Assertion mining and modeling of Cyber-physical Systems

PhD Candidate: Daniele Nicoletti, Advisor: Graziano Pravadelli
University of Verona, Department of Engineering for Innovation Medicine
daniele.nicoletti@univr.it

***Index Terms*—Assertion mining, Hybrid systems, Specification mining, Signal Temporal Logic, STL, Cyber-physical systems, simulation, assertion-based verification**

I. Introduction

In the Electronic Design Automation (EDA) context, temporal logics are widely adopted to formalize the designer's intents in the form of assertions; then an automatic procedure, like model checking or dynamic assertion-based verification (ABV), is used to prove the assertions are actually satisfied by the design implementation. As the manual definition of assertions is an error-prone and time-consuming task, some tools have been developed in the past decade to automatically mine assertions from the actual implementation of the Design Under Verification (DUV) [1], [2], [3], [4], [5]. Following this approach, the mined assertions are then compared against the initial (informal) specifications to verify if the expected behaviors have been implemented in the DUV. Furthermore, by analyzing the mined assertions, the verification engineers can also discover the presence of unexpected behaviors caused by either design errors or malicious code. These Assertion Mining (AM) techniques have been mainly used in the digital domain to generate Linear Time Logic assertions starting from RTL implementation. However, with the ever-increasing diffusion of complex and interconnected cyber-physical systems (e.g., in smart industry, smart cities, automotive, healthcare, etc.) that mix physical and digital parts, mining assertions only on the digital side is no longer sufficient. Therefore, there is a need to support verification engineers of cyber-physical designs with new AM techniques that can mine specifications for hybrid systems, i.e., systems that exhibit both discrete and continuous behaviours [6]. Moreover, CPSs exhibit a level of complexity that demands sophisticated modeling techniques to capture their behavior accurately and ensure their correctness. Such complexity arises from the dynamic nature of CPSs, where discrete computational elements interact with continuous physical processes. Furthermore, identifying and rectifying errors across both digital and physical domains requires specialized tools and expertise; as a result, debugging CPS models presents a challenging task.

A few AM methodologies for the hybrid world are present at the state of the art. They generally adopt the Signal Temporal Logic (STL) [7] to automatically mine assertions that represent behaviors formed by merging time-bounded temporal operators (in a dense-time environment) that predicate over real-valued signals. The already-present methodologies present limitations regarding the mining templates used or in terms of computational cost to obtain the assertions. To fill this gap, I propose an automatic assertions generation tool called SLAM (Signal temporal Logic Assertion Miner). Then, to exploit SLAM, a modeling tool called CHAOS has been created. CHAOS provides a modeling environment based on the formalism of hybrid systems [8] and offers assertion mining and verification features to the user. Both these tools will be explained further in the following sections.

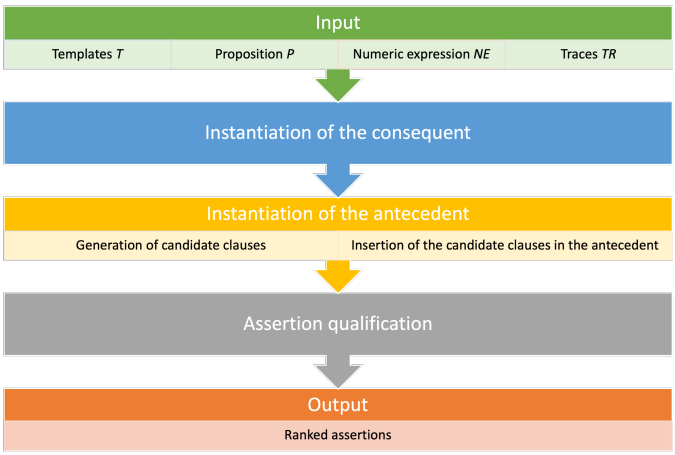

Fig. 1. Overview of the mining procedure.

II. SLAM

As mentioned in the introduction, the objective of SLAM is to infer STL assertions starting from the execution traces of the DUV. Moreover, starting from a set of user-defined hints and the simulation traces of DUV, it is completely agnostic with respect to the design from which the traces were generated. Thus, the DUT source code is not necessary. The user-defined hints involve STL templates, propositions, and ranking metrics that are exploited by the assertion miner to reduce the search space and improve the quality of the generated assertions. This way, the tool supports the work of the verification engineer by including his/her insights in the process of automatically generating assertions. In SLAM the supported templates are in the form $G(antecedent \rightarrow consequent)$.
The methodology is divided into the following three steps, an overview of which is also reported in fig 1:

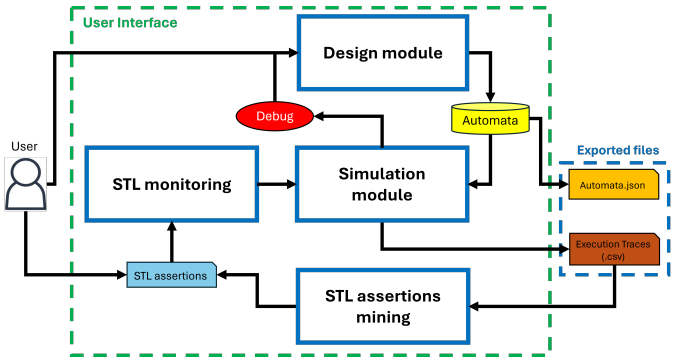

Fig. 2. CHAOS architecture

1) **Instantiation of consequent**: A set of Partially Instantiated Templates (PITs) are generated by instantiating the placeholders included in the *consequent* of the templates belonging to T with the propositions in P. Thus, each resulting PIT corresponds to a different permutation of the propositions that can be instantiated in the placeholders of the *consequent*.
2) **Instantiation of antecedent**: In the second phase, for each PIT generated in phase 1), the tool tries to automatically infer a corresponding *antecedent* using a *Decision Tree* and clustering algorithm to extract the minimal antecedent that makes the whole formula valid on the entire trace.
3) **Assertion qualification**: In the last phase, the mined assertions are filtered and ranked. In this way, assertions that capture irrelevant or trivial behaviours are removed according to interestingness metrics that can be decided by the user. The surviving assertions are then ranked consequently.

SLAM, therefore, stands as a valuable asset in a Verification engineer's toolbox. Providing the user with a powerful tool to improve and speed up the verification process of CPSs.

III. CHAOS

This section details the methodology employed by CHAOS, a tool designed for modeling, simulation, and assertion-based verification of hybrid automata. The tool, whose architecture is shown in fig. 2, is structured into four main modules: Design, Simulation, STL Assertion Mining, and STL Monitoring.

- The Design Module provides a graphical interface for constructing hybrid automata models, enabling users to create complex CPS models without extensive programming knowledge. Users interact with a graphical workspace to define the structure of the hybrid automaton, specifying control modes (states) and control switches (transitions) between these modes. Each switch is governed by jump conditions based on the values of continuous variables. The system is modeled as a collection of parallel automatons, each managing different aspects of the CPS. Once the model is complete, it is exported in JSON format for simulation.
- The Simulation Module facilitates the dynamic analysis of the modeled system, providing insights into system behavior over time and identifying potential design issues. Users specify the total simulation time and frequency, which determine the granularity and duration of the simulation. The module computes the concurrent evolution of all defined automatons, updating control modes based on jump conditions and continuous variables using differential equation solvers. A debugging mode allows users to manually step through each simulation instance, observing variable values and active control modes. The simulation generates execution traces that can be exported for further analysis.
- The STL Assertion Mining module automatically generates STL assertions from execution traces, capturing the behaviors of the CPS model and identifying potential design issues.
- The STL Monitoring module verifies that generated or manually specified STL assertions hold true for the execution traces, facilitating the identification and correction of design errors. Users provide the execution traces and the STL assertions to be checked. If an assertion fails, the tool provides counterexamples, highlighting the time and variable values at the point of failure. Users can re-evaluate assertions after making changes to the model to ensure correctness and identify regressions.

IV. Conclusions

In this paper, I described an assertion generation tool called SLAM and a modeling tool called CHAOS. SLAM presents various advantages w.r.t. the current state-of-the-art approaches in terms of flexibility and ease of use while CHAOS allows to model CPSs in a user-friendly way thanks to the usage of hybrid automatons and a graphical interface while at the same time providing assertion mining and verification features to debug the modeled systems. Future works will focus on refining and extending the SLAM and CHAOS by testing them on real industrial cases.

References

[1] H. Witharana, A. Jayasena, A. Whigham, and P. Mishra, "Automated generation of security assertions for rtl models," *J. Emerg. Technol. Comput. Syst.*, vol. 19, no. 1, jan 2023.
[2] S. Germiniani and G. Pravadelli, "HARM: A hint-based assertion miner," *IEEE Trans. on CAD*, vol. 41, no. 11, pp. 4277–4288, 2022.
[3] A. Danese, N. D. Riva, and G. Pravadelli, "$a-team$: Automatic template-based assertion miner," in *Proc. of ACM/IEEE DAC*, 2017.
[4] S. Vasudevan, D. Sheridan, S. Patel, D. Tcheng, B. Tuohy, and D.Johnson, "Goldmine: automatic assertion generation using data mining and static analysis," in *Proc. of ACM/IEEE DATE*, 2010.
[5] M. R. Heidari Iman, J. Raik, M. Jenihhin, G. Jervan, and T. Ghasempouri, "An automated method for mining high-quality assertion sets," *Microprocessors and Microsystems*, vol. 97, p. 104773, 2023.
[6] M. S. Branicky, *Introduction to Hybrid Systems*. Boston, MA: Birkhäuser Boston, 2005, pp. 91–116.
[7] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Y. Lakhnech and S. Yovine, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 152–166.
[8] T. Henzinger, "The theory of hybrid automata," in *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, 1996, pp. 278–292.