Iterative Model Checking for Safety-Critical Problems in Cyber-Physical Systems

1st Guangyao Chen
ShanghaiTech University
Shanghai, China
chengy2@shanghaitech.edu.cn

2nd Zhihao Jiang
ShanghaiTech University
Shanghai, China
jiangzhih@shanghaitech.edu.cn

***Abstract*—This demonstration presents PyUPPAAL, a Python package designed to facilitate model checking for safety-critical problems in cyber-physical systems (CPS). Through interactive tutorials and case studies, including risk analysis, fault diagnosis, and a real-world electric power system (EPS) case, we showcase how PyUPPAAL can be utilized to enhance the safety and reliability of CPS. Attendees will engage with hands-on coding sessions, gaining practical experience in applying model checking techniques to identify and solve safety-critical issues. The interactive session aims to equip participants with the knowledge and tools necessary to apply PyUPPAAL in their own work, highlighting the advantages of model checking in the development and verification of reliable CPS.**

***Index Terms*—cyber-physical systems, safety-critical systems, risk analysis, fault diagnosis, model checking, python package**

I. Introduction

A. Safety Critical Problems in CPS

Cyber-physical systems (CPS) are integrations of computation, networking, and physical processes, managed and monitored by embedded systems through continuous feedback loops. The safety-critical nature of CPS, such as in automotive systems, medical devices, and power grids, necessitates rigorous validation to prevent failures that could lead to catastrophic outcomes [1]. These problems include:

1) Risk Analysis: Identify the set of ALL possible execution traces T that violate safety requirements φ_0.
2) Fault Identification: Determine whether a fault f has occurred with n observation sequence $o = o_1 o_2 \dots o_n$.
3) Fault Diagnosability: Determine whether a fault f is n-diagnosable in the current system.

Ensuring these systems' safety and reliability is crucial, as failures can significantly impact the environment and health.

B. Model Checking for Safety Critical Problems

Model checking [2] emerges as a powerful formal verification technique, offering a systematic approach to verify whether a model of a given system satisfies a set of specified properties, typically expressed in temporal logic by exploring all possible states of a system. This ensures that the system behaves as intended under all circumstances, thus significantly reducing the risk of undetected errors and enhancing the system's reliability and safety. By integrating model checking into the development process of CPS, we will demonstrate how it can be employed to solve safety-critical problems.

II. Demo Description

This demonstration introduces PyUPPAAL[1], a Python package that seamlessly interfaces with UPPAAL, a leading tool for model checking in real-time systems. Distinguished from other UPPAAL extensions like `verifyta`, `libutap`, and `encaldwell/pyuppaal`, PyUPPAAL stands out by offering comprehensive built-in functions specifically tailored for addressing fundamental research challenges. It is particularly user-friendly, thanks to its thorough documentation and robust testing framework. Our aim is to highlight PyUPPAAL's capacity to effectively tackle safety-critical issues within CPS, blending theoretical depth with practical utility.

Demos are provided for associated problems including risk analysis, fault identification, and fault diagnosability. An Electric Power System case study is also provided to see how PyUPPAAL contributes real-world problems. You can also get access to these demos in PyUPPAAL's documentation[2].

A. Demo1: Risk Analysis for Autonomous Driving

Fig. 1 shows an autonomous car algorithm verification case study modified from [4]. Using PyUPPAAL's iterative model checking, we identify all potential traces that could lead to a collision, demonstrating PyUPPAAL's superiority over traditional UPPAAL tools like verifyta by proving certain crash scenarios are unreachable with existing methods.

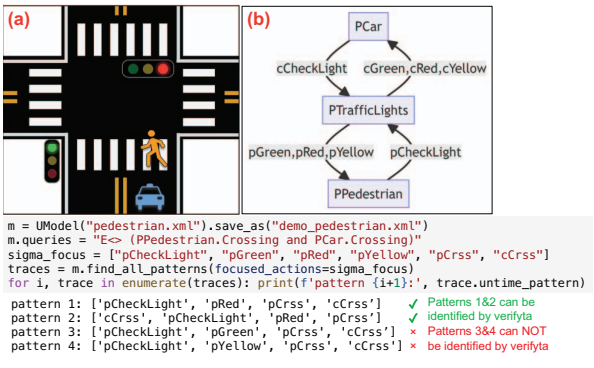

Fig. 1. Demo1: Risk Analysis for Autonomous Driving.

[1]GitHub: https://github.com/jack0chan/pyuppaal.
[2]ReadtheDocs: https://pyuppaal.readthedocs.io.

979-8-3503-6927-4/24/$31.00 ©2024 IEEE
DOI 10.1109/ICCPS61052.2024.00032

B. Demo2: Fault Diagnosability and Identification

Fig. 2 showcases PyUPPAAL's capability determine the diagnosability of a fault f in two distinct models [5], utilizing the `fault_diagnosability()` function. PyUPPAAL explains the reason why Model_A is not 3-diagnosable by showing the execution trace $caba$. It further demonstrates that, although fault f in Model_A is not diagnosable, it can still be identified through a specific observation sequence $o = aaa$, with built-in `fault_identification()` method.

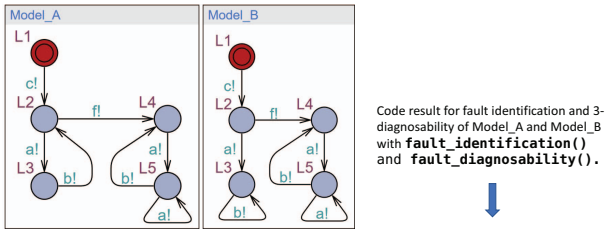

Fig. 2. Demo2: Fault diagnosability and fault identification of the fault f in Model_A and Model_B. Codes are omitted due to the page limit.

C. Demo3: Case Study of Real-World Electric Power Systems

Fig. 3 delves into a real-world Electric Power System (EPS) scenario [6]. It details how PyUPPAAL can be employed to analyze and identify faults within an EPS, based on observations from voltmeters, especially when the system malfunctions. It also evaluates the efficacy of implementing improvement strategies, like adding additional sensors, to enhance fault identification capabilities, concluding that the introduction of sensor $v3$ is more advantageous than $v4$ for fault identification.

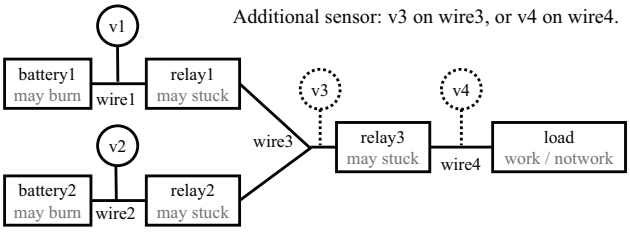

Fig. 3. Demo3: Analyze safety-critical problems in Electric Power Systems.

III. User Experience

Participants will engage with PyUPPAAL through a series of structured demos, each tailored to highlight different aspects of iterative model checking workflow applied to safety-critical problems in cyber-physical systems. This hands-on approach is intended to provide a deep understanding of PyUPPAAL's capabilities and its application in real-world scenarios.

1) **Guided Tutorials:** Attendees will begin their experience with guided tutorials, offering a step-by-step walkthrough of PyUPPAAL's features. This includes creating models, defining properties for verification, running model checks, and interpreting results. These tutorials are designed to cater to all skill levels, ensuring a foundational understanding of model checking concepts and how they can be applied using PyUPPAAL.
2) **Case Studies:** Through the three main demos (Autonomous Driving, Fault Diagnosis, and a Real-World Electric Power System Case), participants will see firsthand the practical applications of PyUPPAAL in diverse CPS domains. These case studies are selected for their relevance to current challenges in the field, providing insights into how model checking can enhance system safety and reliability.
3) **Interactive Problem Solving:** Participants are encouraged to bring their own safety-critical problems. This interactive problem-solving session will enable attendees to use PyUPPAAL to model and analyze their own cases under the guidance of experts. It serves as an excellent opportunity for participants to understand the adaptability of PyUPPAAL to various domains and challenges.

The user experience at this demo aims to leave participants with a solid understanding of model checking as applied to CPS, equipped with the knowledge and skills to implement PyUPPAAL in their own projects. By combining theoretical learning with practical application, the demo seeks to empower attendees with the tools necessary to enhance the safety and reliability of CPS in their domains.

IV. Demo Requirements

In this demo, the following arrangements are required:

1) Display: A large screen (40 inches or larger) for clear viewing of demonstrations.
2) Computers: At least one computer equipped with Python for hands-on coding sessions.
3) Poster Space: A space to display a poster summarizing PyUPPAAL's features and applications.

References

[1] Gunes, Volkan, et al. "A survey on concepts, applications, and challenges in cyber-physical systems." KSII Trans. Internet Inf. Syst. 8.12 (2014): 4242-4268.
[2] Baier, Christel, and Joost-Pieter Katoen. Principles of model checking. MIT press, 2008.
[3] Behrmann, Gerd, Alexandre David, and Kim G. Larsen. "A tutorial on uppaal." Formal methods for the design of real-time systems (2004): 200-236.
[4] Chen, Guangyao, and Zhihao Jiang. "Environment modeling during model checking of cyberphysical systems." Computer 54.9 (2021): 49-58.
[5] Zaytoon, Janan, and Stéphane Lafortune. "Overview of fault diagnosis methods for discrete event systems." Annual Reviews in Control 37.2 (2013): 308-320.
[6] Chen, Guangyao, et al. "Model checking-based decision support system for fault management: A comprehensive framework and application in electric power systems." Expert Systems with Applications, vol. 247, 2024, p. 123371.