems in simulation-based falsification. First, we cut down significantly on execution time by supporting parallel simulations. Second, we allow the simple specification of high-level yet complex scenarios using the interface between dynamic SCENIC and VERIFAI. Third, we support multi-objective specification through the formalism of rulebooks. Lastly, we are able to falsify these multi-objective specifications in a way that is intuitive and scalable using the multi-armed bandit sampler. We hope these extensions prove useful to developers of autonomous systems.

There are a few directions for future work. For example, it might be interesting to see if generating random topological sorts of the rulebooks to create total ordering works well in practice. One could also run covariance analysis on the features to determine if they can be jointly optimized for better active sampling. Further comparison and analysis across other competing active and passive samplers is needed. Lastly, there has been some work in connecting these ideas to real-world testing [19], but especially with multi-objective falsification, this is an interesting future direction. In an industry setting, it may also be worthwhile to scale up parallel falsification even further to run on cloud instances for increased efficiency, which is technically possible but yet to be implemented.

Acknowledgments. This work is partially supported by NSF grants 1545126 (VeHICaL), 1646208 and 1837132, by the DARPA contracts FA8750-18-C-0101 (AA) and FA8750-20-C-0156 (SDCPS), by Berkeley Deep Drive, and by Toyota under the iCyPhy center.

References

1. Apollo: Autonomous Driving Solution. http://apollo.auto/, last accessed: 07-22-2021
2. The upper confidence bound algorithm (Sep 2016), https://banditalgs.com/2016/09/18/the-upper-confidence-bound-algorithm/
3. Abbas, H., Fainekos, G., Sankaranarayanan, S., Ivančić, F., Gupta, A.: Probabilistic temporal logic falsification of cyber-physical systems. ACM Trans. Embed. Comput. Syst. **12**(2s) (May 2013). https://doi.org/10.1145/2465787.2465797, https://doi.org/10.1145/2465787.2465797
4. Annpureddy, Y., Liu, C., Fainekos, G., Sankaranarayanan, S.: S-taliro: A tool for temporal logic falsification for hybrid systems. In: Abdulla, P.A., Leino, K.R.M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 254–257. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
5. Araujo, H., Carvalho, G., Mousavi, M.R., Sampaio, A.: Multi-objective search for effective testing of cyber-physical systems. In: Ölveczky, P.C., Salaün, G. (eds.) Software Engineering and Formal Methods. pp. 183–202. Springer International Publishing, Cham (2019)
6. Arrieta, A., Wang, S., Markiegi, U., Sagardui, G., Etxeberria, L.: Employing multi-objective search to enhance reactive test case generation and prioritization for testing industrial cyber-physical systems. IEEE Transactions on Industrial Informatics **14**(3), 1055–1066 (2018). https://doi.org/10.1109/TII.2017.2788019
7. BerkeleyLearnVerify: Berkeleylearnverify/scenic, https://github.com/BerkeleyLearnVerify/Scenic
8. BerkeleyLearnVerify: Berkeleylearnverify/verifai, https://github.com/BerkeleyLearnVerify/VerifAI
9. Carpentier, A., Lazaric, A., Ghavamzadeh, M., Munos, R., Auer, P.: Upper-confidence-bound algorithms for active learning in multi-armed bandits. In: Kivinen, J., Szepesvári, C., Ukkonen, E., Zeugmann, T. (eds.) Algorithmic Learning Theory. pp. 189–203. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
10. Castro, L.I.R., Chaudhari, P., Tumova, J., Karaman, S., Frazzoli, E., Rus, D.: Incremental sampling-based algorithm for minimum-violation motion planning. CoRR **abs/1305.1102** (2013), http://arxiv.org/abs/1305.1102
11. Censi, A., Slutsky, K., Wongpiromsarn, T., Yershov, D.S., Pendleton, S., Fu, J.G.M., Frazzoli, E.: Liability, ethics, and culture-aware behavior specification using rulebooks. CoRR **abs/1902.09355** (2019), http://arxiv.org/abs/1902.09355
12. Clopper, C.J., Person, E.S.: The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial. Biometrika **26**(4), 404–413 (12 1934). https://doi.org/10.1093/biomet/26.4.404, https://doi.org/10.1093/biomet/26.4.404
13. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning. pp. 1–16 (2017)
14. Dreossi, T., Fremont, D.J., Ghosh, S., Kim, E., Ravanbakhsh, H., Vazquez-Chanlatte, M., Seshia, S.A.: VerifAI: A toolkit for the formal design and analysis of artificial intelligence-based systems. In: 31st International Conference on Computer Aided Verification (CAV) (Jul 2019)
15. Fremont, D.J., Chiu, J., Margineantu, D.D., Osipychev, D., Seshia, S.A.: Formal analysis and redesign of a neural network-based aircraft taxiing system with verifai. CoRR **abs/2005.07173** (2020), https://arxiv.org/abs/2005.07173
16. Fremont, D.J., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Scenic: A language for scenario specification and scene generation. In: Proceedings of the 40th annual ACM SIGPLAN conference on Programming Language Design and Implementation (PLDI) (June 2019)

17. Fremont, D.J., Kim, E., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Scenic: A language for scenario specification and data generation. CoRR **abs/2010.06580** (2020), https://arxiv.org/abs/2010.06580
18. Fremont, D.J., Kim, E., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Scenic: A language for scenario specification and data generation. CoRR **abs/2010.06580** (2020), https://arxiv.org/abs/2010.06580
19. Fremont, D.J., Kim, E., Pant, Y.V., Seshia, S.A., Acharya, A., Bruso, X., Wells, P., Lemke, S., Lu, Q., Mehta, S.: Formal scenario-based testing of autonomous vehicles: From simulation to the real world. In: 23rd IEEE International Conference on Intelligent Transportation Systems (ITSC) (Sep 2020)
20. Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Paul, W., Jordan, M.I., Stoica, I.: Ray: A distributed framework for emerging AI applications. CoRR **abs/1712.05889** (2017), http://arxiv.org/abs/1712.05889
21. Najm, W.G., Smith, J.D., Yanagisawa, M.: Pre-crash scenario typology for crash avoidance research (Apr 2007), https://www.nhtsa.gov/sites/nhtsa.gov/files/pre-crash_scenario_typology-final_pdf_version_5-2-07.pdf
22. Qin, X., Aréchiga, N., Best, A., Deshmukh, J.V.: Automatic testing and falsification with dynamically constrained reinforcement learning. CoRR **abs/1910.13645** (2019), http://arxiv.org/abs/1910.13645
23. Ramezani, Z., Eddeland, J.L., Claessen, K., Fabian, M., Åkesson, K.: Multiple objective functions for falsification of cyber-physical systems. IFAC-PapersOnLine **53**(4), 417–422 (2020)
24. Rong, G., Shin, B.H., Tabatabaee, H., Lu, Q., Lemke, S., Možeiko, M., Boise, E., Uhm, G., Gerow, M., Mehta, S., Agafonov, E., Kim, T.H., Sterner, E., Ushiroda, K., Reyes, M., Zelenkovsky, D., Kim, S.: Lgsvl simulator: A high fidelity simulator for autonomous driving. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC). pp. 1–6 (2020). https://doi.org/10.1109/ITSC45102.2020.9294422
25. Sankaranarayanan, S., Fainekos, G.: Falsification of temporal properties of hybrid systems using the cross-entropy method. In: Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control. p. 125–134. HSCC '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2185632.2185653, https://doi.org/10.1145/2185632.2185653
26. Seshia, S.A., Sadigh, D., Sastry, S.S.: Towards Verified Artificial Intelligence. ArXiv e-prints (July 2016)
27. Viswanadha, K., Indaheng, F., Wong, J., Kim, E., Kalvan, E., Pant, Y., Fremont, D.J., Seshia, S.A.: Addressing the IEEE AV Test Challenge with Scenic and VerifAI. In: The IEEE Third International Conference on Artificial Intelligence Testing
28. Viswanadha, K., Kim, E., Indaheng, F., Fremont, D.J., Seshia, S.A.: Parallel and multi-objective falsification with Scenic and VerifAI. CoRR **abs/2107.04164** (2021), https://arxiv.org/abs/2107.04164
29. Wishart, J., Como, S., Elli, M., Russo, B., Weast, J., Altekar, N., James, E.: Driving safety performance assessment metrics for ADS-equipped vehicles (04 2020). https://doi.org/10.4271/2020-01-1206
30. Zhang, Z., Arcaini, P., Hasuo, I.: Hybrid system falsification under (in) equality constraints via search space transformation. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **39**(11), 3674–3685 (2020)
31. Zhou, X., Gou, X., Huang, T., Yang, S.: Review on testing of cyber physical systems: Methods and testbeds. IEEE Access **6**, 52179–52194 (2018). https://doi.org/10.1109/ACCESS.2018.2869834