Conformal Prediction for STL Runtime Verification

Lars Lindemann*
llindema@usc.edu
University of Southern California
Los Angeles, California, USA

Xin Qin*
xinqin@usc.edu
University of Southern California
Los Angeles, California, USA

Jyotirmoy V. Deshmukh
jyotirmoy.deshmukh@usc.edu
University of Southern California
Los Angeles, California, USA

George J. Pappas
pappasg@seas.upenn.edu
University of Pennsylvania
Philadelphia, Pennsylvania, USA

ABSTRACT

We are interested in predicting failures of cyber-physical systems during their operation. Particularly, we consider stochastic systems and signal temporal logic specifications, and we want to calculate the probability that the current system trajectory violates the specification. The paper presents two predictive runtime verification algorithms that predict future system states from the current observed system trajectory. As these predictions may not be accurate, we construct prediction regions that quantify prediction uncertainty by using conformal prediction, a statistical tool for uncertainty quantification. Our first algorithm directly constructs a prediction region for the satisfaction measure of the specification so that we can predict specification violations with a desired confidence. The second algorithm constructs prediction regions for future system states first, and uses these to obtain a prediction region for the satisfaction measure. To the best of our knowledge, these are the first formal guarantees for a predictive runtime verification algorithm that applies to widely used trajectory predictors such as RNNs and LSTMs, while being computationally simple and making no assumptions on the underlying distribution. We present numerical experiments of an F-16 aircraft and a self-driving car.

CCS CONCEPTS

• **Computer systems organization** → **Robotics**; • **Theory of computation** → **Logic and verification**; *Modal and temporal logics*; • **General and reference** → **Verification**.

KEYWORDS

Predictive runtime verification, stochastic system verification, signal temporal logic, conformal prediction.

ACM Reference Format:
Lars Lindemann, Xin Qin, Jyotirmoy V. Deshmukh, and George J. Pappas. 2023. Conformal Prediction for STL Runtime Verification. In *ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023) (ICCPS '23), May 9–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3576841.3585927

*Both authors contributed equally to this research.

ICCPS '23, May 9–12, 2023, San Antonio, TX, USA

ACM ISBN 979-8-4007-0036-1/23/05…$15.00
https://doi.org/10.1145/3576841.3585927

1 INTRODUCTION

Cyber-physical systems may be subject to a small yet non-zero failure probability, especially when using data-enabled perception and decision making capabilities, e.g., self-driving cars using high-dimensional sensors. Rare yet catastrophic system failures hence have to be anticipated. In this paper, we aim to detect system failures with high confidence early on during the operation of the system.

Verification aims to check the correctness of a system against specifications expressed in mathematical logics, e.g., linear temporal logic [55] or signal temporal logic (STL) [53]. Automated verification tools were developed for deterministic systems, e.g., model checking [7, 23] or theorem proving [68, 69]. Non-deterministic system verification was studied using probabilistic model checking [13, 33, 38, 41] or statistical model checking [42, 43, 83, 84]. Such offline verification techniques have been applied to verify cyber-physical systems, e.g., autonomous race cars [36, 37, 47], cruise controller and emergency braking systems [75, 76], autonomous robots [72], or aircraft collision avoidance systems [8, 9].

These verification techniques, however, are: 1) applied to a system model that may not capture the system sufficiently well, and 2) performed offline and not during the runtime of the system. We may hence certify a system to be safe a priori (e.g., with a probability of 0.99), but during the system's runtime we may observe an unsafe system realization (e.g., belonging to the fraction of 0.01 unsafe realizations). Runtime verification aims to detect unsafe system realizations by using online monitors to observe the current realization (referred to as prefix) to determine if all extensions of this partial realization (referred to as suffix) either satisfy or violate the specification, see [12, 19, 45] for deterministic and [39, 70, 80] for non-deterministic systems. The verification answer can be inconclusive when not all suffixes are satisfying or violating. Predictive runtime verification instead predicts suffixes from the prefix to obtain a verification result more reliably and quickly [6, 56, 81].

We are interested in the predictive runtime verification of a stochastic system, modeled by an unknown distribution $\mathcal{D}$, against a system specification ϕ expressed in STL. Particularly, we want to calculate the probability that the current system execution violates the specification based on the current observed trajectory, see Figure 1. To the best of our knowledge, existing predictive runtime

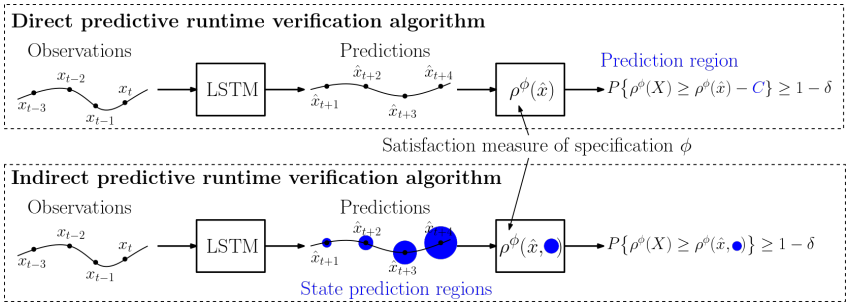

Figure 1: Overview of the proposed STL predictive runtime verification algorithms. Both algorithms use past observations $(x_0, \dots, x_t)$ to obtain state predictions $(\hat{x}_{t+1}, \hat{x}_{t+2}, \dots)$. The direct algorithm calculates the satisfaction measure $\rho^\phi(\hat{x})$ of the specification ϕ based on these predictions, and obtains a prediction region C for the unknown satisfaction measure $\rho^\phi(x)$ using conformal prediction. The indirect method obtains prediction regions for the unknown states $x_{t+1}, x_{t+2}, \dots$ using conformal prediction first, and then obtains a lower of the unknown satisfaction measure $\rho^\phi(x)$ based on the state prediction regions.

verification algorithms do not provide formal correctness guarantees unless restrictive assumptions are placed on the prediction algorithm or the underlying distribution $\mathcal{D}$. We allow the use of complex prediction algorithms such as recurrent neural networks (RNNs) and long short-term memory (LSTM) networks, while making no assumptions on $\mathcal{D}$. Our contributions are as follows:

- We present two predictive runtime verification algorithms that are illustrated in Figure 1 and that use: i) trajectory predictors to predict future system states, and ii) conformal prediction to quantify prediction uncertainty.
- We show that our algorithms enjoy valid verification guarantees, i.e., the verification answer is correct with a user-defined confidence, with minimal assumptions on the predictor and the underlying distribution $\mathcal{D}$. We provide technical proofs of Theorems and Lemmas in an appendix.
- We provide realistic empirical validation of our approach of an F-16 aircraft and a self-driving car, and compare the two proposed runtime verification algorithms.

1.1 Related Work

Statistical model checking. Statistical model checking is a lightweight alternative to computationally expensive probabilistic model checking used to verify black-box systems [42, 43, 83, 84]. The idea is to sample system trajectories and use statistical tools to get valid verification guarantees. Statistical model checking has gained popularity due to the complexity of modern machine learning architectures for which it is difficult to obtain meaningful, i.e., not overly conservative, analytical results.

We focus on signal temporal logic (STL) as a rich specification language [53] that admits robust semantics to quantify how robustly a system satisfies a specification spatially and/or temporally [25, 27, 58]. Statistical model checking under STL specifications was first considered in [10, 11], while [38, 64, 65] proposed a combination of a statistical and a model-based approach. The authors in [59, 78, 79, 87] use statistical testing to derive high confidence bounds on the probability of a cyber-physical system satisfying an STL specification. In [1, 2, 21, 47, 49] risk verification algorithms were proposed using mathematical notions of risk.

Predictive Runtime Verification. Runtime verification complements system verification by observing the current system execution (prefix) to determine if all extensions (suffixes) either satisfy or violate the specification [12, 19, 39, 45, 70, 80]. Runtime verification is an active research area [16, 51, 61], and algorithms were recently proposed for verifying STL properties and hyperproperties in [24, 31, 66] and [29, 32], respectively. While the verification result in runtime verification can be inconclusive, predictive runtime verification predicts a set of possible suffixes (e.g., a set of potential trajectories) to provide a verification result more reliably and quickly. In [3, 40, 54, 81, 85, 86], knowledge of the system is assumed to obtain predictions of system trajectories. However, the system is not always exactly known so that in [5, 6, 28] a system model is learned first, while in [22, 56, 73, 82] future system trajectories are predicted from past observed data using trajectory predictors. To the best of our knowledge, none of these works provide valid verification guarantees unless the system is exactly known or strong assumptions are placed on the prediction algorithm.

Conformal Prediction. Conformal prediction was introduced in [67, 77] as a statistical tool to quantify uncertainty of prediction algorithms. In [52], conformal prediction was used to obtain guarantees on the false negative rate of an online monitor. Conformal prediction was used for verification of STL properties in [57] by learning a predictive model of the STL semantics. For reachable set prediction, the authors in [14, 15, 17] used conformal prediction to quantify uncertainty of a predictive runtime monitor that predicts reachability of safe/unsafe states. However, the works in [14, 15, 17, 57] train task-specific predictors while we use task-independent trajectory predictors to predict future system states from which we infer information about the satisfaction of the task. This is significant as no expensive retraining is required when the specification changes. The authors of the work in [18], which appeared concurrently with our paper, also consider predictive runtime verification under STL specifications. Similar to our work, they provide probabilistic guarantees for the quantitative semantics of STL, but consider a different runtime verification setting in which systems have to be Markovian. Again, their predictors are task-specific while our predictors are task-independent so that we avoid expensive retraining when specifications change.

2 PROBLEM FORMULATION

Let $\mathcal{D}$ be an unknown distribution over system trajectories that describe our system, i.e., let $X := (X_0, X_1 \ldots) \sim \mathcal{D}$ be a random trajectory where X_τ denotes the state of the system at time τ that is drawn from $\mathbb{R}^n$. Modeling stochastic systems by a distribution $\mathcal{D}$ provides great flexibility, and $\mathcal{D}$ can generally describe the motion of Markov decision processes. It can capture stochastic systems whose trajectories follow the recursive update equation $X_{\tau+1} = f(X_\tau, w_\tau)$ where w_τ is a random variable and where the (unknown) function f describes the system dynamics. Stochastic systems can describe the behavior of engineered systems such as robots and autonomous systems, e.g., drones or self-driving cars, but they can also describe weather patterns, demographics, and human motion. We use lowercase letters x_τ for realizations of the random variable X_τ. We make no assumptions on the distribution $\mathcal{D}$, but assume availability of training and calibration data drawn from $\mathcal{D}$.

Assumption 1. *We have access to K independent realizations $x^{(i)} := (x_0^{(i)}, x_1^{(i)}, \ldots)$ of the distribution $\mathcal{D}$ that are collected in the dataset $D := \{x^{(1)}, \ldots, x^{(K)}\}$.*

Informal Problem Formulation. Assume now that we are given a specification ϕ for the stochastic system $\mathcal{D}$, e.g., a safety or performance specification defined over the states X_τ of the system. In "offline" system verification, e.g., in statistical model checking, we are interested in calculating the probability that $(X_0, X_1, \ldots) \sim \mathcal{D}$ satisfies the specification. In runtime verification, on the other hand, we have already observed the partial realization $(x_0, \ldots, x_t)$ of $(X_0, \ldots, X_t)$ online at time t, and we want to use this information to calculate the probability that $(X_0, X_1, \ldots) \sim \mathcal{D}$ satisfies the specification.[1] In this paper, we use predictions $\hat{x}_{\tau|t}$ of future states X_τ for this task in a predictive runtime verification approach.

While in "offline" verification all realizations of $\mathcal{D}$ are taken into account, only a subset of these are relevant in runtime verification. One hence gets different types of verification guarantees, e.g., consider a stochastic system $(X_0, X_1, \ldots) \sim \mathcal{D}$ of which we have plotted ten realizations in Figure 2 (left). In an offline approach, this system satisfies the specification $\inf_{\tau \in [150,250]} X_\tau \in [0, 3] \geq 0$ with a probability of 0.5. However, given an observed partial realization $(x_1, \ldots, x_{100})$, we are able to give a better answer. In this case, we used LSTM predictions $\hat{x}_{\tau|100}$ (red dashed lines), to more confidently say if the specification is satisfied. While the stochastic system in Figure 2 (left) has a simple structure, the same task for the stochastic system in Figure 2 (right) is already more challenging.

2.1 Signal Temporal Logic

To express system specifications, we use signal temporal logic (STL). Let $x := (x_0, x_1, \ldots)$ be a discrete-time signal, e.g., a realization of the stochastic system $(X_0, X_1, \ldots)$. The atomic elements of STL are predicates that are functions $\mu : \mathbb{R}^n \to \{\text{True}, \text{False}\}$. For convenience, the predicate μ is often defined via a predicate function $h : \mathbb{R}^n \to \mathbb{R}$ as $\mu(x_\tau) := \text{True}$ if $h(x_\tau) \geq 0$ and $\mu(x_\tau) := \text{False}$ otherwise. The syntax of STL is recursively defined as

$$\phi \ ::= \ \text{True} \mid \mu \mid \neg\phi' \mid \phi' \wedge \phi'' \mid \phi' U_I \phi'' \mid \phi' \underline{U}_I \phi'' \qquad (1)$$

[1] We note that we consider unconditional probabilities in this paper.

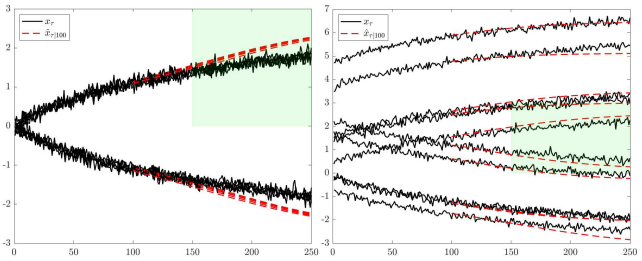

Figure 2: Ten realizations of two stochastic systems (solid lines) and corresponding LSTM predictions at time $t := 100$ (red dashed lines). The specification is that trajectories should be within the green box between 150 and 250 time units.

where ϕ' and ϕ'' are STL formulas. The Boolean operators $\neg$ and $\wedge$ encode negations ("not") and conjunctions ("and"), respectively. The until operator $\phi' U_I \phi''$ encodes that ϕ' has to be true from now on until ϕ'' becomes true at some future time within the time interval $I \subseteq \mathbb{R}_{\geq 0}$. The since operator encodes that ϕ'' was true at some past time within the time interval I and since then ϕ' is true. We can further derive the operators for disjunction ($\phi' \vee \phi'' := \neg(\neg\phi' \wedge \neg\phi'')$), eventually ($F_I \phi := \top U_I \phi$), once ($\underline{F}_I \phi := \top \underline{U}_I \phi$), always ($G_I \phi := \neg F_I \neg\phi$), and historically ($\underline{G}_I \phi := \neg \underline{F}_I \neg\phi$).

To determine if a signal x satisfies an STL formula ϕ that is enabled at time τ_0, we can define the semantics as a relation $\models$, i.e., $(x, \tau_0) \models \phi$ means that ϕ is satisfied. While the STL semantics are fairly standard [53], we recall them in Appendix A. Additionally, we can define robust (sometimes referred to as quantitative) semantics $\rho^\phi(x, \tau_0) \in \mathbb{R}$ that indicate how robustly the formula ϕ is satisfied or violated [25, 27], see Appendix A. Larger and positive values of $\rho^\phi(x, \tau_0)$ indicate that the specification is satisfied more robustly. Importantly, it holds that $(x, \tau_0) \models \phi$ if $\rho^\phi(x, \tau_0) > 0$. We make the following assumption on the class of STL formulas in this paper.

Assumption 2. *We consider bounded STL formulas ϕ, i.e., all time intervals I within the formula ϕ are bounded.*

Satisfaction of bounded STL formulas can be decided by finite length signals [63]. The minimum length is indicated by the formula length L^ϕ, i.e., knowledge of $(x_0, \ldots, x_{\tau_0 + L^\phi})$ is enough to determine if $(x, \tau_0) \models \phi$. We recall the definition of L^ϕ in Appendix A.

2.2 Trajectory Predictors

Given an observed partial sequence $(x_0, \ldots, x_t)$ at the current time $t \geq 0$, we want to predict the states $(x_{t+1}, \ldots, x_{t+H})$ for a prediction horizon of $H > 0$. Our runtime verification algorithm is in general compatible with any trajectory prediction algorithm. Assume therefore that Predict is a measurable function that maps observations $(x_0, \ldots, x_t)$ to predictions $(\hat{x}_{t+1|t}, \ldots, \hat{x}_{t+H|t})$ of $(x_{t+1}, \ldots, x_{t+H})$.

Trajectory predictors are typically learned. We therefore split the dataset D into training and calibration datasets D_{train} and D_{cal}, respectively, and learn Predict from D_{train}.

A specific example of Predict are recurrent neural networks (RNNs) that have shown good performance [50, 62]. For $\tau \leq t$, the recurrent structure of an RNN is given as

$$a_\tau^1 := \mathcal{A}(x_\tau, a_{\tau-1}^1),$$

$$a_\tau^i := \mathcal{A}(x_\tau, a_{\tau-1}^i, a_\tau^{i-1}), \quad \forall i \in \{2, \dots, d\}$$
$$y_{\tau+1|\tau} := \mathcal{Y}(a_\tau^d),$$

where x_τ is the input that is sequentially applied to the RNN and where $\mathcal{A}$ is a function that can parameterize different types of RNNs, e.g., LSTMs [35]. Furthermore, d is the RNN's depth and $a_\tau^1, \dots, a_\tau^d$ are the hidden states. The output $y_{t+1|t} := (\hat{x}_{t+1|t}, \dots, \hat{x}_{t+H|t})$ provides an estimate of $(x_{t+1}, \dots, x_{t+H})$ via the function $\mathcal{Y}$ which typically parameterizes a linear last layer.

2.3 Predictive Runtime Verification

We recall that $(x_0, x_1, \dots)$ denotes a realization of $X := (X_0, X_1, \dots) \sim \mathcal{D}$. Assume that we have observed $x_{\text{obs}} := (x_0, \dots, x_t)$ at time t, i.e., all states up until time t are known, while the realizations of $x_{\text{un}} := (x_{t+1}, x_{t+2}, \dots)$ are not known yet. Consequently, we have that $X := (X_{\text{obs}}, X_{\text{un}})$.[2] In this paper, we are interested in calculating the probability that $(X, \tau_0) \models \phi$ as formally stated next.[3]

Problem 1. *Given a distribution $(X_0, X_1, \dots) \sim \mathcal{D}$, the current time t, the observations $x_{obs} := (x_0, \dots, x_t)$, a bounded STL formula ϕ that is enabled at τ_0, and a failure probability $\delta \in (0, 1)$, determine if $P\big((X, \tau_0) \models \phi\big) \geq 1 - \delta$ holds.*

Several comments are in order. Note that we use the system specification ϕ (and not its negation $\neg\phi$) to determine if ϕ is satisfied. From $P\big((X, \tau_0) \models \phi\big) \geq 1-\delta$, we can infer that $P\big((X, \tau_0) \models \neg\phi\big) \leq \delta$, i.e., we get an upper bound on the probability that the specification is violated. We further remark that, as a byproduct of our solution to Problem 1, we obtain a probabilistic lower bound $\bar{C} \in \mathbb{R}$ on the robust semantics $\rho^\phi(X, \tau_0)$, i.e., so that $P\big(\rho^\phi(X, \tau_0) \geq \bar{C}\big) \geq 1 - \delta$.

We would like to point out two special instances of Problem 1. When $\tau_0 := 0$, we recover the "standard" runtime verification problem in which a specification is enabled at time zero, such as in the example $\inf_{\tau \in [150,250]} x_\tau \in [0, 3]$ shown in Figure 2. When $\tau_0 := t$, the current time coincides with the time the specification is enabled. This may, for instance, be important when monitoring the current quality of a system, e.g., when monitoring the output of a neural network used for perception in autonomous driving.

3 CONFORMAL PREDICTION FOR PREDICTIVE RUNTIME VERIFICATION

In this section, we first provide an introduction to conformal prediction for uncertainty quantification. We then propose two predictive runtime verification algorithms to solve Problem 1. We refer to these algorithms as direct and indirect. This naming convention is motivated as the direct method applies conformal prediction directly to obtain a prediction region for the robust semantics $\rho^\phi(X, \tau_0)$. The indirect method uses conformal prediction to get prediction regions for future states X_τ first, which are subsequently used indirectly to obtain a prediction region for $\rho^\phi(X, \tau_0)$, see Figure 2.

3.1 Introduction to Conformal Prediction

Conformal prediction was introduced in [67, 77] to obtain valid prediction regions for complex prediction algorithms, i.e., neural networks, without making assumptions on the underlying distribution or the prediction algorithm [4, 20, 30, 44, 74].

We first provide a brief introduction to conformal prediction. Let $R^{(0)}, \dots, R^{(k)}$ be $k+1$ independent and identically distributed random variables. The variable $R^{(i)}$ is usually referred to as the nonconformity score. In supervised learning, it may be defined as $R^{(i)} := \|Y^{(i)} - \mu(X^{(i)})\|$ where the predictor μ attempts to predict an output $Y^{(i)}$ based on an input $X^{(i)}$. A large nonconformity score indicates a poor predictive model.

Our goal is to obtain a prediction region for $R^{(0)}$ based on $R^{(1)}, \dots, R^{(k)}$, i.e., the random variable $R^{(0)}$ should be contained within the prediction region with high probability. Formally, given a failure probability $\delta \in (0, 1)$, we want to construct a valid prediction region C that depends on $R^{(1)}, \dots, R^{(k)}$ such that

$$P(R^{(0)} \leq C) \geq 1 - \delta.$$

As C depends on $R^{(1)}, \dots, R^{(k)}$, the probability measure P is defined over the product measure of $R^{(0)}, \dots, R^{(k)}$. This is an important observation as conformal prediction guarantees marginal coverage but not conditional coverage, see [4] for a detailed discussion. By a surprisingly simple quantile argument, see [74, Lemma 1], one can obtain C to be the $(1-\delta)$th quantile of the empirical distribution of the values $R^{(1)}, \dots, R^{(k)}$ and ∞. By assuming that $R^{(1)}, \dots, R^{(k)}$ are sorted in non-decreasing order, and by adding $R^{(k+1)} := \infty$, we can equivalently obtain $C := R^{(p)}$ where $p := \lceil (k+1)(1-\delta) \rceil$, i.e., C is the pth smallest nonconformity score.

3.2 Direct STL Predictive Runtime Verification

Recall that we can obtain predictions $\hat{x}_{\tau|t}$ of x_τ for all future times $\tau > t$ using the Predict function. However, the predictions $\hat{x}_{\tau|t}$ are only point predictions that are not sufficient to solve Problem 1 as they do not contain any information about the uncertainty of $\hat{x}_{\tau|t}$.

We first propose a solution by a direct application of conformal prediction. Let us therefore define $H := \tau_0 + L^\phi - t$ as the maximum prediction horizon that is needed to estimate the satisfaction of the bounded STL specification ϕ. Define now the predicted trajectory

$$\hat{x} := (x_{\text{obs}}, \hat{x}_{t+1|t}, \dots, \hat{x}_{t+H|t}) \tag{2}$$

which is the concatenation of the current observations x_{obs} and the predictions of future states $\hat{x}_{t+1|t}, \dots, \hat{x}_{t+H|t}$. For an a priori fixed failure probability $\delta \in (0, 1)$, our goal is to directly construct a prediction region defined by a constant C so that

$$P\big(\rho^\phi(\hat{x}, \tau_0) - \rho^\phi(X, \tau_0) \leq C\big) \geq 1 - \delta. \tag{3}$$

Note that $\rho^\phi(\hat{x}, \tau_0)$ is the predicted robust semantics for the specification ϕ that we can calculate at time t based on the observations x_{obs} and the predictions $\hat{x}_{t+1|t}, \dots, \hat{x}_{t+H|t}$. Now, if equation (3) holds, then we know that $\rho^\phi(\hat{x}, \tau_0) > C$ is a sufficient condition for $P(\rho^\phi(X, \tau_0) > 0) \geq 1 - \delta$ to hold.

To obtain the constant C, we thus consider the nonconformity score $R := \rho^\phi(\hat{x}, \tau_0) - \rho^\phi(X, \tau_0)$. In fact, let us compute the nonconformity score for each calibration trajectory $x^{(i)} \in D_{\text{cal}}$ as

$$R^{(i)} := \rho^\phi(\hat{x}^{(i)}, \tau_0) - \rho^\phi(x^{(i)}, \tau_0)$$

[2] For convenience, we chose the notations of X_{obs}, X_{un}, and X that do not explicitly reflect the dependence on the current time t.

[3] We remark that the semantics and the robust semantics are measurable so that probabilities over these functions are well defined [11, 47].

where $\hat{x}^{(i)} := (x^{(i)}_{\text{obs}}, \hat{x}^{(i)}_{t+1|t}, \dots, \hat{x}^{(i)}_{t+H|t})$ resembles equation (2), but now defined for the calibration trajectory $x^{(i)}$.[4] A positive nonconformity score $R^{(i)}$ indicates that our predictions are too optimistic, i.e., the predicted robust semantics $\rho^{\phi}(\hat{x}^{(i)}, \tau_0)$ is greater than the actual robust semantics $\rho^{\phi}(x^{(i)}, \tau_0)$ obtained when using the ground truth calibration trajectory $x^{(i)}$. Conversely, a negative value of $R^{(i)}$ means that our prediction are too conservative.

We can now directly obtain a constant C that makes equation (3) valid, and use this C to solve Problem 1, by a direct application of [74, Lemma 1]. Therefore assume, without loss of generality, that the values of $R^{(i)}$ are sorted in non-decreasing order and let us add $R^{(|D_{\text{cal}}|+1)} := \infty$ as the $(|D_{\text{cal}}|+1)$th value.

Theorem 1. *Given a distribution $(X_0, X_1, \dots) \sim \mathcal{D}$, the current time t, the observations $x_{obs} := (x_0, \dots, x_t)$, a bounded STL formula ϕ that is enabled at τ_0, the dataset D_{cal}, and a failure probability $\delta \in (0,1)$. Then the prediction region in equation* (3) *is valid with C defined as*

$$C := R^{(p)} \quad \text{where} \quad p := \left\lceil (|D_{cal}|+1)(1-\delta) \right\rceil, \tag{4}$$

and it holds that $P\big((X, \tau_0) \models \phi\big) \geq 1-\delta$ if $\rho^{\phi}(\hat{x}, \tau_0) > C$.

It is important to note that the direct method, as well as the indirect method presented in the next subsection, do not need to retrain their predictor when the specification ϕ changes, as in existing work such as [14, 57]. This is since we use trajectory predictors to obtain state predictions $\hat{x}_{\tau|t}$ that are specification independent.

Remark 1. *Note that Theorem 1 assumes a fixed failure probability δ. If one wants to find the tightest bound with the smallest failure probability δ so that $P\big((X, \tau_0) \models \phi\big) \geq 1-\delta$ holds, we can (approximately) find the smallest such δ by a simple grid search over $\delta \in (0,1)$ and repeatedly invoke Theorem 1.*

Remark 2. *We emphasize that the prediction regions in equation* (3)*, and hence the result that $P\big((X, \tau_0) > 0 \models \phi\big) \geq 1-\delta$ if $\rho^{\phi}(\hat{x}, \tau_0) > C$, guarantee marginal coverage. This means that the probability measure P is defined over the randomness of the test trajectory X and the randomness of the calibration trajectories in D_{cal}. We thereby obtain probabilistic guarantees for the verification procedure, but we do not obtain guarantees conditional on D_{cal}.*

3.3 Indirect STL Predictive Runtime Verification

We now present the indirect method where we first obtain prediction regions for the state predictions $\hat{x}_{t+1|t}, \dots, \hat{x}_{t+H|t}$, and then use these prediction regions to solve Problem 1. We later discuss advantages and disadvantages between the direct and the indirect method (see Remark 4), and compare them in simulations (see Section 4).

For a failure probability of $\delta \in (0,1)$, our first goal is to construct prediction regions defined by constants C_τ so that

$$P\big(\|X_\tau - \hat{x}_{\tau|t}\| \leq C_\tau, \ \forall \tau \in \{t+1, \dots, t+H\}\big) \geq 1-\delta, \tag{5}$$

i.e., C_τ should be such that the state X_τ is C_τ-close to our predictions $\hat{x}_{\tau|t}$ for all relevant times $\tau \in \{t+1, \dots, t+H\}$ with a probability of at least $1-\delta$. Let us thus consider the following nonconformity score that we compute for each calibration trajectory $x^{(i)} \in D_{\text{cal}}$ as

$$R^{(i)}_\tau := \|x^{(i)}_\tau - \hat{x}^{(i)}_{\tau|t}\|$$

where we recall that $\hat{x}^{(i)}_{\tau|t}$ is the prediction obtained from the observed calibration trajectory $x^{(i)}_{\text{obs}}$. A large nonconformity score indicates that the state predictions $\hat{x}^{(i)}_{\tau|t}$ of $x^{(i)}_\tau$ are not accurate, while a small score indicates accurate predictions. Assume again that the values of $R^{(i)}_\tau$ are sorted in non-decreasing order and define $R^{(|D_{\text{cal}}|+1)}_\tau := \infty$ as the $(|D_{\text{cal}}|+1)$th value. To obtain the values of C_τ that make equation (5) valid, we use the results from [46, 71].

Lemma 1 ([46, 71]). *Given a distribution $(X_0, X_1, \dots) \sim \mathcal{D}$, the current time t, the observations $x_{obs} := (x_0, \dots, x_t)$, the dataset D_{cal}, and a failure probability $\delta \in (0,1)$. Then the prediction regions in equation* (5) *are valid with C_τ defined as*

$$C_\tau := R^{(p)}_\tau \quad \text{where} \quad p := \left\lceil (|D_{cal}|+1)(1-\bar{\delta}) \right\rceil \text{ and } \bar{\delta} := \delta/H. \tag{6}$$

Note the scaling of δ by the inverse of H, as expressed in $\bar{\delta}$. Consequently, the constants C_τ increase with increasing prediction horizon H, i.e., with larger formula length L^{ϕ}, as larger H result in smaller $\bar{\delta}$ and consequently in larger p according to (6).

We can now use the prediction regions of the predictions $\hat{x}_{\tau|t}$ from equation (5) to obtain prediction regions for $\rho^{\phi}(X, \tau_0)$ to solve Problem 1. The main idea is to calculate the worst case of the robust semantics ρ^{ϕ} over these prediction regions. To be able to do so, we assume that the formula ϕ is in positive normal form, i.e., that the formula ϕ contains no negations. This is without loss of generality as every STL formula ϕ can be re-written in positive normal form, see e.g., [63]. Let us next define a worst case version $\bar{\rho}^{\phi}$ of the robust semantics ρ^{ϕ} that incorporates the prediction regions from equation (5). For predicates μ, we define these semantics as

$$\bar{\rho}^{\mu}(\hat{x}, \tau) := \begin{cases} h(x_\tau) & \text{if } \tau \leq t \\ \inf_{\zeta \in \mathcal{B}_\tau} h(\zeta) & \text{otherwise} \end{cases}$$

where we recall the definition of the predicted trajectory $\hat{x}$ in equation (2) and where $\mathcal{B}_\tau := \{\zeta \in \mathbb{R}^n \,|\, \|\zeta - \hat{x}_{\tau|t}\| \leq C_\tau\}$ is a ball of size C_τ centered around the prediction $\hat{x}_{\tau|t}$, i.e., $\mathcal{B}_\tau$ defines the set of states within the prediction region at time τ. The intuition behind this definition is that we know the value of the robust semantics $\rho^{\mu}(X, \tau) = \bar{\rho}^{\mu}(\hat{x}, \tau)$ if $\tau \leq t$ since x_τ is known. For times $\tau > t$, we know that $X_\tau \in \mathcal{B}_\tau$ holds with a probability of at least $1-\delta$ by Lemma 1 so that we compute $\bar{\rho}^{\mu}(\hat{x}, \tau) := \inf_{\zeta \in \mathcal{B}_\tau} h(\zeta)$ to obtain a lower bound for $\rho^{\mu}(X, \tau)$ with a probability of at least $1-\delta$.

Remark 3. *For convex predicate functions h, computing $\inf_{\zeta \in \mathcal{B}_\tau} h(\zeta)$ is a convex optimization problem that can efficiently be solved. However, note that the optimization problem $\inf_{\zeta \in \mathcal{B}_\tau} h(\zeta)$ may need to be solved for different times τ and for multiple predicate functions h. For non-convex functions h, we can obtain lower bounds of $\inf_{\zeta \in \mathcal{B}_\tau} h(\zeta)$ that we can use instead. Particularly, let L_h be the Lipschitz constant of h, i.e., let $|h(\zeta) - h(\hat{x}_{\tau|t})| \leq L\|\zeta - \hat{x}_{\tau|t}\|$. Then, we know that*

$$\inf_{\zeta \in \mathcal{B}_\tau} h(\zeta) \geq h(\hat{x}_{\tau|t}) - L_h C_\tau.$$

For instance, the constraint $h(\zeta) := \|\zeta_1 - \zeta_2\| - 0.5$, which can encode collision avoidance constraints, has Lipschitz constant one.

[4] This means that $\hat{x}^{(i)}$ is the concatenation of the observed calibration trajectory $x^{(i)}_{\text{obs}} := (x^{(i)}_0, \dots, x^{(i)}_t)$ and the predictions $\hat{x}^{(i)}_{t+1|t}, \dots, \hat{x}^{(i)}_{t+H|t}$ obtained from $x^{(i)}_{\text{obs}}$.

The worst case robust semantics $\bar{\rho}^{\phi}$ for the remaining operators (True, conjunctions, until, and since) are defined in the standard way, i.e., the same way as for the robust semantics ρ^{ϕ}, and are summarized in Appendix A for convenience. We can now use the worst case robust semantics to solve Problem 1.

Theorem 2. *Let the conditions of Lemma 1 hold. Given a bounded STL formula ϕ in positive normal form that is enabled at τ_0. Then it holds that $P\big((X, \tau_0) \models \phi\big) \geq 1 - \delta$ if $\bar{\rho}^{\phi}(\hat{x}, \tau_0) > 0$.*

Finally, let us point out conceptual differences with respect to the direct STL predictive runtime verification method.

Remark 4. *The state prediction regions* (5) *obtained in Lemma 1 may lead to conservatism in Theorem 2, especially for larger prediction horizons H due to the scaling of δ with the inverse of H. In fact, we require larger calibration datasets D_{cal} compared to the direct method to achieve $p \leq |D_{cal}|$ (recall that $C_\tau = \infty$ if $p > |D_{cal}|$). On the other hand, the indirect method is more interpretable and allows to identify parts of the formula ϕ that may be violated by analyzing the uncertainty of predicates via the worst case robust semantics $\bar{\rho}^{\mu}(\hat{x}, \tau)$. This information may be helpful and can be used subsequently in a decision making context for plan reconfiguration.*

4 CASE STUDIES

We present two case studies in which we verify an aircraft and a self-driving car. We remark upfront that, in both case studies, we fix the calibration dataset D_{val} a-priori and then evaluate our proposed runtime verification method on several test trajectories. As eluded to in Remark 2, one would technically have to resample a calibration dataset for each test trajectory. This is impractical and, in fact, shown to not be needed when the size of the calibration dataset is large enough, see [4, Section 3.3] for a detailed discussion on this topic.

4.1 F-16 Aircraft Simulator

In our first case study, we consider the F-16 Fighting Falcon, which is a highly-maneuverable aircraft - a brief summary of the system is provided in Appendix D. We use a ground collision avoidance maneuver, and are thus primarily interested in the plane's altitude that we denote by h. We collected $D_{\text{train}} := 1520$ training trajectories, $D_{\text{cal}} := 5680$ calibration trajectories, and $D_{\text{test}} := 100$ test trajectories. From D_{train}, we trained an LSTM of depth two and width 50 to predict future states of h.[5] We show the LSTM performance in predicting h in Figure 3. Particularly, we show plots of the best five and the worst five LSTM predictions, in terms of the mean square error, on the test trajectories D_{test} in Figure 3 (left and left-mid).

We are interested in a safety specification expressed as $\phi := G_{[0,T]}(h \geq 750)$ that is enabled at time $\tau_0 := t$, i.e., a specification that is imposed online during runtime. Hereby, we intend to monitor if the airplane dips below 750 meters within the next $T := 200$ time steps (the sampling frequency is 100 Hz). Additionally, we set $\delta := 0.05$ and fix the current time to $t := 230$.

Let us first use the direct predictive runtime verification algorithm and obtain prediction regions of $\rho^{\phi}(\hat{x}, \tau_0) - \rho^{\phi}(X, \tau_0)$ by calculating C according to Theorem 1. We show the histograms of $R^{(i)}$ over the calibration data D_{cal} in Figure 3 (right-mid). The prediction regions C (i.e., the $R^{(p)}$th nonconformity score) are highlighted as vertical lines. In a next step, we empirically evaluate the results of Theorem 1 by using the test trajectories D_{test}. In Figure 3 (right), we plot the predicted robustness $\rho^{\phi}(\hat{x}^{(i)}, \tau_0)$ and the ground truth robustness $\rho^{\phi}(x^{(i)}, \tau_0)$. We found that for 100 of the $100 = |D_{\text{test}}|$ trajectories it holds that $\rho^{\phi}(\hat{x}^{(i)}, \tau_0) > C$ implies $(x^{(i)}, \tau_0) \models \phi$, confirming Theorem 1. We also validated equation (3) and found that 96/100 trajectories satisfy $\rho^{\phi}(\hat{x}^{(i)}, \tau_0) - \rho^{\phi}(x^{(i)}, \tau_0) \leq C$.

Let us now use the indirect predictive runtime verification algorithm. We first obtain prediction regions of $\|X_\tau - \hat{x}_{\tau|t}\|$ by calculating C_τ according to Lemma 1. We show the histograms for three different τ in Figure 4 (left, left-mid, right-mid). We also indicate the prediction regions C_τ by vertical lines (note that $\bar{\delta} = \delta/200$ in this case). We can observe that larger prediction times τ result in larger prediction regions C_τ. This is natural as the trajectory predictor is expected to perform worse for larger τ. In a next step, we empirically evaluate the results of Theorem 2 by calculating the worst case robust semantic $\bar{\rho}^{\phi}(\hat{x}^{(i)}, \tau_0)$ for the test trajectories D_{test}. In Figure 4 (right), we plot the worst case robustness $\bar{\rho}^{\phi}(\hat{x}^{(i)}, \tau_0)$ and the ground truth robustness $\rho^{\phi}(x^{(i)}, \tau_0)$. We found that for 100 of the $100 = |D_{\text{test}}|$ trajectories it holds that $\bar{\rho}^{\phi}(\hat{x}^{(i)}, \tau_0) > 0$ implies $(x^{(i)}, \tau_0) \models \phi$, confirming Theorem 2.

By a direct comparison of Figures 3 (right) and 4 (right), we observe that the indirect method is more conservative than the direct method in the obtained robustness estimates. Despite this conservatism, the indirect method allows us to obtain more information in case of failure by inspecting the worst case robust semantics $\bar{\rho}^{\phi}(\hat{x}, \tau_t)$ as previously remarked ins Remark 4.

4.2 Autonomous Driving in CARLA

We consider the case study from [47] in which two neural network lane keeping controllers, an imitation learning (IL) controller [60] and a learned control barrier function (CBF) controller [48], are verified within the autonomous driving simulator CARLA [26] using offline trajectory data. The controllers are supposed to keep the car within the lane during a long 180 degree left turn, see Figure 9 (right) in the Appendix. The authors in [47] provide offline probabilistic verification guarantees, and find that not every trajectory satisfies the specification. This motivates our predictive runtime verification approach in which we would like to alert of potential violations of the specification already during runtime. For the analysis, we consider the cross-track error c_e (deviation of the car from the center of the lane) and the orientation error θ_e (difference between the orientation of the car and the lane).

Within CARLA, the control input of the car is affected by additive Gaussian noise and the initial position of the car is drawn uniformly from $(c_e, \theta_e) \in [-1, 1] \times [-0.4, 0.4]$. We obtained 1000 trajectories for each controller, and use $|D_{\text{train}}| := 700$ trajectories to train an LSTM, while we use $|D_{\text{cal}}| := 200$ trajectories to obtain conformal prediction regions. The remaining $|D_{\text{test}}| := 100$ trajectories are used for testing.

We have trained two LSTMs for each controller from D_{train} using the same settings as in the previous section. In Figures 5 and 6, we show the LSTMs performances in predicting c_e and θ_e for each controller, respectively. Particularly, the plots show the best five

[5] We only used the observed sequence of altitudes $(h_0, \ldots, h_t)$ as the input of the LSTM. Additionally using other states is possible and can improve prediction performance.

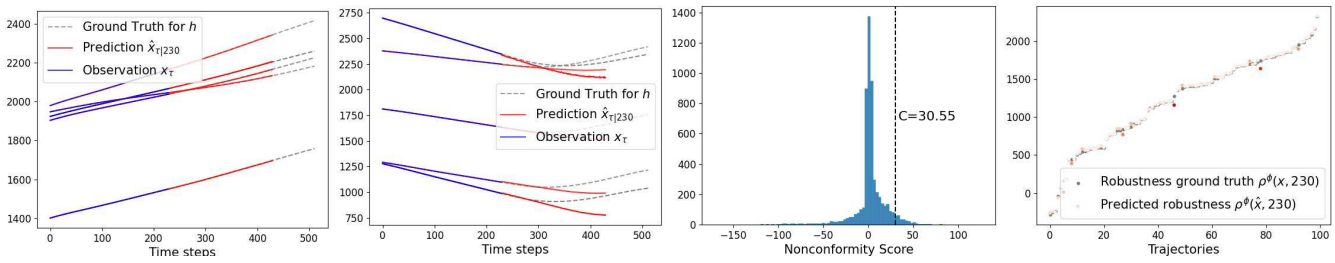

Figure 3: LSTM predictions of the altitude h on D_{test} (left, left-mid) and direct predictive runtime verification method (right-mid, right). Left: five best (in terms of mean square error) predictions on D_{test}, left-mid: five worst predictions on D_{test}, right-mid: histogram of the nonconformal score $R^{(i)}$ on D_{cal} for direct method, right: predicted robustness $\rho^\phi(\hat{x}^{(i)}, \tau_0)$ and ground truth robustness $\rho^\phi(x^{(i)}, \tau_0)$ on D_{test}.

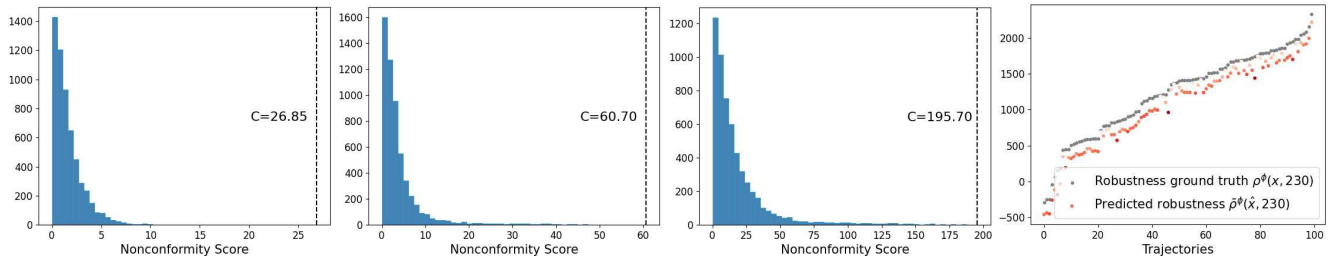

Figure 4: Indirect predictive runtime verification method. Left, left-mid, and right-mid: histograms of the nonconformal scores $R^{(i)}$ of τ step ahead prediction on D_{cal} for $\tau \in \{50, 100, 200\}$ and the indirect method, right: worst case predicted robustness $\bar{\rho}^\phi(\hat{x}^{(i)}, \tau_0)$ and ground truth robustness $\rho^\phi(x^{(i)}, \tau_0)$ on D_{test}.

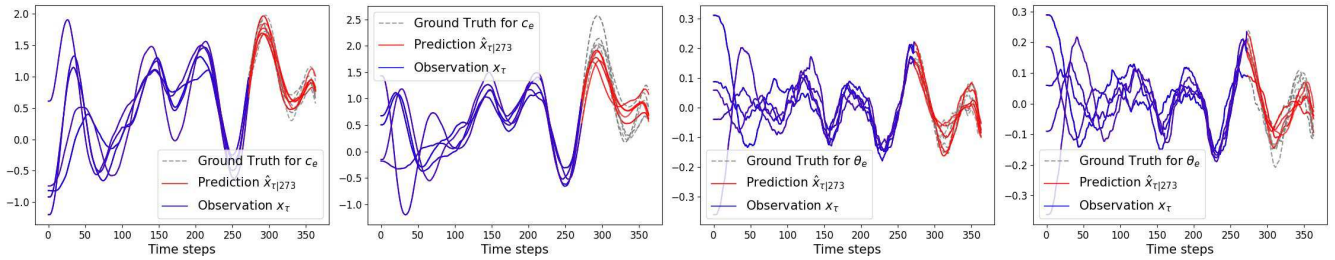

Figure 5: LSTM predictions of the imitation learning controller on D_{test}. Left: five best (in terms of mean square error) c_e predictions, left-mid: five worst c_e predictions, right-mid: five best θ_e predictions, right: five worst θ_e predictions.

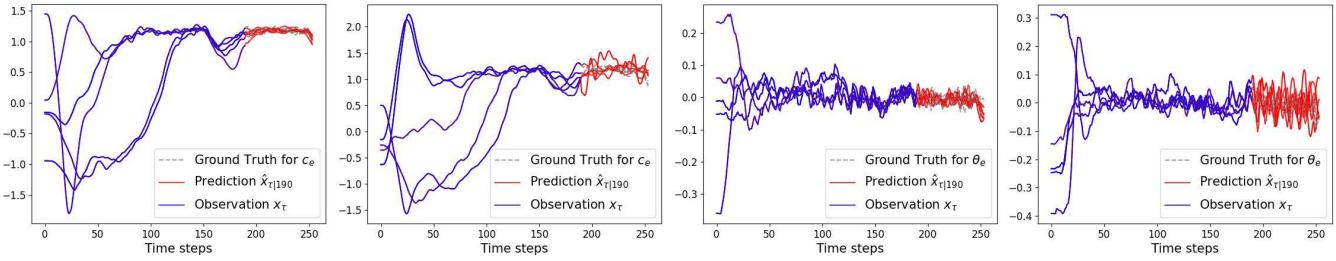

Figure 6: LSTM predictions of the control barrier function controller on D_{test}. Left: five best (in terms of mean square error) c_e predictions, left-mid: five worst c_e predictions, right-mid: five best θ_e predictions, right: five worst θ_e predictions.

and the worst five LSTM predictions (in terms of the mean square error) on the test trajectories D_{test}.

For the verification of the car, we consider the following two STL specifications that are enabled at $\tau_0 := 0$:

$$\phi_1 := G_{[10,\infty)]}\big(|c_e| \leq 2.25\big),$$
$$\phi_2 := G_{[10,\infty)}\big((|c_e| \geq 1.25) \implies F_{[0,5]}G_{[0,5]}(|c_e| \leq 1.25)\big).$$

The first specification is a safety specification that requires the cross-track error to not exceed a threshold of 2.25 in steady-state (after 10 seconds of driving). The second specification is a responsiveness requirement that requires that a cross-track error above 1.25 is followed immediately within the next 5 seconds by a phase of 5 seconds where the cross-track error is below 1.25. As previously mentioned, we can use the same LSTM for both specifications, and we do not need any retraining when the specification changes which is a major advantage of our method over existing works.

We set $\delta := 0.05$ and fix the current time to $t := 273$ for the IL controller and $t := 190$ for the CBF controller. At these times, the cars controlled by each controller are approximately at the same location in the left turn (this difference is caused by different sampling times). As we have limited calibration data D_{cal} available (CARLA runs in real-time so that data collection is time intensive), we only evaluate the direct STL predictive runtime verification algorithm for these two specifications.[6] We hence obtain prediction regions of $\rho^{\phi}(\hat{x}, \tau_0) - \rho^{\phi}(X, \tau_0)$ for each specification $\phi \in \{\phi_1, \phi_2\}$ by calculating C according to Theorem 1.

For the first specification ϕ_1, we show the histograms of $R^{(i)}$ for both controllers over the calibration data D_{cal} in Figure 7 (left: IL, left-mid: CBF). The prediction regions C are again highlighted as vertical lines, and we can see that the prediction regions C for the CBF controller are smaller, which may be caused by an LSTM that predicts the system trajectories more accurately (note that the CBF controller causes less variability in c_e which may make it easier to train a good LSTM). In a next step, we empirically evaluate the results of Theorem 1 by using the test trajectories D_{test}. In Figure 8 (left: IL, left-mid: CBF), we plot the predicted robustness $\rho^{\phi_1}(\hat{x}, \tau_0)$ and the ground truth robustness $\rho^{\phi_1}(X, \tau_0)$. We found that for 99 of the $100 = |D_{\text{test}}|$ trajectories under the IL controller and for 100/100 trajectories under the CBF controller it holds that $\rho^{\phi_1}(\hat{x}^{(i)}, \tau_0) > C$ implies $(x^{(i)}, \tau_0) \models \phi_1$, confirming Theorem 1. We also validated equation (3) and found that 95/100 trajectories under the IL controller and 95/100 trajectories under the CBF controller satisfy $\rho^{\phi_1}(\hat{x}^{(i)}, \tau_0) - \rho^{\phi_1}(x^{(i)}, \tau_0) \leq C$.

For the second specification ϕ_2, we again show the histograms of $R^{(i)}$ for both controllers over the calibration data D_{cal} in Figure 7 (right-mid: IL, right: CBF). We can now observe that the prediction region C for both controllers are relatively small. However, the absolute robustness is also less as in the first specification as can be seen in Figure 8 (right-mid: IL, right: CBF). We again empirically evaluate the results of Theorem 1 by using the test trajectories D_{test}. In Figure 8 (right-mid: CBF, right: IL), we plot the predicted robustness $\rho^{\phi_2}(\hat{x}, \tau_0)$ and the ground truth robustness $\rho^{\phi_2}(X, \tau_0)$. We found that for 99/100 trajectories under the IL controller and for 98/100 trajectories under the CBF controller it holds that $\rho^{\phi_2}(\hat{x}^{(i)}, \tau_0) > C$ implies $(x^{(i)}, \tau_0) \models \phi_2$, confirming Theorem 1. We also validated equation (3) and found that 98/100 trajectories under the IL controller and 92/100 trajectories under the CBF controller satisfy $\rho^{\phi_2}(\hat{x}^{(i)}, \tau_0) - \rho^{\phi_2}(x^{(i)}, \tau_0) \leq C$.

Finally, we would like to remark that we observed that the added Gaussian random noise on the control signals made the prediction task challenging, but the combination of LSTM and conformal prediction were able to deal with this particular type of randomness. In fact, poorly trained LSTMs lead to larger prediction regions.

[6] The indirect STL predictive runtime verification algorithm would require more calibration data, recall the discussion from Remark 4.

5 CONCLUSION

We presented two predictive runtime verification algorithms to compute the probability that the current system trajectory violates a signal temporal logic specification. Both algorithms use i) trajectory predictors to predict future system states, and ii) conformal prediction to quantify prediction uncertainty. The use of conformal prediction enables us to obtain valid probabilistic runtime verification guarantees. To the best of our knowledge, these are the first formal guarantees for a predictive runtime verification algorithm that applies to widely used trajectory predictors such as RNNs and LSTMs, while being computationally simple and making no assumptions on the underlying distribution. An advantage of our approach is that a changing system specification does not require expensive retraining as in existing works. We concluded with experiments of an F-16 aircraft and a self-driving car equipped with LSTMs.

ACKNOWLEDGMENTS

Lars Lindemann and George J. Pappas were generously supported by NSF award CPS-2038873. Xin Qin and Jyotirmoy V. Deshmukh gratefully acknowledge the support by the National Science Foundation through the following grants: SHF-1910088, CAREER award (SHF-2048094), CNS-1932620, funding by Toyota R&D through the USC Center for Autonomy and AI, funding by Airbus Institute for Engineering Research, and gift funding from Northrop Grumman Aerospace Systems. Finally, the authors would like to thank the anonymous reviewers for their feedback.

REFERENCES

[1] Prithvi Akella, Mohamadreza Ahmadi, and Aaron D Ames. 2022. A scenario approach to risk-aware safety-critical system verification. *arXiv preprint arXiv:2203.02595* (2022).
[2] Prithvi Akella, Anushri Dixit, Mohamadreza Ahmadi, Joel W Burdick, and Aaron D Ames. 2022. Sample-Based Bounds for Coherent Risk Measures: Applications to Policy Synthesis and Verification. *arXiv preprint arXiv:2204.09833* (2022).
[3] Matthias Althoff and John M Dolan. 2014. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics* 30, 4 (2014), 903–918.
[4] Anastasios N Angelopoulos and Stephen Bates. 2021. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511* (2021).
[5] Reza Babaee, Vijay Ganesh, and Sean Sedwards. 2019. Accelerated learning of predictive runtime monitors for rare failure. In *International Conference on Runtime Verification*. Springer, 111–128.
[6] Reza Babaee, Arie Gurfinkel, and Sebastian Fischmeister. 2018. Prevent: A Predictive Run-Time Verification Framework Using Statistical Learning. In *International Conference on Software Engineering and Formal Methods*. Springer, 205–220.
[7] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking* (1 ed.). The MIT Press, Cambridge, MA.
[8] Stanley Bak, Changliu Liu, and Taylor Johnson. 2021. The second international verification of neural networks competition (vnn-comp 2021): Summary and results. *arXiv preprint arXiv:2109.00498* (2021).

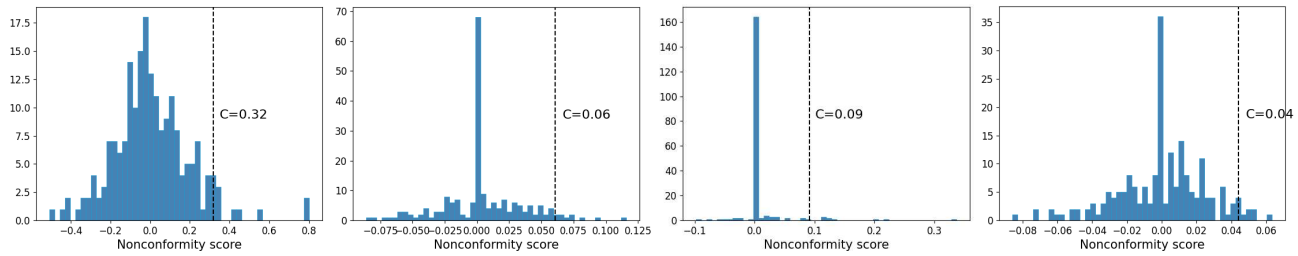

Figure 7: Histograms of the nonconformal scores $R^{(i)}$ on D_{cal} and prediction region C. Left: IL controller and ϕ_1, left-mid: CBF controller and ϕ_1, right-mid: IL controller and ϕ_2, right: CBF controller and ϕ_2.

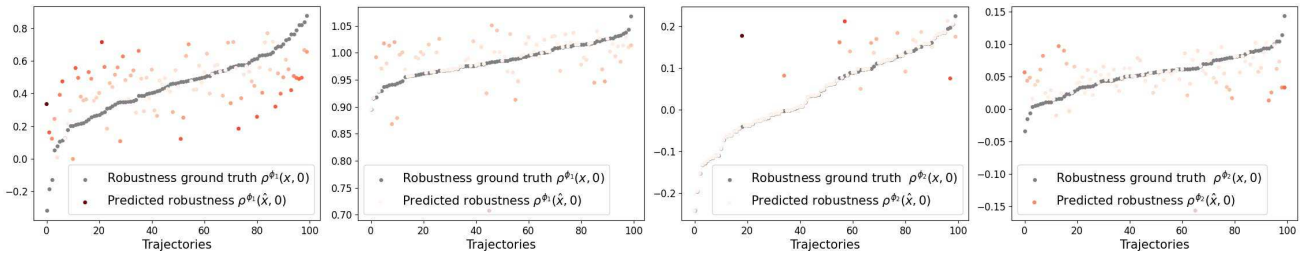

Figure 8: Predicted robustness $\rho^{\phi}(\hat{x}^{(i)}, \tau_0)$ and ground truth robustness $\rho^{\phi}(x^{(i)}, \tau_0)$ on D_{test}. Left: IL controller and ϕ_1, left-mid: CBF controller and ϕ_1, right-mid: IL controller and ϕ_2, right: CBF controller and ϕ_2.

[9] Stanley Bak and Hoang-Dung Tran. 2022. Neural Network Compression of ACAS Xu Early Prototype Is Unsafe: Closed-Loop Verification Through Quantized State Backreachability. In *NASA Formal Methods Symposium*. Springer, 280–298.

[10] Ezio Bartocci, Luca Bortolussi, Laura Nenzi, and Guido Sanguinetti. 2013. On the robustness of temporal properties for stochastic models. In *Proc. Int. Workshop Hybrid Syst. Biology*. Taormina, Italy, 3–19.

[11] Ezio Bartocci, Luca Bortolussi, Laura Nenzi, and Guido Sanguinetti. 2015. System design of stochastic models using robustness of temporal properties. *Theoret. Comp. Science* 587 (2015), 3–25.

[12] Andreas Bauer, Martin Leucker, and Christian Schallhart. 2011. Runtime verification for LTL and TLTL. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 20, 4 (2011), 1–64.

[13] Andrea Bianco and Luca de Alfaro. 1995. Model checking of probabilistic and nondeterministic systems. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*. Springer, 499–513.

[14] Luca Bortolussi, Francesca Cairoli, Nicola Paoletti, Scott A Smolka, and Scott D Stoller. 2019. Neural predictive monitoring. In *International Conference on Runtime Verification*. Springer, 129–147.

[15] Luca Bortolussi, Francesca Cairoli, Nicola Paoletti, Scott A Smolka, and Scott D Stoller. 2021. Neural predictive monitoring and a comparison of frequentist and Bayesian approaches. *International Journal on Software Tools for Technology Transfer* 23, 4 (2021), 615–640.

[16] Dimitrios Boursinos and Xenofon Koutsoukos. 2021. Assurance monitoring of learning-enabled cyber-physical systems using inductive conformal prediction based on distance learning. *AI EDAM* 35, 2 (2021), 251–264.

[17] Francesca Cairoli, Luca Bortolussi, and Nicola Paoletti. 2021. Neural predictive monitoring under partial observability. In *International Conference on Runtime Verification*. Springer, 121–141.

[18] Francesca Cairoli, Nicola Paoletti, and Luca Bortolussi. 2022. Conformal Quantitative Predictive Monitoring of STL Requirements for Stochastic Processes. *arXiv preprint arXiv:2211.02375* (2022).

[19] Ian Cassar, Adrian Francalanza, Luca Aceto, and Anna Ingólfsdóttir. 2017. A survey of runtime monitoring instrumentation techniques. *arXiv preprint arXiv:1708.07229* (2017).

[20] Maxime Cauchois, Suyash Gupta, Alnur Ali, and John C Duchi. 2020. Robust validation: Confident predictions even when distributions shift. *arXiv preprint arXiv:2008.04267* (2020).

[21] Margaret P Chapman, Riccardo Bonalli, Kevin M Smith, Insoon Yang, Marco Pavone, and Claire J Tomlin. 2021. Risk-sensitive safety analysis using Conditional Value-at-Risk. *IEEE Trans. Automat. Control* (2021).

[22] Yi Chou, Hansol Yoon, and Sriram Sankaranarayanan. 2020. Predictive runtime monitoring of vehicle models using Bayesian estimation and reachability analysis. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2111–2118.

[23] Edmund M Clarke. 1997. Model checking. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*. Springer, 54–56.

[24] 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.

[25] Alexandre Donzé and Oded Maler. 2010. Robust Satisfaction of Temporal Logic over Real-valued Signals. In *Proc. Int. Conf. FORMATS*. Klosterneuburg, Austria, 92–106.

[26] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*. PMLR, 1–16.

[27] Georgios E Fainekos and George J Pappas. 2009. Robustness of temporal logic specifications for continuous-time signals. *Theoret. Comp. Science* 410, 42 (2009), 4262–4291.

[28] Angelo Ferrando and Giorgio Delzanno. 2021. Incrementally Predictive Runtime Verification. In *CILC*. 92–106.

[29] Bernd Finkbeiner, Christopher Hahn, Marvin Stenger, and Leander Tentrup. 2019. Monitoring hyperproperties. *Formal Methods in System Design* 54, 3 (2019), 336–363.

[30] Matteo Fontana, Gianluca Zeni, and Simone Vantini. 2023. Conformal prediction: A unified review of theory and new challenges. *Bernoulli* 29, 1 (2023), 1 – 23.

[31] Luis Gressenbuch and Matthias Althoff. 2021. Predictive monitoring of traffic rules. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 915–922.

[32] Christopher Hahn. 2019. Algorithms for monitoring hyperproperties. In *International Conference on Runtime Verification*. Springer, 70–90.

[33] Hans Hansson and Bengt Jonsson. 1994. A logic for reasoning about time and reliability. *Formal aspects of computing* 6, 5 (1994), 512–535.

[34] 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.

[35] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[36] Radoslav Ivanov, Taylor J Carpenter, James Weimer, Rajeev Alur, George J Pappas, and Insup Lee. 2020. Case study: verifying the safety of an autonomous racing car with a neural network controller. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*. 1–7.

[37] Radoslav Ivanov, James Weimer, Rajeev Alur, George J Pappas, and Insup Lee. 2019. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid*

Systems: Computation and Control. 169–178.
[38] John Jackson, Luca Laurenti, Eric Frew, and Morteza Lahijanian. 2021. Formal verification of unknown dynamical systems via Gaussian process regression. *arXiv preprint arXiv:2201.00655* (2021).
[39] Manfred Jaeger, Kim G Larsen, and Alessandro Tibo. 2020. From statistical model checking to run-time monitoring using a bayesian network approach. In *International Conference on Runtime Verification*. Springer, 517–535.
[40] Markus Koschi, Christian Pek, Mona Beikirch, and Matthias Althoff. 2018. Set-based prediction of pedestrians in urban environments considering formalized traffic rules. In *2018 21st international conference on intelligent transportation systems (ITSC)*. IEEE, 2704–2711.
[41] 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.
[42] Axel Legay, Benoît Delahaye, and Saddek Bensalem. 2010. Statistical model checking: An overview. In *International conference on runtime verification*. Springer, 122–135.
[43] Axel Legay, Anna Lukina, Louis Marie Traonouez, Junxing Yang, Scott A Smolka, and Radu Grosu. 2019. Statistical model checking. In *Computing and Software Science*. Springer, 478–504.
[44] Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J Tibshirani, and Larry Wasserman. 2018. Distribution-free predictive inference for regression. *J. Amer. Statist. Assoc.* 113, 523 (2018), 1094–1111.
[45] Martin Leucker and Christian Schallhart. 2009. A brief account of runtime verification. *The journal of logic and algebraic programming* 78, 5 (2009), 293–303.
[46] Lars Lindemann, Matthew Cleaveland, Gihyun Shim, and George J Pappas. 2022. Safe Planning in Dynamic Environments using Conformal Prediction. *arXiv preprint arXiv:2210.10254* (2022).
[47] Lars Lindemann, Lejun Jiang, Nikolai Matni, and George J Pappas. 2022. Risk of Stochastic Systems for Temporal Logic Specifications. *arXiv preprint arXiv:2205.14523* (2022).
[48] Lars Lindemann, Alexander Robey, Lejun Jiang, Stephen Tu, and Nikolai Matni. 2021. Learning Robust Output Control Barrier Functions from Safe Expert Demonstrations. *arXiv preprint arXiv:2111.09971* (2021).
[49] Lars Lindemann, Alena Rodionova, and George Pappas. 2022. Temporal Robustness of Stochastic Signals. In *25th ACM International Conference on Hybrid Systems: Computation and Control*. 1–11.
[50] Zachary C Lipton, John Berkowitz, and Charles Elkan. 2015. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019* (2015).
[51] Anna Lukina, Christian Schilling, and Thomas A Henzinger. 2021. Into the unknown: Active monitoring of neural networks. In *International Conference on Runtime Verification*. Springer, 42–61.
[52] Rachel Luo, Shengjia Zhao, Jonathan Kuck, Boris Ivanovic, Silvio Savarese, Edward Schmerling, and Marco Pavone. 2021. Sample-efficient safety assurances using conformal prediction. *arXiv preprint arXiv:2109.14082* (2021).
[53] O. Maler and D. Nickovic. 2004. Monitoring temporal properties of continuous signals. In *Proc. Int. Conf. FORMATS FTRTFT*. Grenoble, France, 152–166.
[54] Srinivas Pinisetty, Thierry Jéron, Stavros Tripakis, Yliès Falcone, Hervé Marchand, and Viorel Preoteasa. 2017. Predictive runtime verification of timed properties. *Journal of Systems and Software* 132 (2017), 353–365.
[55] Amir Pnueli. 1977. The temporal logic of programs. In *Proc. Annual Symp. Found. Comp. Sci.* Washington, DC, 46–57.
[56] Xin Qin and Jyotirmoy V Deshmukh. 2020. Clairvoyant Monitoring for Signal Temporal Logic. In *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 178–195.
[57] Xin Qin, Yuan Xian, Aditya Zutshi, Chuchu Fan, and Jyotirmoy V Deshmukh. 2022. Statistical Verification of Cyber-Physical Systems using Surrogate Models and Conformal Inference. In *Proceedings of the International Conference on Cyber-Physical Systems*. Milan, Italy, 116–126.
[58] Alëna Rodionova, Lars Lindemann, Manfred Morari, and George J. Pappas. 2022. Temporal Robustness of Temporal Logic Specifications: Analysis and Control Design. *ACM Trans. Embed. Comput. Syst.* (July 2022).
[59] Nima Roohi, Yu Wang, Matthew West, Geir E Dullerud, and Mahesh Viswanathan. 2017. Statistical verification of the Toyota powertrain control verification benchmark. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*. Pittsburgh, Pennsylvania, 65–70.
[60] Stéphane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*. Sardinia, Italy, 661–668.
[61] Ivan Ruchkin, Matthew Cleaveland, Radoslav Ivanov, Pengyuan Lu, Taylor Carpenter, Oleg Sokolsky, and Insup Lee. 2022. Confidence Composition for Monitors of Verification Assumptions. In *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 1–12.
[62] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. 2020. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research* 39, 8 (2020), 895–935.
[63] S. Sadraddini and C. Belta. 2015. Robust temporal logic model predictive control. In *Proceedings of the 53rd Annual Allerton Conference on Communication, Control, and Computing*. Monticello, IL, 772–779. https://doi.org/10.1109/ALLERTON.2015.7447084
[64] Ali Salamati, Sadegh Soudjani, and Majid Zamani. 2020. Data-Driven Verification under Signal Temporal Logic Constraints. *IFAC-PapersOnLine* 53, 2 (2020), 69–74.
[65] Ali Salamati, Sadegh Soudjani, and Majid Zamani. 2021. Data-driven verification of stochastic linear systems with signal temporal logic constraints. *Automatica* 131 (2021), 109781.
[66] Daniel Selvaratnam, Michael Cantoni, JM Davoren, and Iman Shames. 2022. MITL Verification Under Timing Uncertainty. *arXiv preprint arXiv:2204.10493* (2022).
[67] Glenn Shafer and Vladimir Vovk. 2008. A Tutorial on Conformal Prediction. *Journal of Machine Learning Research* 9, 3 (2008).
[68] Mary Sheeran, Satnam Singh, and Gunnar Stålmarck. 2000. Checking safety properties using induction and a SAT-solver. In *International conference on formal methods in computer-aided design*. Springer, 127–144.
[69] Yasser Shoukry, Pierluigi Nuzzo, Alberto L Sangiovanni-Vincentelli, Sanjit A Seshia, George J Pappas, and Paulo Tabuada. 2017. SMC: Satisfiability modulo convex optimization. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*. 19–28.
[70] A Prasad Sistla, Miloš Žefran, and Yao Feng. 2011. Runtime monitoring of stochastic cyber-physical systems with hybrid state. In *International Conference on Runtime Verification*. Springer, 276–293.
[71] Kamile Stankeviciute, Ahmed M Alaa, and Mihaela van der Schaar. 2021. Conformal time-series forecasting. *Advances in Neural Information Processing Systems* 34 (2021), 6216–6228.
[72] 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.
[73] Minghu Tan, Hong Shen, Kang Xi, and Bin Chai. 2022. Trajectory prediction of flying vehicles based on deep learning methods. *Applied Intelligence* (2022), 1–22.
[74] Ryan J Tibshirani, Rina Foygel Barber, Emmanuel Candes, and Aaditya Ramdas. 2019. Conformal prediction under covariate shift. *Advances in neural information processing systems* 32 (2019).
[75] 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.
[76] 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.
[77] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. 2005. *Algorithmic learning in a random world*. Springer Science & Business Media.
[78] Yu Wang, Mojtaba Zarei, Borzoo Bonakdarpoor, and Miroslav Pajic. 2021. Probabilistic conformance for cyber-physical systems. In *Proceedings of the Conference on Cyber-Physical Systems*. Nashville, Tennessee, 55–66.
[79] Yu Wang, Mojtaba Zarei, Borzoo Bonakdarpour, and Miroslav Pajic. 2019. Statistical verification of hyperproperties for cyber-physical systems. *ACM Transactions on Embedded Computing Systems (TECS)* 18, 5s (2019), 1–23.
[80] Cristina M Wilcox and Brian C Williams. 2010. Runtime verification of stochastic, faulty systems. In *International Conference on Runtime Verification*. Springer, 452–459.
[81] Hansol Yoon, Yi Chou, Xin Chen, Eric Frew, and Sriram Sankaranarayanan. 2019. Predictive runtime monitoring for linear stochastic systems and applications to geofence enforcement for UAVs. In *International Conference on Runtime Verification*. Springer, 349–367.
[82] Hansol Yoon and Sriram Sankaranarayanan. 2021. Predictive runtime monitoring for mobile robots using logic-based bayesian intent inference. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 8565–8571.
[83] Håkan LS Younes and Reid G Simmons. 2002. Probabilistic verification of discrete event systems using acceptance sampling. In *International Conference on Computer Aided Verification*. Springer, 223–235.
[84] Håkan LS Younes and Reid G Simmons. 2006. Statistical probabilistic model checking with a focus on time-bounded properties. *Information and Computation* 204, 9 (2006), 1368–1409.
[85] Xinyi Yu, Weijie Dong, Xiang Yin, and Shaoyuan Li. 2022. Model Predictive Monitoring of Dynamic Systems for Signal Temporal Logic Specifications. *arXiv preprint arXiv:2209.12493* (2022).
[86] Xinyi Yu, Weijie Dong, Xiang Yin, and Shaoyuan Li. 2022. Online Monitoring of Dynamic Systems for Signal Temporal Logic Specifications with Model Information. *arXiv preprint arXiv:2203.16267* (2022).
[87] Mojtaba Zarei, Yu Wang, and Miroslav Pajic. 2020. Statistical verification of learning-based cyber-physical systems. In *Proceedings of the Conference on Hybrid Systems: Computation and Control*. 1–7.

A SEMANTICS OF SIGNAL TEMPORAL LOGIC

For a signal $x := (x_0, x_1, \ldots)$, the semantics of an STL formula ϕ that is enabled at time τ_0, denoted by $(x, \tau_0) \models \phi$, can be recursively computed based on the structure of ϕ using the following rules:

$$\begin{aligned}
(x,\tau) &\models \text{True} && \text{iff} && \text{True},\\
(x,\tau) &\models \mu && \text{iff} && h(x_\tau) \geq 0,\\
(x,\tau) &\models \neg\phi && \text{iff} && (x,\tau) \not\models \phi,\\
(x,\tau) &\models \phi' \wedge \phi'' && \text{iff} && (x,\tau) \models \phi' \text{ and } (x,\tau) \models \phi'',\\
(x,\tau) &\models \phi' U_I \phi'' && \text{iff} && \exists \tau'' \in (\tau \oplus I) \cap \mathbb{N} \text{ s.t. } (x,\tau'') \models \phi''\\
& && && \text{and } \forall \tau' \in (\tau,\tau'') \cap \mathbb{N}, (x,\tau) \models \phi',\\
(x,\tau) &\models \phi' \underline{U}_I \phi'' && \text{iff} && \exists \tau'' \in (\tau \ominus I) \cap \mathbb{N} \text{ s.t. } (x,\tau'') \models \phi''\\
& && && \text{and } \forall \tau' \in (\tau'',\tau) \cap \mathbb{N}, (x,\tau) \models \phi'.
\end{aligned}$$

The robust semantics $\rho^\phi(x, \tau_0)$ provide more information than the semantics $(x, \tau_0) \models \phi$, and indicate how robustly a specification is satisfied or violated. We can again recursively calculate $\rho^\phi(x, \tau_0)$ based on the structure of ϕ using the following rules:

$$\begin{aligned}
\rho^{\text{True}}(x,\tau) &:= \infty,\\
\rho^{\mu}(x,\tau) &:= h(x_\tau)\\
\rho^{\neg\phi}(x,\tau) &:= -\rho^{\phi}(x,\tau),\\
\rho^{\phi'\wedge\phi''}(x,\tau) &:= \min(\rho^{\phi'}(x,\tau), \rho^{\phi''}(x,\tau)),\\
\rho^{\phi' U_I \phi''}(x,\tau) &:= \sup_{\tau'' \in (\tau\oplus I)\cap\mathbb{N}} \Big(\min\big(\rho^{\phi''}(x,\tau''), \inf_{\tau' \in (\tau,\tau'')\cap\mathbb{N}} \rho^{\phi'}(x,\tau')\big)\Big),\\
\rho^{\phi' \underline{U}_I \phi''}(x,\tau) &:= \sup_{\tau'' \in (\tau\ominus I)\cap\mathbb{N}} \Big(\min\big(\rho^{\phi''}(x,\tau''), \inf_{\tau' \in (\tau'',\tau)\cap\mathbb{N}} \rho^{\phi'}(x,\tau')\big)\Big).
\end{aligned}$$

The formula length $L^{\neg\phi}$ of a bounded STL formula ϕ can be recursively calculated based on the structure of ϕ using the following rules:

$$\begin{aligned}
L^{\text{True}} = L^{\mu} &:= 0\\
L^{\neg\phi} &:= L^{\phi}\\
L^{\phi'\wedge\phi''} &:= \max(L^{\phi'}, L^{\phi''})\\
L^{\phi' U_I \phi''} &:= \max\{I \cap \mathbb{N}\} + \max(L^{\phi'}, L^{\phi''})\\
L^{\phi' \underline{U}_I \phi''} &:= \max(L^{\phi'}, L^{\phi''}).
\end{aligned}$$

Lastly, we define the worst case robust semantics $\bar{\rho}^\phi(\hat{x}, \tau_0)$, which are again recursively defined as follows:

$$\begin{aligned}
\bar{\rho}^{\text{True}}(\hat{x},\tau) &:= \infty,\\
\bar{\rho}^{\mu}(\hat{x},\tau) &:= \begin{cases} h(x_\tau) & \text{if } \tau \leq t\\ \inf_{\zeta\in\mathcal{B}_\tau} h(\zeta) & \text{otherwise}\end{cases}\\
\bar{\rho}^{\neg\phi}(\hat{x},\tau) &:= -\bar{\rho}^{\phi}(\hat{x},\tau),\\
\bar{\rho}^{\phi'\wedge\phi''}(\hat{x},\tau) &:= \min(\bar{\rho}^{\phi'}(\hat{x},\tau), \bar{\rho}^{\phi''}(\hat{x},\tau)),\\
\bar{\rho}^{\phi' U_I \phi''}(\hat{x},\tau) &:= \sup_{\tau'' \in (\tau\oplus I)\cap\mathbb{N}} \Big(\min\big(\bar{\rho}^{\phi''}(\hat{x},\tau''), \inf_{\tau' \in (\tau,\tau'')\cap\mathbb{N}} \bar{\rho}^{\phi'}(\hat{x},\tau')\big)\Big),\\
\bar{\rho}^{\phi' \underline{U}_I \phi''}(\hat{x},\tau) &:= \sup_{\tau'' \in (\tau\ominus I)\cap\mathbb{N}} \Big(\min\big(\bar{\rho}^{\phi''}(\hat{x},\tau''), \inf_{\tau' \in (\tau'',\tau)\cap\mathbb{N}} \bar{\rho}^{\phi'}(\hat{x},\tau')\big)\Big).
\end{aligned}$$

B PROOF FOR THEOREM 1

The nonconformity scores $R^{(i)}$ are independent and identically distributed by their definition and Assumption 1. By [74, Lemma 1], we hence know that equation (3) is valid by the specific choice of C in equation (4). Consequently, we have that

$$P\big(\rho^\phi(X,\tau_0) \geq \rho^\phi(\hat{x},\tau_0) - C\big) \geq 1-\delta.$$

If now $\rho^\phi(\hat{x},\tau_0) > C$, it holds that $P(\rho^\phi(X,\tau_0) > 0) \geq 1-\delta$ by which it follows that

$$P\big((X,\tau_0) \models \phi\big) \geq 1-\delta$$

since $\rho^\phi(X,\tau_0) > 0$ implies $(X,\tau_0) \models \phi$ [25, 27].

C PROOF FOR THEOREM 2

Note first that $X_\tau \in \mathcal{B}_\tau$ for all times $\tau \in \{t+1, \ldots, t+H\}$ with a probability of at least $1-\delta$ by Lemma 1. For all predicates μ in the STL formula ϕ and for all times $\tau \in \{0, \ldots, t+H\}$, it hence holds that $\rho^\mu(X,\tau) \geq \bar{\rho}^\mu(\hat{x},\tau)$ with a probability of at least $1-\delta$ by the definition of $\bar{\rho}^\mu$. Since the formula ϕ does not contain negations[7], it is straightforward to show (inductively on the structure of ϕ) that $\rho^\phi(X,\tau) \geq \bar{\rho}^\phi(\hat{x},\tau)$ with a probability of at least $1-\delta$. Consequently, if $\bar{\rho}^\phi(\hat{x},\tau) > 0$, it holds that $P\big((X,\tau_0) \models \phi\big) \geq 1-\delta$ since $\rho^\phi(X,\tau_0) > 0$ implies $(X,\tau_0) \models \phi$ [25, 27].

D F-16 AIRCRAFT CASE STUDY DESCRIPTION

The F-16 has been used as a verification benchmark, and the authors in [34] provide a high-fidelity simulator for various maneuvers such as ground collision avoidance, see Figure 9 (left). The F-16 aircraft is modeled with 6 degrees of freedom nonlinear equations of motion, and the aircraft control system consists of an outer and an inner control-loop. The outer loop encodes the logic of the maneuver in a finite state automaton and provides reference trajectories to the inner loop. In the inner loop, the aircraft (modeled by 13 continuous states) is controlled by low-level integral tracking controllers (adding 3 additional continuous states), we refer the reader to [34] for details. In the simulator, we introduce randomness by uniformly sampling the initial conditions of the air speed, angle of attack, angle of sideslip, roll, pitch, yaw, roll rate, pitch rate, yaw rate, and altitude from a compact set.

[7] Negations would in fact flip the inequality in an unfavorable direction, e.g., for $\neg\mu$ it would hold that $\rho^{\neg\mu}(X,\tau) \leq \bar{\rho}^{\neg\mu}(\hat{x},\tau)$ with a probability of at least $1-\delta$.

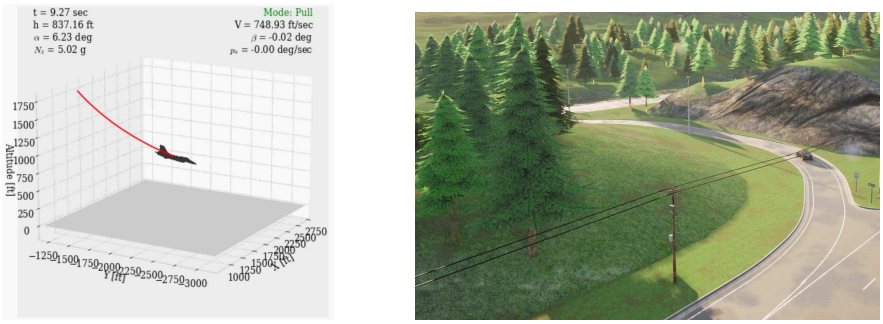

Figure 9: Left: F-16 Fighting Falcon within the high fidelity aircraft simulator from [34]. Right: Self-driving car within the autonomous driving simulator CARLA [26].