SBFT Tool Competition 2024 - Cyber-Physical Systems Track

Matteo Biagiola
Università della Svizzera italiana
Lugano, Switzerland
matteo.biagiola@usi.ch

Stefan Klikovits
Institute for Business Informatics - Software Engineering
Johannes Kepler University
Linz, Austria
stefan.klikovits@jku.at

ABSTRACT

This report summarizes the results of the fourth edition of the 2024 Cyber-Physical Systems tool competition, held as part of the SBFT'24 workshop. Three tools (AmbieGenVAE, CRAG, and OptAngle) competed with the aim of triggering out-of-bounds errors of two autonomous driving agents. The competitors were evaluated based on the effectiveness in exposing failures and the diversity of the discovered faulty tests. As in previous years, we report on the experiment methodology, the competitors and the results.

KEYWORDS

Autonomous Vehicles, Search-Based Software Testing

ACM Reference Format:
Matteo Biagiola and Stefan Klikovits. 2024. SBFT Tool Competition 2024 - Cyber-Physical Systems Track . In *2024 ACM/IEEE International Workshop on Search-Based and Fuzz Testing (SBFT '24), April 14, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3643659.3643932

1 INTRODUCTION

Being among the most promising cyber-physical systems (CPSs), self-driving vehicles experience continuous attention software engineering research community [16] and industry [8]. Tool competitions offer the opportunity to benchmark new testing approaches and improve the state of the art.

As in previous years, we organized the fourth edition of the SBFT Testing Tool Competition's CPS track. This edition received three submissions: AmbieGenVAE, CRAG, and OptAngle. This is half as many submissions as in the previous edition, which we attribute to the availability of a similar track targeting testing of UAVs (a.k.a. Drones) [13]. Two teams that joined the previous edition improved their tools for this year's competition. The OptAngle team participated for the first time in the competition, confirming the continued interest in this competition despite the availability of new, alternative tracks.

Similar to previous editions, we provided the participants with an updated version of the open-source test framework [15] together with the documentation on how to use it (tutorials, instructions, sample driving agents, and test generators). The test framework encapsulates the definition of complex driving scenarios and their execution in a simulator.

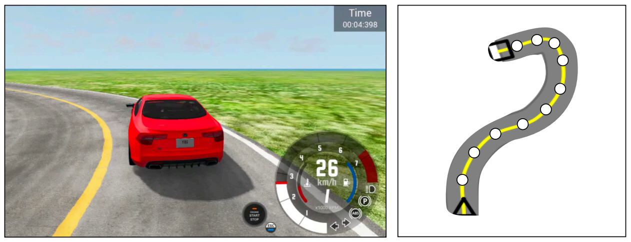

Figure 1: A driving task for lane keeping assist systems (right); test simulation in BeamNG.tech (left).

This year, we decided to re-use the same competition setup as last year, hence reusing our test framework and the same version of the BeamNG.tech simulator [4]. We also introduced a clustering-based diversity metric based on the structural and behavioral features of the tests we defined in the previous edition [5]. Clustering is a common way to measure failure diversity in software testing, as recent literature suggests [1, 6, 21]. We use such diversity metric alongside the feature map coverage [19, 20] we used in the previous edition. For the test subjects, we selected (1) BeamNG.AI, the driving agent included with the BeamNG.tech simulator, and (2) a DL-based driving agent based on the Dave-2 architecture proposed by Bojarski et al. [7]. Both test subjects have been used in previous research as well [10, 12, 17, 18].

As in previous years [5, 9, 14], we compared the competing tools by running them multiple times on each test subject in order to account for the stochastic nature of the tools. For the BeamNG.AI agent, AmbieGenVAE by Humeniuk and Khomh [11] achieved the best results, being minimally ahead of CRAG by Arcaini and Cetinkaya [2] in all metrics. However, for Dave-2, CRAG achieved by far the highest scores for all coverage metrics. OptAngle by Babikian and Varró [3] generated the most test cases and valid roads, and managed to discover the highest absolute number of out-of-bounds errors for BeamNG.AI, but did not achieve as diverse results as the other tools.

In the final ranking, CRAG placed first, ahead of AmbieGenVAE and OptAngle.

2 EXPERIMENTAL COMPARISON

2.1 Competition Goal

The participants provided test generators that searched for road shapes to challenge the test subjects off the road, i.e., out of bounds (OOB). More specifically, the test subjects ("agents") are two lane-keeping assist systems (LKASs), which were tasked to follow a two-lane road from the starting point to the end point, without leaving the right lane [5].

A test, i.e., a specification of a virtual road, is a sequence of road points (○) on a two-dimensional map (cf. Figure 1). These points are then interpolated as cubic splines, such that the first and last road points are the starting (△) and goal position (□), respectively. The code pipeline is then tasked to initialize the given road in the simulator, execute the test subject, and return the results of the simulation to the test generator. Note that, as in previous years, only *valid* roads are considered. Hence, a road should (i) not self-intersect; (ii) not contain overly-sharp turns; and (iii) fully lie within a fixed-size map. Competitors are subsequently evaluated based on the diversity of their OOB-inducing tests, i.e., valid virtual roads that cause the ego-car controlled by the agent under test to drive off the lane.

2.2 Metrics

For this instance of the competition, we extended the evaluation to include an additional diversity metric. More specifically, we considered feature maps, as in 2023, and a newly introduced clustering-based metric. Both metrics characterize each OOB using both structural features (characteristics of the road) and behavioral features (characteristics of the output of the simulator). Specifically, we selected **Direction Coverage** and **Maximum curvature** as *structural* features, and **Standard Deviation of the Steering Angle** and **Mean Lateral Position** as *behavioral* features. Such features have been empirically asssessed by by Zohdinasab et al. [20]. We computed such features only on a relevant portion of the road input, i.e., 30 meters before and after the OOB takes place.

Feature Map-based Metric. As in our previous edition, we used these features to define a four-dimensional *feature map* where we place road inputs causing a failure according to their features, such that inputs with similar features are close in the feature map. Measuring the coverage of such feature map, allows us to quantify how diverse are the failure-inducing road inputs of a test generator. We refer the reader to our previous report [5] for a detailed description of how the feature map coverage is computed.

Clustering-based Metric. For each test subject, we considered all the features of all the runs of all the tools for clustering. Before clustering, we reduced each vector of features to two dimensions using the UMAP dimensionality reduction technique [1], to obtain better clustering scores. As clustering algorithm we used K-means together with a Silhouette score analysis to choose the optimal number of clusters K^* [6]. For each run i, we computed the coverage of each tool T, i.e., $C_{cov}^{(i)}$, as the number of clusters that are covered by at least one failure-inducing input generated by T, divided by K^*. Regarding the entropy, for each run i, we computed the probability $p_j^{(i)}$ of finding a failure generated by T in cluster c_j ($j = 1, \ldots, K^*$), multiplied by $\log_2(p_j^{(i)})$. The entropy of T for each run, i.e., $C_{ent}^{(i)}$, is the sum of all such quantities normalized by $\log_2(K^*)$. Entropy complements coverage as it quantifies how distributed are the failures across the clusters. The final coverage metric of T for a certain test subject, i.e., C_{cov}, is the average of the coverage metrics across all the runs; this equivalently holds for the final entropy C_{ent}.

Subsequently, we merged the metrics to calculate a weighted average. The feature map-based (*FM*) and clustering-based metrics are weighted equally, and within clustering, coverage C_{cov} and entropy C_{ent} metrics are also weighted equally. Hence, the final diversity score D of a tool is calculated as:

$$Div = FM * 0.5 + C_{\text{cov}} * 0.25 + C_{\text{ent}} * 0.25$$

2.3 Test Subjects

As the previous year [5], we chose two test subjects widely used in the software testing literature: BeamNG.AI, a rule-based driving agent shipped with the BeamNG.tech simulator[1], and Dave-2, a DL-based driving agent. The competitors had access to both test subjects, but we did not disclose our final experimental setup.

2.4 Tools

We evaluated a total of three tools. Below, we briefly describe the main characteristics of each of them. More information can be found in the corresponding reports:

AmbieGenVAE [11] is a test generation approach that leverages optimization with evolutionary search in a latent space of a pre-trained variational autoencoder (VAE). The VAE is trained on a dataset of diverse and valid test scenarios.

CRAG [2] is a generator of road scenarios for autonomous driving testing combining combinatorial testing and search to generate roads in which the car drives off the lane. The former explores high level road configurations, while the latter samples concrete road geometries in these configurations.

OptAngle [3] is a test generator for autonomous vehicles that leverages meta-heuristic search over a road representation based on relative angles between fixed-size road segments. It derives virtual roads by optimizing for road structure validity, failures during test execution, and test case diversity.

Table 1: Setups of the experiments for each subject.

Subject	Map Size (m x m)	Speed Limit (km / h)	Time Budget (h)	OOB Tol.
BeamNG.AI	200 × 200	70	3	0.85
Dave-2	200 × 200	25	3	0.10

2.5 Experimental Procedure

We ran each tool 6 times for BeamNG.AI and Dave-2. The experiment setups are described in Table 1. The BeamNG.AI agent has a speed limit of 70 km/h with an OOB tolerance value of 0.85 and a time budget of 3 h of real time. We executed the Dave-2 agent with the same time budget, a speed limit of 25 km/h, and an OOB tolerance of 0.1. Thus, the Dave-2 agent drives more slowly but a lower tolerance is used to trigger failures. To ensure a fair comparison, we ran each tool the same number of times in each experiment setup. We ran all experiments using version v.0.26.2.0 of BeamNG.tech, on a desktop PC running Microsoft Windows 11 Pro and featuring an eight-core Intel i9-9900K CPU @ 3.60 GHz, 64 GB of RAM, and an Nvidia Quadro RTX 4000 GPU.

[1]BeamNG.tech was kindly provided to all participants by BeamNG.GmbH

Table 2: Statistics of tests generated by each tool and subject. Bold indicates the best values.

Subject	Tool	#TCs	%Val.	OOBs
BeamNG.AI	AmbieGenVAE	342.8	96.9 %	48.8
	CRAG	311.8	97.8 %	41.2
	OptAngle	**366.3**	**98.2 %**	**167.2**
Dave-2	AmbieGenVAE	231.5	96.3 %	2.3
	CRAG	204.2	96.8 %	**16.8**
	OptAngle	**242.5**	**99.8 %**	3.0

Table 3: Final ranking. Coverage values for each tool and subject. Bold indicates the best values.

Tool	BeamNG.AI				Dave-2				*Div*
	FM	C_{cov}	C_{ent}	Div_B	*FM*	C_{cov}	C_{env}	Div_D	
CRAG	0.11	0.24	0.68	0.29	**0.14**	**0.96**	**0.91**	**0.54**	**0.41**
AmbieGenVAE	**0.13**	**0.26**	**0.70**	**0.30**	0.02	0.35	0.31	0.18	0.24
OptAngle	0.05	0.15	0.54	0.20	0.02	0.13	0.10	0.07	0.13

3 RESULTS

3.1 Test Generation Effectiveness and Efficiency

Table 2 reports the number of generated test cases (*#TCs*), the percentage of valid roads (*%Val.*), and the number of failure-inducing inputs (*OOBs*) produced by each tool (*Tool*) for each subject (*Subject*). The reported values are averages over the runs with the same configuration.

We observe that OptAngle produces the highest number of test cases for both test subjects, thus being the most efficient test generator in this competition. It also excels in producing the highest percentage of valid roads. Finally, we also see that for the BeamNG.AI agent, OptAngle also produced the highest absolute number of OOBs. Nonetheless, for Dave-2, CRAG showed a higher number of OOBs than OptAngle. Furthermore, we see that for both test subjects, CRAG produces the lowest number of total test cases, and for BeamNG.AI even the lowest number of OOBs.

3.2 Final Scores

Table 3 reports the final ranking of the tools, computed as the average diversity *Div* of each tool. The table shows that, for BeamNG.AI, AmbieGenVAE produces the best scores, leading in all three diversity measures (*FM*, C_{cov}, C_{ent}); CRAG is just shortly behind (0.29 vs 0.30). OptAngle shows the lowest diversity scores for BeamNG.AI. For Dave-2 we see that CRAG reached very high diversity scores. This can be attributed to the fact, that it also discovered the highest number of OOBs. AmbieGenVAE ranks second, and OptAngle third.

As for the final scores, the dominance in the Dave-2 competition leads CRAG to win with a final score of 0.41, before AmbieGenVAE (Div = 0.24), and OptAngle (Div = 0.13). As shown in Figure 2,

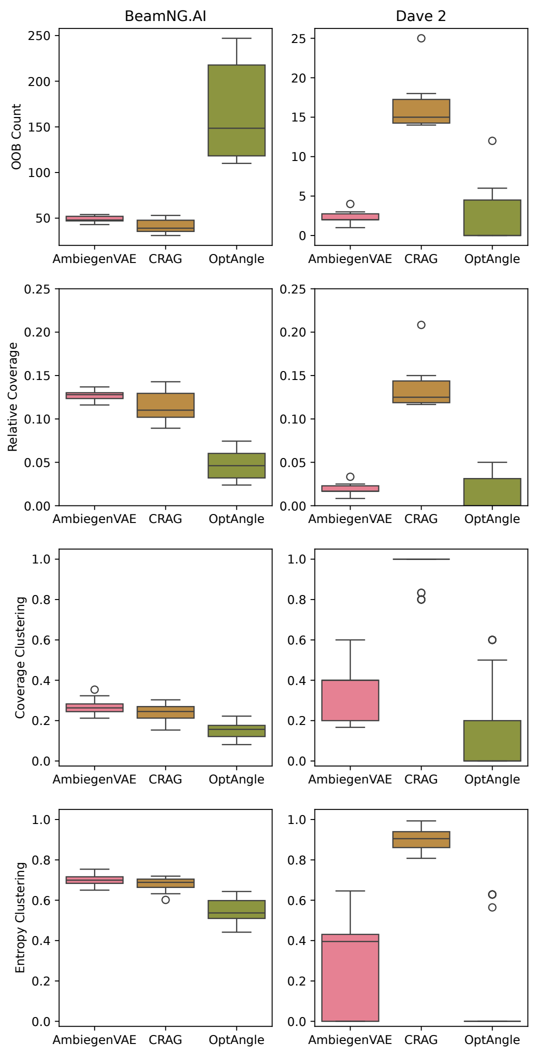

Figure 2: Benchmark results. The box plots report, top-to-bottom, the number of detected failures, the achieved Relative OOB Coverage (FM), the coverage clustering and the entropy clustering in each configuration (BeamNG.AI left, Dave-2 right).

the feature map coverage (second row) agrees with the clustering-based coverage metrics (third and fourth rows) for both test subjects. Figure 3 shows the results of clustering the failures generated by all the tools for the Dave-2 agent. We observe that CRAG (+) covers all the clusters (Run # 5) and such failures are well distributed across clusters (i.e., high entropy). On the other hand, OptAngle (○) covers only the red cluster in Run # 5, while AmbieGenVAE (★) uniformly covers three clusters, resulting in a high entropy.

4 CONCLUSIONS AND FINAL REMARKS

The 2024 SBFT CPS testing tool competition focused on the challenge of evaluating and comparing test generators for autonomous driving. In this fourth edition, three tools competed by testing two

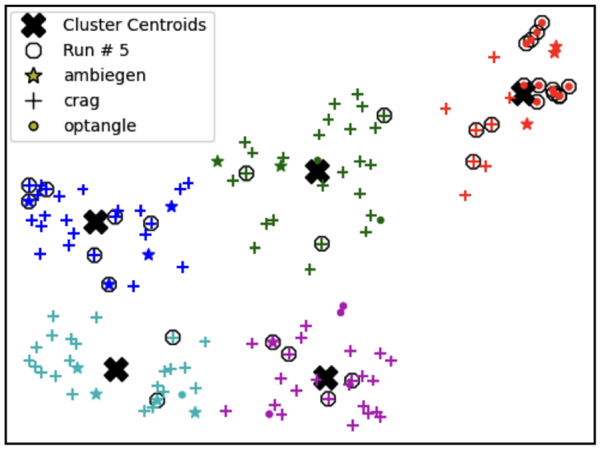

Figure 3: Results of clustering the failures of each tool for the Dave-2 subject. Each color represents a different cluster identified by the cluster centroid. The failures of the last run are highlighted.

test subjects (BeamNG.AI and Dave-2) and generated inputs that triggered failures of both systems. OptAngle generated the highest number of valid roads for both subjects, and also found the highest number of OOBs for BeamNG.AI. AmbieGenVAE and CRAG, on the other hand, produced fewer, but more diverse roads.

Overall, the competition results showed that different search approaches can be highly effective in discovering diverse test suites, balancing the trade-off between exploration and exploitation. In the final ranking, across all settings and evaluation metrics, CRAG ranks first, ahead of AmbieGenVAE and OptAngle, who place second and third, respectively.

ACKNOWLEDGMENTS

We thank the participants for their invaluable contributions and feedback, which support the evolution and maturity of automated testing strategies for CPSs. We thank BeamNG GmbH for providing the driving simulator to the participants. We would also like to thank the IEEE Technical Community on Software Engineering (TCSE) and ACM Special Interest Group on Software Engineering (SIGSOFT) for sponsoring SBFT'24. Matteo Biagiola was partially supported by the H2020 project PRECRIME, funded under the ERC Advanced Grant 2017 Program (ERC Grant Agreement n. 787703).

REFERENCES

[1] Zohreh Aghababaeyan, Manel Abdellatif, Lionel C. Briand, Ramesh S, and Mojtaba Bagherzadeh. 2023. Black-Box Testing of Deep Neural Networks through Test Case Diversity. *IEEE Trans. Software Eng.* 49, 5 (2023), 3182–3204. https://doi.org/10.1109/TSE.2023.3243522

[2] Paolo Arcaini and Ahmet Cetinkaya. 2024. CRAG at the SBFT 2024 Tool Competition – Cyber-Physical Systems Track. In *17th IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT 2024, Lisbon, Portugal, April 14, 2024.*

[3] Aren A. Babikian and Dániel Varró. 2024. OptAngle at the SBFT 2023 Tool Competition - Cyber-Physical Systems Track. In *17th IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT 2024, Lisbon, Portugal, April 14, 2024.*

[4] BeamNG GmbH. 2024. *BeamNG.tech*. https://beamng.tech/

[5] Matteo Biagiola, Stefan Klikovits, Jarkko Peltomäki, and Vincenzo Riccio. 2023. SBFT Tool Competition 2023 - Cyber-Physical Systems Track. In *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2023, Melbourne, Australia, May 14, 2023.* IEEE, 45–48. https://doi.org/10.1109/SBFT59156.2023.00010

[6] Matteo Biagiola and Paolo Tonella. 2023. Testing of Deep Reinforcement Learning Agents with Surrogate Models. *ACM Trans. Softw. Eng. Methodol.* (nov 2023). https://doi.org/10.1145/3631970 Just Accepted.

[7] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. 2016. End to End Learning for Self-Driving Cars. *CoRR* abs/1604.07316 (2016), 1–9.

[8] Markus Borg. 2021. The AIQ Meta-Testbed: Pragmatically Bridging Academic AI Testing and Industrial Q Needs. In *Software Quality: Future Perspectives on Software Engineering Quality - 13th International Conference, SWQD 2021, Vienna, Austria, January 19-21, 2021, Proceedings (Lecture Notes in Business Information Processing, Vol. 404)*, Dietmar Winkler, Stefan Biffl, Daniel Méndez, Manuel Wimmer, and Johannes Bergsmann (Eds.). Springer, 66–77. https://doi.org/10.1007/978-3-030-65854-0_6

[9] Alessio Gambi, Gunel Jahangirova, Vincenzo Riccio, and Fiorella Zampetti. 2022. SBST Tool Competition 2022. In *15th IEEE/ACM International Workshop on Search-Based Software Testing, SBST@ICSE 2022, Pittsburgh, PA, USA, May 9, 2022.* IEEE, 25–32. https://doi.org/10.1145/3526072.3527538

[10] Alessio Gambi, Marc Müller, and Gordon Fraser. 2019. Automatically testing self-driving cars with search-based procedural content generation. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2019, Beijing, China, July 15-19, 2019*, Dongmei Zhang and Anders Møller (Eds.). ACM, 318–328. https://doi.org/10.1145/3293882.3330566

[11] Dmytro Humeniuk and Foutse Khomh. 2024. AmbieGenVAE at the SBFT 2024 Tool Competition — Cyber-Physical Systems Track. In *17th IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT 2024, Lisbon, Portugal, April 14, 2024.*

[12] Gunel Jahangirova, Andrea Stocco, and Paolo Tonella. 2021. Quality Metrics and Oracles for Autonomous Vehicles Testing. In *14th IEEE Conference on Software Testing, Verification and Validation, ICST 2021, Porto de Galinhas, Brazil, April 12-16, 2021.* IEEE, 194–204. https://doi.org/10.1109/ICST49551.2021.00030

[13] Sajad Khatiri, Prasun Saurabh, Timothy Zimmermann, Charith Munasinghe, Christian Birchler, and Sebastiano Panichella. 2024. SBFT Tool Competition 2024 - CPS-UAV Track. In *17th IEEE/ACM International Workshop on Search-Based And Fuzz Testing, SBFT @ ICSE 2024, Lisbon, Portugal, April 14, 2024.*

[14] Sebastiano Panichella, Alessio Gambi, Fiorella Zampetti, and Vincenzo Riccio. 2021. SBST Tool Competition 2021. In *14th IEEE/ACM International Workshop on Search-Based Software Testing, SBST 2021, Madrid, Spain, May 31, 2021.* IEEE, 20–27. https://doi.org/10.1109/SBST52555.2021.00011

[15] Vincenzo Riccio, Matteo Biagiola, Alessio Gambi, Stefan Klikovits, and Jarkko Peltomäki. 2022. SBST CPS Tool Competition Infrastructure. https://github.com/sbft-cps-tool-competition/cps-tool-competition.

[16] Vincenzo Riccio, Gunel Jahangirova, Andrea Stocco, Nargiz Humbatova, Michael Weiss, and Paolo Tonella. 2020. Testing machine learning based systems: A systematic mapping. *Empir. Softw. Eng.* 25, 6 (2020), 5193–5254.

[17] Vincenzo Riccio and Paolo Tonella. 2020. Model-based exploration of the frontier of behaviours for deep learning system testing. In *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, Prem Devanbu, Myra B. Cohen, and Thomas Zimmermann (Eds.). ACM, 876–888. https://doi.org/10.1145/3368089.3409730

[18] Andrea Stocco, Michael Weiss, Marco Calzana, and Paolo Tonella. 2020. Misbehaviour prediction for autonomous driving systems. In *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, Gregg Rothermel and Doo-Hwan Bae (Eds.). ACM, 359–371. https://doi.org/10.1145/3377811.3380353

[19] Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo Tonella. 2021. DeepHyperion: exploring the feature space of deep learning-based systems through illumination search. In *ISSTA '21: 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, Virtual Event, Denmark, July 11-17, 2021*, Cristian Cadar and Xiangyu Zhang (Eds.). ACM, 79–90. https://doi.org/10.1145/3460319.3464811

[20] Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo Tonella. 2023. Efficient and Effective Feature Space Exploration for Testing Deep Learning Systems. *ACM Trans. Softw. Eng. Methodol.* 32, 2 (2023), 49:1–49:38. https://doi.org/10.1145/3544792

[21] Tahereh Zohdinasab, Vincenzo Riccio, and Paolo Tonella. 2023. DeepAtash: Focused Test Generation for Deep Learning Systems. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2023, Seattle, WA, USA, July 17-21, 2023*, René Just and Gordon Fraser (Eds.). ACM, 954–966. https://doi.org/10.1145/3597926.3598109