A systematic literature review on counterexample explanation – Summary

Arut Prakash Kaleeswaran[1], Arne Nordmann[1], Thomas Vogel[2], Lars Grunske[2]

Abstract: In this extended abstract, we summarize our systematic literature review on counterexample explanation published in the journal Information and Software Technology (IST) in 2022 [Ka22].

Keywords: Model checking; Counterexample explanation; Safety

Summary. In order to assure the safety of the systems, automotive systems are often created in accordance with standards like ISO 26262. These standards use time-consuming and error-prone manual safety analysis techniques, such as Failure Mode and Effect Analysis, Fault Tree Analysis, and Hazard and Operability, to undertake safety analysis. To overcome these challenges, formal methods is a potential option, e.g., model checking, an automated verification method. Model checkers take the system model and specifications as input and verify whether the specifications are satisfied by the system model. If not, then a counterexample is provided by the model checker that illustrates the violation with the execution path and system states. Nevertheless, comprehending the counterexample is challenging because it is cryptic, relevant erroneous states and variables are not highlighted, and debugging is performed manually. These challenges call for a method *counterexample explanation* to ease the error comprehension. Thus, the systematic literature review (SLR) is performed on counterexample explanation to provide an overview on the state of the art, mainly focusing on the types of counterexample representation, the methods to transform or optimize a counterexample, to provide an explanation, influence of the input system and requirement on counterexample explanation, and the different domains and applications to evaluate approaches to counterexample explanation. The SLR has been conducted as part of a research project between BOSCH and HU Berlin on making model-checking results in contract-based design of safety-critical systems easier to understand [Ka20].

The survey is performed by collecting answers for the eight specific research questions from 116 primary studies. By surveying 116 primary studies, four counterexample representations types are identified: graphical, textual, tabular, and trace representations. Among these, graphical and trace formats are predominantly used (89 of 116 primary studies, 77%) to represent a counterexample. For instance, the counterexample is presented in a SysML or UML diagram. A trace representation is a modified form of a counterexample with the addition or removal of (sub)-traces. Trace representation is further categorized into three types based

[1] Bosch Corporate Sector Research, Renningen, Germany. ArutPrakash.Kaleeswaran@bosch.com
[2] Humboldt-Universität zu Berlin, Software Enginering Group, Unter den Linden 6, 10099 Berlin, Germany. {thomas.vogel, grunske}@informatik.hu-berlin.de

on processing a counterexample: minimized counterexample, multiple counterexamples, and witness traces (traces that satisfy the failed specification, which is contrary to the counterexample). Among 54 primary studies that process the counterexample, majority (38, 70%) use minimized counterexample approach. Providing additional information along with the counterexample explanation improves error comprehension, for example, highlighting erroneous state transitions in the state-machine. However, only 8 of 116 primary studies are found to provide additional information along with the counterexample explanation.

In this study, we also investigated whether counterexample explanations are represented and explained with the user given input domain or those different from the input domain. Among 81 primary studies (excluding 31 studies that use trace representation), 60 studies represent the counterexample different from the given input domain, and only 21 studies represent the counterexample in the given input domain. Furthermore, we have also collected verification tools and frameworks used for verification and explaining counterexamples, as well as temporal logic and specification properties used to express system specifications. Mostly used verification tools are NuSMV/nuXmv, PRISM, and SPIN, frameworks are ASSERT, DiPro, and MODCHK, temporal logics are LTL and CTL, and specification properties are safety properties. Finally, focusing on the evaluation methods, evaluation aspects, and applications used for the evaluation, the majority employs use-case based evaluation method focusing on effectiveness by using industrial applications.

This survey reveals several open points and future directions to enhance the counterexample explanation approach. One of the main findings is still the counterexample explanation needs improvement for non-experts in formal methods. We found many frameworks for counterexample explanation and a considerable number of these are open-source. Therefore, it seems advisable to build upon existing frameworks and improve them. To improve the effectiveness and usability of counterexample explanations, user studies need to be performed where one could gain insights on the degree of error comprehension. Such user studies will provide evidence for the effectiveness and usability of such approaches from a user's point of view and hence can guide future research on counterexample explanation.

Data Availability. We published an online appendix [Ka21] on Zenodo (`https://doi.org/10.5281/zenodo.5679227`) that lists the data items to be extracted and a bibliography of the primary studies, and tabulates primary studies and extracted data for the quantitative items.

Bibliography

[Ka20] Kaleeswaran, Arut Prakash; Nordmann, Arne; Vogel, Thomas; Grunske, Lars: Counterexample Interpretation for Contract-Based Design. In: 7th Intl. Symposium on Model-Based Safety and Assessment, IMBSA. volume 12297 of LNCS. Springer, pp. 99–114, 2020.

[Ka21] Kaleeswaran, Arut Prakash; Nordmann, Arne; Vogel, Thomas; Grunske, Lars: Appendix of the paper: A Systematic Literature Review on Counterexample Explanation. 2021.

[Ka22] Kaleeswaran, Arut Prakash; Nordmann, Arne; Vogel, Thomas; Grunske, Lars: A systematic literature review on counterexample explanation. Information and Software Technology, 145:106800, 2022.