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.

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) (ICCP'S '23), May 9–12, 2023, San Antonio, TX, USA. ACM, New York, NY, USA, 12 pages. 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 \( D \), against a system specification \( \phi \) 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 veriﬁying 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 \( \rho^\phi(x) \) 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.
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 $X_{t}$ denotes the state of the system at time $t$ 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 $X_{t+1} = f(X_{t}, w_{t})$ 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 $\mathcal{D}$, but assume availability of training and calibration data drawn from $\mathcal{D}$.

**Assumption 1.** We have access to $K$ independent realizations $x^{(i)} := (x^{(i)}_{0}, x^{(i)}_{1}, \ldots)$ of the distribution $\mathcal{D}$ that are collected in the dataset $\mathcal{D} := \{x^{(1)}, \ldots, x^{(K)}\}$.

**Informal Problem Formulation.** Assume now that we are given a specification $\phi$ for the stochastic system $\mathcal{D}$, 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 $\{X_{0}, X_{1}, \ldots\} \sim \mathcal{D}$ 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. In this paper, we use predictions $\hat{x}_{t|\tau}$ of future states $X_{t}$ 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 $\inf_{\mathcal{D}} \{150, 250\}$ $X_{t} \in [0, 3] \geq 0$ 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

$$\phi := \text{True} \mid \mu \mid \neg \phi' \mid \phi' \land \phi'' \mid \phi' U \phi'' \mid \phi' U_{\tau} \phi''$$

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. We can further derive the operators for disjunction ($\phi' \lor \phi'' := \neg(\neg \phi' \land \neg \phi'')$), eventually ($F_{\epsilon} \phi := \top \cup I \phi$), once ($E_{\epsilon} \phi := \top \cup I_{\phi}$), always ($G_{\epsilon} \phi := \neg F_{\epsilon} \neg \phi$), and historically ($G_{\epsilon} \phi := \neg E_{\epsilon} \neg \phi$).

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 [33], we recall them in Appendix A. Additionally, we can define robust (sometimes referred to as quantitative) semantics $\rho^{(\phi)}(x, t_{0}) \in \mathbb{R}$ that indicate how robustly the formula $\phi$ is satisfied or violated [25, 27], see Appendix A. Larger and positive values of $\rho^{(\phi)}(x, t_{0})$ indicate that the specification is satisfied more robustly. Importantly, it holds that $(x, t_{0}) \models \phi$ if $\rho^{(\phi)}(x, t_{0}) > 0$. 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 $(x_{0}, \ldots, x_{L^{\phi}+\epsilon})$ 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 $(x_{t+1}, \ldots, x_{t+H})$ of $(x_{t+1}, \ldots, x_{t+H})$.

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

A specific example of $\text{Predict}$ 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

$$a_{t}^{\tau} := \mathcal{A}(x_{t}, a_{t-\tau}^{\tau-1})$$
where $x_t$ 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 $a^0, \ldots, a^d$ are the hidden states. The output $y_{t+1} := y(a^d)$ provides an estimate of $(x_{t+1}, \ldots, x_{t+H})$ via the function $\mathcal{Y}$ which typically parameterizes a linear last layer.

### 2.3 Predictive Runtime Verification

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

**Problem 1.** *Given a distribution $(X_0, X_1, \ldots) \sim D$, the current time $t$, the observations $x_{\text{obs}} := (x_0, x_1)$, a bounded STL formula $\phi$ that is enabled at $\tau_0$, and a failure probability $\delta \in (0, 1)$, determine if $P((X, \tau_0) \models \phi) \geq 1 - \delta$ holds.*

Several comments are in order. Note that we use the system specification $\phi$ (and not its negation $\neg \phi$) to determine if $\phi$ is satisfied. From $P((X, \tau_0) \models \phi) \geq 1 - \delta$, we can infer that $P((X, \tau_0) \models \neg \phi) \leq \delta$, 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 $\tilde{C} \in \mathbb{R}$ on the robust semantics $\rho^\phi(X, \tau_0)$, i.e., so that $P(\rho^\phi(X, \tau_0) \geq \tilde{C}) \geq 1 - \delta$.

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 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 $X_t$ 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 $R^{(0)}, \ldots, R^{(k)}$ 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 $R^{(i)} := ||X^{(i)} - \mu(||X^{(i)}||)||$ where the predictor $\mu$ 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 $R^{(1)}, \ldots, R^{(k)}$, 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 $R^{(1)}, \ldots, R^{(k)}$ such that

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

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

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 $\phi$. Define now the predicted trajectory

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

which is the concatenation of the current observations $x_{\text{obs}}$ 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 $x_{\text{obs}}$ 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 $x^{(i)} \in D_{\text{cal}}$ as

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

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

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

and it holds that $P([X(t) \equiv \phi] \geq 1 - \delta$ if $\rho^B(\hat{x}, \tau_0) > C$.

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 $\hat{x}_t$ 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 $P([X(t) \equiv \phi] \geq 1 - \delta$ 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 $P([X(t) \equiv \phi] > 0 \equiv \phi) \geq 1 - \delta$ if $\rho^B(\hat{x}, \tau_0) > C$, 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 $\text{cal}$. We thereby obtain probabilistic guarantees for the verification procedure, but we do not obtain guarantees conditional on $\text{cal}$.

### 3.3 Indirect STL Predictive Runtime Verification

We now present the indirect method where we first obtain prediction regions for the state predictions $\hat{x}_{t+1}, \ldots, \hat{x}_{t+H|t}$, 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 $C_\tau$ so that

$$P(\|X_t - \hat{x}_{t|t}\| \leq C_\tau, \forall t \in \{t + 1, \ldots, t + H\}) \geq 1 - \delta$$

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

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

where we recall that $\hat{x}^{(i)}$ is the prediction obtained from the observed calibration trajectory $x^{(i)}$. A large nonconformity score indicates that the state predictions $\hat{x}^{(i)}$ 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 $R^{(\text{cal} + 1)} := \infty$ as the $(\text{cal} + 1)$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 $x_{\text{obs}} := (x_0, \ldots, x_t)$, the dataset $\text{cal}$, and a failure probability $\delta \in (0, 1)$. Then the prediction regions in equation (5) are valid with $C_\tau$ defined as

$$C_\tau := R^{(p)} \text{ where } p := \left(\left|\text{cal} + 1\right| - \delta\right)$$

and $\delta$ is 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 $L^B$, 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 $\hat{x}_{t|t}$ from equation (5) to obtain prediction regions for $\rho^B(x, \tau)$ to solve Problem 1. The main idea is to calculate the worst case of the robust semantics $\rho^B$ 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 $\phi$ 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 $\rho^B$ of the robust semantics $\rho^B$ that incorporates the prediction regions from equation (5). For predicates $\mu$, we define these semantics as

$$\rho^B(\hat{x}, \tau) := \begin{cases} h(x_t) & \text{if } \tau \leq t \\ \inf_{\xi \in \mathbb{R}} h(\xi) & \text{otherwise} \end{cases}$$

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

**Remark 3.** For convex predicate functions $h$, computing $\inf_{\xi \in \mathbb{R}} h(\xi)$ is a convex optimization problem that can efficiently be solved. However, note that the optimization problem $\inf_{\xi \in \mathbb{R}} h(\xi)$ 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 $\inf_{\xi \in \mathbb{R}} h(\xi)$ that we can use instead. Particularly, let $L_\mu$ be the Lipschitz constant of $h$, i.e., let $\|h(\xi) - h(\hat{x}_{t|t})\| \leq L \|\xi - \hat{x}_{t|t}\|$. Then, we know that

$$\inf_{\xi \in \mathbb{R}} h(\xi) \geq h(\hat{x}_{t|t}) - L_\mu C_\tau.$$

For instance, the constraint $h(\xi) := \|\xi - \xi_0\| - 0.5$, which can encode collision avoidance constraints, has Lipschitz constant one.
The worst case robust semantics $\bar{\rho}$ 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 $\rho$, 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 $P(X(t, t_0) \models \phi) \geq 1 - \delta$ if $\bar{\rho}(\hat{x}, t_0) > 0$.

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 $D\text{cal}$ compared to the direct method to achieve $p \leq |D\text{cal}|$ (recall that $C_\varepsilon = \infty$ if $p > |D\text{cal}|$). 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 $\bar{\rho}(\hat{x}, t)$. 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 $D\text{cal}$ 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-manoeuvrable 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 $D\text{train}$, 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 $D\text{test}$ in Figure 3 (left and left-mid).

We are interested in a safety expression specified 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 $\bar{\rho}(\hat{x}, t_0) - \rho(\hat{x}, t_0)$ by calculating $C$ according to Theorem 1. We show the histograms of $R(i)$ over the calibration data $D\text{cal}$ 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 $D\text{test}$. In Figure 3 (right), we plot the predicted robustness $\bar{\rho}(\hat{x}, t_0)$ and the ground truth robustness $\rho(\hat{x}, t_0)$. We found that for $100$ of the $100 = |D\text{test}|$ trajectories it holds that $\rho(\hat{x}, t_0) > C$ implies $(x(t_0), \tau_0) \models \phi$, confirming Theorem 1. We also validated equation (5) and found that $96/100$ trajectories satisfy $\bar{\rho}(\hat{x}, t_0) - \rho(\hat{x}, t_0) \leq C$.

Let us now use the indirect predictive runtime verification algorithm. We first obtain prediction regions of $\|X - \hat{x}_t\|$ 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 $\bar{\rho}(\hat{x}, t_0)$ for the test trajectories $D\text{test}$. In Figure 4 (right), we plot the worst case robustness $\bar{\rho}(\hat{x}, t_0)$ and the ground truth robustness $\rho(\hat{x}, t_0)$. We found that for $100 = |D\text{test}|$ trajectories it holds that $\bar{\rho}(\hat{x}, t_0) > 0$ implies $(x(t_0), \tau_0) \models \phi$, 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 $\bar{\rho}(\hat{x}, t_0)$ 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 $c_x$ (deviation of the car from the center of the lane) and the orientation error $c_\theta$ (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 $(c_x, c_\theta) \in [-1, 1] \times [-0.4, 0.4]$. 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 $D\text{train}$ using the same settings as in the previous section. In Figures 5 and 6, we show the LSTMs performances in predicting $c_x$ and $c_\theta$ 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 $\rho^\phi(\hat{x}^{(i)}, \tau_0)$ 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 $r$ step ahead prediction on $D_{cal}$ for $r \in \{50, 100, 200\}$ 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 $\theta_e$ 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 $t_0 := 0$:

$$
\phi_1 := G_{[10,\infty)}(|x| \leq 2.25),
$$
$$
\phi_2 := G_{[10,\infty)}(|x| \geq 2.5) \Rightarrow F_{[0.5]}G_{[0.5]}(|x| \leq 1.25).
$$

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 2.5 is followed immediately within the next 5 seconds by a phase of 5 seconds where the cross-track error is below 2.25. 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 $D_{\text{cal}}$ 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. We hence obtain prediction regions of $\rho^\phi(x, t_0)$ 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 $c_i$ which may make it easier to train a good CBF). 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 $\rho^{\phi_1}(x, t_0)$ and the ground truth robustness $\rho^{\phi_1}(x, t_0)$. We found that for 99 of 100 trajectories under the IL controller and for 100/100 trajectories under the CBF controller it holds that $\rho^{\phi_1}(x, t_0) \geq \phi_1$, confirming Theorem 1. We also validated equation (3) and found that 99/100 trajectories under the IL controller and 95/100 trajectories under the CBF controller satisfy $\rho^{\phi_1}(x, t_0) - \rho^{\phi_1}(x, t_0) \leq C$.

For the second specification $\phi_2$, we again show the histograms of $R(i)$ for both controllers over the calibration data $D_{\text{cal}}$ in Figure 7 (right-mid: IL, right: CBF). We can now observe that 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 CNS-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 Bahaei, 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 Bahaei, 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:2108.00498 (2021).
Francesca Cairoli, Luca Bortolussi, and Nicola Paoletti. 2021. Neural predictive monitoring of vehicle models using Bayesian estimation and reachability analysis.

Andreas Bauer, Martin Leucker, and Christian Schallhart. 2011. Runtime verification based on distance learning. In Theoret. Comput. Science 387 (2015), 3–25.

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.

Ezio Bartocci, Luca Bortolussi, Laura Nenzi, and Guido Sanguinetti. 2015. System design of stochastic models using robustness of temporal properties. Theoret. Comput. Science 587 (2015), 1–25.

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.

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.

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.

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.

Dimitrios Bourouisos 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.

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

Francesca Cairoli, Nicola Paoletti, and Luca Bortolussi. 2022. Conformal Predictive Monitoring of STL Requirements for Stochastic Processes. arXiv preprint arXiv:2211.02975 (2022).

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

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

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

Yi Chou, Hansol Yoon, and Siriram Santhanakrishnan. 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.

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

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

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

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

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

Alessandro della Monica, Andrea Passarella, and Giorgio Delzanno. 2011. Incrementally Predictive Runtime Verification. In CILC. 92–106.

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

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

Luis Greiner-Büch and Matthias Ahlhoff. 2021. Predictive monitoring of traffic rules. In 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). IEEE, 915–922.

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

Hans Hansson and Bengt Jonsson. 1994. A logic for reasoning about time and reliability. Formal aspects of computing 6, 5 (1994), 512–555.

Peter Heindlflasch, 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.

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

Radoslav Ivanov, Taylor J Carpenter, James Weinert, 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.

Radoslav Ivanov, James Weinert, 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
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:

\[
(x, t) \models \text{True} \quad \text{iff} \quad \text{True},
\]
\[
(x, t) \models \mu \quad \text{iff} \quad h(x_t) \geq 0,
\]
\[
(x, t) \models \neg \phi \quad \text{iff} \quad \phi \notin (x, t),
\]
\[
(x, t) \models \phi \land \psi' \quad \text{iff} \quad (x, t) \models \phi' \quad \text{and} \quad (x, t) \models \psi' ,
\]
\[
(x, t) \models \phi \lor \psi' \quad \text{iff} \quad \phi \lor \psi \text{is satisfied or violated. We can again recursively calculate}
\]
\[
\rho \phi \left|_{\bar{x}, \tau} \right. (x, t) \models \phi \lor \psi \quad \text{iff} \quad \phi \lor \psi \text{ is satisfied or violated.}
\]

Lastly, we define the worst case robust semantics \( \rho_{\phi} (x, t_0) \) based on the structure of \( \phi \) using the following rules:

\[
\rho_{\phi} \left( x, t \right) = \infty, \quad \rho_{\phi} \left( x, t \right) = h(x_t), \quad \rho_{\phi} \left( x, t \right) = -\rho_{\phi} (x, t), \quad \rho_{\phi} \left( x, t \right) = \max \{ \rho_{\phi} (x, t), \rho_{\psi} (x, t) \},
\]
\[
\rho_{\phi} \left|_{\bar{x}, \tau} \right. \left( x, t \right) = \sup \{ \min \{ \rho_{\phi} (x, t'), \inf \rho_{\phi} (x, t') \} \}, \quad \rho_{\phi} \left|_{\bar{x}, \tau} \right. \left( x, t \right) = \sup \{ \min \{ \rho_{\phi} (x, t'), \inf \rho_{\phi} (x, t') \} \},
\]

The formula length \( L_\phi \) of a bounded STL formula \( \phi \) can be recursively calculated based on the structure of \( \phi \) using the following rules:

\[
L_{\phi} = L_{\phi} := 0, \quad L_{\phi} = L_{\phi}, \quad L_{\phi} = \max \{ L_{\phi}, L_{\phi} \},
\]
\[
L_{\phi} = \max \{ L_{\phi}, L_{\phi} \}, \quad L_{\phi} = \max \{ L_{\phi}, L_{\phi} \}.
\]

Lastly, we define the worst case robust semantics \( \rho_{\phi} (x, t_0) \), which are again recursively defined as follows:

\[
\rho_{\phi} \left( x, t \right) = \infty, \quad \rho_{\phi} \left( x, t \right) = h(x_t), \quad \rho_{\phi} \left( x, t \right) = -\rho_{\phi} (x, t), \quad \rho_{\phi} \left( x, t \right) = \max \{ \rho_{\phi} (x, t), \rho_{\psi} (x, t) \},
\]
\[
\rho_{\phi} \left|_{\bar{x}, \tau} \right. \left( x, t \right) = \sup \{ \min \{ \rho_{\phi} (x, t'), \inf \rho_{\phi} (x, t') \} \}, \quad \rho_{\phi} \left|_{\bar{x}, \tau} \right. \left( x, t \right) = \sup \{ \min \{ \rho_{\phi} (x, t'), \inf \rho_{\phi} (x, t') \} \}.
\]

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

\[
P(\rho_{\phi} (x, t_0) \geq \rho_{\phi} (x, t_0) - C) \geq 1 - \delta.
\]

If now \( \rho_{\phi} (x, t_0) > C \), 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 \( (x, t_0) \models \phi \) [25, 27].

C PROOF FOR THEOREM 2

Note first that \( X_t \in S_t \) for all times \( t \in \{ t + 1, \ldots, t + H \} \) with a probability of at least \( 1 - \delta \) by Lemma 1. For all predicates \( \mu \) in the STL formula \( \phi \) and for all times \( t \in \{ t, \ldots, t + H \} \), it hence holds that \( \rho_{\phi} (x, t) \geq \rho_{\phi} (x, t) \) with a probability of at least \( 1 - \delta \) by the definition of \( L_\phi \). Since the formula \( \phi \) does not contain negations\(^1\), it is straightforward to show (inductively on the structure of \( \phi \)) that \( \rho_{\phi} (x, t) \geq \rho_{\phi} (x, t) \) with a probability of at least \( 1 - \delta \). Consequently, if \( \rho_{\phi} (x, t) > 0 \), it holds that \( P((x, t_0) \models \phi) \geq 1 - \delta \) since \( \rho_{\phi} (x, t_0) > 0 \) implies \( (x, t_0) \models \phi \) [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.

\(^1\)Negations would in fact flip the inequality in an unfavorable direction, e.g., for \( \neg \mu \) it would hold that \( \rho_{\phi} (x, t) \leq \rho_{\phi} (x, t) \) 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].