Extended Abstract of Poster: STARS: Tree-Based Classification and Testing of Feature Combinations in the Automated Robotic Domain

Till Schallau *, Dominik Schmid *, Nick Pawlinorz *,
Stefan Naujokat *, and Falk Howar *†
till.schallau@tu-dortmund.de, dominik.schmid@tu-dortmund.de, nick.pawlinorz@tu-dortmund.de,
stefan.naujokat@tu-dortmund.de, falk.howar@tu-dortmund.de
*TU Dortmund University, 44227 Dortmund, Germany
†Fraunhofer Institute for Software and Systems Engineering, 44147 Dortmund, Germany

***Abstract*—Testing complex systems is crucial for ensuring safety, especially in automated driving, where diverse data sources and variable environments pose challenges. Here, robust safety validation is critical but exhaustive n-way combinatorial testing is impractical due to the vast number of test cases. The STARS framework uses tree-based scenario classifiers to limit feature combinations in a given domain.**

***Index Terms*—feature coverage, scenario classification, n-way combinatorial testing, automated driving**

Automated systems, such as those used in automated driving, aviation, and industrial robotics, are inherently complex and must meet stringent safety standards, particularly when operating in environments shared with humans. Ensuring such systems' safety requires systematic approaches to detect and eliminate software errors.

While combinatorial testing is practical in well-defined systems with predictable input parameters [1], its application becomes challenging in highly automated systems operating in dynamic and complex environments. For these systems, the exploration space quickly becomes unmanageable due to the high cost of real-world testing. This challenge is particularly evident in automated driving, where systems must interact with a mix of human drivers, cyclists, pedestrians, and other automated vehicles. The unpredictable behavior of human road users significantly complicates the testing process and increases the potential test space, rendering exhaustive testing infeasible [2]. Moreover, the complexity of these environments complicates efforts to evaluate safety using statistics, such as fatalities per million miles driven [3], [4].

To address these challenges, safety testing for automated driving systems typically combines *on-road testing*, *simulation testing*, and *scenario-based testing* [5]. While real-world testing accounts for unpredictable factors, simulation-based evaluations enable extensive coverage under controlled conditions. A key challenge in scenario-based testing lies in ensuring that the analyzed scenarios capture diverse and relevant feature combinations.

Our previous work addressed the need to systematically classify real-world and simulated test drives through the introduction of a tree-based scenario classifier (TSC) [6]. A TSC organizes scenario features into a hierarchical structure, significantly reducing the combinatorial explosion of feature combinations by filtering out impossible combinations like red traffic lights on a highway. This classification serves as a critical first step in testing complex systems, providing a structured representation of existing data while identifying gaps in feature combinations that require further exploration or generation. Kitamura et al. [7] proposed a method to transform tree-structured test models with propositional logic constraints into flat structures for combinatorial testing, improving computational efficiency and expressiveness. Unlike their approach, our framework operates directly on hierarchical scenario classifications, eliminating the need for transformations. For this process, we present the STARS framework[1], which integrates the tree-based approach to classify feature combinations and evaluate their coverage.

The STARS Framework

The STARS framework provides a post-hoc evaluation framework for recordings of automated robotic systems [8]. Recordings are divided into segments (e.g., based on map regions), with each segment capturing the timed states of the robotic system and its surroundings. The framework then classifies segments as scenarios based on a formal definition of the operational design domain, while simultaneously monitoring compliance with safety requirements and the system's intended functionality. To fully utilize the STARS framework, users must provide three key inputs for their specific use case:

Domain Data: Users must supply domain data in an arbitrary but time-structured format. This data is segmented into distinct parts for classification. The type of data depends on the use case, ranging from abstract data classes to raw sensor outputs. For example, the automotive domain relies heavily on camera and radar data, whereas robots in smart factories exchange networked data from various sensors. The STARS framework processes raw domain data by merging it into a coherent data structure and segmenting it into semantic segments.

[1]https://github.com/tudo-aqua/stars

979-8-3315-3467-7/25 © 2025 IEEE

ICST Workshops 2025, Naples, Italy
IWCT 2025

Domain Knowledge: The operational environment and relevant scenario features must be formalized. Scenarios are modeled and identified in segments of recorded data using features specified in temporal logics, such as Linear Temporal Logic (LTL), Metric Temporal Logic (MTL), and Counting Metric First-Order Temporal Binding Logic (CMFTBL) [6]. The TSC then represents features and their sub-features as nodes in a hierarchical tree structure, reflecting the taxonomy and semantics of features within the system's operational design domain (ODD). The tree enforces that certain feature combinations are invalid and that some features are dependent on the presence of others. This hierarchical organization restricts the set of classifiable scenarios to only feasible feature combinations, significantly reducing the search space compared to exhaustive n-way combinatorial testing.

Requirements: Users may specify the requirements to be tested, such as specifications, regulations, or other criteria. These are represented as monitor definitions. Depending on the use case, these requirements may come from legal texts, government regulations or industry standards.

Using the TSC, STARS classifies the data segments into scenarios by top-down determining which node formulas hold. If a formula evaluates to *false*, the evaluation of all its sub-features can be omitted, since parent nodes are always required preconditions. A key insight of this approach is that specific feature combinations, classified as scenarios, can act as triggering conditions for system failures. By systematically linking scenarios to observed failures, we gain valuable insights into the root causes of these failures, enabling targeted interventions to improve system safety. By classifying scenarios, STARS also tracks the coverage of observed feature combinations and provides insights into their distributions.

Secondly, all requirements, formalized as monitors, are evaluated for each segment, producing a verdict for each. If a monitor fails for a given segment, the failure is recorded and linked to the corresponding scenario. This linkage helps identify the cause of the monitor failure based on the associated scenario. By combining these results, correlations between specific features in the failed monitors can be identified to trace the triggering condition that caused the system failure or functional insufficiency analogous to the statistical evaluations used in n-way combinatorial testing.

Evaluation

We contrast the hierarchical structure of the tree-based scenario classifier with n-way testing of scenario features. For this purpose, we utilize previous analysis results [6] and compare them to n-way testing of the features[2].

The evaluation of both approaches, the hierarchical structure of the TSC and the n-way projection, yields identical observed feature combinations. However, the number of *possible* feature combinations is significantly higher without the structural information and bounds, given by the TSC. The number of combinations for the n-way approach can be calculated as 2^n, since every feature combination must be tested separately. To test all possible feature combinations, n has to be selected as $|features|$ for a strong safety argument about the system under test and to make it comparable with the TSC approach. The calculation for the hierarchical TSC is more complex and depends of the bounds and the fan-out of each node. The exact formula is described in [6]. In our experiments, involving 34 features, the n-way approach yielded a possible scenario count of $1.7 \cdot 10^{10}$ while the TSC, using the structural information of the ODD, reduces the number to $5,040$.

[2]https://github.com/tudo-aqua/stars-combinatorial-testing

Conclusion

In this paper we demonstrated that a hierarchical approach for feature combination testing outperforms n-way combinatorial testing in addressing the complexity of feature interactions in automated systems. By leveraging the tree-based feature combination approach, that enforces constraints and reduces infeasible combinations,TSCs achieve higher feature combination coverage by maintaining a manageable test space.

For future work, we plan to extend our approach by generating tailored scenarios in simulation environments based on missing feature combinations identified during coverage analysis [9]. This will enable testing under unexamined conditions and facilitate simulation- or hardware-in-the-loop evaluations. Additionally, we intend to generate further test cases focusing on feature combinations where monitors have failed, as these failures indicate specific malfunctions within the system, to further investigate their source.

References

[1] R. C. Bryce, Y. Lei, D. R. Kuhn, and R. Kacker, "Combinatorial testing," in *Handbook of Research on Software Engineering and Productivity Technologies: Implications of Globalization*. IGI Global Scientific Publishing, 2010, doi: 10.4018/978-1-60566-731-7.ch014.

[2] J. Wang, L. Su, S. Han, D. Song, and F. Miao, "Towards safe autonomy in hybrid traffic: Detecting unpredictable abnormal behaviors of human drivers via information sharing," *ACM Transactions on Cyber-Physical Systems*, 2024, doi: 10.1145/3616398.

[3] P. Junietz, W. Wachenfeld, K. Klonecki, and H. Winner, "Evaluation of different approaches to address safety validation of automated driving," in *ITSC*. IEEE, 2018, doi: 10.1109/ITSC.2018.8569959.

[4] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transp. Res. A: Policy Pract.*, 2016, doi: 10.1016/j.tra.2016.09.010.

[5] F. Khan, M. Falco, H. Anwar, and D. Pfahl, "Safety testing of automated driving systems: A literature review," *IEEE Access*, 2023, doi: 10.1109/ACCESS.2023.3327918.

[6] T. Schallau, S. Naujokat, F. Kullmann, and F. Howar, "Tree-based scenario classification," in *NASA Formal Methods*, 2024, doi: 10.1007/978-3-031-60698-4_15.

[7] T. Kitamura, A. Yamada, G. Hatayama, C. Artho, E.-H. Choi, N. T. B. Do, Y. Oiwa, and S. Sakuragi, "Combinatorial testing for tree-structured test models with constraints," in *QRS*, 2015, doi: 10.1109/QRS.2015.29.

[8] T. Schallau, D. Mäckel, S. Naujokat, and F. Howar, "STARS: A tool for measuring scenario coverage when testing autonomous robotic systems," in *EDCC 2024 Workshops*. Springer, 2024, doi: 10.1007/978-3-031-56776-6_6.

[9] T. Schallau and S. Naujokat, "Validating behavioral requirements, conditions, and rules of autonomous systems with scenario-based testing," *Electron. Commun. EASST*, 2023, doi: 10.14279/tuj.eceasst.82.1222.