Conformal Prediction for STL Runtime Verification

Lars Lindemann∗
xin@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
pappas@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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICCPS ’23, May 9–12, 2023, San Antonio, TX, USA © 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 978-8-4007-0036-1/23/05... $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 Ω, 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
verification algorithms do not provide formal correctness guarantees unless restrictive assumptions are placed on the prediction algorithm or the underlying distribution 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 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 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.

![Figure 1: Overview of the proposed STL predictive runtime verification algorithms. Both algorithms use past observations \((x_0, \ldots, x_t)\) to obtain state predictions \((\hat{x}_{t+1}, \hat{x}_{t+2}, \ldots)\). The direct algorithm calculates the satisfaction measure \(\rho^{\phi}(\hat{x})\) of the specification \(\phi\) 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 \(x_{t+1}, x_{t+2}, \ldots\) using conformal prediction first, and then obtains a lower of the unknown satisfaction measure \(\rho^{\phi}(x)\) based on the state prediction regions.](image)
2 PROBLEM FORMULATION

Let $D$ be an unknown distribution over system trajectories that describe our system, i.e., let $X := (X_0, X_1, \ldots) \sim D$ be a random trajectory where $X_t$ denotes the state of the system at time $t$ that is drawn from $\mathbb{R}^n$. Modeling stochastic systems by a distribution $D$ provides great flexibility, and $D$ can generally describe the motion of Markov decision processes. It can capture stochastic systems whose trajectories follow the recursive update equation $X_{t+1} = f(X_t, w_t)$ where $w_t$ 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 $x_t$ for realizations of the random variable $X_t$. We make no assumptions on the distribution $D$, but assume availability of training and calibration data drawn from $D$.

ASSUMPTION 1. We have access to $K$ independent realizations $x^{(i)} := (x^{(i)}_0, x^{(i)}_1, \ldots)$ of the distribution $D$ that are collected in the dataset $D := \{x^{(1)}, \ldots, x^{(K)}\}$.

Informal Problem Formulation. Assume now that we are given a specification $\phi$ for the stochastic system $D$, e.g., a safety or performance specification defined over the states $X_t$ of the system. In “offline” system verification, e.g., in statistical model checking, we are interested in calculating the probability that $(X_0, X_1, \ldots) \sim D$ satisfies the specification. In runtime verification, on the other hand, we have already observed the partial realization $(x_0, x_1, \ldots, x_t)$ of $(X_0, \ldots, X_t)$ online at time $t$, and we want to use this information to calculate the probability that $(X_0, X_1, \ldots) \sim D$ satisfies the specification.\footnote{We note that we consider unconditional probabilities in this paper.}

While in “offline” verification all realizations of $D$ 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 $(X_0, X_1, \ldots) \sim D$ of which we have plotted ten realizations in Figure 2 (left). In an offline approach, this system satisfies the specification $\inf_{x \in [0,250]} X_t \in [0,3] \geq 0$ 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 $\hat{x}_{t|100}$ (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 $\mu : \mathbb{R}^n \rightarrow \{\text{True}, \text{False}\}$. For convenience, the predicate $\mu$ is often defined via a predicate function $h : \mathbb{R}^n \rightarrow \mathbb{R}$ as $\mu(x) := \text{True}$ if $h(x) \geq 0$ and $\mu(x) := \text{False}$ otherwise. The syntax of STL is recursively defined as

$$\phi := \text{True} \mid \neg \phi' \mid \phi' \land \phi'' \mid \phi' U \phi'' \mid \phi' W \phi''$$

where $\phi'$ and $\phi''$ are STL formulas. The Boolean operators $\neg$ and $\land$ encode negations (“not”) and conjunctions (“and”), respectively. The until operator $\phi' U \phi''$ encodes that $\phi''$ has to be true from now on until $\phi''$ becomes true at some future time within the time interval $I \subseteq \mathbb{R}_{\geq 0}$. The since operator encodes that $\phi''$ was true at some past time within the time interval $I$ and since then $\phi'$ is true.

To determine if a signal $x$ satisfies an STL formula $\phi$ that is enabled at time $t_0$, we can define the semantics as a relation $\models$, i.e., $(x, t_0) \models \phi$ means that $\phi$ is satisfied. While the STL semantics are fairly standard [53], we recall them in Appendix A. Importantly, it holds that $(x, t_0) \models \phi$ if $\phi^D(x, t_0) > 0$. We make the following assumption on the class of STL formulas in this paper.

ASSUMPTION 2. We consider bounded STL formulas $\phi$, i.e., all time intervals $I$ within the formula $\phi$ are bounded.

Satisfaction of bounded STL formulas can be decided by finite length signals [63]. The minimum length is indicated by the formula length $L^\phi$, i.e., knowledge of $(x_0, \ldots, x_{t+L^\phi})$ is enough to determine if $(x, t_0) \models \phi$. We recall the definition of $L^\phi$ 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 $\text{Predict}$ 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 $D_{\text{train}}$ and $D_{\text{cal}}$, respectively, and learn $\text{Predict}$ from $D_{\text{train}}$.

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

$$a^{1}_r := A(x_r, a^{1}_{r-1}),$$

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2.png}
\caption{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.}
\end{figure}
We recall that \( y \phi \) is enabled at \( \tau \). This may, for instance, be important when monitoring the current time coincides with the time the specification is enabled. \( \phi \) is a bounded STL specification that does not contain any information about the uncertainty of the current time. To obtain a prediction region for \( \phi \), we would like to point out two special instances of Problem 1. As \( C \) depends on \( R^{(1)}, \ldots, R^{(k)} \), the probability measure \( P \) is defined over the product measure of \( R^{(0)}, \ldots, R^{(k)} \). 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 \( R^{(1)}, \ldots, R^{(k)} \) and \( \infty \). By assuming that \( R^{(1)}, \ldots, R^{(k)} \) are sorted in non-decreasing order, and by adding \( R^{(k+1)}\) \( = \infty \), we can equivalently obtain \( C = R^{(p)} \) where \( p = [(k+1)(1-\delta)], i.e., \( C \) is the pth smallest nonconformity score.

### 3.2 Direct STL Predictive Runtime Verification

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

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

\[
\hat{x} := (x_{obs}, \hat{x}_{t+1}[t], \ldots, \hat{x}_{t+H}[t])
\]

which is the concatenation of the current observations \( x_{obs} \) and the predictions of future states \( \hat{x}_{t+1}[t], \ldots, \hat{x}_{t+H}[t] \). 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

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

Note that \( \rho^\phi(\hat{x}, \tau_0) \) is the predicted robust semantics for the specification \( \phi \) that we can calculate at time \( t \) based on the observations \( x_{obs} \) and the predictions \( \hat{x}_{t+1}[t], \ldots, \hat{x}_{t+H}[t] \). 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 \( x^{(i)} \in D_{cal} \) as

\[
R^{(i)} := \rho^\phi(\hat{x}^{(i)}, \tau_0) - \rho^\phi(x^{(i)}, \tau_0)
\]
where \( \hat{x}^{(i)} := (x_{obs}^{(i)}, x_{t+1}^{(i)}, \ldots, x_{t+H}^{(i)}) \) resembles equation (2), but now defined for the calibration trajectory \( x^{(i)} \). A positive nonconformity score \( R^{(i)} \) indicates that our predictions are too optimistic, i.e., the predicted robust semantics \( R^{\phi} (\hat{x}^{(i)}, \tau_0) \) is greater than the actual robust semantics \( \hat{x}^{(i)} (x^{(i)}, \tau_0) \) 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 \( R^{(|D_{cal} + 1|)} := \infty \) as the \( (|D_{cal}| + 1) \)th value.

**Theorem 1.** Given a distribution \( (X_0, X_1, \ldots) \sim D \), the current time \( t \), the observations \( x_{obs} := (x_0, \ldots, x_t) \), a bounded STL formula \( \phi \) that is enabled at \( \tau_0 \), the dataset \( D_{cal} \), and a failure probability \( \delta \in (0, 1) \). Then the prediction region in equation (3) is valid if \( C \) defined as

\[
C := R^{(p)} \quad \text{where} \quad p := \left[ (|D_{cal}| + 1) (1 - \delta) \right],
\]

and it holds that \( P(\{ (X, \tau_0) \mid \phi \} \geq 1 - \delta) \) if \( R^{(p)} (\hat{x}, \tau_0) > C \).

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 \( \phi \) changes, as in existing work such as [14, 57]. This is since we use trajectory predictors to obtain state predictions \( \hat{x}_{t|\tau} \) that are specification independent.

**Remark 1.** Note that Theorem 1 assumes a fixed failure probability \( \delta \). If one wants to find the tightest with the smallest failure probability \( \delta \) so that \( P(\{ (X, \tau_0) \mid \phi \} \geq 1 - \delta) \) holds, we can (approximately) find the smallest such \( \delta \) 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 \( P(\{ (X, \tau_0) \mid \phi \} > 0) \geq 1 - \delta \) 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 \( \hat{x}_{t|\tau} \), \( \hat{x}_{t+1|\tau}, \ldots, \hat{x}_{t+H|\tau} \), 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 \( C_{\tau} \) so that

\[
P(\| X_{\tau, t+1} - \hat{x}_{t|\tau} \| \leq C_{\tau} \quad \forall \tau \in \{ t + 1, \ldots, t + H \}) \geq 1 - \delta.
\]

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

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

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

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

\[
C_{\tau} := R^{(p)} \quad \text{where} \quad p := \left[ (|D_{cal}| + 1) (1 - \delta) \right] \text{ and } \delta = \delta / H.
\]

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

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

For predicates \( \mu \), we define these semantics as

\[
\rho^{\mu} (\hat{x}, \tau) := \begin{cases} h(\hat{x}_t) & \text{if } \tau \leq t, \\ \inf_{\hat{x} \in B_t} h(\hat{x}) & \text{otherwise} \end{cases}
\]

where we recall the definition of the predicted trajectory \( \hat{x} \) in equation (2) and where \( B_t := \{ \hat{x} \in \mathbb{R}^n \mid \| \hat{x} - \hat{x}_{t|\tau} \| \leq C_{\tau} \} \) is a ball of size \( C_{\tau} \) centered around the prediction \( \hat{x}_{t|\tau} \), i.e., \( B_t \) defines the set of states within the prediction region at time \( \tau \). The intuition behind this definition is that we know the value of the robust semantics \( \rho^{\mu} (X, \tau_0) = \rho^{\phi} (\hat{x}, \tau) \) if \( \tau \leq t \) since \( \hat{x}_t \) is known. For times \( \tau > t \), we know that \( X_t \in B_{t} \) holds with a probability of at least \( 1 - \delta \) by Lemma 1 so that we compute \( \rho^{\phi} (\hat{x}, \tau) := \inf_{\hat{x} \in B_t} h(\hat{x}) \) to obtain a lower bound for \( \rho^{\phi} (X, \tau) \) with a probability of at least \( 1 - \delta \).

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

\[
\inf_{\hat{x} \in B_t} h(\hat{x}) \geq h(\hat{x}_t) - L_h C_{\tau}.
\]

For instance, the constraint \( h(\hat{x}) := \| \hat{x} - \hat{x}_t \| > 0.5 \), which can encode collision avoidance constraints, has Lipschitz constant one.
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 $\rho^\phi$, 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 $\phi$ in positive normal form that is enabled at $t_0$. Then it holds that $P((X, t_0) \models \phi) \geq 1 - \delta$ if $\bar{\rho}^\phi((X, t_0)) > 0$.

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 $\delta$ 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 $C_H = \infty$ if $p > |D_{cal}|$). On the other hand, the indirect method is more interpretable and allows to identify parts of the formula $\phi$ that may be violated by analyzing the uncertainty of predicates via the worst case robust semantics $\bar{\rho}^\phi((X, t))$. 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 $D_{cal}$ 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 $D_{train} := 1520$ training trajectories, $D_{cal} := 5680$ calibration trajectories, and $D_{test} := 100$ test trajectories. From $D_{train}$, we trained an LSTM of depth two and width 50 to predict future states of $h$. 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 $t_0 := t$, 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 $\rho^\phi((X, t_0)) - \rho^\phi((X, t_0))$ 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 $R^{(i)}$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 $\rho^\phi((X, t_0))$ and the ground truth robustness $\rho^\phi((X, t_0))$. We found that for $100 = |D_{test}|$ trajectories it holds that $\rho^\phi((X, t_0)) > C$ implies $(X, t_0) \models \phi$, confirming Theorem 1. We also validated equation (5) and found that $96/100$ trajectories satisfy $\rho^\phi((X, t_0)) - \rho^\phi((X, t_0)) \leq C$.

Let us now use the indirect predictive runtime verification algorithm. We first obtain prediction regions of $\|X_{t} - \hat{X}_{t}|\|$ by calculating $C_T$ according to Lemma 1. We show the histograms for three different $T$ in Figure 4 (left, left-mid, right-mid). We also indicate the prediction regions $C_T$ by vertical lines (note that $\delta = \delta/200$ in this case). We can observe that larger prediction times $T$ result in larger prediction regions $C_T$. This is natural as the trajectory predictor is expected to perform worse for larger $T$. In a next step, we empirically evaluate the results of Theorem 2 by calculating the worst case robust semantic $\bar{\rho}^\phi((X, t_0))$ for the test trajectories $D_{test}$. In Figure 4 (right), we plot the worst case robustness $\rho^\phi((X, t_0))$ and the ground truth robustness $\rho^\phi((X, t_0))$. We found that for $100 = |D_{test}|$ trajectories it holds that $\bar{\rho}^\phi((X, t_0)) > 0$ implies $(X, t_0) \models \phi$, 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 $\bar{\rho}^\phi((X, t))$ as previously remarked in 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 $e_c$ (deviation of the car from the center of the lane) and the orientation error $e_{\theta}$ (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 $[e_c, e_{\theta}] \in [-1, 1] \times [-0.4, 0.4]$. We obtained 1000 trajectories for each controller, and use $|D_{train}| := 700$ trajectories to train an LSTM, while we use $|D_{cal}| := 200$ trajectories to obtain conformal prediction regions. The remaining $|D_{test}| := 100$ 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 $e_c$ and $e_{\theta}$ for each controller, respectively. Particularly, the plots show the best five
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 $\rho^\phi(x^{(i)}, \tau_0)$ 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 $\tau$ step ahead prediction on $D_{cal}$ for $\tau \in \{50, 100, 200\}$ and the indirect method, right: worst case predicted robustness $\bar{\rho}^\phi(x^{(i)}, \tau_0)$ 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 $\theta_e$ predictions, right: five worst $\theta_e$ 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 $\theta_e$ predictions, right: five worst $\theta_e$ predictions.
and the worst five LSTM predictions (in terms of the mean square error) on the test trajectories $D_{\text{test}}$.

For the verification of the car, we consider the following two STL specifications that are enabled at $t_0 := 0$:

$$\phi_1 := G_{[10,\infty)}(|c_e| \leq 2.25),$$

$$\phi_2 := G_{[10,\infty)}(|c_e| > 2.25) \implies f_{[0.5]}G_{[0.5]}(|c_e| \leq 1.25).$$

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 2.25 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 $t := 273$ 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 $D_{\text{cal}}$ 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 $\hat{\rho}^\phi(\hat{x}, t_0)$ and the ground truth robustness $\rho^\phi(X, t_0)$ for each specification $\phi \in \{\phi_1, \phi_2\}$ by calculating $C$ according to Theorem 1.

For the first specification $\phi_1$, we show the histograms of $R^{(i)}$ for both controllers over the calibration data $D_{\text{cal}}$ 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 $D_{\text{test}}$. In Figure 8 (left: IL, left-mid: CBF), we plot the predicted robustness $\hat{\rho}^\phi(\hat{x}, t_0)$ and the ground truth robustness $\rho^\phi(X, t_0)$. We found that for 99 of the 100 $|D_{\text{test}}|$ trajectories under the IL controller and for 99/100 trajectories under the CBF controller it holds that $\rho^\phi(X, t_0) > \hat{\phi}_2$, 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 $\rho^\phi(\hat{x}(i), t_0) - \rho^\phi(\hat{x}(i), t_0) \leq C$.

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.

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 on 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


Figure 7: Histograms of the nonconformal scores $R_i^{(i)}$ on $D_{\text{cal}}$ and prediction region $C$. Left: IL controller and $\phi_1$, left-mid: CBF controller and $\phi_2$, right: CBF controller and $\phi_2$.

Figure 8: Predicted robustness $\rho^\phi(x^{(i)}, \tau_0)$ and ground truth robustness $\rho^\phi(x^{(i)}, \tau_0)$ on $D_{\text{test}}$. Left: IL controller and $\phi_1$, left-mid: CBF controller and $\phi_1$, right-mid: IL controller and $\phi_2$, right: CBF controller and $\phi_2$.
A SEMANTICS OF SIGNAL TEMPORAL LOGIC

For a signal \( x := (x_0, x_1, \ldots) \), the semantics of an STL formula \( \phi \) that is enabled at time \( t_0 \), denoted by \( (x, t_0) \models \phi \), can be recursively computed based on the structure of \( \phi \) using the following rules:

\[
\begin{align*}
(x, t) &\models 0 & \text{iff} & 0, \\
(x, t) &\models \mu & \text{iff} & h(x_t) \geq 0, \\
(x, t) &\models \neg \phi & \text{iff} & (x, t) \not\models \phi, \\
(x, t) &\models \phi \land \phi'' & \text{iff} & (x, t) \models \phi' \land (x, t) \models \phi'', \\
(x, t) &\models \phi U \phi'' & \text{iff} & \exists \tau' \in (\tau \oplus I) \cap \mathbb{N} \text{ such that } (x, t') \models \phi'' \\
& & & \text{and } \forall \tau' \in (\tau, \tau') \cap \mathbb{N}, (x, t) \models \phi', \\
(x, t) &\models \phi U \phi'' & \text{iff} & \exists \tau'' \in (\tau \oplus I) \cap \mathbb{N} \text{ such that } (x, t'') \models \phi'' \\
& & & \text{and } \forall \tau'' \in (\tau', \tau'') \cap \mathbb{N}, (x, t) \models \phi'.
\end{align*}
\]

The robust semantics \( \rho^\phi(x, t_0) \) provide more information than the semantics \( (x, t_0) \models \phi \), and indicate how robustly a specification is satisfied or violated. We can again recursively calculate \( \rho^\phi(x, t_0) \) based on the structure of \( \phi \) using the following rules:

\[
\begin{align*}
\rho^\phi_{\text{True}}(x, t) &:= \infty, \\
\rho^\phi_{\mu}(x, t) &:= h(x_t), \\
\rho^\phi_{\neg \phi}(x, t) &:= -\rho^\phi(x, t), \\
\rho^\phi_{\phi \land \phi''}(x, t) &:= \min(\rho^\phi(x, t), \rho^\phi(x, t)), \\
\rho^\phi_{\phi U \phi''}(x, t) &:= \sup_{\tau' \in (\tau \oplus I) \cap \mathbb{N}} \min(\rho^\phi(x, t'), \inf_{\tau'' \in (\tau, \tau') \cap \mathbb{N}} \rho^\phi(x, t'')), \\
\rho^\phi_{\phi U \phi''}(x, t) &:= \sup_{\tau' \in (\tau \oplus I) \cap \mathbb{N}} \min(\rho^\phi(x, t'), \inf_{\tau'' \in (\tau', \tau'') \cap \mathbb{N}} \rho^\phi(x, t'')).
\end{align*}
\]

The robust semantics \( \rho^\phi(x, t_0) \) 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

\[
P(\rho^\phi(X, t_0) \geq C) \geq 1 - \delta.
\]

If now \( \rho^\phi(x, t_0) \geq C \), it holds that \( P(\rho^\phi(X, t_0) \geq 0) \geq 1 - \delta \) by which it follows that

\[
P((X, t_0) \models \phi) \geq 1 - \delta
\]

since \( \rho^\phi(X, t_0) > 0 \) implies \( (X, t_0) \models \phi \).  

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

\[
P(\rho^\phi(X, t_0) \geq C) \geq 1 - \delta.
\]

If now \( \rho^\phi(x, t_0) \geq C \), it holds that \( P(\rho^\phi(X, t_0) > 0) \geq 1 - \delta \) by which it follows that

\[
P((X, t_0) \models \phi) \geq 1 - \delta
\]

since \( \rho^\phi(X, t_0) > 0 \) implies \( (X, t_0) \models \phi \). 

C PROOF FOR THEOREM 2

Note first that \( X_r \in \mathcal{B}_r \) for all times \( r \in \{t + 1, \ldots, t + H\} \) with a probability of at least \( 1 - \delta \) by Lemma 1. For all predicates \( \mu \) in the STL formula \( \phi \) and for all times \( r \in \{0, \ldots, t + H\} \), it hence holds that \( \rho^\mu(X, t) \geq \rho^\mu(\hat{x}, \tau) \) with a probability of at least \( 1 - \delta \) by the definition of \( \rho^\mu \). Since the formula \( \phi \) does not contain negations, it is straightforward to show (inductively on the structure of \( \phi \)) that \( \rho^\phi(x, t) \geq \rho^\phi(\hat{x}, \tau) \) with a probability of at least \( 1 - \delta \). Consequently, if \( \rho^\phi(\hat{x}, \tau) > 0 \), it holds that \( P((X, t_0) \models \phi) \geq 1 - \delta \). 

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.
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].