Probabilistic Conformance for Cyber-Physical Systems

Yu Wang, Mojtaba Zarei, Borzoo Bonakdarpoor, and Miroslav Pajic

Abstract—Conformance is a key concept in the analysis of cyber-physical systems (CPS). It indicates that two models simultaneously satisfy the same specifications of interest practically so that the result of analyzing one model automatically transfers to the other. In this paper, we propose a new concept of probabilistic conformance for CPS, extending previous study to the probabilistic domain. It is defined by the approximately equal satisfaction probabilities for a given parameterized signal temporal logic specification. Then we propose the first statistical verification method for probabilistic conformance of temporal logic specifications for grey-box CPS. To this end, we introduce a novel extended Kolmogorov-Smirnov test that can check approximately equal of two probability distributions for any desired confidence level (≤ 1). We apply our technique to verify the probabilistic conformance of: (1) the startup time of the widely-used full and simplified models of the Toyota Powertrain system, (2) the settling time of lane-keeping controllers based on model predictive control and neural network (NN)-based lane-keeping controllers of different sizes for an autonomous car, and (3) the maximal deviation of DC voltage between the full and simplified models of a power grid system.

I. INTRODUCTION

The conformance [1]–[6] of two models for a set of given specifications (e.g., reachability or input-output relation) is an important property in analysis of cyber-physical systems (CPS). Roughly, it indicates that two models satisfy the given specifications simultaneously, such that the results derived by analyzing one model can automatically transfer to the other for the specifications of interest. The term “conformance” may also be used to refer to the consistency between a system and a design specification of interest (e.g., [7], [8]) — this is out of the scope of this work.

In CPS, many important design specifications are captured by temporal logics such as the signal temporal logic [9]. Following the line of work [10], [11], we focus on the conformance of CPS for temporal logics. This notion of conformance is more general than the conformance for reachability [4], [5], since reachability can generally be captured by temporal logic formulas.

The conformance can be used for two different models derived from the same system under different conditions. This implies that the system executes in the same way under different conditions (e.g., two different inputs). A well-known example of nonconformity is the Volkswagen emissions scandal [12], where the emission control software was deliberately configured to work differently in lab testing and driving conditions; this was done to bypass the emission test without actually reducing the pollution generated from the cars while driving. Similar undesirable nonconformity exists in printers [13], where the software drivers are configured to work differently, in favor of their own cartridge brand. To prevent such software doping [14], it is necessary to verify the conformance of the system under different conditions/settings with respect to the specifications of interest.

The conformance can also be used for two models derived from two different systems operating under the same conditions. This implies that replacing one system with another would not result in violating the specifications of interest. For instance, recently there is a great interest in replacing precise but computationally expensive controllers based on model predictive control (MPC) by neural network (NN) controllers for applications such as lane-keeping systems in autonomous cars [15]. To migrate from an MPC controller to an NN controller without significantly changing the responsiveness of the controller, it is necessary to check the conformance of the two controllers for their settling time, especially considering the fragility of AI-based controllers. While we focus on the latter type of conformance in our case studies, our approach can be also directly used for the former.

Since CPS such as autonomous cars are frequently subject to randomness (e.g., system/network/environment noise), we propose a probabilistic notion of conformance for these systems. We use the notion of probabilistic uncertain systems (PUSs) proposed in [16] to capture a wide class of CPS. Generally, they are grey-box probabilistic dynamical systems with unknown dynamics on a given state space. They subsume commonly studied models such as continuous-time Markov chains and hybrid I/O automata [17] with probabilistic parameters, which model the probabilistic version of the Toyota Powertrain [18]. A PUS incorporates nondeterminism as its input and probabilistic behavior as its parameters, both of which are time functions of values of general types, including real, integer, or categorical/Boolean. Given the values of the input and the parameters, a time-dependent sample path is generated, which can also be of general types.

As illustrated in Figure 1, we define the notion of conformance through a parameterized signal temporal logic (STL) formulas [19] by requiring that the satisfaction probabilities are approximately equal for all values of the parameters; these are effectively infinitely many STL formulas. For example, for the probabilistic conformance of two models $\mathcal{M}_1$ and $\mathcal{M}_2$ of
For monotonically parameterized STL formulas, our general idea for checking conformance is to use sampling to estimate the satisfaction probabilities across the value of the parameters, and to make assertions with high confidence levels from sufficient samples. Due to the monotonicity, these satisfaction probabilities over the values of the parameters can form two probability distributions. The conformance of two PUSs requires the two distributions to be approximately equal.

We develop a novel extended Kolmogorov-Smirnov (EKS) statistical test to check the approximate equality of two distributions with provable confidence levels. Our EKS test is based on the classic Kolmogorov-Smirnov (KS) test [23] and its multivariate generalization [24] for checking the exact equality of two distributions. This allows us to develop a statistical verification method that can check either the probabilistic conformance or nonconformance of two PUSs for any desired confidence level \(< 1\).

We apply the proposed statistical verification method to check the probabilistic conformance of three complex CPS for different specifications of interest to show that our method can be used to a wide range of applications. First, we check the probabilistic conformance of the widely used full and simplified models of the Toyota powertrain system [18], [25], with respect to the startup time for their air to fuel ratio to reach a working region. Our results show that the two models do not probabilistically conform, suggesting the simplified model may not capture certain important aspects of the system. Second, we check the probabilistic conformance of the settling time of an MPC-based lane-keeping controller and several NN-based lane-keeping controllers of different sizes for an autonomous car [26]. We show that NN-based controllers conform to the MPC-based controller, as their size increases; however, a small NN design may result in nonconformity. This suggests that an MPC-based controller can be replaced with an NN-based controller of sufficient sizes, to satisfyingly control the settling time. Finally, we check the probabilistic conformance of the maximal deviation of DC voltage between the full model and a simplified model of a power grid system [27]. Our results show that the two models do not probabilistically conform — i.e., the simplified model again may not capture certain important aspects of the system.

This paper is organized as follows. After preliminaries in Section II, in Section III, we formalize the problem and our definition of probabilistic conformance with respect to an STL formula set. We present our extended Kolmogorov-Smirnov test and the statistical verification for the probabilistic conformance in Section IV. In Section V, we apply the statistical verification to three real-world case-studies, before discussing related work in Section VI, and concluding in Section VII.

Notation: We denote the sets of natural, real, and non-negative real numbers by \(\mathbb{N}\), \(\mathbb{R}\), and \(\mathbb{R}_{\geq 0}\). We define \(\mathbb{R}_\infty = \mathbb{R} \cup \{-\infty, \infty\}\), and \([n]\) = \{1, \ldots, n\}, for \(n \in \mathbb{N}\). The cardinality and the power set of a set \(S\) are denoted by \(|S|\) and \(2^S\).

II. Problem Formulation

We consider a general system model for CPS called probabilistic uncertain systems (PUSs) [16]. This model captures continuous-time probabilistic dynamics on a hybrid state-space of discrete and continuous values, as well as generalizes common probabilistic models such as continuous-time Markov chains (CTMC) and probabilistic hybrid I/O automata [16]. Since we adopt a statistical approach, we mainly view a PUS as a grey-box that generates random samples (Figure 3).
Definition 1. A PUS is a tuple \( \mathcal{M} = (\mathcal{X}, \mathcal{X}_{init}, \mathcal{I}, \mathcal{D}, \{ D(t) \}_{t \in \mathbb{R}_{\geq 0}}, T) \), where

- \( \mathcal{X} = X_1 \times \cdots \times X_n \) is the state space with each \( X_i \) being either \( \mathbb{R} \) or a discrete set \([n]\);
- \( \mathcal{X}_{init} \in \mathcal{X} \) is the initial state;
- \( \mathcal{I} = I_1 \times \cdots \times I_m \) is the range of inputs with each \( I_i \) being either \( \mathbb{R} \) or a discrete set \([n]\);
- \( \mathcal{D} = D_1 \times \cdots \times D_l \) is the range of parameters with each \( D_i \) being either \( \mathbb{R} \) or a discrete set \([n]\);
- \( \{ D(t) \}_{t \in \mathbb{R}_{\geq 0}} \) is a random process on \( \mathcal{D} \) (for a properly defined probability space), defining the random change of the parameter over time;
- \( T : (\mathbb{R}_{\geq 0} \rightarrow \mathcal{I}) \times (\mathbb{R}_{\geq 0} \rightarrow \mathcal{D}) \rightarrow (\mathbb{R}_{\geq 0} \rightarrow \mathcal{X}) \) defines the transition of the system; given the (time-dependent) value of the input and parameter, the system deterministically generates a path.

Given the value of the (time-dependent) input \( I : \mathbb{R}_{\geq 0} \rightarrow \mathcal{I} \), the PUS can generate a random signal \( \sigma(t) = T(I(t), D(t)) \), where the randomness comes from the parameter \( D(t) \). We denote by \( \sigma \sim \mathcal{M}_I \) if the signal \( \sigma \) is randomly generated from the system \( \mathcal{M} \) for the given input \( I \). We also write \( \sigma \sim \mathcal{M} \) if \( I \) is clear from the context.

Note that there is no assumption on the dynamics of a PUS, such as Markovian, causal, etc. Common probabilistic models including discrete-time or continuous-time Markov chains [28], and probabilistic hybrid I/O automata [29], [30] are subsumed by the notion of a PUS (see [16] for details).

Example 1. A simple example of a PUS is a bouncing ball with random gravitational acceleration, as shown in Figure 4. Its state is the height and velocity \((x, v)\). For \( x > 0 \), the state evolves by \( \dot{x} = v, \dot{v} = g \); for \( x = 0 \), it jumps by \( x \mapsto x, v \mapsto -v \). The parameter \( g \) is randomly drawn from a normal distribution \( N(g_0, \sigma^2) \) for some \( g_0, \sigma > 0 \). The initial state is \((x_0, 0)\). The input space is empty.

Finally, note that although by Definition 1, a PUS has a unique initial state, it allows for defining conformance of paths from different initial states \( X_1 \) and \( X_2 \) of the PUS. This is done by adding a new initial state \( X_0 \) to the PUS, and model the transition from \( X_0 \) to \( X_1 \) and \( X_2 \) as part of the input.

Signal Temporal Logic: We use the signal temporal logic (STL) [9] to capture the temporal specifications of interest for the random signals generated by the PUS. STL can be viewed as the counterpart of linear temporal logic (LTL) in the real-time domain with real-valued constraints. An STL formula is defined inductively by the syntax

\[
\varphi ::= f > 0 \mid \neg \varphi \mid \varphi \land \varphi \mid \varphi U [t_1, t_2] \varphi, \tag{1}
\]

where \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is a given function. To simplify further discussion, we let \( t_1, t_2 \in \mathbb{R}_{\infty} \), instead of taking values in nonnegative rational numbers. We call \( f > 0 \) an atomic proposition and \( U [t_1, t_2] \) the “until” operator. Other temporal and logic operators are defined as usual; for example,

- (false/true) \( False = \varphi \land (\neg \varphi) \) and \( True = \neg False \),
- (finally) \( \diamond [t_1, t_2] \varphi = True U [t_1, t_2] \varphi \), and
- (always) \( \Box [t_1, t_2] \varphi = \neg (\diamond [t_1, t_2] \neg \varphi) \).

For a concrete signal \( \sigma : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^n \) generated by a PUS, the satisfaction relation for STL formulas is defined recursively by the semantics

\[
\sigma \models f > 0 \quad \iff \quad f(\sigma(0)) > 0
\]

\[
\sigma \models \neg \varphi \quad \iff \quad \sigma \not\models \varphi
\]

\[
\sigma \models \varphi_1 \land \varphi_2 \quad \iff \quad \sigma \models \varphi_1 \text{ and } \sigma \models \varphi_2
\]

\[
\sigma \models \varphi_1 U [t_1, t_2] \varphi_2 \quad \iff \quad \text{there exists } t \in [t_1, t_2] \text{ such that } \sigma(t) \models \varphi_2 \text{ and for any } 0 \leq t' < t,
\]

\[
\text{it holds that } \sigma(t') \models \varphi_1,
\]

where \( \sigma(t) \) denotes the \( t \)-shift of the signal, defined by \( \sigma(t') = \sigma(t + t') \) for any \( t' \in \mathbb{R}_{\geq 0} \). We make the convention that \( \varphi_1 U [t_1, t_2] \varphi_2 \) is equivalent to \( False \), if \( t_2 < t_1, t_1 < 0, \) or \( t_2 < 0 \).

Example 2. The following STL formula

\[
\varphi = \Box \left( |x| > 0.5 \rightarrow \diamond [0.0, 0.15] |x| \right)
\]

requires that it is always the case that if \( |x| > 0.5 \), then within 0.6 time units \( |x| \) settles under value 0.5 for 1.5 time units.

III. Probabilistic Conformance

Following the line of work from [10], [11], we focus on a class of conformance properties for CPS for an (infinite) set of STL formulas. Mathematically, we say that two PUSs probabilistically conform, if for any STL formula from the set, the satisfaction probabilities are approximately equal for two random signals drawn respectively from the two PUSs. This can be viewed as a probabilistic generalization of [10], [11].

Definition 2 (Conformance). Let \( \Phi \) be an infinite set of STL formulas. For two PUSs \( \mathcal{M}_1 \) and \( \mathcal{M}_2 \) from Definition 1 and some given \( c > 0 \), we say that \( \mathcal{M}_1 \) and \( \mathcal{M}_2 \) \( c \)-approximately probabilistically conform for \( \Phi \) (for the same given input), if for any STL formula \( \phi \in \Phi \), it holds that

\[
| \text{Pr}_{\sigma_1 \sim \mathcal{M}_1}(\sigma_1 \models \phi) - \text{Pr}_{\sigma_2 \sim \mathcal{M}_2}(\sigma_2 \models \phi) | < c,
\]

where \( \sigma_i \sim \mathcal{M}_i \) is a random path drawn from the PUS \( \mathcal{M}_i \), for \( i \in \{1, 2\} \).

In Definition 2, we only require the satisfaction probabilities to be approximately equal for the STL formulas for interest.
instead of exactly equal, since the latter is usually unnecessary in applications (see for example the case studies in Section V). Also, we note that the conformance from Definition 2 cannot be expressed by single formulas in any common temporal logic, as a parameterized formula is effectively an infinite number of STL formulas. For any fixed values of the employed parameters, the property can be expressed in HyperPSTL [16].

Depending on the choice of the class (i.e., set) of temporal properties $\Phi$, different notions for the conformance of PUSs are derived, including probabilistic reach-set conformance and probabilistic trace conformance. Commonly, an STL formula set $\Phi$ can be derived by parametrizing a single STL formula $\phi$ by [19]

$$\Phi = \{ \phi_d : d \in \mathbb{R}^K \}. \quad (2)$$

Effectively, $\phi_d$ represents infinitely many STL formulas, as the parameter $d$ can take infinitely many values. For example, the STL formula set

$$\Phi_1 = \{ \square_{[0,1]}(\sigma > a) : a \in \mathbb{R} \} \quad (3)$$

is derived by parametrizing the threshold $a$. It contains an infinite set of reachability specifications for the parametrized threshold $a$ within a fixed time interval $[0, 1]$. The conformance of the two PUSs $M_1$ and $M_2$ for the set $\Phi_1$ means that, for any threshold $a$ the probability of reaching the threshold should be approximately equal for two random signals respectively from $M_1$ and $M_2$.

Similarly, the STL formula set

$$\Phi_2 = \{ \square_{[0,t]}(\sigma > 0) : t \in \mathbb{R} \} \quad (4)$$

is derived by parametrizing the time horizon $t$. It contains an infinite set of reachability specifications for the fixed threshold 0, within a parameterized time interval $[0, t]$. The conformance of the two PUSs $M_1$ and $M_2$ for the set $\Phi_2$ means that the probability of reaching the threshold 0 (i.e., $> 0$) within any time interval $[0, t]$ should be approximately equal for two random signals respectively from $M_1$ and $M_2$.

Considering that the PUSs can have complex dynamics that may be even unknown in practice, in this work, we propose to statistically verify the conformance of PUSs from Definition 2, as this method exhibits better scalability than the exhaustive approaches and can handle unknown dynamics [20], [21]. There are infinitely many STL formulas of interest in (2), so the proposed statistical verification method should be able to handle an infinite set of STL specifications. This is very challenging since all existing statistical verification techniques can only handle single STL specifications [20], [22]. To solve this, we focus on the conformance for the following class of monotonically parameterized STL formulas:

**Monotonically parameterized formulas:** An important class of parameterized STL formulas is the monotonically parameterized formulas [19]. Generally, the parametrized formula $\phi_d$ (where $d$ captures the vector of parameters) is monotone if the satisfaction probability on a model is preserved for the order of the parameters, i.e., the satisfaction probability changes monotonically with the parameter. While statistically verifying the probabilistic conformance for an arbitrary STL formula set is very difficult, handling a monotonically parameterized formula set can be done by exploiting the formula’s monotonicity.

**Definition 3 (Monotonically Parameterized Formula).** A parameterized formula $\phi_d$ with $d \in \mathbb{R}^K$ is monotone for a PUS $M$ if for any given path $\sigma$ from $M$ and $i \in [K]$, and

- for any $d, d’$ such that $d \preceq_i d’$, it holds that $\sigma \models \phi_d$ implies $\sigma \models \phi_{d’}$, OR
- for any $d, d’$ such that $d \preceq_i d’$, it holds that $\sigma \models \phi_{d’}$ implies $\sigma \models \phi_d$.

where $d \preceq_i d’$ denotes that the entries of $d$ and $d’$ are equal except for $d_i \leq d’_i$.

Following Definition 3, the monotonicity of an STL formula is preserved under parameter alternations.

**Definition 4 (Alternation).** We call the function $\pi(d) = d’$ an alternation, if for all $i \in [K]$, $d’_i = d_i$, or $d’_i = -d_i$. The set of all $2^K$ K-dimensional alternations in $\mathbb{R}^K$, is denoted by $\Pi_K$.

From the previous definitions, the following directly holds.

**Lemma 1.** If $\phi_d$ is a monotonically parameterized STL formula, then so is $\phi_{\pi(d)}$, where $\pi$ is an alternation.

In addition to Definition 3, monotonicity of a parameterized STL formula may depend on the model $M$. For example, the parameterized STL formula from (4) is monotone for $t \in \mathbb{R}$ for any PUS, since from STL semantics, $\square_{[0,t]}(\sigma > 0)$ always implies $\square_{[0,t_2]}(\sigma > 0)$ for any $t_2 \geq t_1$ (including negative $t_1$, $t_2$). On the other hand, the parameterized STL formula from (3) is monotone for $a \in \mathbb{R}$, if any signal $\sigma(t)$ from the PUS $M$ is monotone – i.e., for any $t_2 \geq t_1$, it holds that $\sigma(t_2) \geq \sigma(t_1)$ or $\sigma(t_2) \leq \sigma(t_1)$. As illustrated in Figure 5, if signal $x(t)$ is non-decreasing then for any $a_1 \geq a_2$, it holds

$$\left( \square_{[0,1]}(\sigma > a_1) \right) \rightarrow \left( \square_{[0,1]}(\sigma > a_2) \right). \quad (5)$$

This is generally not true for an arbitrary signal $\sigma(t)$.

**IV. Statistical Verification for Probabilistic Conformance**

In this section, we propose a statistical verification algorithm for verifying the probabilistic conformance of two PUSs for a monotonically parametrized STL formula. Due to the monotonicity, the satisfaction probabilities on a PUS of the parametrized formula can be captured by an (unknown) cumulative distribution function (CDF). To check the probabilistic conformance for this monotonically parametrized formula, it suffices to check the equivalence of two (unknown) CDFs. To achieve this, we propose an extended Kolmogorov-Smirnov (EKS) statistical test, based on the classic Kolmogorov-Smirnov test [23]. In Section IV-A, we introduce the EKS test that can statistically check whether two general unknown CDFs are approximately equal. In Section IV-B, we employ the EKS test to verify probabilistic conformance.
A. Extended Kolmogorov-Smirnov Test

Consider two $K$-dimensional random vectors $X = (X_1, \ldots, X_K)$ and $Y = (Y_1, \ldots, Y_K)$. For each $K$-dimensional alternation $\pi \in \Pi_K$ we define

$$F^\pi(a) = \text{Pr}_X \left( \pi(X)_1 \leq \pi(a)_1, \ldots, \pi(X)_K \leq \pi(a)_K \right),$$

$$G^\pi(a) = \text{Pr}_Y \left( \pi(Y)_1 \leq \pi(a)_1, \ldots, \pi(Y)_K \leq \pi(a)_K \right),$$

where $\pi(X)_i$ is the $i^{th}$ entry of $\pi(X)$, and the probabilities $\text{Pr}_X$ and $\text{Pr}_Y$ are taken for the random vectors $X$ and $Y$, respectively. If $\pi$ is the identity map, then $F^\pi$ and $G^\pi$ are respectively the cumulative distribution functions (CDFs) of $X$ and $Y$, which we denote by $F$ and $G$ for simplicity. Otherwise, $F^\pi$ and $G^\pi$ are the complimentary CDFs of $X$ and $Y$.

To measure the difference between the probability distributions of $X$ and $Y$, let $\gamma_{X,Y} = \max_{\pi \in \Pi_K} \|F^\pi - G^\pi\|_\infty$, with $\| \cdot \|_\infty$ standing for the $L_\infty$ function norm. If $\gamma_{X,Y} = 0$, then $X$ and $Y$ have the same probability distributions. The alternation $\pi$ in (7) is necessary, since two different multidimensional probability distributions may have the same CDFs (but different ECDFs). Then the approximate equality of the probability distributions of $X$ and $Y$ is formulated as the hypothesis testing problem

$$H_0: \gamma_{X,Y} < c \quad H_1: \gamma_{X,Y} > c,$$

where $c > 0$ is a given parameter for approximate equality.

Remark 1. The hypothesis testing problem (8) cannot be handled by the classic Kolmogorov-Smirnov (KS) test [23] and its multivariate generalization [24], since they can only check for the exact equality of two probability distributions, i.e., the hypothesis testing problem

$$H'_0: \gamma_{X,Y} = 0 \quad H'_1: \gamma_{X,Y} > 0.$$ 

To solve problem (8), we build on the classic KS test and propose an extended Kolmogorov-Smirnov (EKS) statistical test. To facilitate presentation, we start from the scalar case and then move to the vector case.

1) Scalar: If $X$ and $Y$ are scalar, then from (8), we have that $\gamma_{X,Y} = \|F - G\|_\infty$, where $F$ and $G$ are the CDFs of $X$ and $Y$, respectively. Given two sets of independent and identically distributed (i.i.d.) samples

$$X^{[n]} = \{X^{(1)}, \ldots, X^{(n)} \}$$

and

$$Y^{[m]} = \{Y^{(1)}, \ldots, Y^{(m)} \}$$

drawn respectively from $X$ and $Y$, the empirical cumulative distribution functions (ECDFs) of the samples are

$$F_{X^{[n]}}(x) = \frac{1}{n} \sum_{i=1}^{n} \delta(X^{(i)} \leq x),$$

$$G_{Y^{[m]}}(y) = \frac{1}{m} \sum_{i=1}^{m} \delta(Y^{(i)} \leq y),$$

where $\delta(\cdot)$ is the indicator function. Intuitively, the different $\gamma_{X,Y}$ can be statistically estimated by (as shown in Figure 6)

$$\delta_{X^{[n]}, Y^{[m]}} = \|F_{X^{[n]}} - G_{Y^{[m]}}\|_\infty.$$ 

When the numbers of samples $n, m \to \infty$, the ECDFs converge to the CDFs: $F_{X^{[n]}} \to F$ and $G_{Y^{[m]}} \to G$, and thus, $\delta_{X^{[n]}, Y^{[m]}} \to \gamma_{X,Y}$. Therefore, for the hypothesis testing problem (8), we propose the statistics assertion

$$A(X^{[n]}, Y^{[m]}) = \begin{cases} H_0, & \text{if } \delta_{X^{[n]}, Y^{[m]}} < c \\ H_1, & \text{if } \delta_{X^{[n]}, Y^{[m]}} > c \end{cases}$$

**Remark 2.** The assertion (12) in our EKS test may look similar to the classic KS test [23]. However, the meaning of the threshold $c$ is different: in the EKS test, $c$ is the parameter (of approximate equality) from (9), while in the classic KS test, $c$ is related to the confidence level.

For random samples $X^{[n]}$ and $Y^{[m]}$, the probability $\alpha$ that the assertion (12) agrees with the correct answer to the hypothesis testing problem (8) is called the confidence level. It depends on the discrepancy between $\gamma_{X,Y}$ and $\delta_{X^{[n]}, Y^{[m]}}$, which is bounded by

$$d_{X^{[n]}, Y^{[m]}} = \| (F_{X^{[n]}} - F) - (G_{Y^{[m]}} - G) \|_\infty$$

due to the triangle inequality

$$|\delta_{X^{[n]}, Y^{[m]}} - \gamma_{X,Y}| \leq d_{X^{[n]}, Y^{[m]}}.$$ 

When the numbers of samples $n, m \to \infty$, the discrepancy $d_{X^{[n]}, Y^{[m]}} \to 0$ with probability 1. But the probability distribution of the rescaled discrepancy $d_{X^{[n]}, Y^{[m]}} \sqrt{mn/(m+n)}$ (for random samples $X^{[n]}, Y^{[m]}$) is asymptotically invariant of $n, m$ and is independent of the CDFs $F$ and $G$, as formally stated below.

**Lemma 2** (Section 7.9 of [23]). The CDF $H(x)$ of the $d_{X^{[n]}, Y^{[m]}} \sqrt{mn/(m+n)}$ from (13) obeys the Kolmogorov-Smirnov distribution

$$H(x) = 1 - 2 \sum_{i=1}^{\infty} (-1)^{i-1} e^{-2i^2x^2} \approx 1 - 2e^{-2x^2}.$$ 

Following Lemma 2 and (14), for any $\lambda > 0$ and random samples $X^{[n]}$ and $Y^{[m]}$, we have

$$\text{Pr}_{X^{[n]}, Y^{[m]}} (|\delta_{X^{[n]}, Y^{[m]}} - \gamma_{X,Y}| < \lambda) \leq \text{Pr}_{X^{[n]}, Y^{[m]}} (d_{X^{[n]}, Y^{[m]}} < \lambda) = H(\lambda \sqrt{mn/(m+n)}).$$

Thus, given any value of $\delta_{X^{[n]}, Y^{[m]}}$ that is less than $c$, the confidence level for asserting $\gamma_{X,Y} < c$ is at least $H((c - \delta_{X^{[n]}, Y^{[m]}}) \sqrt{mn/(m+n)})$. Similarly, given any value of

1For this scalar case, we write $X$ and $Y$ as $X$ and $Y$.

2This is generally not true for multi-dimension.

3More precisely, this is convergence in distribution.
Algorithm 1 EKS test.

Require: Desired confidence \(\alpha_d > 0\), \(c \in (0, 1)\), \(k_1, k_2 \in \mathbb{N}\).
1: Sample sizes \(n, m \leftarrow 0\), \(\alpha \leftarrow 0\).
2: while \(\alpha < \alpha_d\) do
3: \hspace{1em} Draw \(k_1, k_2\) new samples from \(X, Y\), respectively.
4: \hspace{1em} \(n \leftarrow n + k_1, m \leftarrow m + k_2\)
5: \hspace{1em} Update \(\delta^{X, Y}_n, m\) by (11).
6: \hspace{1em} Update \(\alpha\) by (16).
7: end while
8: return \(A\) by (12).

\(\delta^{X, Y}_n, m\) that is greater than \(c\), the confidence level for asserting \(\gamma_{X, Y} > c\) is at least \(H(\delta^{X, Y}_n, m - c)\sqrt{mn/(m+n)}\). In sum, the confidence level \(\alpha\) for the assertion (12) satisfies

\[
\alpha \geq H(\delta^{X, Y}_n, m - c)\sqrt{mn/(m+n)} \tag{16}
\]

Based on this, for the desired confidence level \(\alpha_d\), the EKS test is deployed sequentially as in Algorithm 1. Iteratively, the algorithm draws \(k_1\) and \(k_2\) new samples from the two CDFs \(F\) and \(G\), respectively, and then computes the actual confidence level \(\alpha\) from (16). It terminates when \(\alpha \leq \alpha_d\), and then returns the assertion by (12). This is formally captured by the following theorem.

**Theorem 1.** Algorithm 1 terminates with probability 1 and has the confidence level \(\alpha_d\).

**Proof.** Termination: As \(n, m \to \infty\), we have \(\delta^{X, Y}_n, m\to \gamma_{X, Y} \neq c\), i.e., \(\delta^{X, Y}_n, m\) converges to some value that is not \(c\) with probability 1, so either \(H_0\) or \(H_1\) holds. Therefore, Algorithm 1 terminates with probability 1.

Correctness: Let \(\tau\) be the step Algorithm 1 terminates and \(A\) be “the assertion \(A\) from (12) is correct”, then \(\Pr(A) = \sum_{i \in \mathbb{N}} \Pr(A|\tau = i)\Pr(\tau = i)\). From (16), for any \(i \in \mathbb{N}\), we have that \(\Pr(A|\tau = i) > \alpha_d\). In addition, by Termination, we have that \(\sum_{i \in \mathbb{N}} \Pr(\tau = i) = 1\). Therefore, it holds that \(\Pr(A) \geq \alpha_d\).

**Remark 3.** Our EKS test is implemented sequentially to achieve any given significance level the classic KS test is for fixed samples, while the classic KS test is used for a fixed number of samples and only guarantees the confidence level when the two test distributions are the same.

2) Multidimension: Similarly to the scalar case, for random vectors \(\mathcal{X}\) and \(\mathcal{Y}\), let \(\{X^{(1)}, \ldots, X^{(n)}\}\) and \(\{Y^{(1)}, \ldots, Y^{(m)}\}\) be two sets of i.i.d. samples from \(\mathcal{X}\) and \(\mathcal{Y}\), respectively. Then, we can define the ECDF and the complimentary ECDFs by

\[
F^n_K(a) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(\pi(X^{(i)}) \leq \pi(a))_1, \tag{17}
\]

\[
\ldots, \pi(X^{(i)})K \leq \pi(a)K. \tag{18}
\]

for each \(K\)-dimensional alternation \(\pi \in \Pi_K\) (given by Definition 4). Similarly, we can define \(G^n_m(a)\) for \(Y\).

Following [24], [31], we note that generally \(\|F^n_K - G^n_m\|_\infty\) are not equal for all \(\pi \in \Pi_K\). Thus, defining the test statistics by only using the CDFs \(F_n\) and \(G_m\) by \(\delta^{X, Y}_n, m = \|F_n - G_m\|_\infty\), as in (13), is not enough. Instead, the test statistics should take all the CDFs and complimentary CDFs by

\[
\delta^{X, Y}_n, m = \max_{\pi \in \Pi_K} \|F^n_K - G^n_m\|_\infty. \tag{18}
\]

Then, by [24], [31], the test statistics \(\delta^{X, Y}_n, m\) satisfies Lemma 2 and asymptotically obeys the Kolmogorov-Smirnov distribution from (15). Therefore, the classical test (12) extends to the multidimensional case by using \(\delta^{X, Y}_n, m\) from (18). The confidence level for (18) can be computed from [24], [31].

**B. Verification of Probabilistic Conformance Using EKS Test**

Using the EKS test introduced in Section IV-A, we return to the problem of checking the probabilistic conformance of a monotonically parametrized STL formula on a PUS with given inputs (formulated in Section III), and propose a statistical verification algorithm solve it. To demonstrate our algorithm, and as with most other works (e.g., [20], [21]), we focus on bounded-time properties; handling unbounded-time properties is more involving, and is an avenue for future work.

Following Definition 2, for a monotonically parametrized STL formula \(\phi_d\) with \(d \in \mathbb{R}^K\) and for each \(K\)-dimensional alternation \(\pi \in \Pi_K\) (from Definition 4), let

\[
F^n_{\pi}(d) = \Pr_{\pi_1 \sim \mathcal{M}_1, \sigma_1 \leftarrow \phi_{\pi}(d)) \tag{19}
\]

where the monotonicity of \(\phi_d\) and \(\gamma_{X, Y}\). Similarly, we can define the ECDF and the complimentary ECDFs by

\[
G^n_{\pi}(d) = \Pr_{\pi_2 \sim \mathcal{M}_2, \sigma_2 \leftarrow \phi_{\pi}(d)) \tag{19}
\]

By the monotonicity of \(\phi_d\) from Definition 2, for each \(\pi \in \Pi_K\), the multivariate functions \(F^n_{\pi}\) is the CDF or a complementary CDF of the satisfaction probability of \(\phi_d\) for the parameter \(d\), and the same holds for \(G^n_{\pi}\).

From Definition 2, the two PUSs \(\mathcal{M}_1\) and \(\mathcal{M}_2\) conform with respect to the monotonically parametrized formula \(\phi_d\), if these CDFs and complimentary CDFs are approximately equal i.e.,

\[
|\Pr_{\pi_1 \sim \mathcal{M}_1, \sigma_1 \leftarrow \phi} - \Pr_{\pi_2 \sim \mathcal{M}_2, \sigma_2 \leftarrow \phi}| < c
\]

if and only if

\[
\gamma_{X, Y} = \max_{\pi \in \Pi_K} \|F^n_{\pi} - G^n_{\pi}\|_\infty < c. \tag{20}
\]

This can be tested by the EKS test introduced in Section IV-A.

Specifically, for two sets of sample paths \(\{\sigma_1^{(1)}, \ldots, \sigma_1^{(n)}\}\) and \(\{\sigma_2^{(1)}, \ldots, \sigma_2^{(m)}\}\) from the PUSs \(\mathcal{M}_1\) and \(\mathcal{M}_2\), respectively, we define the empirical approximations of \(F(d)\) and \(G(d)\) by

\[
F^n_{\pi}(d) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(\sigma_1^{(i)} \leftarrow \phi_{\pi}(d)) \tag{21}
\]

\[
G^n_{\pi}(d) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}(\sigma_2^{(i)} \leftarrow \phi_{\pi}(d)) \tag{21}
\]
Algorithm 2 Statistical verification for conformance.

Require: Desired confidence level $\alpha_d$, threshold $c > 0$
1: Sample sizes $n, m \leftarrow 0$, $\alpha \leftarrow 0$
2: while $\alpha < \alpha_d$ do
3: Draw new samples from $\mathcal{M}_1, \mathcal{M}_2$ and update $n, m$.
4: Update $F_{n}^{\pi}, G_{m}^{\pi}$ by (21) and compute $\delta_{n, m}$ by (22).
5: Update $\alpha$ by [24], [31].
6: end while
7: return $\mathcal{A}$ by (12).

where $\pi \in \Pi_K$ and $I(\cdot)$ is the indicator function. Similarly to (18), the test statistics

$$\delta_{n, m} = \max_{\pi \in \Pi_K} \| F_{n}^{\pi} - G_{m}^{\pi} \|_{\infty},$$

(22)
satisfies Lemma 2 and obeys the Kolmogorov-Smirnov distribution from (15) (asymptotically for $K \geq 2$); hence, the statistical test (12) applies. Given the sample paths, $\delta_{n, m}$ is the $L_\infty$ norm of two known step functions, and thus can be computed explicitly.

To summarize, we propose Algorithm 2 for checking probabilistic conformance. It terminates with probability 1 and can achieve any desired confidence level $\alpha_d < 1$. The proof follows from that of Theorem 1.

V. CASE STUDIES AND EVALUATION

We evaluated our statistical verification algorithms on three CPS benchmarks with complex dynamics from a wide range of application domains: (1) Toyota Powertrain, (2) lane-keeping Assistant (LKA) Controllers, and (3) 100kW Grid-Connected Photo Voltaic (PV) Array, to demonstrate the utility of our statistical verification method for probabilistic conformance of parameterized STL formulas. We find the case study in Section V-B particularly important since, to the best of our knowledge, previously there are few comparative studies between NN-based and conventional techniques in cyber-physical and embedded systems.

The Toyota powertrain model is derived from [25]. The LKA is implemented in MATLAB using the MPC, Deep Learning, and Reinforcement Learning Toolboxes [32]. The PV Array is implemented using the Simscape Power Systems toolbox [33]. All implementations are available at [34].

Evaluations are performed on a laptop with Intel Xeon E-2176M CPU @ 2.7GHz and 16 GB RAM. For each benchmark, we run Algorithm 2 with different indifference parameter $c$ and desired confidence level $\alpha_d$ (i.e., the probability for Algorithm 2 to return the correct assertion is at least $\alpha_d$). We report the test statistics $\delta_{n, m}$, number of samples, total algorithm execution time, and the assertion $\mathcal{A}$ when the algorithm terminates.

A. Toyota Powertrain

We use the Simulink models for the Toyota Powertrain with a four-mode embedded controller and 15 state variables from [25]. It is challenging to show that complex embedded/CPS with hybrid dynamics, such as the powertrain, satisfy strict performance requirements. On the one hand, the available benchmark model must capture a reasonable portion of behaviors of the real powertrain to enable us to assess, evaluate, and verify the designs against requirements. On the other hand, the simulation time for a simpler model that sufficiently conforms with the real system is significantly lower.

In [25], two models of the Toyota Powertrain are presented. A detailed but complex model contains an air to fuel (A/F) ratio controller and an average model of the engine dynamics, such as the throttle and intake manifold air dynamics. Due to the complexity of this detailed model and limitations of existing verification tools, in [25], a simpler abstract model as a hybrid I/O automaton is also introduced to facilitate system analysis, including formal verification.

Conformance: For the Toyota powertrain, the A/F ratio control problem is of key interest. Hence, we study the conformance for the A/F deviations $e_{A/F}$ for the detailed and abstract models for an RPM of 1600 (the system input). When the nominal input RPM is subject to Gaussian system noise $N(0, 18^2)$, (samples of) the change of $e_{A/F}$ over time for the two models are given in Figure 7. The conformance requires that, under this system noise, the A/F deviations $e_{A/F}$ of the detailed and abstract models enter some desired working region $(|e_{A/F}| < 0.05)$ in any time interval $[0.22, \tau]$ with approximately the same probability; i.e., the STL specification $	op_{[0.22, \tau]}(|e_{A/F}| < 0.05)$ holds with approximately the same probability for any $\tau$ between the two models, as formally captured below.

$$\forall \tau \geq 0. \Pr_{\sigma\sim\mathcal{M}_f} (|e_{A/F}| < 0.05) \approx_c \Pr_{\sigma\sim\mathcal{M}_d} (|e_{A/F}| < 0.05).$$

(23)

Here, the constant $c > 0$, the approximate equality $\approx_c$ means the difference is less than $c$, the subscripts $f$ and $d$ stand for

$^4$More precisely for $\tau \geq 0.22$ from (23). Otherwise, the satisfaction probability is trivially 0.
the complex and abstracted models, respectively, \( e_{A/F} \) is the percentage deviation of A/F ratio, and \( \tau \) is the time bound.

**Result Analysis:** We analyzed (23) using Algorithm 2 with the confidence level \( \alpha \in \{0.95, 0.99\} \) and the conformance parameter \( c \in \{0.2, 0.15, 0.10, 0.05\} \) (see Table I). The results are derived with relatively small numbers of samples for all confidence and indifference parameters. The results indicate that the two employed models do not conform for the requirement (23), although it is claimed in [25] that the abstract model is a representative of the detailed model.

Starting from the initial RPM values, the predicted A/F ratio in the complex model would take more time to reach the desired working region than in most cases in the abstracted model. This agrees with Figure 7, as the A/F ratio of the abstracted model would remain inside the desired area, while in the complex model, this value exceeds the desired region’s boundary in most of the cases. Furthermore, from Table I, the value of the test statistics \( \delta_{n,m} \) is almost 1 in all the cases, when Algorithm 2 terminates. This implies that for the detailed and abstracted models, the distribution of the startup time for their A/F ratio to reach working region are very different — this agrees with the algorithm assertion.

### B. Replacing MPC with NN

The controller of the LKA system is commonly based on model predictive control (MPC) or more recently neural networks (NN). The conventional MPC-based controllers solve a constrained quadratic programming optimization problem from the observed state of a plant in a open-loop fashion. This approach is usually computationally ineffective in real-time. Recently, NN-based controllers are employed to imitate the control rules of MPC-based controller from samples, in order to improve the real-time computation efficiency. In this case study, we check the conformance of a NN-based controller and an MPC-based controller for the LKA system in MATLAB/Simulink [32].

In the LKA system, the sensors measure the lateral deviation, relative yaw angle between the center-line of a lane and the vehicle, current lane curvature, and its derivative. The objective of the controller is to keep the lateral error and relative yaw angle close to zero. To dynamics of the vehicle is given by the three Degrees-of-Freedom (DoF) bicycle model [35] as

\[
\begin{bmatrix}
\dot{V}_y(t) \\
\psi(t)
\end{bmatrix} = 
\begin{bmatrix}
-\frac{2C_l f + 2C_r r}{m V_x} - V_x - \frac{2C_l f - 2C_r r}{I_x V_x} \\
\frac{C_l f}{I_x}
\end{bmatrix} u(t) 
+ 2 \frac{C_l m}{I_x} \left[ \frac{C_l m}{I_x} \right] V_y.
\]

Here, \( V_x \) is the longitudinal velocity, \( m \) is the total vehicle mass, \( l_x \) is the yaw moment of inertia of the vehicle, \( l_f \) and \( l_r \) are the longitudinal distance from the center of gravity to the front and rear tires, and \( C_f \) and \( C_r \) are the cornering stiffness of the front and rear tires, respectively. The system state consist of the lateral velocity \( V_y \) and yaw angle rate \( \psi \), and the front steering angle \( u(t) \) is the system input.

**MPC:** The MPC-based controller is derived from the MPC toolbox in MATLAB. The values of the variables are set as follows: \( V_x = 15 \) m/s, \( m = 1575 \) kg, \( I_x = 2875 \) m\cdot N\cdot s\(^2\), \( l_f = 1.2 \) m, \( l_r = 1.6 \) m, \( C_f = 19000 \) N/rad, and \( C_r = 33000 \) N/rad. The controller output is confined within the interval [\(-\pi/3, \pi/3\)] rad. The predictive time horizon and control time horizon are set to \( h_p = 20 \) and \( h_c = 20 \).

**DNN Replacement:** We train a NN controller to replace the MPC controller, by sampling from the MPC based controller for randomly generated states, last control action, and measured disturbances. The samples are divided into the training and validation testing data, and are used to train several NNs with similar structure, but different numbers of neurons per layer (30, 45, 60, and 300 neuron per layer). All middle layers are fully connected with ReLU activation function and the output layer is a fully connected layer with tanh activation function and a scalar layer. The maximum number of epoch to stop the training is set to 30. The structure of the NNs is shown in Figure 8.

**Conformance:** For the input of the same reference path of the vehicle (given by the Matlab Toolbox), we expect that using the NN controller the lateral deviation of the vehicle under random values of the initial states should be similar to the output of the MPC-based closed-loop system. Thus, we assign an upper bound to the error of the lateral deviation and check when the designed controller reaches this boundary. With fixed values of initial states, we run the closed-loop system with two NNs and the reference MPC. Then, we compare the time that the absolute value of the lateral deviation falls below the desired value for the NNs controller and the MPC’s controller; this is formally captured by the STL formula

\[
\Box_{[0,\tau]}(|e_y| < \gamma)
\]

monotonically parametrized by \( \tau \). Accordingly, the conformance between the MPC-controlled and NN-controlled LKA systems for this parametrized specification is

\[
\forall \tau \geq 0, \Pr_{\sigma_1 \sim \text{NN}}(\sigma_1 \models \Box_{[0,\tau]}(|e_y|_{\text{NN}} < \gamma)) 
\approx_c \Pr_{\sigma_2 \sim \text{MPC}}(\sigma_2 \models \Box_{[0,\tau]}(|e_y|_{\text{MPC}} < \gamma)),
\]

where the constants \( c, \gamma > 0 \), the approximate equality \( \approx_c \) means the difference is less than \( c \), and \( e_y \) is the lateral deviation of the intended controller. The random signals \( \sigma_1 \) and \( \sigma_2 \) are derived as follows. The initial conditions

| \( c \) | \( 1 - \alpha \) | \( \delta_{n,m} \) | Samples | Time (sec.) | \( \mathcal{A} \) |
|---|---|---|---|---|---|
| 0.40 | 0.01 | 1.00 | 3.9e+01 | 1.8e-02 | False |
| 0.40 | 0.05 | 1.00 | 1.9e+01 | 4.4e-03 | False |
| 0.25 | 0.01 | 1.00 | 2.5e+01 | 4.6e-03 | False |
| 0.25 | 0.05 | 1.00 | 1.3e+01 | 2.2e-03 | False |
| 0.10 | 0.01 | 1.00 | 1.8e+01 | 3.6e-03 | False |
| 0.10 | 0.05 | 1.00 | 9.0e+00 | 1.6e-03 | False |
| 0.05 | 0.01 | 1.00 | 1.6e+01 | 2.8e-03 | False |
| 0.05 | 0.05 | 1.00 | 8.0e+00 | 1.3e-03 | False |
of the system such as the lateral velocity $V_p$, yaw angle rate $\dot{\psi}$, lateral deviation $e_1$, relative yaw angle $e_2$, last steering angle $u$, and the measured road yaw rate $V_{\phi}$. They are drawn randomly using the uniform distribution from intervals $[-2, 2]$ m/s, $[-\pi/3, \pi/3]$ rad/s, $[-1, 1]$ m, $[-\pi/4, \pi/4]$ rad, $[-\pi/3, \pi/3]$ rad, and $[-0.01, 0.01]$, respectively. The minimum road reduces is 100 m.

Result Analysis: The results for applying Algorithm 2 with parameters $\alpha \in \{0.95, 0.99\}$, and $c \in \{0.40, 0.25, 0.10, 0.05\}$ are shown in Table II for NN controllers with 30 and 45 neurons per layer. As can be seen, the NN controllers with 45 neurons per layer conforms much better with the MPC controller than the NN controllers with 30 neurons per layer for the requirement (25). The results for 60 and 300 neurons per layer are similar, so they are omitted due to space limit. All these results are achieved with a relatively small number of samples (at most a few thousands samples for each setup).

The results of Table II implies that increasing the size of the NN-based controllers improves the conformance with the MPC controller. To check this observation and confirm the results of Table II, we show in Figure 9 the ECDFs of the settling time for the MPC controller and the NN controllers with 30, 45, 60, and 300 neurons per layer. The conformance for the requirement (25) is visually demonstrated by the closeness of the ECDFs. To derive the same conclusion, each ECDF uses 200 samples, which is significantly more than the samples required by Algorithm 2, as shown in Table II. As shown in Figure 9, increasing the number neurons beyond 45 does not lead to considerable change in the CDF of the settling times for NN based controllers. Comparing to NN3000, the NN60 controller has better conformance with the MPC. The latter implies that NN300 controller has the over-fitting problem. For the NN-based controllers of different sizes, the test statistics upon algorithm termination is $\delta_{NN300} = 0.98$, $\delta_{NN300} = 0.31$, $\delta_{NN60} = 0.31$, and $\delta_{NN300} = 0.35$.

C. Power Plant

Finally, we compare the detailed and average models of a 100kW array connected to a 25kV grid via a DC-DC boost converter and a three-phase three-level Voltage Source Converter (VSC), from the MATLAB Simscape Electrical Toolbox [27]. Both models include a Photo Voltaic (PV) array that delivers the maximum power of 100 kW at 1000 W/m² sun irradiance, a DC-DC boost converter, 3-level 3-phase VSC, capacitor bank, three-phase coupling transformer, and a given utility grid. The models use the Simulink model of a boost converter to implement the Maximum Power Point Tracking (MPPT). The MPPT optimizes the match between the solar array (PV panels) and the utility grid. The models have differences such as employed technique to implement MPPT, DC-DC, and VSC converters’ structure [36].

The VSC converts the 500V DC link voltage to 260V AC and keeps unity power factor. To this end, two control loops are employed: one control loop regulates DC link voltage to ±250V (external controller) and the other control loop regulates active and reactive grid currents (internal controller). The active current reference is the output of the DC voltage external controller. The latter controller is a PI controller whose input is the error of the DC voltage.

Conformance: We consider the deviation of the DC voltage $e_{dc}$, when the sun irradiance $\varepsilon_{dc}$ and environment temperature are subject to changes. For an arbitrary threshold $\gamma$, we use the STL specification $\square_{0.5, 2}(|e_{dc}| < \gamma)$, which is monotonically parametrized by $\gamma$, to capture that $e_{dc}$ is always below $\gamma$ within the time interval $[0.5, 2]$ of interest. Accordingly, the conformance between the detailed and average models for this parametrized specification is captured by

$$\forall \gamma \geq 0. \Pr_{\sigma_d \sim M_d}(\sigma_d \models \square_{0.5, 2}(|e_{dc}| < \gamma)) \approx_c \Pr_{\sigma_a \sim M_a}(\sigma_a \models \square_{0.5, 2}(|e_{dc}| < \gamma)).$$

(26)

where the constant $c > 0$, the approximate equality $\approx_c$ means the difference is less than $c$, and the detailed and average models are denoted by $d$ and $a$, respectively.
TABLE II: Statistical verification results for the conformance property (25) and the test statistics \( \delta_{n,m} \) upon Algorithm 2 termination for different conformance parameter \( c \) and confidence level \( \alpha \).

| \( c \) | \( 1 - \alpha \) | \( \delta_{n,m} \) | Samples | \( T(s) \) | \( A \) | \( \delta_{n,m} \) | Samples | \( T(s) \) | \( A \) |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0.40 | 0.01 | 0.98 | 4.3e+01 | 7.4e-03 | False | 0.36 | 1.0e+04 | 9.6e+00 | True |
| 0.40 | 0.05 | 1.00 | 1.9e+01 | 3.1e-03 | False | 0.36 | 3.6e+03 | 2.0e+00 | True |
| 0.25 | 0.01 | 1.00 | 2.5e+01 | 4.1e-03 | False | 0.37 | 9.5e+02 | 3.2e-01 | False |
| 0.25 | 0.05 | 1.00 | 1.3e+01 | 2.1e-03 | False | 0.42 | 2.5e+02 | 5.9e-02 | False |
| 0.10 | 0.01 | 1.00 | 1.8e+01 | 3.0e-03 | False | 0.36 | 2.1e+02 | 4.2e-02 | False |
| 0.10 | 0.05 | 1.00 | 9.0e+00 | 1.6e-03 | False | 0.35 | 1.2e+02 | 2.2e-02 | False |
| 0.05 | 0.01 | 1.00 | 1.6e+01 | 2.7e-03 | False | 0.38 | 1.3e+02 | 2.5e-02 | False |
| 0.05 | 0.05 | 1.00 | 8.0e+00 | 1.2e-03 | False | 0.36 | 7.3e+01 | 1.4e-02 | False |

TABLE III: Statistical verification results of the conformance property (26) and the test statistics \( \delta_{n,m} \) upon Algorithm 2 termination, for different conformance parameter \( c \) and confidence level \( \alpha \).

We applied Algorithm 2 with parameters \( \alpha \in \{0.95, 0.99\} \) and \( c \in \{0.001, 0.005, 0.01, 0.05\} \). For both the models, we consider the standard test conditions (initial temperature and irradiance are 25° and 1000 W/m², respectively) with the following scenario (i.e., the input to the models):

1) At \( t = 0.3 \)s MPPT starts to regulate PV voltage.
2) In time interval \([0.6, 1.1]\)s, the sun irradiance linearly is ramped to a minimum value. Also, the environment temperature start increasing to a maximum value, simultaneously.
3) In time interval \([1.1, 1.2]\)s, the sun irradiance and environment temperature stay constant. The minimum value of the irradiance is drawn randomly from a distribution \( \mathbb{N}_{ir} (650, 10^2) \) and the maximum temperature is \( 20 - 0.02 \times \mathbb{N}_{ir} (650, 10^2) \).
4) In time interval \([1.2, 1.7]\)s, the sun irradiance and temperature are linearly restored back to 1000W/m² and 25°, respectively; from then onward, remain constant.

Result Analysis: Table III contains the results that demonstrate the nonconformance of the detailed and average models for the requirement (26), although it is commonly believed that the average model is generally a good approximation of the detailed model [27]. This result is achieved with a relatively small number of samples (at most a few dozen samples for each setup). The results for the considered specification reveals that two models do not have conformance for any values of \( c \). To confirm the results of Table III, we plot in Figure 10 the ECDFs of the maximum deviation \( |e_{V_{dc}}| \) of the detailed and average models; the discrepancy of the two ECDFs demonstrates the nonconformance of two models for the requirement (26). Each ECDF uses 100 samples, which is significantly more than the samples required by Algorithm 2, as shown by Table III.

![ECDFs comparison](image-url)

Fig. 10: The ECDFs for the maximum deviation of \( V_{dc} \) in the detailed and average models for 100 samples. The two distributions of the maximum errors for the models are noticeably different.

VI. Related Work

Conformance of CPS for different types of specifications of interest is studied in [1]–[6]. As in [10], [11], in this work, we focus on a class of conformance properties for CPS that are specified by temporal logic formulas. Our notion of conformance can be viewed as the probabilistic extension of [10], [11], that is needed to allow for capturing the conformance between a wide class of probabilistic CPS (which we model as PUSs). Since reachability properties can be in general captured by temporal logic formulas, our notion of conformance is more general than the conformance for reachability from [4], [5].

Existing works on conformance for temporal logic specifications mainly focus on non-probabilistic models [3], [5], [6], [10], [11]. On the other hand, in this work, we focus on a probabilistic notion of conformance – the satisfaction probability of the specifications of interest should be approximately equal. In [10], [11], conformance builds a relation between two models such that if any STL formula holds on one model, then the corresponding formula should automatically hold on the other model. Conceptually, our notion of conformance is less stringent, as it only involves a given set of STL formulas of interest. Furthermore, our notion of conformance is conceptually more general than [4], [5], where the conformance is only for reachability. Our notion of conformance can specify the conformance of probabilistic reachability for two models.

Conformance is different from the simulation/bisimulation [37] in two aspects. Conceptually, conformance focuses on the level of functionality, and only captures the similarity between two models with respect to a certain set of specifications of interest. That is, the behavior of the two models may
be very different for other specifications (not of interest). On the other hand, the simulation/bisimulation focuses on the level of executions, and requires an execution-wise correspondence between the two models. Also, the two concepts have slightly different domains of applications [3], [10], [11]. Conformance is commonly only used for cyber-physical and embedded control systems, while simulation/bisimulation may be used for both discrete models [37] and cyber-physical/embedded control systems [38], [39].

To the best of our knowledge, this is the first work on statistically verifying the probabilistic conformance of CPS with complex dynamics (formally captured as probabilistic uncertain systems from Definition 1), while providing provable confidence levels (i.e., false positive/negative ratios). Existing model-based methods for conformance, such as [3], [5], [10], [11] cannot directly handle such systems with complex or even unknown dynamics in practice. On the other hand, existing testing methods for conformance for temporal logic specifications [4], [6] or for other types of conformance [1], [2] cannot provide the probabilistic guarantees like the presented method. Therefore, those methods are not directly comparable with ours for the case studies presented in Section V.

VII. CONCLUSION AND FUTURE WORK

In this paper, we proposed a new concept called probabilistic conformance for CPS. This notion is based on approximately equal satisfaction probabilities for a given (infinite) set of signal temporal logic (STL) formulas. We proposed a verification algorithm for the probabilistic conformance of grey-box CPS, modeled by probabilistic uncertain systems. Our statistical verification algorithm is based on an extended Kolmogorov-Smirnov test that can check if two probability distributions are equal for any desired confidence level (lower than 1). Finally, we applied the proposed statistical verification algorithm to check the probabilistic conformance of (1) the startup time of the full and simplified models of the Toyota powertrain system, (2) the settling time of a model predictive control (MPC) based lane-keeping controller and several neural networks (NN) based lane-keeping controllers of different sizes for an autonomous car, and (3) the maximal deviation of DC voltage between the full model and a simplified model of a power grid system.

REFERENCES

[1] M. Ryabtsev and O. Strichman, “Translation validation: From simulink to c,” in International Conference on Computer Aided Verification. Springer, 2009, pp. 696–701.
[2] R. Majumdar, I. Saha, K. Ueda, and H. Yazarel, “Compositional equivalence checking for models and code of control systems,” in 52nd IEEE Conference on Decision and Control, 2013, pp. 1564–1571.
[3] N. Khakpour and M. R. Mousavi, “Notions of Conformance Testing for Cyber-Physical Systems: Overview and Roadmap,” in 26th Int. Conf. on Concurrency Theory (CONCUR 2015), vol. 42, 2015, pp. 18–40.
[4] H. Boehm, J. Oehlerking, M. Woehrle, and M. Althoff, “Reachset Conformance Testing of Hybrid Automata,” in 19th Int. Conf. on Hybrid Systems: Computation and Control (HSCC), 2016, pp. 277–286.
[5] S. B. Liu and M. Althoff, “Reachset Conformance of Forward Dynamic Models for the Formal Analysis of Robots,” in IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS), 2018, pp. 370–376.
[6] A. Graf-Brill and H. Hermanns, “Component-aware input-output conformance,” in International Conference on Formal Techniques for Distributed Objects, Components, and Systems, 2019, pp. 111–128.
[7] L. Heerink and J. Tretmans, “Formal methods in conformance testing: A probabilistic refinement,” in International Workshop on Testing of Communicating Systems. Springer US, 1996, pp. 261–276.
[8] N. López, M. Núñez, and I. Rodríguez, “Specification, testing and implementation relations for symbolic-probabilistic systems,” Theoretical Computer Science, vol. 353, no. 1-3, pp. 228–248, 2006.
[9] O. Maler and D. Nickovic, “Monitoring temporal properties of continuous signals,” in Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems. Springer, 2004, pp. 152–166.
[10] H. Abbas, H. Mittelmann, and G. Fainekos, “Formal property verification in a conformance testing framework,” in 12th ACM/IEEE Conf. on Formal Methods and Models for Codeign (MEMOCODE), 2014, pp. 155–164.
[11] J. V. Deshmukh, R. Majumdar, and V. S. Prabhu, “Quantifying conformance using the Skorokhod metric,” Formal Methods in System Design, vol. 50, no. 2, pp. 168–206, 2017.
[12] Wikipedia, “Volkswagen emissions scandal,” https://en.wikipedia.org/wiki/Volkswagen_emissions_scandal, 2020, accessed: 2020-1-15.
[13] G. Barthe, P. R. D’Argenio, B. Finkbeiner, and H. Hermanns, “Facets of Software Doping,” arXiv:1803.10154 [cs], 2018.
[14] S. Pinisetty, G. Schneider, and D. Sands, “Runtime verification of hyperproperties for deterministic programs,” in Proceedings of the 6th Conference on Formal Methods in Software Engineering - FormaliSSE ’18, 2018, pp. 20–29.
[15] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, Y. Yan, E. Theodorou, and B. Boots, “Agile autonomous driving using end-to-end deep imitation learning,” arXiv preprint arXiv:1709.07174, 2017.
[16] Y. Wang, M. Zarei, B. Bonakdarpour, and M. Pajic, “Statistical verification of hyperproperties for cyber-physical systems,” ACM Transactions on Embedded Computing Systems, vol. 18, no. 5s, pp. 1–23, 2019.
[17] T. A. Henzinger, “The theory of hybrid automata,” in Verification of Hybrid and Digital Systems. Springer, 2000, pp. 265–292.
[18] N. Rouhi, Y. Wang, M. West, G. E. Dullerud, and M. Viswanathan, “Statistical Verification of the Toyota Powertrain Control Verification Benchmark,” in Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control, 2017, pp. 65–70.
[19] E. Asarin, A. Donzé, O. Maler, and D. Nickovic, “Parametric identification of temporal properties,” in International Conference on Runtime Verification, 2011, pp. 147–160.
[20] G. Agha and K. Palmkosk, “A Survey of Statistical Model Checking,” ACM Trans. Model. Comput. Simul., vol. 28, no. 1, pp. 6:1–6:39, 2018.
[21] K. G. Larsen and A. Legay, “Statistical Model Checking: Past, Present, and Future,” in Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques, 2016, pp. 3–15.
[22] A. Legay, B. Delahaye, and S. Bensalem, “Statistical Model Checking: Hyperproperties and Beyond,” in Runtime Verification, 2010, pp. 641–648.
[23] J. V. Deshpande, U. Naik-Nimbalkar, and I. Dewan, Nonparametric Statistics: Theory and Methods, 2018.
[24] J. A. Peacock, “Two-dimensional goodness-of-fit testing in astronomy,” Monthly Notices of the Royal Astronomical Society, vol. 202, no. 3, pp. 615–627.
[25] X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. Butts, “Powertrain control verification benchmark,” in The 17th International Conference on Hybrid Systems: Computation and Control, 2014, pp. 253–262.
[26] MathWorks, Inc., “Model predictive control toolbox,” https://www.mathworks.com/help/mpc/ug/lane-keeping-assist-system-using-model-predictive-control.html, 2019, accessed: 2019-7-15.
[27] ——, “Powersim,” https://www.mathworks.com/help/physmod/sps/industrial/examples/250-kw-grid-connected-pv-array.html, 2019, accessed: 2019-7-15.
[34] anonymized authors, “Conformance case studies,” anonymized_url, 2020, accessed: 2020-1-15.
[35] T. D. Gillespie, Fundamentals of vehicle dynamics. Society of automotive engineers Warrendale, PA, 1992, vol. 400.
[36] M. A. De Brito, L. P. Sampaio, G. Luigi, G. A. e Melo, and C. A. Canesin, “Comparative analysis of MPPT techniques for PV applications,” in 2011 International Conference on Clean Electrical Power, 2011, pp. 99–104.
[37] D. Dams and O. Grumberg, “Abstraction and Abstraction Refinement,” in Handbook of Model Checking. Springer International, pp. 385–419.
[38] A. A. Julius and G. J. Pappas, “Approximations of Stochastic Hybrid Systems,” IEEE Transactions on Automatic Control, vol. 54, no. 6, pp. 1193–1203, 2009.
[39] Y. Wang, N. Roohi, M. West, M. Viswanathan, and G. E. Dullerud, “Statistical Verification of Dynamical Systems Using Set Oriented Methods,” in Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, 2015, pp. 169–178.