MOSAIC: Model-based Safety Analysis for AI-enabled Cyber Physical System

XUAN XIE, University of Alberta, Canada

JIAYANG SONG, University of Alberta, Canada

ZHEHUA ZHOU, University of Alberta, Canada

FUYUAN ZHANG, The University of Tokyo, Japan

LEI MA, The University of Tokyo, Japan, and University of Alberta, Canada

Cyber-physical systems (CPSs) are now widely deployed in many industrial domains, e.g., manufacturing and autonomous vehicles. To further enhance the applicability of CPSs, there comes a recent trend from both academia and industry to utilize learning-based artificial intelligence (AI) controllers for the system control process, resulting in an emerging class of AI-enabled cyber-physical systems (AI-CPSs). Although such AI-CPSs could achieve obvious performance enhancement, due to the random exploration nature and lack of systematic explanations, such AI-based techniques also bring uncertainties and safety risks to the controlled system, posing an urgent need for effective safety analysis techniques for AI-CPSs. Hence, in this work, we propose Mosaic, a model-based safety analysis framework for AI-CPSs. Mosaic first constructs a Markov decision process (MDP) model as an abstract model of the AI-CPS, which tries to characterize the behaviors of the system. Then, based on the derived model, safety analysis is designed in two aspects: online safety monitoring and offline model-guided falsification. The usefulness of Mosaic is evaluated on four industry-level AI-CPSs, the results of which demonstrate that Mosaic is effective in providing safety monitoring to AI-CPSs and able to outperform the state-of-the-art falsification techniques, providing the basis for advanced safety analysis of AI-CPSs. To enable further research along this direction to build better AI-enabled CPS, we made all of the code and experimental results data publicly available at https://sites.google.com/view/ai-cps-mosaic.

CCS Concepts: • **Software and its engineering → Software reliability**; • **Computing methodologies → Artiicial intelligence**; • **Computer systems organization → Embedded and cyber-physical systems**.

Additional Key Words and Phrases: Cyber-physical systems, AI controllers, Safety analysis, Safety monitoring, Falsification

ACM Reference Format:

Xuan Xie, Jiayang Song, Zhehua Zhou, Fuyuan Zhang, and Lei Ma. 2018. MOSAIC: Model-based Safety Analysis for AI-enabled Cyber Physical System. 1, 1 (December 2018), 30 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Cyber-physical systems (CPSs) are commonly and broadly defined as systems that integrate digital computational components and physical components (also referred to as plants). Benefiting from modern advances in digitization over the past decade, nowadays CPSs have been widely deployed and have emerged as fundamental pillars within crucial industrial and social infrastructures across various domains, e.g., industrial manufacturing systems [62], robotic

Authors' addresses: Xuan Xie, University of Alberta, Canada, xxie9@ualberta.ca; Jiayang Song, University of Alberta, Canada, jiayan13@ualberta.ca; Zhehua Zhou, University of Alberta, Canada, zhehua1@ualberta.ca; Fuyuan Zhang, The University of Tokyo, Japan, fuyuanzhang@163.com; Lei Ma, The University of Tokyo, Japan, and University of Alberta, Canada, ma.lei@acm.org.

systems [26], computerized vehicle and aircraft controls [42, 47, 63], smart grids [78], medical devices [57], etc. CPSs leverage the ability to establish communication between digital devices and physical processes, thereby enabling the accomplishment of complex tasks. However, traditional CPS controllers usually rely on accurate modelling of the system behavior, in which complicated tasks are often hard to achieve [18]. Moreover, designing a controller based on a specific model limits its capability of performing well in different environments and tasks, which further diminishes its applicability and generalizability [35]. Hence, there remains a research challenge in realizing efficient and flexible control processes for CPSs.

Inspired by the impressive performance of artificial intelligence (AI) techniques in solving intricate real-world problems, e.g., image recognition [5, 44] and decision-making [58, 68], there is a recent trend of exploring the potential of employing AI-based approaches to overcome the limitations of traditional CPS controllers. This has led to the emergence of a new class of CPSs, referred to as AI-enabled CPSs (AI-CPSs) [76, 97]. AI-CPSs can learn from data, deal with complex system structures, adapt to dynamic environments, and make intelligent decisions, leading to improved overall system performance, including effectiveness, efficiency, flexibility, and adaptability [12, 15, 50, 72, 73]. Some practical examples of the advantages of AI-CPSs over traditional controllers are: (1) they can achieve end-to-end control using raw inputs, such as steering commands based on camera images in autonomous driving systems [15]; (2) AI techniques can improve storage efficiency in the case of memory limitation, e.g., in aircraft collision avoidance systems [50]; (3) AI-CPSs excel in complex control tasks, e.g., manipulating deformable or soft objects [61].

However, due to the difficulties in explaining the AI controllers' behaviors, the employment imposes an obvious potential drawback and risk on AI-CPSs, i.e., the lack of promising safety analysis [76, 97]. This raises concerns for their wider adoption, especially in safety-critical domains. Therefore, there is an urgent need for a general technique and framework for the safety analysis of AI-CPSs, which can serve as the foundation for building safe and trustworthy AI-CPSs.

The traditional and *de facto* approach to analyzing the safety of CPSs often relies on expert experience and a transparent understanding of the system behavior [59]. Various safety analysis methods have been developed to tackle the safety challenges of CPSs, such as fault tree analysis [24], failure modes and effects analysis [34], model-based engineering [8], and verified controller executables [14]. However, these techniques are, in general, not applicable to AI-CPSs due to reasons such as the low explainability of AI components, the limited testing samples, and the lack of a comprehensive model to describe system characteristics. Moreover, existing falsification techniques designed for traditional CPSs are also still limited in falsifying and detecting safety issues in AI-CPSs [76]. Therefore, new safety analysis methods that are able to reveal the behavior of AI components are urgently needed to realize a safe and reliable deployment of AI-CPSs.

In this paper, we propose Mosaic, a model-based safety analysis framework for AI-CPSs. It consists of three key components: *data collection*, *model abstraction*, and *safety analysis* (see Fig. 1). The central idea is that, through using simulation data that represents the safety properties of the AI-CPS under analysis, we first construct a Markov decision process (MDP) [70] model as an abstract model of the system. Such an abstract model captures the essential characteristics of the system with reduced state, input, and output spaces, enabling efficient safety analysis. Then, based on the constructed abstract MDP model, we propose safety analysis techniques for AI-CPSs from two directions: (1) online safety monitoring by utilizing probabilistic model checking (PMC) [56]; and (2) offline model-guided falsification. A more detailed overview of Mosaic is presented in Section 2.2. To demonstrate the usefulness of our proposed framework, we perform an extensive evaluation of diverse and representative AI-CPSs across various domains. The

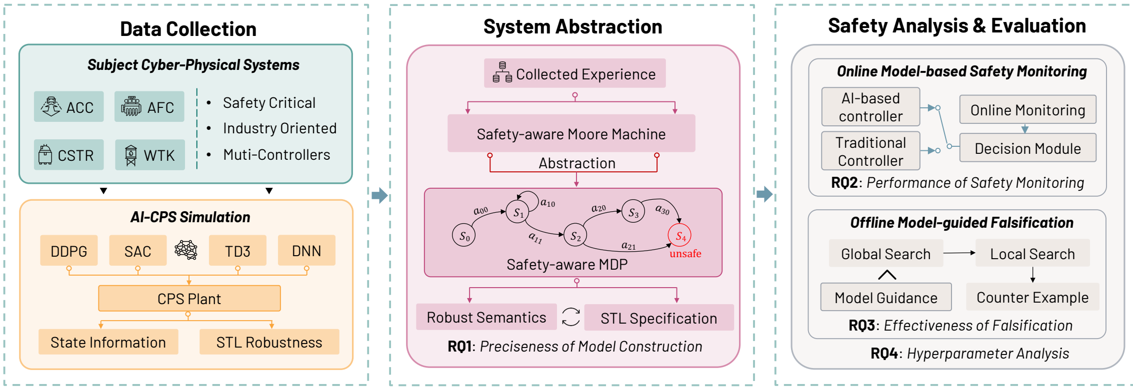


Fig. 1. Overview of MOSAIC: a model-based safety analysis framework for AI-CPS.

results demonstrate that MOSAIC is effective and efficient in providing safety analysis to AI-CPSs, serving as a foundation for the development of advanced AI-CPSs.

In summary, the key contributions of this work are as follows:

- We propose a novel model-based safety analysis framework for AI-CPSs that leverages model abstraction techniques. The constructed abstract model is facilitated by its probabilistic nature and safety awareness, which represents the safety-related behavior of the system.
- Based on the abstract model, we propose two techniques for safety analysis of AI-CPS, i.e., (1) online safety monitoring and (2) offline abstract model-guided falsification.
- The model-based safety monitoring technique provides online safety advice to mitigate potential risks and hazards in AI-CPSs, ensuring safer operation and behavior.
- Our offline model-guided falsification technique, which combines model-guided exploration and optimization-based search, performs offline analysis of an AI-CPS to discover cases that have the potential to violate the specified safety specifications.
- The effectiveness and usefulness of MOSAIC are demonstrated by our in-depth evaluation and analysis from four perspectives. (1) We show that our constructed abstract model is precise in terms of state labelling, which is essential for safety analysis. (2) We also illustrate that the online *safety monitoring* can increase the safety of the system while maintaining a satisfactory level of performance, which indicates its effectiveness. (3) For *falsification*, we demonstrate that MOSAIC outperforms three state-of-the-art falsification techniques with different optimization algorithms. (4) Additionally, we perform hyper-parameter analysis to show that the selection of the hyper-parameters influences the size of the model, the preciseness, the verification time, and the falsification success rate.

To the best of our knowledge, this paper represents an early contribution with a specific focus on the safety analysis of AI-CPSs. As the adoption of AI in CPSs continues to grow, there is a need to address the potential risks and hazards associated with this integration, including issues such as uncertainty and behavior interpretability. These concerns can hinder the widespread adoption of AI-CPSs, emphasizing the importance of developing a safety analysis framework. Our proposed framework serves two key purposes: firstly, it provides a technique and tool for analyzing and detecting safety issues in AI-CPSs. Secondly, it lays the foundation for further research in this critical area, enabling the design of advanced safety and trustworthiness techniques that can facilitate the broader adoption of AI-CPSs. To encourage

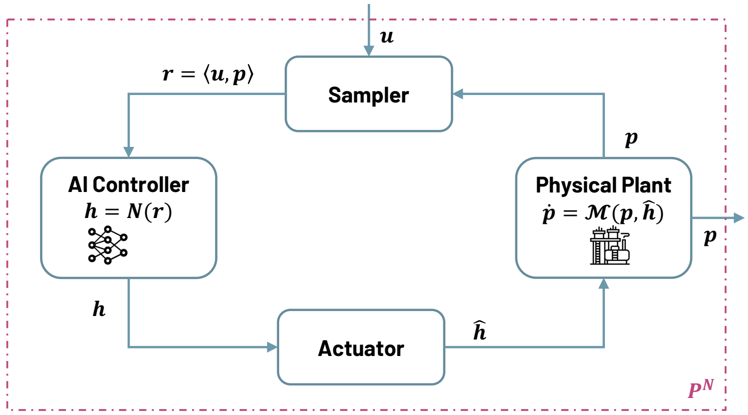


Fig. 2. The common workflow of AI-CPS.

and support further research studies in this direction, we have made all of our source code, benchmarks, and detailed evaluation results publicly available at https://sites.google.com/view/ai-cps-mosaic.

The Contributions to the Software Engineering Field. CPSs are complex systems that encompass various fields, including mechanical engineering, control, and software engineering (SE) [39, 60, 66]. The role of SE in CPSs is pivotal, as it orchestrates the integration of various components during the system's creation and implementation, allowing for a synergy of components to achieve the system's intended features. In this work, our focus is to provide a quality assurance framework for CPSs, which is one of the predominant tasks in SE [20, 40, 46, 101], and improve the safety and resilience, by model-based online safety monitoring and offline model-guided falsification.

2 OVERVIEW

In this section, we first present a brief introduction to the structure of AI-CPS considered in this work. Then, an overview of the proposed safety analysis framework is given, together with the high-level research questions (RQs) that we would investigate.

2.1 AI-CPS

Due to its various advantages, e.g., enhanced storage efficiency and end-to-end control capabilities [15, 50], AI-CPS has captivated the interest of numerous researchers. Typically, an AI-CPS, denoted as P^{N}, integrates a physical plant with a AI controller. By facilitating communication between a powerful AI-based controller and the plant, AI-CPS achieves remarkable performance, making it highly competitive in various domains. As depicted in Fig. 2, it consists of four main components: a *sampler*, a *plant* P, an *actuator*, and an AI *controller* N. The sampler combines the external signal u from the environment and the state of the plant p as its output data $r = \langle u, p \rangle$. The AI controller then takes the sample data r as input and determines an action h, which is sent to the actuator to produce a continuous signal $\hat{h}$ for controlling the plant. Based on the control commands and signals it receives, the plant undergoes physical processes, such as interactions with the environment, and evolves to new states, gradually accomplishing the designated task. From a system perspective, the AI-CPS P^{N} can be considered as a black box function that describes the physical process and maps a system input u (from the environment) to a system output $\mathsf{P}^{\mathsf{N}}(u)$.

It is worth mentioning that deriving learning-based AI controllers typically involves two main categories of state-of-the-art techniques: *supervised learning* and *deep reinforcement learning (DRL)*. In supervised learning, training data is collected from traditional control-theoretical controllers, e.g., proportional-integral-derivative (PID) [48] controllers and model predictive controllers (MPC) [17]. The input to the AI controllers, e.g., Deep Neural Network (DNN), is the collected system states, while the ground-truth labels correspond to the outputs of the traditional controllers. The AI controller learns from the collected data, and attempts to approximate the performance of the traditional control-theoretical controller. In the case of DRL, the AI controller is trained through direct interaction with the environment, aiming to learn an optimal policy that maximizes a predefined reward function based on observed system states. Various DRL algorithms exist, including deep deterministic policy gradient (DDPG), soft actor-critic (SAC), and twin-delayed deep deterministic policy gradient (TD3). For more details about DRL, we refer readers to [6].

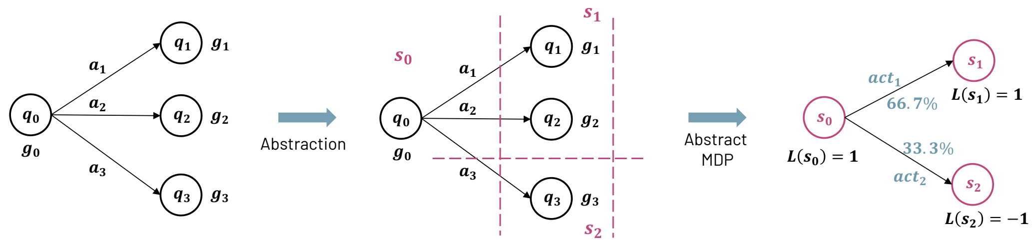
Example 1. *A typical example of AI-CPS is the Adaptive Cruise Control (ACC) system [63, 76, 97], which is designed to regulate the velocity of the ego car while ensuring collision avoidance with a lead car. In this system, the sampler receives external signals u, i.e., the distance between the two cars, and the speed of the ego car p, as the input and sends them to the controller. The AI controller decides the acceleration of the ego car h, and conveys the corresponding actions to the actuator. The actuator converts the discrete actions into continuous signals $\hat{h}$, which are then sent to the plant for execution.*

2.2 Overview of Mosaic and RQ Design

Safety analysis for AI-CPSs refers the procedure of *analyzing the safety of the system by identifying potential hazards, assessing risks, and the mitigation to the safety threats*. In this work, we propose a model-based safety analysis framework, Mosaic, towards enhancing the development of the safety analysis for AI-CPSs. Figure 1 presents the workflow of Mosaic, which contains three key parts: *data collection*, *model abstraction*, and *safety analysis*.

First, as the preparation step, we simulate the AI-CPS under examination and gather relevant data that includes system states, traces, and their associated safety properties. Then, this collected data is used to construct a Moore machine [53], which serves as a suitable representation of the AI-CPS's behaviors for subsequent safety analysis. In practice, the state, input, and output spaces of such a Moore machine are often high-dimensional and continuous, presenting computational challenges during safety analysis. Moreover, we would like to equip an explainable way to illustrate the system state. Therefore, to overcome this challenge, we propose the creation of an abstract model from the Moore machine as an MDP by performing abstraction in terms of state, transition, action, and labelling. This abstract MDP model preserves critical safety properties while enabling efficient analysis of AI-CPSs. Since the preciseness of the constructed abstract model, i.e., the labelling accuracy of abstract model, is imperative for performing further analysis, the first research question that we would like to investigate is, **RQ1:** *How precise are the constructed abstract models?*

Building upon the abstract MDP model, we further propose safety analysis techniques from two directions: *online safety monitoring* and *offline falsification*. As the first direction, we propose an online safety monitoring method that aims to enhance system safety while maintaining comparable performance to the original system. In particular, the monitoring module intelligently computes online safety predictions by observing the system's status and performing probabilistic model checking on the abstract MDP model. Then, based on these safety predictions, the employed controller is dynamically switched between the efficient AI-based controller and a predefined safety controller to improve the safety of the AI-CPS. To assess the effectiveness of online safety monitoring in enhancing the safety of AI-CPS while maintaining comparable performance to the original system, we would like to investigate **RQ2:** *Can Mosaic provide effective safety monitoring?*



(a) AI-CPS as Moore machine (b) Abstraction Modelling (c) Abstract as MDP

Fig. 3. Model abstraction illustration.

As the second direction, we further propose a novel offline model-guided falsification technique specially designed for AI-CPS. Falsification is a well-established safety validation technique that explores the behavior of the CPS to search for a counterexample that violates the given specification. However, traditional falsification is ineffective in AI-CPS since it easily falls into the local optimum [76]. To address this problem, we design and develop a novel falsification technique that combines model-guided exploration and optimization-based search to effectively detect counterexamples for AI-CPS. To assess whether the proposed technique is useful and outperforms existing state-of-the-art falsification techniques for AI-CPSs, we perform a comparative study to demonstrate, **RQ3:** *Is Mosaic effective in guiding the falsification procedure?*

Furthermore, to understand the influence of different parameters on the constructed abstract model, we select multiple parameter sets, and examine how they affect the number of states and transitions, the preciseness, the time of probabilistic model checking (in online safety monitoring), and the falsification success rate (in offline falsification). This leads to the last RQ that we would like to investigate, **RQ4:** *What is the influence of the parameters of the abstraction process on the effectiveness of the safety analysis?*

3 ABSTRACT MODEL CONSTRUCTION

In this section, we discuss how to construct the abstract model of the AI-CPS to empower the safety analysis. We first briefly introduce how to use a Moore machine to represent the behavior of AI-CPS in Section 3.1. Then, we propose a method to construct a representative abstract MDP model from collected data accordingly (Section 3.2).

3.1 AI-CPS as Moore Machine

To conduct the safety analysis for AI-CPS, it is crucial to accurately represent the system's behavior, which directly indicates its level of safety. Considering its stateful nature and safety awareness, we adopt *Moore machine* [69] as our formal model to depict the system's behavior.

Definition 1 (Moore Machine). *A Moore machine is a tuple* $(\mathrm{Q}, q_0, \Sigma, \mathrm{O}, \Xi, \mathrm{G})$*, where* Q *is a finite set of system states and* $q_0 \in \mathrm{Q}$ *is the initial state.* Σ *and* O *are finite sets and are referred to as input and output alphabets, respectively.* $\Xi : \mathrm{Q} \times \Sigma \times \mathrm{Q}$ *is a set of transition.* $\mathrm{G} : \mathrm{Q} \to \mathrm{O}$ *is the output function mapping a state to the output alphabet.*

In order to build the Moore machine, we first simulate the system extensively to gather sufficient data to describe the behavior of the system. Next, we profile the system using the simulation data and the AI-CPS as inputs. This profiling process extracts important information such as system states, controller outputs, and Signal Temporal Logic (STL) semantics (introduced below), which effectively capture the system's behavior. We construct the Moore machine by mapping the profiling data to its corresponding components. More concretely, Q represents the system state space, and q_0 is the starting point[1]. Σ represents the controller's actions, which decides how the system behaves in the environment. Ξ describes how the system's state changes in response to each control action, and G maps the system's state to the STL robust semantics. For O, we use the robust semantics of STL as the output of the system state.

With the ability to describe safety-related temporal behaviors, STL is extensively employed as the specification language of CPSs [9, 27, 30, 94, 97, 98]. It is empowered with *always* ($\Box$) and *eventually* ($\Diamond$) temporal operators, which could be used to describe diverse type of temporal behavior of CPSs. Moreover, it is enriched by *quantitative robust semantics* [30]. This semantics assigns a quantitative value to indicate the level of satisfaction of the specification, enabling a wide range of safety analysis techniques, including falsification [36, 93, 95, 102] and monitoring [27, 94]. A positive/negative semantics value indicate that the safety specification is satisfied/violated, and a higher absolute value denotes a stronger satisfaction or violation. Readers who are interested can find more explanations of the robust semantics in the Appendix as well as in [30]. The following example illustrates the case of using STL to describe a safety specification of the ACC system.

Example 2. *An example of an STL safety specification for the ACC system (recall Example 1) is* $\Box_{[0,30]}(\texttt{speed} \leq 60)$, *which stands for "within 30 seconds, the speed of the ego car should not exceed 60 km/h". If the robust semantics returns a negative number, it means that the system violates the given speed requirement; otherwise, the requirement is satisfied.*

In this work, we focus on the safety analysis of AI-CPS, which can be well described by using STL robust semantics as the degree of satisfaction of the system w.r.t. given safety specifications. By incorporating this robust semantics into the states of the Moore machine, we enable the safety awareness of the model. Therefore, we leverage the quantitative robust semantics of the STL specification as the output G of the Moore machine.

Example 3. *Recall that q is the state of the AI-CPS. Say, the state space of* ACC *(in Example 1) is two dimensional, and we collect three simulation traces with two time steps:* $\{q_0 = (65.0000, 7.0000), q_1 = (65.0411, 6.8030)\}$, $\{q_0 = (65.0000, 7.0000), q_2 = (65.0411, 6.8047)\}$, *and* $\{q_0 = (65.0000, 7.0000), q_3 = (64.9912, 7.8110)\}$. *Moreover, the controller actions a and STL robustness g are also collected, which are* $a_1 = \{-0.9200\}$, $a_2 = \{-0.9150\}$, $a_3 = \{1.9200\}$, *and* $\{g_0 = 1, g_1 = 0.9800\}$, $\{g_0 = 1, g_2 = 0.9809\}$, $\{g_0 = 1, g_3 = -1.3200\}$, *respectively. The Moore machine is thus given as:* Q, Σ, *and* O *are* $\{q_0, q_1, q_2, q_3\}$, $\{a_1, a_2, a_3\}$ *and* $\{g_1, g_2, g_3\}$, *respectively; the set of transitions is* $\{(q_0, a_1, q_1), (q_0, a_2, q_2), (q_0, a_3, q_3)\}$; *the mapping of the output function is* $\mathrm{G}(q_0) = g_0, \mathrm{G}(q_1) = g_1, \mathrm{G}(q_2) = g_2, \mathrm{G}(q_3) = g_3$.

3.2 MDP Model Abstraction

In Section 3.1, we represent the behavior of AI-CPS as a Moore machine. However, it is often computationally expensive to perform safety analysis directly on the Moore machine model of AI-CPS, mainly due to the high-dimensional and continuous nature of the state, input, and output spaces. To address this challenge, we propose constructing an abstract MDP model from the Moore machine as a surrogate for the AI-CPS, which enables more efficient safety analysis. The definition of MDP is given as follows.

Definition 2 (Markov Decision Process). *An MDP $\mathcal{M}$ can be represented as a tuple* $(\mathsf{S}, s_0, \mathsf{Act}, \Theta, \delta, \mathsf{AP}, \mathsf{L})$ *consisting of a finite set of states* S, *an initial state* $s_0 \in \mathsf{S}$, *a finite set of actions* Act, *a finite set of transitions* $\Theta : \mathsf{S} \times \mathsf{Act} \rightarrow \mathsf{S}$, *a transition probability function* $\delta : \mathsf{S} \times \mathsf{Act} \times \mathsf{S} \rightarrow [0, 1]$, *a set of atomic propositions* AP, *and a labelling function* $\mathsf{L} : \mathsf{S} \rightarrow 2^{\mathsf{AP}}$.

[1]The internal structure, e.g., neuron activation values, of the AI controller is abstracted away in this work, as suggested by Bensalem et al. [10].

The MDP model $\mathcal{M}$ is derived based on the abstraction that considers *state*, *transition*, *action*, and *labelling*. We denote the state, transition, action, and labelling abstraction functions as $\zeta_S, \zeta_\Theta, \zeta_{Act}, \zeta_L$, respectively, the details are presented in the rest of this subsection.

State Abstraction. Given a Moore machine state q, the state abstraction function ζ_S maps it to an MDP state s, i.e., ζ_S $(q) = s$. The abstraction procedure consists of two steps: *automated dimension reduction* and *equal interval partition*. We first apply automated dimension reduction to resolve the problem of high dimensionality in the Moore machine states Q. Specifically, this is achieved by employing Principal Component Analysis (PCA) [1] that transforms the Moore machine state $q \in \mathbb{R}^l$ to a low dimensional state $\hat{q} \in \mathbb{R}^k$ with $l > k$, which aims at uncovering the correlations among states. Mathematically, we have $\hat{q} = f(q)$, where f is the process of PCA.

With the k-dimensional reduced states as the input, we further partition them into c^k regular grids [80], i.e., each dimension is equally partitioned into c intervals. We denote the i-th interval on j-th dimension as d_i^j. An MDP state s thus contains the Moore machine states $\{q_1, \ldots, q_n\}$ that fits in the same grid (see Fig. 3). Formally, we have

$$s_{\langle l_1,\ldots,l_k\rangle} = \{q_i | \hat{q}_i^1 \in d_{l_1}^1 \wedge \cdots \wedge \hat{q}_i^k \in d_{l_k}^k, \hat{q}_i = f(q_i), \\ i \in \{1, \ldots, n\}\}.$$

Remark 1 As discussed in Section 5, the choice of parameters c and k has an impact on both the preciseness of the abstract model and the performance of the safety analysis. In practical applications, these parameters should be selected based on the specific task requirements, e.g., the desired efficiency of the analysis. Moreover, alternative partitioning methods, such as multi-layered partitioning [43] or clustering-based partitioning [99], can be employed instead of regular grids-based partitioning. Our abstraction method is compatible with these different partitioning approaches, allowing users the flexibility to choose the most suitable method for their specific needs. However, as the discussion of the advantages and limitations of different partitioning methods is beyond the scope of this paper, we focus on the regular grids-based partition in our method to provide a clear explanation.

Transition Abstraction. The transitions between MDP states are derived using the transition abstraction. We utilize the transition abstraction function ζ_Θ to map a Moore machine transition ξ to an MDP transition θ, i.e., ζ_Θ $(\xi) = \theta$. If there exists a Moore machine transition $\xi \in \Xi$ between $q \in s$ and $q' \in s'$, then an MDP transition is established accordingly between the corresponding MDP states s and s'. Hence, an MDP transition encompasses all Moore machine transitions that share the same starting and destination MDP states (see Fig. 3).

Moreover, to enable probabilistic safety analysis for AI-CPS, we introduce the transition probability $\rho(s, \text{act}, s')$ for each transition. It is calculated based on the number of Moore machine transitions from $q \in s$ to $q' \in s'$ with input $\sigma \in \text{act}$, relative to the total number of outgoing transitions from state s. We therefore have

$$\rho(s, \text{act}, s') = \frac{|\{(q, \sigma, q') | q \in s, \sigma \in \text{act}, q' \in s'\}|}{|\{(q, \sigma, \_) | q \in s, \sigma \in \text{act}\}|}.$$

Action Abstraction. The action abstract function ζ_{Act} is designed to transform the input of the Moore machine σ into a corresponding low-dimensional MDP action act, i.e., ζ_{Act} $(\sigma) = \text{act}$. Considering computational efficiency, we use the round function, which takes the integer part of σ, as the abstraction of the input. In other words, given an input of the Moore machine σ, the MDP action act is abstracted as $\text{act} = \zeta_{Act}$ $(\sigma) = \texttt{round}(\sigma)$.

Labeling Abstraction. Finally, we perform abstraction on the output of the Moore machine, which is mapped to the labelling of the MDP. Recall that the output G represents the robust semantics of the STL specification, which gives a continuous value where positive (negative) values indicate a safe (unsafe) system status. Considering that an MDP state s may contain multiple states $q_1 \dots q_n$, we define the labelling abstraction function as

$$\zeta_L(\mathrm{G}(q)) := \begin{cases} -1 & \text{if } \min_{i=1}^{n} \mathrm{G}(q_n) < \varepsilon, q \in s = \{q_1, \dots, q_n\} \\ +1 & \text{otherwise} \end{cases}$$

Intuitively, if the minimum value of the output over $\{q_1, \dots q_n\}$ is smaller than a predefined threshold $\varepsilon > 0$, the system is close to or already in the dangerous status (see Fig. 3). In such a case, we label the output of s as -1, otherwise, we label it as $+1$.

Example 4. *Continued from Example 3, suppose that after the abstraction, the MDP is given as Fig. 3:* S *contains three states* $s_0 = \{q_0\}$, $s_1 = \{q_1, q_2\}$, *and* $s_2 = \{q_3\}$; $\mathsf{Act} = \{\mathsf{act}_1 = -1, \mathsf{act}_2 = 2\}$; *the set of abstract transitions is* $\{\theta_1 = (s_0, \mathsf{act}_1, s_1), \theta_2 = ((s_0, \mathsf{act}_2, s_2))\}$ *with transition probability* $\rho(\theta_1) = 66.7\%$ *and* $\rho(\theta_2) = 33.3\%$; *the labelling function* $\mathsf{L}(s_0) = \mathsf{L}(s_1) = 1$, $\mathsf{L}(s_2) = -1$.

Remark 2 The purpose of abstraction is to create a more compact model that facilitates safety analysis. In addition, the abstraction also serves as a *human-explainable* model [28, 33], allowing the system's condition to be expressed in terms of the level of satisfaction with respect to the specification. This aids developers in intuitively understanding the system's characteristics and controller behaviors. Moreover, our model is relatively *robust* in the presence of noise, i.e., small perturbation observed by the sampler from the environment. This is due to the fact that one abstract state includes a region of concrete state space, where the perturbed states are included.

4 ONLINE AND OFFLINE SAFETY ANALYSIS

The constructed abstract MDP model provides the possibility to further perform efficient safety analysis. In this section, we introduce novel safety analysis techniques in two directions: online safety monitoring (Section 4.1) and offline model-guided falsification (Section 4.2). These two diverse applications shows the potential of Mosaic in addressing the quality assurance challenges of AI-CPS. Additionally, our methods initialize an early step to provide safety issue detection of AI-CPS and potentially increase the reliability of a deployed AI-CPS.

4.1 Online Safety Monitoring

Our objective in online safety monitoring is to offer safety suggestions and potential countermeasures for mitigating safety hazards in *real-time* control processes of AI-CPS. To this end, we propose an effective model-based safety monitoring technique that utilizes the derived MDP model. In this technique, we employ Probabilistic Model Checking (PMC) [56] as the underlying approach for performing the online safety analysis. The advantages of using PMC are two-fold. First, it allows for the detection of potential risks during runtime and can help in identifying problems before they escalate into critical failures [2, 51]. Second, by analyzing the system's behavior probabilistically, it can provide insights into the likelihood of certain failures occurring. The early detection enables the usage of proactive measures to mitigate potential risks [100]. A brief overview of PMC is presented as follows.

As an automated verification technique, PMC focuses on providing not only Boolean results of the model checking but also *quantitative* guarantees for systems that exhibit probabilistic and randomized behaviors. It adopts Probabilistic Computation Tree Logic (PCTL) [21] as the foundation for the verification. PCTL is designed to describe the behavior of Markov models. An example formula is $\mathcal{P}_{>0.5}[\mathcal{X}(rob = -1)]$, which means "the probability of reaching a state where

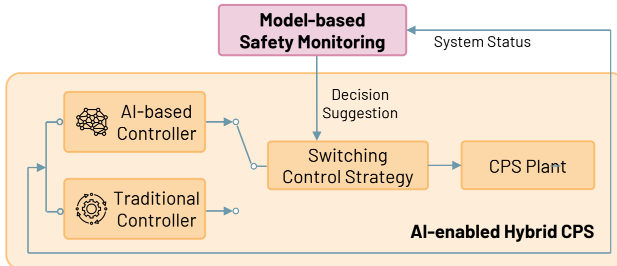


Fig. 4. Overview of the model-based safety monitoring.

variable p is true is greater than or equal to 0.8" By taking an MDP model $\mathcal{M}$, a state s ∈ S, and a PCTL formula ϕ as the inputs, PMC outputs "yes" if $s \models \phi$, or otherwise, a counterexample (error path).

The online safety monitoring is performed by employing PMC to compute the output of a given PCTL formula based on the abstract MDP model, which describes the safety status of the system and is referred to as the *safety query* in this work. An example of the safety query could be $s \models \mathcal{P}_{>0.5}[\mathcal{X}(rob = -1)]$, which means that "Is the probability that the robustness value will be -1 in the next timestamp, given the current state s, greater than 50%?". If the PMC returns "yes", we consider the system to be in an unsafe status. Otherwise, the result indicates that the system is in a safe condition.

Based on the results of PMC, we further introduce a corresponding switching control strategy that switches between the AI controller and a predefined traditional safety controller to maintain the safety of the AI-CPS (see Fig. 4). If the PMC outputs "unsafe", the traditional safety controller will be activated to ensure the safety of the system. Otherwise, the AI controller continues to determine the actually applied actions for increasing the overall efficiency of the AI-CPS [76, 97]. By using the proposed online safety monitoring technique, we can potentially design a hybrid control system that takes advantage of both the efficient AI controller and the traditional controller.

Remark 3 The switching control mechanism is a well-established technique in control theory, including approaches like sliding mode control [91] and supervisory control [55]. While careful consideration is required to design the switching criteria for ensuring system stability, this mechanism offers the opportunity to leverage the benefits of both the AI controller and the traditional control concurrently, thus enhancing the overall performance of the system.

4.2 Offline Model-guided Falsification

Algorithm 1 Optimization-based falsification

Require: A Cyber-Physical System P, an STL formula φ, and a simulation budget t_g

Let $\mathsf{rb}_{\min} \leftarrow \infty, u \leftarrow \varnothing, t \leftarrow 0$ ▷ record minimum robustness and falsifying input
while $t < t_g$ **do**
  $t \leftarrow t + 1$
  $u_i \leftarrow \textsc{Opt}(\{\langle u_k, \mathsf{rb}_k \rangle\}_{k=0,\ldots,t-1})$ ▷ sample signal with minimal robustness
  $\mathsf{rb}_i \leftarrow [\![\mathrm{P}(u_i), \varphi]\!]$ ▷ compute robustness
  if $\mathsf{rb}_i < \mathsf{rb}_{\min}$ **then**
    $u \leftarrow u_i,\ \mathsf{rb}_{\min} \leftarrow \mathsf{rb}_i$
return $\begin{cases} u & \text{if } \mathsf{rb}_{\min} < 0 \\ \varnothing & \text{otherwise, the budget } t_g \text{ is run out} \end{cases}$

Falsification [36, 95, 97] has been widely utilized in the safety validation of CPSs to identify a temporal signal that lead to system behaviors violating the safety specification. The quantitative robust semantics of STL empowers the *optimization-based falsification*, which aims to minimize $[\![\mathsf{P}(u), \varphi]\!]$ to find an input signal u such that $[\![\mathsf{P}(u), \varphi]\!] < 0$, where $\mathsf{P}(u)$ is the system output and φ is the desired specification. The input signal u which triggers the system violation is called a *falsifying input*. Algorithm 1 outlines the classical falsification process which initiate the current minimal robust semantics rb_{min}, the corresponding signal u, and the current iteration t (Line 1). The iteration should be larger than the predefined budget t_g (Line 2) and the iteration counter increases one at the beginning of the loop (Line 3). Then, taking the history of the searched signals as input, it iteratively searches the input signal space (Line 4) and checks the robustness rb_i of an input signal u_i (Line 5). If u_i is a falsifying input, it is returned; otherwise, if no falsifying input is found within the allocated budget, a timeout is reported (Line 8). The search process in Line 4 is usually implemented using optimization techniques [67, 74]. The optimization is employed to search for inputs or parameter values that optimize the objective function, i.e., to minimize the robustness value. However, existing optimization-based falsification techniques developed for traditional CPS have been recently found to be ineffective when applied to AI-CPS [76]. Thus, as another direction of performing the safety analysis, in this subsection, we present a novel offline model-guided falsification method designed for AI-CPS.

An overview of the proposed offline model-guided falsification is presented in Fig. 5, which consists of two stages: a *global* and a *local* search stages. In the global search, a candidate signal is dequeued from a search queue, where two types of signals are collected: 1) randomly generated signals at the beginning of the algorithm; 2) potentially unsafe signals obtained from the local search. The candidate signal is then put into the local search procedure. The local search employs stochastic optimization techniques, e.g., hill-climbing optimization [74], on the candidate signal and tries to find the signal with minimal robust semantics. At each intermediate step of the optimization, PMC, on the abstract model w.r.t. the safety query, is conducted to examine the corresponding intermediate signal. If the PMC returns "unsafe", indicating that the signal could lead to an *unsafe region*, the signal is considered as potentially unsafe and put into the search queue for the next round of global search, aiming to use the constructed model as a exploration guidance.

Algorithm 2 summarizes the detailed offline model-guided falsification process. The inputs to the algorithm are the AI-CPS P^{N} controlled by the AI controller N, the abstract MDP model $\mathcal{M}$, the desired STL specification φ for the system, a PCTL specification ϕ for the safety query, the initial size k of the seed queue Q, a global budget t_g, and a local budget t_l. The algorithm follows these steps:

- First, Q is initialized by sampling k input signals randomly in the system input space (Line 1), and the algorithm enters the outer loop (Line 2);
- Then, the algorithm enters an inner loop (Line 4), in which input signals are iteratively selected and evaluated. Specifically, the inner loop performs the following steps:
  - at iteration i, an input signal u_i is selected by: i) popping the head of Q if $i = 1$, or ii) running hill-climbing optimization Opt (the detailed optimization algorithm will be introduced in Section 5.2), taking the sampling history of input signals as the input and outputting the next optimized signal u_i, which acts as the exploitation for falsifying inputs (Line 8);
  - the robustness rb_i of the output signal $\mathsf{P}^{\mathsf{N}}(u_i)$ w.r.t. φ is computed (Line 9), and if rb_i is negative, u_i is returned as a falsifying input;
  - the PMC of u_i is conducted on $\mathcal{M}$ w.r.t. ϕ to decide whether it can lead to a potentially unsafe region (Line 12). If so, enqueue u_i to Q as a possible guide to future system behavior exploration (Line 14);

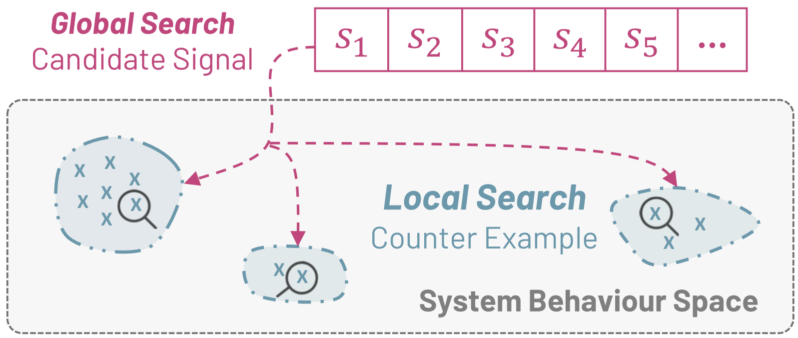


Fig. 5. Overview of model-guided falsification for AI-CPS.

Algorithm 2 The offline model-guided falsification algorithm

Require: an AI-CPS P^{N} controlled by N, the MDP $\mathcal{M}$, an STL specification φ, a PCTL specification ϕ, the initial size k of the seed queue Q, a global budget t_g and a local budget t_l.

1: Initialize t with 0, and Q with k randomly sampled input signals
2: **while** $t < t_g$ **do**
3: $\quad t \leftarrow t + 1$
4: $\quad$ **for** $i \in \{1, ..., t_l\}$ **do**
5: $\quad\quad$ **if** $i = 1$ **then**
6: $\quad\quad\quad u_i \leftarrow \textsc{DeQueue}(\mathrm{Q})$
7: $\quad\quad$ **else**
8: $\quad\quad\quad u_i \leftarrow \textsc{Opt}(\{u_l, \mathtt{rb}_l\}_{l=0,...,i-1})$ ▷ optimization for falsifying inputs (exploitation)
9: $\quad\quad \mathtt{rb}_i \leftarrow [\![\mathsf{P}^{\mathsf{N}}(u_i), \varphi]\!]$ ▷ check the robustness of x_i
10: $\quad\quad$ **if** $\mathtt{rb}_i < 0$ **then**
11: $\quad\quad\quad$ **return** u_i ▷ u_i is a falsifying input, return it
12: $\quad\quad \mathtt{flag} \leftarrow \mathrm{PMC}(u_i, \phi, \mathcal{M})$ ▷ check if x_i could lead to unsafe region (exploration)
13: $\quad\quad$ **if** `flag` is TRUE **then**
14: $\quad\quad\quad \mathrm{Q} \leftarrow \textsc{EnQueue}(\mathrm{Q}, u_i)$ ▷ u_i may lead to falsification, push it to Q

- The algorithm can also terminate if no falsifying input is found within the global budget t_g (Line 2).

Note that, Algorithm 2 incorporates the *exploration* (PMC procedure to explore the regions that have not been deeply visited) and *exploitation* (hill-climbing optimization) principle from optimization theory [19]. Moreover, the idea of using Queue Q in our global search is inspired by the *power scheduling* in *fuzzing* [13, 88], which aims to re-explore the region/space that has not been visited for a while and avoid the generation of excessive test cases that repeatedly exercise the same region.

5 EXPERIMENTAL EVALUATION

To demonstrate the effectiveness, efficiency, and potential usefulness of Mosaic, we perform a comprehensive evaluation and conduct an in-depth analysis of the results using multiple representative CPSs. In particular, we design experiments that aim to answer the following research questions (see also Section 2.2):

- RQ1: How precise are the constructed abstract models?
- RQ2: Can Mosaic provide effective safety monitoring?

Table 1. The subject CPSs with target domains, introductions, and the number of blocks.

Subject CPS	Domain	Description	#Blocks
Adaptive Cruise System (ACC)	Driving assistant	Cruise at user-set speed while maintain a safe distance	297
Abstract Fuel Control (AFC)	Powertrain	Keep air-to-fuel ratio inside a cylinder at a reference level	281
Exothermic Chemical Reactor (CSTR)	Chemical reactor	Maintain a target conversion rate in a reactor	316
Water Tank (WTK)	Water storage	Keep the water level at an user-set level	919

- RQ3: Is Mosaic effective in guiding the falsification procedure?
- RQ4: What is the influence of the parameters of the abstraction process on the effectiveness of the safety analysis?

5.1 Benchmark Systems and the Specifications

Adaptive Cruise Control (ACC), *Abstract Fuel Control* (AFC), *Exothermic Chemical Reactor* (CSTR), and *Water Tank* (WTK) (see Table 1) are representative CPSs that are widely used in CPS safety assurance studies, e.g., [47, 63, 76, 95–97].

Adaptive Cruise Control (ACC). The system, developed by Mathworks [63], is designed to control the acceleration of the ego vehicle to keep the relative distance $\mathtt{D_{rel}}$ to the lead vehicle greater than a safety distance $\mathtt{D_{safe}}$. The safety distance is dynamically changed based on the relative velocity between the two vehicles. In addition, when the safety distance is assured, the ego vehicle should approach to user-set cruising velocity $\mathtt{v_{target}}$.

- φ^1_{ACC} requires that "Over a 50 seconds period, the relative distance between the two vehicles $\mathtt{D_{rel}}$, should be greater than or equal to the safe distance, which is defined as the sum of $\mathtt{D_{safe}}$ and the braking distance of the ego vehicle ($1.4 * \mathtt{v_{ego}}$)".

$$\varphi^1_{\mathrm{ACC}} \;=\; \Box_{[0,50]}(\mathtt{D_{rel}} \geq \mathtt{D_{safe}} + 1.4 * \mathtt{v_{ego}})$$

- φ^2_{ACC} expects that "Over a 50 seconds period, if the distance between the two vehicles $\mathtt{D_{rel}}$ is smaller than the safe distance, which is defined as the sum of $\mathtt{D_{safe}}$ and the braking distance of the ego vehicle ($1.4 * \mathtt{v_{ego}}$), the system should be able to return to a safe status in five seconds".

$$\varphi^2_{\mathrm{ACC}} \;=\; \Box_{[0,50]}\left(\begin{array}{l}(\mathtt{D_{rel}} < \mathtt{D_{safe}} + 1.4 * \mathtt{v_{ego}}) \rightarrow \\ \Diamond_{[0,5]}(\mathtt{D_{rel}} > \mathtt{D_{safe}} + 1.4 * \mathtt{v_{ego}})\end{array}\right)$$

Abstract Fuel Control (AFC). AFC is released by Toyota [47], and it simulates the powertrain system of a vehicle. It receives external inputs such as pedal angle and engine speed and controls the fuel injection rate to a fuel cylinder. The air-fuel-ratio AF inside the cylinder is required to be maintained at a reference value AFref in order to achieve complete gasoline fuel combustion. The optimized stoichiometric air-fuel-ratio is approximately 14.7 [3], and the deviation from this reference value should be limited to within 0.1.

- φ^1_{AFC} specifies that "Over a 30 seconds period, the air-to-fuel ratio AF should not deviate from the reference value AFref by 10% more".

$$\varphi^1_{\mathrm{AFC}} \;=\; \Box_{[0,30]}\left(\left|\frac{\mathtt{AF} - \mathtt{AFref}}{\mathtt{AFref}}\right| < 0.1\right)$$

- φ^2_{AFC} requires that "Between the period of 10 and 30 seconds, if the air-fuel-ratio ratio AF deviates by 10% or more from the reference value AFref, within 1.5 seconds, the system should be able to bring the ratio back to the permitted

Table 2. Abstract models details.

Benchmark	ACC-DDPG	ACC-SAC	ACC-TD3	ACC-DNN$_1$	ACC-DNN$_2$	AFC-DNN$_1$
#states	376	1507	629	1091	337	783
#transitions	1983	23396	12857	94574	2729	2875

Benchmark	AFC-DNN$_2$	CSTR-DDPG	CSTR-TD3	WTK-DDPG	WTK-TD3
#states	367	724	323	940	1182
#transitions	1208	9465	2087	48258	99840

range ($|\frac{\mathtt{AF}-\mathtt{AFref}}{\mathtt{AFref}}| < 0.1$)".

$$\varphi^2_{\mathtt{AFC}} = \square_{[10,30]} \left(\begin{array}{l} (\left|\frac{\mathtt{AF}-\mathtt{AFref}}{\mathtt{AFref}}\right| > 0.1) \rightarrow \\ (\Diamond_{[0,1.5]} \left|\frac{\mathtt{AF}-\mathtt{AFref}}{\mathtt{AFref}}\right| < 0.1) \end{array} \right)$$

Exothermic Chemical Reactor (CSTR). The CSTR system, initially released by Mathworks [62], is a widely used system in various chemical industry domains. The objective of this system is to maintain the concentration of the reagent in the exit stream of the reactor at a specific setpoint. When the reactor is triggered from a low transformation ratio to a high transformation ratio, the concentration setpoint will change accordingly. In addition, the error of the concentration should be less than 0.35 during the entire transformation process.

- $\varphi^1_{\mathtt{CSTR}}$ specifies that "Between the period of 27 and 30 seconds, the error of reactor conversion rate should be less than or equal to 0.35".

$$\varphi^1_{\mathtt{CSTR}} = \square_{[27,30]} (|error| \leq 0.35)$$

Water Tank (WTK). WTK [64] is a commonly utilized system in various industrial domains, including the chemical industry, as a container for regulating the inflow and outflow of water. The water inflow rate from the top of the tank varies according to the voltage applied to the pump. On the other hand, water exits the tank through a hole in the base at a rate that is proportional to the square root of the water level within the tank. The inclusion of the square root in the water flow rate introduces non-linearity to the system.

- $\varphi^1_{\mathtt{WTK}}$ requires that "During specific periods ([5, 6], [11, 12], [17, 18], [23, 24]), the absolute value of the error should be no more than 0.4."

$$\begin{aligned} \varphi^1_{\mathtt{WTK}} = & \square_{[5,6]} (|error| < 0.4) \wedge \square_{[11,12]} (|error| < 0.4) \\ & \wedge \square_{[17,18]} (|error| < 0.4) \wedge \square_{[23,24]} (|error| < 0.4) \end{aligned}$$

5.2 Experimental Setup

By considering the RQs that we plan to investigate, we design the evaluation experiments accordingly. Details about the experimental setup are given as follows.

RQ1. First, we would like to investigate the preciseness of the abstract model in terms of labelling, which has potential influences on the subsequent safety analysis. For this purpose, we randomly sample 1,000 traces for each derived abstract model. Then, for all concrete states q contained in the trace, we check whether the output G (q) (the robust semantics) matches the labelling of the corresponding abstract state s, i.e., we verify if $\mathsf{L}(s) = \zeta_{\mathsf{L}}(\mathsf{G}(q))$, where $q \in s$. The average percentage of the matched states is computed for evaluation.

RQ2. In this RQ, we examine whether Mosaic is able to improve the safety of AI-CPS while keeping a similar functional performance compared to the original system. The corresponding safety and performance metrics used in the evaluation are given in Table 3. We run the simulations of each AI-CPS with 100 randomly generated signals and record the total number of time steps t that satisfy the both specifications in a simulation. The average percentage of safe and well-performed time steps over 100 simulations is computed for evaluation.

Moreover, we would also like to check the False Positive (FP) rate and the False Negative (FN) rate of the safety monitoring. More concretely, for the case of FP, we run the simulation of the original AI controller and Mosaic at the same time. We examine instances where the decision module determines a switch to the traditional controller, indicating a potentially unsafe state. We then verify whether actual violations occur in the future. Given that the system behavior may change after switching, we focus on evaluating the initial switching point only. We also apply a similar setting to assess FN rate. We perform experiments on three cases, ACC-DDPG, ACC-SAC, and ACC-TD3. The episode of checking a violation is five seconds, and the number of simulations is 500.

RQ3. In this RQ, we try to explore the effectiveness of Mosaic in conducting falsification on AI-CPS. Recall that Mosaic requires a local optimization process (Algorithm 2 line 8). Here, we choose three classical optimization algorithms, which are Covariance Matrix Adaptation Evolution Strategy (CMAES) [41], Genetic Algorithm (GA) [67], and Simulated Annealing (SA) [11]. They have been widely used in the research community [36, 93, 95]. We denote them as Mosaic-CMAES, Mosaic-GA, and Mosaic-SA, and try to examine the performance of Mosaic when equipping with diverse optimization solvers.

We also aim to investigate if Mosaic outperforms existing state-of-the-art falsification techniques for AI-CPSs. For this, we compare the performance of Mosaic with three falsification approaches, i.e., Random, Breach, and Mosaic$_{rand}$.

- Random is a simple falsification method that randomly samples input signals from the input space and applies them to the system for simulation.
- Breach is a state-of-the-art falsification tool that employs the classic optimization-based falsification method [29]. We select CMAES, GA, and SA as the optimization solvers for Breach in our experiments, denoted as Breach-CMAES, Breach-GA, and Breach-SA.
- Mosaic$_{rand}$ is a variant of the proposed Mosaic, where the enqueue operation based on PMC (Line 14 in Algorithm 2) is replaced with random sampling in the input space. We introduce Mosaic$_{rand}$ for better analyzing the effect of the guidance of the model in the falsification procedure.

The effectiveness of different falsification techniques is evaluated by using the following metrics: (i) falsification success rate (*FSR*): the number of successful runs, out of 30, where the falsification approach finds a falsifying input for the given specification; (ii) *time*: the average time that takes for successful falsification trials, i.e., a counterexample is detected; (iii) *#sim*: the average number of simulations conducted during successful falsification trials. Based on these metrics, we perform evaluations on the four selected benchmark systems with 11 AI controllers and six STL specifications, which will be explained in the remaining part of this subsection.

RQ4. In order to evaluate the influence of hyper-parameters on the effectiveness of safety analysis, we perform the experiment on ACC and consider three AI controllers: DDPG, SAC, and TD3. We specifically manipulate the abstraction parameters, namely c (the number of partition intervals) with $\{10, 20, 40\}$ and k (the abstraction dimension) with $\{2, 3, 4\}$. We conduct experiments using nine different parameter configurations and evaluate the number of states and transitions, the preciseness, and the time of probabilistic model checking, i.e., the time that it takes that the model checker returns a results for the safety specification, and the falsification success rates.

Traditional & AI Controllers. We use model predictive controller [62, 63] for ACC and CSTR, PI and feedforward controller [47] for AFC, and PID controller [64] for WTK, which are all from the original benchmark systems. As mentioned in Section 2.1, we investigate both DRL and supervised learning-based AI controllers in this work. In particular, we adopt the controllers from two standard benchmarks [76, 97], which have been utilized in the state-of-the-art quality assurance studies of AI-CPS. Therefore, for ACC, we perform experiments with five AI controllers: DDPG, SAC, TD3 from Song et al. [76], and two DNN controllers (provided by Zhang et al. [97]) with the following configurations: DNN_1 with 3 layers and 50 hidden neurons in each layer, and DNN_2 with 4 layers and 30 hidden neurons in each layer. for AFC, we use two DNN controllers, which have the following number of layers and hidden neurons: DNN_1 with 3 layers and 15 hidden neurons, and DNN_2 with 4 layers and 10 hidden neurons. for CSTR, we mainly evaluate two DRL controllers, i.e., DDPG and TD3. In total, nine AI controllers are implemented in the experiments.

Details of Abstract Models. For RQ1, RQ2 and RQ3, the state abstraction parameters c and k are selected as 10 and 3, respectively. The number of states and transitions of the abstract model is summarized in Table 2. For building each abstract model, we collect 20,000 traces by using random sampling in the input space. For the labelling of MDP, we choose φ^1_{ACC}, φ^1_{AFC}, φ^1_{CSTR}, and φ^1_{WTK}, as described in Section 5.1, for the four selected CPSs, which reflect the safety requirements of the system.

Safety Query. For the safety query, we use the following specification for the PMC.

$$s \models \mathcal{P}_{>0.8}[\mathcal{F}_{\leq 10}(rob = -1)]$$

which means "*Is the probability that for a given state s, the robustness is -1 in the future 10 steps, greater than 80%?*". To perform safety monitoring, we select the state s at every five-second interval during the simulation and conduct PMC based on this selected state. For offline falsification, we choose the state at the five-second mark of the simulation as the state for PMC analysis.

Software Dependencies & Hardware Platforms. We implement Mosaic in MATLAB with the library of Simulink, and Python. For PMC, we adopt Prism [56] as the verification engine. The falsification algorithm of Mosaic is implemented on top of Breach [29], which is also considered as one baseline approach for falsification in our comparisons. The ACC model requires the *Model Predictive Control* and *Control System* MATLAB toolboxes.

For the abstract MDP model construction, we run the approaches on a server with Intel(R) Core(TM) i9-10940X CPU @ 3.30GHz Processor, 28 CPUs, 62G RAM with two NVIDIA RTX A6000. For the collection of sampling traces, the falsification trials, the safety monitoring evaluation, and the hyper-parameter analysis, we run the experiment with a Lambda Tensorbook, which is equipped with an Intel(R) Core(TM) i7-11800H @ 2.30GHz Processor with 8 CPUs and 64G RAM, and an NVIDIA RTX 3080 Max-Q GPU with 16 GB VRAM.

5.3 RQ1. The preciseness of Abstract Model

To evaluate the efficacy and preciseness of the abstract model in reducing the concrete state space while maintaining accurate labelling, we conduct experiments and obtain the experimental results, which are presented in Table 4. The average preciseness across all four benchmark systems is 92.83%. And for ACC, AFC, and CSTR, WTK, the average preciseness is 86.04%, 99.99%, 99.91%, and 95.58%, respectively. It can be observed that ACC has a lower preciseness

Table 3. The STL specifications for evaluating safety monitoring (RQ2). Safety follows the pattern: $\Box_I(\varphi_1)$, where I is the time interval; Performance follows the pattern: $\Box_I(\varphi_2)$.

Systems	Safety	Performance
ACC	$I = [0, 50]$ $\varphi_1 \equiv \mathtt{D_{rel}} \geq \mathtt{D_{safe}}$	$I = [0, 50]$ $\varphi_2 \equiv \lvert \mathtt{v_{ego}} - \mathtt{v_{target}} \rvert \leq 0.2$
AFC	$I = [0, 30]$ $\varphi_1 \equiv \left\lvert \frac{\mathtt{AF-AFref}}{\mathtt{AFref}} \right\rvert \leq 0.02$	$I = [0, 30]$ $\varphi_2 \equiv \left\lvert \frac{\mathtt{AF-AFref}}{\mathtt{AFref}} \right\rvert \leq 0.016$
CSTR	$I = [25, 30]$ $\varphi_1 \equiv \lvert error \rvert \leq 0.3$	$I = [25, 30]$ $\varphi_2 \equiv \lvert error \rvert = 0.24$
WTK	$I = [5, 6] \cup [11, 12] \cup [17, 18] \cup [23, 24]$ $\varphi_1 \equiv \lvert error \rvert \leq 0.05$	$I = [5, 6] \cup [11, 12] \cup [17, 18] \cup [23, 24]$ $\varphi_1 \equiv \lvert error \rvert \leq 0.04$

Table 4. RQ1 – The preciseness of the abstract models.

Benchmark	ACC-DDPG	ACC-SAC	ACC-TD3	ACC-DNN_1	ACC-DNN_2	AFC-DNN_1
Preciseness	89.73%	90.58%	83.36%	84.15%	82.96%	99.99%
Benchmark	AFC-DNN_2	CSTR-DDPG	CSTR-TD3	WTK-DDPG	WTK-TD3	
Preciseness	100%	99.85%	99.96%	97.71%	93.45%	

compared to the other three systems, which have a value above 95%. This difference in preciseness could be attributed to the need for more fine-tuning of the abstraction parameters, such as c and k, in order to achieve desirable preciseness. We will further investigate this aspect in Section 5.6.

Answer to RQ1: The average preciseness of the four benchmark systems is 92.83%, which indicates that labelling characteristics is accurately captured.

5.4 RQ2. Online Safety Monitoring

Table 5 shows the results of the analysis of online safety monitoring, revealing three key observations. First, MOSAIC demonstrates its capability to enhance the safety of the system while maintaining comparable functional performance to the original system. In four systems (ACC-SAC, ACC-TD3, CSTR-DDPG, CSTR-TD3), both safety and performance are improved, which might due to The reason might be that the control switching strategy absorbs the advantages from both sides. Second, for ACC with DNN_1 and DNN_2, we observe that although the proposed online safety monitoring does increase safety, it comes at the expense of performance. The safety metrics of ACC with traditional controller, DNN_1, and DNN_2 are 0.9744, 0.5109, 0.5311, while the performance metrics for the three systems are 0.6475, 0.9101, 0.8157, respectively. This indicates that AI-based controllers are with better performance metrics, leading to compromised safety. On the other hand, the traditional controller exhibits the opposite behavior, typically safer but sacrificing performance. Consequently, when switching to the traditional controller for safety, the system's performance inevitably decreases. Third, for AFC with DNN_1 and DNN_2, MOSAIC performs worse compared to the original AI-controlled systems. This could be primarily because the traditional controllers have lower safety and performance metrics compared to the

Table 5. RQ2 – Experiment results for the online safety monitoring.

Benchmark	Controller	Safety	Performance
ACC-trad.	traditional	0.9744	0.6475
ACC-DDPG	AI	0.9880	0.2934
	Mosaic	0.9880	0.3134
ACC-SAC	AI	0.9299	0.9390
	Mosaic	0.9559	0.9435
ACC-TD3	AI	0.8537	0.9418
	Mosaic	0.9242	0.9606
ACC-DNN$_1$	AI	0.5109	0.9101
	Mosaic	0.9753	0.6363
ACC-DNN$_2$	AI	0.5311	0.8157
	Mosaic	0.9733	0.6924
AFC-trad.	traditional	0.7177	0.6143
AFC-DNN$_1$	AI	0.8449	0.7700
	Mosaic	0.7589	0.6658
AFC-DNN$_2$	AI	0.7863	0.6896
	Mosaic	0.7001	0.6038
CSTR-trad.	traditional	1	1
CSTR-DDPG	AI	0.8361	0.6988
	Mosaic	1	1
CSTR-TD3	AI	1	0.9643
	Mosaic	1	1
WTK-trad.	traditional	0.4411	0.3407
WTK-DDPG	AI	0.8830	0.8240
	Mosaic	0.8955	0.8070
WTK-TD3	AI	0.4555	0.3865
	Mosaic	0.4687	0.3532

AI controllers. Specifically, the safety and performance metrics of the traditional PI and feedforward controllers are 0.7177 and 0.6143, respectively, which are lower than those of AFC with DNN$_1$ and DNN$_2$. In cases where the traditional controller is already inferior both in terms of safety and functional performance, the switching control strategy fails to improve the system's safety.

Moreover, the FP rates of ACC-DDPG, ACC-SAC, and ACC-TD3 are 3.2%, 0%, 0.9% and the FN rates are 2.1%, 0%, 0%, respectively. This indicates that our safety monitoring is precise for predicting potential future violations, with low FP rate and low FN rate.

Answer to RQ2: Mosaic is effective in providing online safety monitoring while keeping a comparable performance to the original AI-controlled system. In certain cases, the safety or performance decreases, which is mainly due to the limitation of traditional controllers.

Table 6. RQ3 – Experiment results for falsification trials of ACC, AFC, CSTR, and WTK with their specifications. (*FSR*: the number of successful falsification trials that found falsifying inputs (out of 30). *time*: average time cost of successful trials, in seconds. *#sim*: average number of simulations of successful trials. CMAES: Covariance Matrix Adaptation Evolution Strategy. GA: Genetic Algorithm. SA: Simulated Annealing.) We highlight the best results (with the best FSR) in gray.

	ACC-DDPG-φ^1_{ACC}			ACC-SAC-φ^1_{ACC}			ACC-TD3-φ^1_{ACC}			ACC-DNN$_1$-φ^1_{ACC}		
Alg.	FSR	time	#sim	FSR	time	#sim	FSR	time	#sim	FSR	time	#sim
Random	0	-	-	6	90.4	71.7	10	95.9	47.0	5	368.5	83
Breach-CMAES	0	-	-	9	65.6	51.7	18	39.0	34.1	0	-	-
Breach-GA	0	-	-	1	7.46	12.0	1	36.51	60.0	0	-	-
Breach-SA	5	35.58	55.6	16	87.47	135.06	20	68.17	108.7	10	112.04	133.2
Mosaic$_{rand}$	9	560.4	90.6	15	248.5	67.8	11	307.6	87.2	10	169.6	78.6
Mosaic-CMAES	8	187.01	84.25	13	175.93	68.54	18	119.42	59.11	26	59.80	61.73
Mosaic-GA	1	89.84	39.0	10	171.26	65.9	18	169.81	79.56	25	49.67	51.08
Mosaic-SA	2	121.70	47.0	17	111.32	39.24	24	78.45	33.08	26	23.64	25.23

	ACC-DNN$_2$-φ^1_{ACC}			ACC-DDPG-φ^2_{ACC}			ACC-SAC-φ^2_{ACC}			ACC-TD3-φ^2_{ACC}		
Alg.	FSR	time	#sim	FSR	time	#sim	FSR	time	#sim	FSR	time	#sim
Random	2	370.0	70	0	-	-	0	-	-	1	49.7	25.0
Breach-CMAES	4	192.3	41.8	8	143.0	86.8	12	95.7	72.9	14	88.9	50.1
Breach-GA	10	160.01	211.0	0	-	-	0	-	-	7	135.29	226.43
Breach-SA	20	65.46	88.6	13	67.54	107.62	25	33.95	54.44	18	63.54	103.61
Mosaic$_{rand}$	19	160.4	60.7	4	203.2	132.5	7	197.4	113.7	10	105.9	70.6
Mosaic-CMAES	22	52.50	65.05	4	97.70	44.0	11	158.66	68.27	12	87.84	39.33
Mosaic-GA	10	49.50	61.9	0	-	-	4	200.74	80.5	3	181.95	75.67
Mosaic-SA	17	27.83	33.41	15	183.82	68.47	27	147.06	52.26	23	183.83	68.89

	ACC-DNN$_1$-φ^2_{ACC}			ACC-DNN$_2$-φ^2_{ACC}			AFC-DNN$_1$-φ^1_{AFC}			AFC-DNN$_2$-φ^1_{AFC}		
Alg.	FSR	time	#sim	FSR	time	#sim	FSR	time	#sim	FSR	time	#sim
Random	0	-	-	0	-	-	6	136.6	50.3	8	232.5	85.8
Breach-CMAES	0	-	-	21	245.7	56.6	7	323.6	105.8	6	553.5	212.5
Breach-GA	19	126.68	187.16	12	156.08	207.42	14	43.72	72.21	7	56.00	96.29
Breach-SA	25	34.63	51.86	20	38.62	52.75	20	80.77	126.1	11	78.83	126.73
Mosaic$_{rand}$	9	195.4	71.5	18	185.6	59.1	19	190.7	71.2	10	199.3	76.7
Mosaic-CMAES	20	76.05	73.45	22	206.5	59.5	16	224.23	75.75	12	90.72	61.08
Mosaic-GA	15	76.32	81.6	2	96.29	126.5	6	228.61	76.33	9	129.14	82.11
Mosaic-SA	29	52.74	56.55	28	38.55	49.27	10	199.26	63.2	7	112.65	67.0

	AFC-DNN$_1$-φ^2_{AFC}			AFC-DNN$_2$-φ^2_{AFC}			CSTR-DDPG-φ^1_{CSTR}			CSTR-TD3-φ^1_{CSTR}		
Alg.	FSR	time	#sim	FSR	time	#sim	FSR	time	#sim	FSR	time	#sim
Random	0	-	-	0	-	-	30	34.2	23.8	1	141.7	71.0
Breach-CMAES	4	673.2	186.0	0	-	-	18	64.8	46.5	1	35.6	31.0
Breach-GA	1	143.34	217.0	1	171.96	296.0	30	5.74	8.83	12	105.72	156.17
Breach-SA	5	56.99	90.0	0	-	-	30	7.79	11.5	28	78.17	114.43
Mosaic$_{rand}$	3	98.8	45.0	0	-	-	30	38.6	13.3	12	110.9	60.1
Mosaic-CMAES	4	186.4	26.2	1	277.27	97.0	30	20.05	11.4	19	52.54	56.63
Mosaic-GA	2	163.16	98.0	0	-	-	30	15.01	8.6	4	85.45	-
Mosaic-SA	2	175.23	99.5	5	197.90	59.0	30	13.75	7.8	20	40.42	46

	WTK-DDPG-φ^1_{WTK}			WTK-TD3-φ^1_{WTK}		
Alg.	FSR	time	#sim	FSR	time	#sim
Random	0	-	-	0	-	-
Breach-CMAES	0	-	-	30	19.10	64.0
Breach-GA	0	-	-	24	64.21	198.38
Breach-SA	0	-	-	11	42.41	119.91
Mosaic$_{rand}$	0	-	-	0	-	-
Mosaic-CMAES	1	177.7	153.0	18	156.05	74.78
Mosaic-GA	0	-	-	6	236.41	109.83
Mosaic-SA	0	-	-	13	96.71	43.92

5.5 RQ3. Falsification

The experimental results of falsification are presented in Table 6. We run 30 falsification trials for each falsification approach and report the number of successful trials, i.e., *FSR*, as the indicator of effectiveness. We highlight in the table the best approach, w.r.t. FSR, in each benchmark system. In the case of the same FSR, the best approach is selected based on #sim. Based on the results, we can observe that:

- First, Mosaic, i.e., Mosaic with CMAES, GA, and SA, outperforms the other three falsification techniques. In 13 out of 18 falsification problems, Mosaic achieves the most FSR. There are 10 benchmark systems where Mosaic is strictly better than all other approaches, i.e., the FSR is strictly greater than all other methods.
- Additionally, the choice of optimization algorithm in Mosaic makes a difference. In ten benchmarks, Mosaic-SA achieves the best falsification performance among all falsification algorithms, while Mosaic-CMAES is the best in three benchmarks. This suggests that, in practice, the developers should try different optimization algorithms to achieve the most effective falsification results.
- Moreover, there is one case, i.e., ACC-DDPG with φ_1^{ACC}, where Mosaic does not perform as well as Mosaic$_{rand}$. One possible reason for this is that, Mosaic sometimes spends more time searching for suspicious regions. There are also four cases where Breach performs the best, i.e., AFC-DNN$_1$ with φ_1^{AFC}, AFC-DNN$_1$ with φ_2^{AFC}, CSTR-TD3 with $\varphi_1^{\mathsf{CSTR}}$, and WTK-TD3 with φ_1^{WTK}.
- Random fails to outperform any other methods in the falsification trials. This indicates that Random, which relies solely on random exploration in the input space, is the least effective falsification approach.

Answer to RQ3: Mosaic outperforms the other three falsification methods, i.e., Breach, Mosaic$_{rand}$, and Random (13 out of 18 cases). Mosaic is effective in providing guidance for falsification.

5.6 RQ4. Hyper-parameter Analysis

In this RQ, we would like to assess the impact of the hyper-parameters. Table 7, Table 8, and Table 9 present the details of the abstract model in three systems. We observe that larger values of c and k result in an increase in the number of abstract states and abstract transitions. For example, in ACC-SAC, when $c = 10$ and $k = 4$ change to $c = 20$ and $k = 4$, the number of abstract states increases from 3441 to 17834, and abstract transitions increase from 49965 to 211643. Additionally, the preciseness of the abstract model also improves with a larger model. For instance, for ACC-DDPG, the most precise model achieves a preciseness of 96.04%, while one of the most imprecise models has a preciseness of 85.35%. However, it is worth noting that larger abstract models also require more verification time, as indicated by the "PMC time" column. With the increase in abstract state size, the PMC time could increase in order of magnitude, which is expected of the current model checking tools (the search space for them becomes quickly intractable with the increase in the number of states). For example, in ACC-DDPG, the PMC time varies from 0.43 to 1361.71 seconds. A too big abstract model might make the deployment of online safety monitoring impractical. Thus, in practice, users should take a balance between preciseness and PMC time when selecting hyper-parameters.

Figure 6 shows the falsification success rate results in three benchmarks with different optimization algorithms with different hyper-parameters. The impact of hyper-parameters on FSR is noticeable, with the largest difference of 9 between the best and worst falsification cases (SAC-SA). The effect on DDPG is less pronounced, with a maximum difference of 3. These findings confirm that the selection of hyper-parameters can affect the size and preciseness of the constructed abstract models, thereby influencing the performance of offline falsification since it relies on abstract models for PMC. Moreover, different systems and variations in controllers are non-trivial factors when selecting and tuning hyper-parameters. For instance, in ACC, even with identical hyper-parameter settings, the DDPG controller yields relatively fewer abstract states compared to SAC controllers. We consider the stochastic nature of SAC controller can explore more distinct corner states and transitions during the simulations.

Answer to RQ4: There is indeed an impact of the hyper-parameters on the size of the model, the preciseness of the model, the verification time, and the falsification success rate. In practice, it is recommended to perform parameter tuning for the safety analysis to achieve a better result.

Table 7. The details of the abstract model with different hyper-parameters for ACC-DDPG. Verification time in seconds.

c	k	Abstract states	Abstract transitions	Preciseness	PMC time
10	2	78	351	85.35%	0.43
20	2	217	998	92.57%	0.57
40	2	442	2250	91.73%	1.15
10	3	375	1981	89.73%	1.04
20	3	1900	10131	93.11%	19.79
40	3	4327	26837	94.20%	169.77
10	4	853	4530	89.86%	3.37
20	4	4710	23921	95.45%	139.95
40	4	11954	62035	96.04%	1361.71

Table 8. The details of the abstract model with different hyper-parameters for ACC-SAC. Verification time in seconds.

c	k	Abstract states	Abstract transitions	Preciseness	PMC time
10	2	242	3448	86.73%	0.51
20	2	679	10176	93.40%	0.62
40	2	1383	22663	95.91%	1.51
10	3	1506	23394	90.58%	1.50
20	3	6999	97467	93.96%	27.52
40	3	16944	229743	96.71%	259.18
10	4	3441	49965	97.82%	6.78
20	4	17834	211643	98.09%	235.07
40	4	48908	484429	98.21%	2625.34

Table 9. The details of the abstract model with different hyper-parameters for ACC-TD3. Verification time in seconds.

c	k	Abstract states	Abstract transitions	Preciseness	PMC time
10	2	312	5108	72.02%	0.51
20	2	942	16378	80.36%	1.41
40	2	1857	34142	84.92%	2.98
10	3	1742	31829	83.36%	2.35
20	3	8648	153237	88.10%	69.01
40	3	17525	337181	89.51%	602.97
10	4	2874	55623	86.93%	7.81
20	4	15122	263676	88.53%	316.67
40	4	31554	521804	90.96%	3500.55

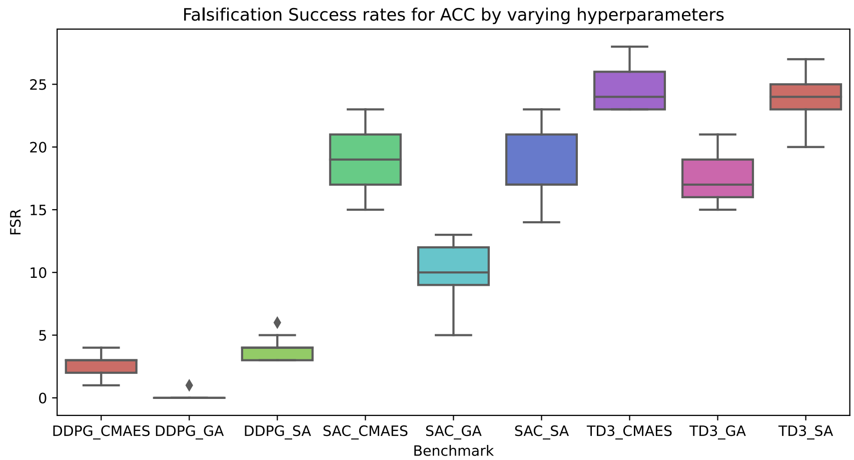


Fig. 6. RQ4 – The hyper-parameter Impact on FSR Effectiveness.

6 THREATS TO VALIDITY

External Validity. Diverse system behaviors and various operation requirements among different AI-CPSs can be an external factor that impacts the validity of our results since the proposed safety analysis method may not be effective on other CPSs. To mitigate this threat, we select CPSs from diverse and representative industry domains with different control requirements and operation specifications. Moreover, we have considered the adaptability and generalizability during the design of our safety analysis framework; namely, we first leverage Moore Machine to accurately capture and represent the system behavior and then construct a corresponding abstract MDP model to compensate for the high-dimension and continuous state, input and action spaces. Given the broad applicability of these two methods, we believe that our framework possesses generalizability to other AI-CPS applications, e.g., the system with deep perception modules or deep radar modules.

Internal Validity. One potential threat is that CPSs can have varying performance due to different environmental parameters, especially when controlled by AI controllers. To mitigate this threat and ensure consistency throughout the entire experiment, we use the same parameters for sample collection, model checking, falsification and evaluation. We configure the environment setting and confirm the system behaviors with reference to the source documentation and related research works.

Construct Validity. The evaluation metrics we used may not fully reflect the effectiveness of our approach. To address this threat, we employ various metrics to quantitatively assess the performance of online safety monitoring and falsification, respectively. In particular, we leverage multiple specifications from the source documentation to evaluate the effectiveness of safety monitoring from both safety and performance perspectives. In terms of falsification, three different metrics (i.e., FSR, time, and #sim) are proposed to assess the performance of different falsification methods.

7 RELATED WORK

Quality Assurance for AI-enabled Cyber-Physical System. Quality assurance of AI-CPSs is of urgent need since many of its applications are in safety-critical domains. Therefore, there comes an increasing trend of relevant research activities in this direction recently [4, 39, 40, 60, 65, 66, 75, 76, 97, 98, 101]. Among those methods, the three most popular are *monitoring* [10, 51, 92, 94, 100], *falsification* [23, 31, 83], and *verification* [7, 45, 49, 79, 82].

Monitoring [25, 51, 94, 100] plays a crucial role in ensuring the reliability and safety of AI-CPS at runtime. The key idea of runtime monitoring is to continuously observe the system's behavior during operation and detect deviations from expected or desired behavior in real-time. Based on Gaussian process regression theory, Yel and Bezzo [90] propose a fast reachability analysis to predict future states in real-time. Cairoli et al. [16] introduces quantitative predictive monitoring, the first predictive monitoring method to support stochastic processes and STL specifications. Junges et al. [51] develop a monitor calculates the risk of the unobservable present system state using partial system state information that is provided by observations during runtime. Cleaveland et al. [22] construct probabilistic automata and calculate a bounded time safety estimate at runtime. Different from these work, Mosaic concentrates on AI-CPS and combines the abstract MDP and PMC to provide runtime prediction for potential future violation to the safety specfications. Falsification [37, 84, 89, 93, 95, 96, 102] tries to show the violation of a specification rather than proving its satisfaction. Dreossi et al. [31] introduce a compositional falsification framework for AI-CPS models, utilizing a machine learning analyzer to abstract feature spaces, approximate classifiers, and provide falsifying inputs. Tuncali et al. [83] propose a testing framework that evaluates test cases against STL formula with system including cameras, lidar, and radar sensors. Yaghoubi and Fainekos develop a gradient-based method for searching the input space of a closed-loop control system to find adversarial samples against system-level requirements. Ernst [37] et al. propose an adaptive probabilistic search based falsification approach, which makes an uninformed probabilistic choice between simple strategies to extend the input signal through exploration or exploitation at each decision point within a single falsification trial. Zolfagharian et al. [101] leverage ML models and genetic algorithms to test the reliability of DRL agents towards faulty episodes. In contrast, Mosaic is a general safety analysis framework that can be applied to various complex CPSs. Our framework not only provides guidance to the offline falsification but also supports online safety monitoring for AI-CPS. The results confirm that Mosaic is effective in guiding the search process towards the violation of the specification, which leads to an efficient falsification technique. Moreover, our model-based falsification enables a deeper understanding of the system's dynamics and allows for more systematic identification of potential vulnerabilities.

Verification [38, 45, 45, 71, 79, 81, 82] techniques provide formal guarantees of correctness and reliability for AI-enabled CPS by analyzing system models against specified properties or requirements. These methods rigorously ensure that the system operates safely and accurately under various conditions. NNV [82] collect a series of reachability analysis algorithms with diverse kinds of set representations, e.g., polyhedra and zonotopes. NNV supports exact and over-approximate reachability analysis schemes for linear plant models and FFNN controllers with piecewise-linear activation functions. For nonlinear plant models, NNV supports over-approximate analysis by combining star set analysis with zonotope-based analysis. Tran et al. [81] uses two efficient, exact, and over-approximate reachability algorithms for verifying neural network control systems using star sets, an efficient representation of polyhedra. Ivanov et al. [45] convert NNs into equivalent hybrid systems, allowing for existing tools to solve the problem. Mosaic focuses on providing online safety monitoring and offline falsification, instead of providing formal and rigorous analysis for AI-CPS.

Abstraction-based Analysis for AI models. Building an abstract model for DNNs recently attracted researchers' interest in facilitating the explainability of the AI model [28, 32, 33, 52, 77, 87, 101]. Considering that AI systems, particularly AI-CPSs, can come with large continuous state space, such characteristics bring significant challenges to the implementations of many traditional approaches [54, 101]. Therefore, several abstraction approaches have been introduced to address this problem by simplifying and narrowing the system down to a relatively small scale while preserving selected features/properties for further analysis.

Up to the present, most of the abstraction methods utilize deterministic finite automata (DFA) or the Markov model to cover the AI systems under a regular format. For example, Weiss et al. [86] design an active learning algorithm to extract a DFA, in which the refinement process is performed by using the counterexamples returned from equivalence query, based on Angluin's algorithm. Wang et al. [85] propose a method that applies learning and abstraction on system log traces to automatically enable formal verification of discrete-time systems. Later, Du et al. [32] proposed MARBLE, a quantitative adversarial robustness analysis technique for recurrent neural networks (RNNs). It extracts an abstract model from RNN to quantitatively measure the robustness information of RNN, which shows its effectiveness in the detection of adversarial examples. Khmelnitsky et al. [52] propose to combine statistical model checking and active automata learning to perform robustness certification for a recurrent neural network.

The abstraction methods conducted by the studies mentioned above are constrained to certain conditions or scenarios; namely, some of them target specific types of ML models, and others are only applicable to environments with discrete state/action space. Thus, modification and adaptation are needed to apply these approaches to different tasks. In contrast, our work focuses on constructing a general-purpose safety analysis framework and performing the safety guidance and defect detection for AI-CPS, which is more on the system level, where DNNs behave as the key components in the system. By leveraging Moore machine and MDP model abstraction, our method has the potential to get deployed on a large spectrum of systems with different dynamics and operation requirements. Moreover, the proposed online monitoring and offline falsification techniques are not limited by specific abstraction approaches. Namely, our safety analysis methods can be adapted to any model as long as the probabilistic information is preserved. The design of advanced abstraction methods that explicitly fit the complex characteristics of AI-CPSs is left as a future work.

8 CONCLUSION

In this work, we present Mosaic, a model-based safety analysis framework for AI-CPS. Mosaic first abstracts the system as an MDP, which is a representative model to empower effective safety analysis. With the abstract model, we further design two safety analysis approaches: online safety monitoring and offline model-guided falsification. The evaluation demonstrates that Mosaic is effective in performing safety monitoring and finding falsifying inputs. In future work, we aim to expand the capabilities of Mosaic by developing additional safety analysis techniques. For instance, we plan to explore the possibility of repairing AI controllers by leveraging the violation episodes identified in the abstract model. By constructing traces that avoid hazards and lead to safe system states, we can provide valuable feedback for the retraining process of AI controllers. This approach holds promise for improving the overall safety and reliability of AI-CPS.

ACKNOWLEDGMENT

This work was supported in part by Canada First Research Excellence Fund as part of the University of Alberta's Future Energy Systems research initiative, Canada CIFAR AI Chairs Program, Amii RAP program, the Natural Sciences and Engineering Research Council of Canada (NSERC No.RGPIN-2021-02549, No.RGPAS-2021-00034, No.DGECR-2021-00019), as well as JSPS KAKENHI Grant No.JP20H04168, No.JP21H04877, JST-Mirai Program Grant No.JPMJMI20B8.

REFERENCES

[1] Hervé Abdi and Lynne J Williams. 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics* 2, 4 (2010), 433–459.

[2] Luca Aceto, Antonis Achilleos, Elli Anastasiadi, Adrian Francalanza, Anna Ingólfsdóttir, Karoliina Lehtinen, and Mathias Ruggaard Pedersen. 2022. On probabilistic monitorability. In *Principles of Systems Design: Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday*. Springer, 325–342.

[3] Shamil Ahmed Flamarz Al-Arkawazi. 2019. Analyzing and predicting the relation between air–fuel ratio (AFR), lambda (λ) and the exhaust emissions percentages and values of gasoline-fueled vehicles using versatile and portable emissions measurement system tool. *SN Applied Sciences* 1, 11 (2019), 1370.

[4] Raid Rafi Omar Al-Nima, Tingting Han, Saadoon Awad Mohammed Al-Sumaidaee, Taolue Chen, and Wai Lok Woo. 2021. Robustness and performance of deep reinforcement learning. *Applied Soft Computing* 105 (2021), 107295.

[5] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S Awwal, and Vijayan K Asari. 2018. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164* (2018).

[6] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* 34, 6 (2017), 26–38.

[7] Kyungmin Bae and Jia Lee. 2019. Bounded model checking of signal temporal logic properties using syntactic separation. *Proceedings of the ACM on Programming Languages* 3, POPL (2019), 1–30.

[8] Ayan Banerjee, Krishna K Venkatasubramanian, Tridib Mukherjee, and Sandeep Kumar S Gupta. 2011. Ensuring safety, security, and sustainability of mission-critical cyber–physical systems. *Proc. IEEE* 100, 1 (2011), 283–299.

[9] Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donzé, Georgios Fainekos, Oded Maler, Dejan Ničković, and Sriram Sankaranarayanan. 2018. Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. In *Lectures on Runtime Verification*. Springer, 135–175.

[10] Saddek Bensalem, Panagiotis Katsaros, Dejan Ničković, Brian Hsuan-Cheng Liao, Ricardo Ruiz Nolasco, Mohamed Abd El Salam Ahmed, Tewodros A Beyene, Filip Cano, Antoine Delacourt, Hasan Esen, et al. 2023. Continuous engineering for trustworthy learning-enabled autonomous systems. In *International Conference on Bridging the Gap between AI and Reality*. Springer, 256–278.

[11] Dimitris Bertsimas and John Tsitsiklis. 1993. Simulated annealing. *Statistical science* 8, 1 (1993), 10–15.

[12] Aude Billard and Danica Kragic. 2019. Trends and challenges in robot manipulation. *Science* 364, 6446 (2019), eaat8414.

[13] Marcel Böhme, Van-Thuan Pham, and Abhik Roychoudhury. 2016. Coverage-based greybox fuzzing as markov chain. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 1032–1043.

[14] Brandon Bohrer, Yong Kiam Tan, Stefan Mitsch, Magnus O Myreen, and André Platzer. 2018. VeriPhy: verified controller executables from verified cyber-physical system models. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 617–630.

[15] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

[16] Francesca Cairoli, Nicola Paoletti, and Luca Bortolussi. 2023. Conformal quantitative predictive monitoring of stl requirements for stochastic processes. In *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control*. 1–11.

[17] Eduardo F Camacho and Carlos Bordons Alba. 2013. *Model predictive control*. Springer science & business media.

[18] Konstantinos Chatzilygeroudis, Bernardo Fichera, Ilaria Lauzana, Fanjun Bu, Kunpeng Yao, Farshad Khadivar, and Aude Billard. 2020. Benchmark for bimanual robotic manipulation of semi-deformable objects. *IEEE Robotics and Automation Letters* 5, 2 (2020), 2443–2450.

[19] Jie Chen, Bin Xin, Zhihong Peng, Lihua Dou, and Juan Zhang. 2009. Optimal contraction theorem for exploration–exploitation tradeoff in search and optimization. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 39, 3 (2009), 680–691.

[20] Yuqi Chen, Christopher M Poskitt, Jun Sun, Sridhar Adepu, and Fan Zhang. 2019. Learning-guided network fuzzing for testing cyber-physical system defences. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 962–973.

[21] Frank Ciesinski and Marcus Größer. 2004. On probabilistic computation tree logic. In *Validation of Stochastic Systems*. Springer, 147–188.

[22] Matthew Cleaveland, Oleg Sokolsky, Insup Lee, and Ivan Ruchkin. 2023. Conservative Safety Monitors of Stochastic Dynamical Systems. In *NASA Formal Methods Symposium*. Springer, 140–156.

[23] Anthony Corso, Robert Moss, Mark Koren, Ritchie Lee, and Mykel Kochenderfer. 2021. A survey of algorithms for black-box safety validation of cyber-physical systems. *Journal of Artificial Intelligence Research* 72 (2021), 377–428.

[24] Baudouin Dafflon, Nejib Moalla, and Yacine Ouzrout. 2021. The challenges, approaches, and used techniques of CPS for manufacturing in Industry 4.0: a literature review. *The International Journal of Advanced Manufacturing Technology* 113 (2021), 2395–2412.

[25] Ankush Desai, Shromona Ghosh, Sanjit A Seshia, Natarajan Shankar, and Ashish Tiwari. 2019. SOTER: a runtime assurance framework for programming safe robotics systems. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 138–150.

[26] Jyotirmoy Deshmukh, Marko Horvat, Xiaoqing Jin, Rupak Majumdar, and Vinayak S Prabhu. 2017. Testing cyber-physical systems through bayesian optimization. *ACM Transactions on Embedded Computing Systems (TECS)* 16, 5s (2017), 1–18.

[27] Jyotirmoy V Deshmukh, Alexandre Donzé, Shromona Ghosh, Xiaoqing Jin, Garvit Juniwal, and Sanjit A Seshia. 2017. Robust online monitoring of signal temporal logic. *Formal Methods in System Design* 51, 1 (2017), 5–30.

[28] Guoliang Dong, Jingyi Wang, Jun Sun, Yang Zhang, Xinyu Wang, Ting Dai, Jin Song Dong, and Xingen Wang. 2020. Towards interpreting recurrent neural networks through probabilistic abstraction. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 499–510.

[29] Alexandre Donzé. 2010. Breach, a toolbox for verification and parameter synthesis of hybrid systems.. In *CAV*, Vol. 10. Springer, 167–170.

[30] Alexandre Donzé and Oded Maler. 2010. Robust satisfaction of temporal logic over real-valued signals. In *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 92–106.

[31] Tommaso Dreossi, Alexandre Donzé, and Sanjit A Seshia. 2019. Compositional falsification of cyber-physical systems with machine learning components. *Journal of Automated Reasoning* 63, 4 (2019), 1031–1053.

[32] Xiaoning Du, Yi Li, Xiaofei Xie, Lei Ma, Yang Liu, and Jianjun Zhao. 2020. Marble: model-based robustness analysis of stateful deep learning systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 423–435.

[33] Xiaoning Du, Xiaofei Xie, Yi Li, Lei Ma, Yang Liu, and Jianjun Zhao. 2019. Deepstellar: Model-based quantitative analysis of stateful deep learning systems. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 477–487.

[34] Charles E Ebeling. 2019. *An introduction to reliability and maintainability engineering*. Waveland Press.

[35] SMS Elattar. 2008. Automation and robotics in construction: opportunities and challenges. *Emirates journal for engineering research* 13, 2 (2008), 21–26.

[36] Gidon Ernst, Paolo Arcaini, Ismail Bennani, Alexandre Donze, Georgios Fainekos, Goran Frehse, Logan Mathesen, Claudio Menghi, Giulia Pedrinelli, Marc Pouzet, et al. 2020. Arch-comp 2020 category report: Falsification. *EPiC Series in Computing* (2020).

[37] Gidon Ernst, Sean Sedwards, Zhenya Zhang, and Ichiro Hasuo. 2021. Falsification of hybrid systems using adaptive probabilistic search. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 31, 3 (2021), 1–22.

[38] Jiameng Fan, Chao Huang, Xin Chen, Wenchao Li, and Qi Zhu. 2020. Reachnn*: A tool for reachability analysis of neural-network controlled systems. In *International Symposium on Automated Technology for Verification and Analysis*. Springer, 537–542.

[39] Carlos A González, Mojtaba Varmazyar, Shiva Nejati, Lionel C Briand, and Yago Isasi. 2018. Enabling model testing of cyber-physical systems. In *Proceedings of the 21th ACM/IEEE international conference on model driven engineering languages and systems*. 176–186.

[40] Liping Han, Shaukat Ali, Tao Yue, Aitor Arrieta, and Maite Arratibel. 2023. Uncertainty-aware Robustness Assessment of Industrial Elevator Systems. *ACM Transactions on Software Engineering and Methodology* 32, 4 (2023), 1–51.

[41] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation* 11, 1 (2003), 1–18.

[42] Peter Heidlauf, Alexander Collins, Michael Bolender, and Stanley Bak. 2018. Verification Challenges in F-16 Ground Collision Avoidance and Other Automated Maneuvers.. In *ARCH@ ADHS*. 208–217.

[43] Kyle Hsu, Rupak Majumdar, Kaushik Mallik, and Anne-Kathrin Schmuck. 2018. Lazy abstraction-based control for safety specifications. In *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 4902–4907.

[44] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv:1602.07360* (2016).

[45] Radoslav Ivanov, Taylor J Carpenter, James Weimer, Rajeev Alur, George J Pappas, and Insup Lee. 2020. Verifying the safety of autonomous systems with neural network controllers. *ACM Transactions on Embedded Computing Systems (TECS)* 20, 1 (2020), 1–26.

[46] Yifan Jia, Jingyi Wang, Christopher M Poskitt, Sudipta Chattopadhyay, Jun Sun, and Yuqi Chen. 2021. Adversarial attacks and mitigation for anomaly detectors of cyber-physical systems. *International Journal of Critical Infrastructure Protection* 34 (2021), 100452.

[47] Xiaoqing Jin, Jyotirmoy V Deshmukh, James Kapinski, Koichi Ueda, and Ken Butts. 2014. Powertrain control verification benchmark. In *HSCC*. 253–262.

[48] Michael A Johnson and Mohammad H Moradi. 2005. *PID control*. Springer.

[49] Taylor T Johnson, Diego Manzanas Lopez, Luis Benet, Marcelo Forets, Sebastián Guadalupe, Christian Schilling, Radoslav Ivanov, Taylor J Carpenter, James Weimer, and Insup Lee. 2021. ARCH-COMP21 Category Report: Artificial Intelligence and Neural Network Control Systems (AINNCS) for Continuous and Hybrid Systems Plants.. In *ARCH@ ADHS*. 90–119.

[50] Kyle D Julian, Mykel J Kochenderfer, and Michael P Owen. 2019. Deep neural network compression for aircraft collision avoidance systems. *Journal of Guidance, Control, and Dynamics* 42, 3 (2019), 598–608.

[51] Sebastian Junges, Hazem Torfah, and Sanjit A Seshia. 2021. Runtime monitors for markov decision processes. In *International Conference on Computer Aided Verification*. Springer, 553–576.

[52] Igor Khmelnitsky, Daniel Neider, Rajarshi Roy, Xuan Xie, Benoît Barbot, Benedikt Bollig, Alain Finkel, Serge Haddad, Martin Leucker, and Lina Ye. 2021. Property-directed verification and robustness certification of recurrent neural networks. In *International Symposium on Automated Technology for Verification and Analysis*. Springer, 364–380.

[53] Zvi Kohavi and Niraj K Jha. 2009. *Switching and finite automata theory*. Cambridge University Press.

[54] George Konidaris. 2019. On the necessity of abstraction. *Current opinion in behavioral sciences* 29 (2019), 1–7.

[55] Xenofon D Koutsoukos, Panos J Antsaklis, James A Stiver, and Michael D Lemmon. 2000. Supervisory control of hybrid systems. *Proc. IEEE* 88, 7 (2000), 1026–1049.

[56] Marta Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *International conference on computer aided verification*. Springer, 585–591.

[57] Insup Lee, Oleg Sokolsky, Sanjian Chen, John Hatcliff, Eunkyoung Jee, BaekGyu Kim, Andrew King, Margaret Mullen-Fortino, Soojin Park, Alexander Roederer, et al. 2011. Challenges and research directions in medical cyber–physical systems. *Proc. IEEE* 100, 1 (2011), 75–90.

[58] Yuxi Li. 2017. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274* (2017).

[59] Xiaorong Lyu, Yulong Ding, and Shuang-Hua Yang. 2019. Safety and security risk assessment in cyber-physical systems. *IET Cyber-Physical Systems: Theory & Applications* 4, 3 (2019), 221–232.

[60] Tao Ma, Shaukat Ali, and Tao Yue. 2021. Testing self-healing cyber-physical systems under uncertainty with reinforcement learning: an empirical study. *Empirical Software Engineering* 26 (2021), 1–54.

[61] Jan Matas, Stephen James, and Andrew J Davison. 2018. Sim-to-real reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*. PMLR, 734–743.

[62] Mathworks. 2018. Nonlinear Model Predictive Control of an Exothermic Chemical Reactor. https://www.mathworks.com/help/mpc/ug/nonlinear-model-predictive-control-of-exothermic-chemical-reactor.html

[63] Mathworks. 2021. Adaptive Cruise Control System Using Model Predictive Control. https://www.mathworks.com/help/mpc/ug/adaptive-cruise-control-using-model-predictive-controller.html

[64] Mathworks. 2021. Adaptive Cruise Control System Using Model Predictive Control. https://www.mathworks.com/help/slcontrol/gs/watertank-simulink-model.html

[65] Claudio Menghi, Shiva Nejati, Lionel Briand, and Yago Isasi Parache. 2020. Approximation-refinement testing of compute-intensive cyber-physical models: An approach based on system identification. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 372–384.

[66] Claudio Menghi, Enrico Viganò, Domenico Bianculli, and Lionel C Briand. 2021. Trace-checking CPS properties: Bridging the cyber-physical gap. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 847–859.

[67] Seyedali Mirjalili and Seyedali Mirjalili. 2019. Genetic algorithm. *Evolutionary Algorithms and Neural Networks: Theory and Applications* (2019), 43–55.

[68] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[69] Edward F Moore et al. 1956. Gedanken-experiments on sequential machines. *Automata studies* 34 (1956), 129–153.

[70] Martin L Puterman. 1990. Markov decision processes. *Handbooks in operations research and management science* 2 (1990), 331–434.

[71] Xin Qin, Yuan Xia, Aditya Zutshi, Chuchu Fan, and Jyotirmoy V Deshmukh. 2022. Statistical verification of cyber-physical systems using surrogate models and conformal inference. In *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 116–126.

[72] Petar Radanliev, David De Roure, Razvan Nicolescu, Michael Huth, and Omar Santos. 2022. Digital twins: Artificial intelligence and the IoT cyber-physical systems in Industry 4.0. *International Journal of Intelligent Robotics and Applications* 6, 1 (2022), 171–185.

[73] Petar Radanliev, David De Roure, Max Van Kleek, Omar Santos, and Uchenna Ani. 2021. Artificial intelligence in cyber physical systems. *AI & society* 36 (2021), 783–796.

[74] Bart Selman and Carla P Gomes. 2006. Hill-climbing search. *Encyclopedia of cognitive science* 81 (2006), 82.

[75] Seung Yeob Shin, Karim Chaouch, Shiva Nejati, Mehrdad Sabetzadeh, Lionel C Briand, and Frank Zimmer. 2021. Uncertainty-aware specification and analysis for hardware-in-the-loop testing of cyber-physical systems. *Journal of Systems and Software* 171 (2021), 110813.

[76] Jiayang Song, Deyun Lyu, Zhenya Zhang, Zhijie Wang, Tianyi Zhang, and L. Ma. 2021. When Cyber-Physical Systems Meet AI: A Benchmark, an Evaluation, and a Way Forward. *ArXiv* abs/2111.04324 (2021).

[77] Jiayang Song, Xuan Xie, and Lei Ma. 2023. SIEGESIEGE: A Semantics-Guided Safety Enhancement Framework for AI-Enabled Cyber-Physical Systems. *IEEE Transactions on Software Engineering* 49, 8 (2023), 4058–4080. https://doi.org/10.1109/TSE.2023.3282981

[78] Mark J Stanovich, Isaac Leonard, K Sanjeev, Mischa Steurer, Thomas P Roth, Stephen Jackson, and Matthew Bruce. 2013. Development of a smart-grid cyber-physical systems testbed. In *2013 IEEE PES Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 1–6.

[79] Xiaowu Sun, Haitham Khedr, and Yasser Shoukry. 2019. Formal verification of neural network controlled autonomous systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. 147–156.

[80] Joe F Thompson, Bharat K Soni, and Nigel P Weatherill. 1998. *Handbook of grid generation*. CRC press.

[81] Hoang-Dung Tran, Feiyang Cai, Manzanas Lopez Diego, Patrick Musau, Taylor T Johnson, and Xenofon Koutsoukos. 2019. Safety verification of cyber-physical systems with reinforcement learning control. *ACM Transactions on Embedded Computing Systems (TECS)* 18, 5s (2019), 1–22.

[82] Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor T Johnson. 2020. NNV: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In *International Conference on Computer Aided Verification*. Springer, 3–17.

[83] Cumhur Erkan Tuncali, Georgios Fainekos, Danil Prokhorov, Hisahiro Ito, and James Kapinski. 2019. Requirements-driven test generation for autonomous vehicles with machine learning components. *IEEE Transactions on Intelligent Vehicles* 5, 2 (2019), 265–280.

[84] Eric Vin, Shun Kashiwa, Matthew Rhea, Daniel J Fremont, Edward Kim, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L Sangiovanni-Vincentelli, and Sanjit A Seshia. 2023. 3D Environment Modeling for Falsification and Beyond with Scenic 3.0. In *International Conference on Computer Aided Verification*. Springer, 253–265.

[85] Jingyi Wang, Jun Sun, Shengchao Qin, and Cyrille Jegourel. 2018. Automatically 'Verifying'Discrete-Time Complex Systems through Learning, Abstraction and Refinement. *IEEE Transactions on Software Engineering* 47, 1 (2018), 189–203.

[86] Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. Extracting automata from recurrent neural networks using queries and counterexamples. In *International Conference on Machine Learning*. PMLR, 5247–5256.

[87] Xiaofei Xie, Wenbo Guo, Lei Ma, Wei Le, Jian Wang, Lingjun Zhou, Yang Liu, and Xinyu Xing. 2021. RNNrepair: Automatic RNN repair via model-based analysis. In *International Conference on Machine Learning*. PMLR, 11383–11392.

[88] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. 2019. Deephunter: a coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 146–157.

[89] Shakiba Yaghoubi and Georgios Fainekos. 2019. Gray-box adversarial testing for control systems with machine learning components. In *HSCC*. 179–184.

[90] Esen Yel and Nicola Bezzo. 2019. Fast run-time monitoring, replanning, and recovery for safe autonomous system operations. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1661–1667.

[91] K David Young, Vadim I Utkin, and Umit Ozguner. 1999. A control engineer's guide to sliding mode control. *IEEE transactions on control systems technology* 7, 3 (1999), 328–342.

[92] Zhenya Zhang, Jie An, Paolo Arcaini, and Ichiro Hasuo. 2023. Online Causation Monitoring of Signal Temporal Logic. In *International Conference on Computer Aided Verification*. Springer, 62–84.

[93] Zhenya Zhang, Paolo Arcaini, and Ichiro Hasuo. 2020. Hybrid system falsification under (in) equality constraints via search space transformation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 11 (2020), 3674–3685.

[94] Zhenya Zhang, Paolo Arcaini, and Xuan Xie. 2022. Online Reset for Signal Temporal Logic Monitoring. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 11 (2022), 4421–4432.

[95] Zhenya Zhang, Gidon Ernst, Sean Sedwards, Paolo Arcaini, and Ichiro Hasuo. 2018. Two-layered falsification of hybrid systems guided by monte carlo tree search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 11 (2018), 2894–2905.

[96] Zhenya Zhang, Deyun Lyu, Paolo Arcaini, Lei Ma, Ichiro Hasuo, and Jianjun Zhao. 2021. Effective hybrid system falsification using monte carlo tree search guided by QB-robustness. In *International Conference on Computer Aided Verification*. Springer, 595–618.

[97] Zhenya Zhang, Deyun Lyu, Paolo Arcaini, Lei Ma, Ichiro Hasuo, and Jianjun Zhao. 2022. FalsifAI: Falsification of AI-Enabled Hybrid Control Systems Guided by Time-Aware Coverage Criteria. *IEEE Transactions on Software Engineering* 01 (2022), 1–17.

[98] Zhehua Zhou, Jiayang Song, Xuan Xie, Zhan Shu, Lei Ma, Dikai Liu, Jianxiong Yin, and Simon See. 2023. Towards Building AI-CPS with NVIDIA Isaac Sim: An Industrial Benchmark and Case Study for Robotics Manipulation. *arXiv preprint arXiv:2308.00055* (2023).

[99] Derui Zhu, Jinfu Chen, Weiyi Shang, Xuebing Zhou, Jens Grossklags, and Ahmed E Hassan. 2021. DeepMemory: Model-based Memorization Analysis of Deep Neural Language Models. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 1003–1015.

[100] Amirhossein Zolfagharian, Manel Abdellatif, Lionel C Briand, et al. 2023. SMARLA: A Safety Monitoring Approach for Deep Reinforcement Learning Agents. *arXiv preprint arXiv:2308.02594* (2023).

[101] Amirhossein Zolfagharian, Manel Abdellatif, Lionel C Briand, Mojtaba Bagherzadeh, and S Ramesh. 2023. A Search-Based Testing Approach for Deep Reinforcement Learning Agents. *IEEE Transactions on Software Engineering* (2023).

[102] Aditya Zutshi, Jyotirmoy V Deshmukh, Sriram Sankaranarayanan, and James Kapinski. 2014. Multiple shooting, cegar-based falsification for hybrid systems. In *Proceedings of the 14th International Conference on Embedded Software*. 1–10.

SIGNAL TEMPORAL LOGIC

With the ability to describe safety-related temporal behaviors, STL is extensively employed as the specification language of CPSs [30, 94, 97]. The STL *syntax* is explained as follows.

Definition 3 (STL Syntax). *The syntax of STL is composed of the atomic proposition β and the formula φ, which are defined as $\beta ::\equiv f(s) > 0$ and $\varphi ::\equiv \beta \mid \bot \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \,\mathcal{U}_I\, \varphi_2 \mid \Box_I \varphi \mid \Diamond_I \varphi$, respectively. Here, f is a function that maps a signal s to a real value, and I is a time interval $[a, b]$ with $a, b \in \mathbb{R}$ and $a < b$. $\mathcal{U}, \Box$ and $\Diamond$ are typical modalities in temporal logic that denote* until, always *and* eventually *operators respectively.*

In terms of semantics, the conventional STL employs Boolean semantics, which establishes a binary relationship between the signal and the formula, indicating whether the formula is satisfied or not. However, a more advanced approach called *quantitative robust semantics*[30] has been developed. This semantics assigns a quantitative value to indicate the degree of satisfaction of the specification, enabling a wide range of safety analysis techniques, including falsification[36, 95] and monitoring [27, 94].

Definition 4 (STL Quantitative Robust Semantics). *Suppose $u : [t_0, t_1] \to \mathbb{R}^n$ is a signal, where t_0 is the starting time point, and t_1 is the ending time point. The robust semantics $[\![u, \varphi]\!] \in \mathbb{R} \cup \{+\infty, -\infty\}$ is defined as follows.*

$$
\begin{aligned}
[\![u, \varphi]\!] &:= f(u(t_0)) \qquad [\![u, \bot]\!] := -\infty \\
[\![u, \neg\varphi]\!] &:= -[\![u, \varphi]\!] \\
[\![u, \varphi_1 \wedge \varphi_2]\!] &:= \min\left([\![u, \varphi_1]\!], [\![u, \varphi_2]\!]\right) \\
[\![u, \Box_I \varphi]\!] &:= \inf_{t \in I} [\![u(t), \varphi]\!] \\
[\![u, \varphi_1 \,\mathcal{U}_I\, \varphi_2]\!] &:= \sup_{t \in I} \min \begin{pmatrix} [\![u(t), \varphi_2]\!], \\ \inf_{t' \in [t_0, t)} [\![u(t'), \varphi_1]\!] \end{pmatrix}
\end{aligned}
$$

PROBABILISTIC COMPUTATION TREE LOGIC

Probabilistic Model Checking adopts Probabilistic Computation Tree Logic (PCTL) [21] as the foundation for the verification, which is defined as follows.

Definition 5 (Probabilistic Computation Tree Logic). *PCTL is a variant of temporal logic and is composed of the state formula ϕ and the path formula ψ, which are defined as*

$$\phi ::\equiv true \mid \alpha \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \mathcal{P}_{\sim p}[\psi]$$

$$\psi ::\equiv \mathcal{X}\phi \mid \Box\phi \mid \Diamond\phi \mid \phi_1 \,\mathcal{U}^{\leq k}\, \phi_2 \mid \phi_1 \,\mathcal{U}\, \phi_2 \mid \mathcal{F}_{\leq k}\phi$$

where $\mathcal{P}$ is the probabilistic operator, I is the instantaneous operator, C is the cumulative operator, $\mathcal{F}$ is the eventually operator, $\mathcal{X}$ is the next operator, $\mathcal{U}$ is the until operator, α is an atomic proposition, $p \in [0, 1]$ is the probability bound. We have $\sim \in \{<, \leq, >, \geq\}$ and $k \in \mathbb{N}$. $s \models \mathcal{P}_{\sim p}[\psi]$ means that the probability from state s, that ψ is true for an outgoing path satisfies $\sim p$.

Remark 4 It is worth mentioning that PCTL also includes a reward operator $\mathcal{R}_{\sim r}[]$, which calculates the expected value of a formula satisfying $\sim r$. This operator can also be used in safety queries. For example, the question "Is the expected cumulative robustness larger than 3 up to time-step k?" can be expressed as $\mathcal{R}_{robustness>3}[C \leq k]$. Users have the flexibility to define their own safety queries based on their specific usage scenarios.