Learning-Based Approaches to Predictive Monitoring with Conformal Statistical Guarantees

Francesca Cairoli[1], Luca Bortolussi[1], and Nicola Paoletti[2]

[1] University of Trieste, Italy
[2] King's College London, United Kingdom

Abstract. This tutorial focuses on efficient methods to predictive monitoring (PM), the problem of detecting at runtime future violations of a given requirement from the current state of a system. While performing model checking at runtime would offer a precise solution to the PM problem, it is generally computationally expensive. To address this scalability issue, several lightweight approaches based on machine learning have recently been proposed. These approaches work by learning an approximate yet efficient surrogate (deep learning) model of the expensive model checker. A key challenge remains to ensure reliable predictions, especially in safety-critical applications. We review our recent work on predictive monitoring, one of the first to propose learning-based approximations for CPS verification of temporal logic specifications and the first in this context to apply conformal prediction (CP) for rigorous uncertainty quantification. These CP-based uncertainty estimators offer statistical guarantees regarding the generalization error of the learning model, and they can be used to determine unreliable predictions that should be rejected. In this tutorial, we present a general and comprehensive framework summarizing our approach to the predictive monitoring of CPSs, examining in detail several variants determined by three main dimensions: system dynamics (deterministic, non-deterministic, stochastic), state observability, and semantics of requirements' satisfaction (Boolean or quantitative).

1 Introduction

Verification of temporal properties for a cyber-physical systems (CPS) is of paramount importance, especially with CPSs having become ubiquitous in safety-critical domains, from autonomous vehicles to medical devices [1]. We focus on *predictive monitoring* (PM), that is, the problem of predicting, at runtime, if a safety violation is imminent from the current CPS state. In this context, PM has the advantage, compared to traditional monitoring [9], of detecting potential safety violations before they occur, in this way enabling preemptive countermeasures to steer the system back to safety (e.g. switching to a failsafe mode as done in the Simplex architecture [37]). Thus, effective PM must balance between prediction accuracy, to avoid errors that can jeopardize safety, and computational efficiency, to support fast execution at runtime.

We focus on correctness specifications given in Signal Temporal Logic (STL) [42,29], a popular language for formal reasoning about CPS. An advantage of STL is that it admits two semantics, the usual Boolean semantics and a quantitative (robust) semantics, which quantifies the degree of satisfaction of a property. When using the latter, we speak of *quantitative PM (QPM)*.

Performing model checking of STL requirements at run-time would provide a precise solution to the PM problem (precise up to the accuracy of the system's model), but such

a solution is computationally expensive in general, and thus infeasible for real-world applications. For this reason, a number of approximate PM techniques based on machine learning have been recently proposed (see e.g. [14,17,19,40,43]).

In this paper, we review our work on learning-based methods for PM, developed under the name of *Neural Predictive Monitoring* [13]. The core idea is that when a satisfaction (SAT) oracle is at our disposal, we can approximate it using deep learning models trained using a set of oracle-labeled examples. The resulting learning-based model overcomes the scalability issues faced by the original oracle: a forward pass of a (reasonably sized) neural network is most often more efficient than computing satisfaction of an STL property, especially if the underlying system is non-deterministic or stochastic. However, such a solution is inherently approximate, and so it becomes essential – especially in safety-critical domains – to offer assurances regarding the generalization performance of our approximation. To this purpose, we rely on *conformal prediction (CP) [61]*, a technique that allows us to complement model predictions with uncertainty estimates that enjoy (finite-sample, i.e., non-asymptotic) statistical guarantees on the model's generalization error. CP requires only very mild assumptions on the data[3] and it is flexible enough to be applied on top of most predictors. Furthermore, computing CP-based uncertainty estimates is highly efficient, meaning that our approach can offer statistical guarantees on the PM predictions without affecting performance and runtime applicability.

In this tutorial, we present a general and comprehensive framework summarizing several variants of the neural predictive monitoring approach, variants determined by the following three dimensions:

1. *Dynamics*: The CPS dynamics can be either deterministic, non-deterministic, or stochastic, depending on whether the future behavior of the system is uniquely determined by its current state, is uncertain or exhibits randomness.
2. *State Observability*: The current CPS state can be fully observable, or we may only have access to partial and noisy measurements of the state, making it more challenging to obtain accurate predictions of the CPS evolution.
3. *Satisfaction*: The type of property satisfaction can be either Boolean or quantitative. In the former case, the PM outcome is a "yes" or "no" answer (the CPS either satisfies the property or does not). In the latter case, the outcome is a quantitative degree of satisfaction, which quantifies the robustness of STL (Boolean) satisfaction to perturbations (in space or time) of the CPS trajectories.

By considering the above dimensions, we can design accurate and reliable PM solutions in a variety of scenarios accounting for a vast majority of CPS models.

Overview of the paper. This paper is structured as follows. We start by presenting the background theory in Sect. 2. Sect. 3 states rigorous formalizations of the predictive monitoring problems. Sect. 4 provides background on methods used to estimate predictive uncertainty and obtain statistical guarantees. Sect. 5 defines the different declinations of learning-based PM approaches. Related works are discussed in Sect. 6. Conclusions are drawn in Sect. 7.

---

[3] The only assumption is exchangeability, a weaker version of the independent and identically distributed assumption. A collection of N values is exchangeable if the $N!$ different orderings are equally likely, i.e. have the same joint probability.

2 Background

A cyber-physical system (CPS) is a system combining physical and digital components. Hybrid systems (HS), whose dynamics exhibit both continuous and discrete dynamics, can well capture the mixed continuous and discrete behaviour of a CPS. An HS has both flows, described by differential equations, and jumps, described by a state machine or an automaton. The continuous behaviour depends on the discrete state and discrete jumps are similarly determined by the continuous state. The state of an HS is defined by the values of the continuous variables and by a discrete mode. The continuous flow is permitted as long as a so-called invariant holds, while discrete transitions can occur as soon as given jump conditions are satisfied.

Remark 1. The approaches proposed in this paper are applicable to any black-box system for which a satisfaction (SAT) oracle is available. That said, HS represent a very useful and expressive class of models for which several SAT oracles (model checkers) have been developed. Therefore, we will focus on this class of models for the rest of the paper.

A hybrid automaton (HA) is a formal model that mathematically describes the evolution in time of an HS.

Definition 1 (Hybrid automaton). *A* hybrid automaton *(HA) is a tuple* $\mathcal{M} = (Loc,$ $Var, Init, Flow, Trans, Inv)$, *where Loc is a finite set of discrete* locations *(or* modes*);* $Var = \{v_1, ..., v_n\}$ *is a set of continuous* variables, *evaluated over a* continuous domain $Var \subseteq \mathbb{R}^n$; $Init \subseteq S(\mathcal{M})$ *is the set of* initial states, *where* $S(\mathcal{M}) = Loc \times Var$ *is the* state space *of* $\mathcal{M}$; $Flow : Loc \rightarrow (Var \rightarrow Var)$ *is the* flow *function, defining the continuous dynamics at each location; Trans is the* transition relation, *consisting of tuples of the form* (l, g, r, l'), *where* $l, l' \in Loc$ *are* source and target locations, *respectively,* $g \subseteq Var$ *is the* guard, *and* $r : Var \rightarrow Var$ *is the* reset; $Inv : Loc \rightarrow 2^{Var}$ *is the* invariant *at each location.*

In general, HA dynamics can be either deterministic, non-deterministic, or stochastic. *Deterministic* HAs are a special case where the transition relation is a function of the source location. On the other hand, when the continuous flow or the discrete transitions happen according to a certain probability distribution, we have a *stochastic* HA. In this case, the dynamics is represented as the combination of continuous stochastic flows probability $Flow : (Loc \times Var) \rightarrow (Var \rightarrow [0,1])$ and discrete jump probability $Trans : (Loc \times 2^{Var}) \rightarrow ((Loc \times (Var \times Var)) \rightarrow [0,1])$. In particular, $Flow(v' \mid l, v)$ denotes the probability of a change rate of v' when in state (l, v) and $Trans(r, l' \mid l, g)$ is the probability of applying reset r and jumping into l' through a transition starting from l with guard g. We often prefer to avoid transitions with both non-determinism and stochasticity and so, for each stochastic HA location, we define its invariant and guards so that they form a partition of Var.

We define a *signal* as a function $\vec{s} : \mathbb{T} \rightarrow \mathbb{V}$, where $\mathbb{T} \subset \mathbb{R}^+$ is the time domain, whereas $\mathbb{V}$ determines the nature of the signal. If $\mathbb{V} = \mathbb{B} := \{true, false\}$, we have a *Boolean signal*. If $\mathbb{V} = \mathbb{R}$, we have a *real-valued signal*. We consider signals that are solutions of a given HA. Let $\tau = \{[t_i, t_{i+1}] \mid t_i \leq t_{i+1}, i = 1, 2, ...\}$ be a hybrid time trajectory. If τ is infinite the last interval may be open on the right. Let $\mathbb{T}$ denote the set of hybrid time

trajectories. Then, for a given $\tau \in \mathbb{T}$, a *hybrid signal* $\vec{s} : \tau \to S(\mathcal{M})$ defined on τ with values in a generic hybrid space $S(\mathcal{M})$ is a sequence of functions

$$\vec{s} = \{\vec{s}_i : [t_i, t_{i+1}] \to S(\mathcal{M}) \mid [t_i, t_{i+1}] \in \tau\}.$$

In practice, it can be seen as a pair of hybrid signals $\vec{v} : \tau \to Var$ and $\vec{l} : \tau \to Loc$ with $\tau \in \mathbb{T}$, such that $(\vec{l}_1(t_1), \vec{v}_1(t_1)) \in Init$, and for any $[t_i, t_{i+1}] \in \tau$, $\vec{l}_i$ is constant and there exist g and r such that $(l_i, g, r, l_{i+1}) \in Trans$, $\vec{v}_i(t_{i+1}^-) \in g$, and $\vec{v}_{i+1}(t_{i+1}) = r(\vec{v}_i(t_{i+1}^-))$. Moreover, for every $t \in [t_i, t_{i+1}]$, it must hold that $\vec{v}_i(t) \in Inv(l_i)$ and

$$\vec{v}_i(t) = \vec{v}_i(t_i) + \int_{t_i}^{t} Flow\left(\vec{l}_i(t'), \vec{v}_i(t')\right) dt'.$$

When the HA is stochastic, hybrid signals must have a non-zero probability, that is, for every piece i of the signal, there exists g, r such that $Trans(r, l_{i+1} \mid l_i, g) > 0$, $\vec{v}_i(t_{i+1}^-) \in g$, and $\vec{v}_{i+1}(t_{i+1}) = r(\vec{v}_i(t_{i+1}^-))$. Moreover, for every $t \in [t_i, t_{i+1})$, it must hold that $Flow(\dot{\vec{v}}_i(t) \mid l_i, \vec{v}_i(t)) > 0$, where $\dot{\vec{v}}_i(t) = \lim_{dt \to 0}(\vec{v}_i(t+dt) - \vec{v}_i(t))/\vec{v}_i(t)$.

Remark 2. We note that HA induces Markovian dynamics, that is, the evolution of the HA depends only on the current state. We do not see this as a restriction as most systems of interest are Markovian or can be made so by augmenting the state space.

2.1 Signal Temporal Logic (STL)

Signal temporal logic (STL) [42] was originally developed in order to specify and monitor the behaviour of physical systems, including temporal constraints between events. STL allows the specification of properties of dense-time, real-valued signals, and the automatic generation of monitors for testing these properties on individual simulation traces. The rationale of STL is to transform hybrid signals into Boolean ones, using predicates built on the following *STL syntax*:

$$\varphi := true \mid \mu_\ell \mid \mu_g \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathcal{U}_{[a,b]}\varphi, \tag{1}$$

where $[a,b] \subseteq \mathbb{T}$ is a bounded temporal interval. For a hybrid signal $\vec{s}[t]$, μ_g denotes atomic predicates over continuous variables, with $g : Var \to \mathbb{R}$, whereas μ_ℓ denotes atomic predicates over discrete variables, with $\ell \in Loc$. From this essential syntax, it is easy to define other operators, used to abbreviate the syntax in a STL formula: $false := \neg true$, $\varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi)$, $\Diamond_{[a,b]}\varphi := true \, \mathcal{U}_{[a,b]}\varphi$ and $\Box_{[a,b]}\varphi := \neg\Diamond_{[a,b]}\neg\varphi$.

Boolean semantics. The satisfaction of a formula φ by a signal $\vec{s}$ at time t is defined as:

- $(\vec{s},t) \models \mu_g \iff g(\vec{v}[t]) > 0$;
- $(\vec{s},t) \models \mu_\ell \iff \vec{l}[t] = \ell$;
- $(\vec{s},t) \models \varphi_1 \wedge \varphi_2 \iff (\vec{s},t) \models \varphi_1 \wedge (\vec{s},t) \models \varphi_2$;
- $(\vec{s},t) \models \neg\varphi \iff \neg((\vec{s},t) \models \varphi))$;
- $(\vec{s},t) \models \varphi_1 \mathcal{U}_{[a,b]}\varphi_2 \iff \exists t' \in [t+a, t+b]$ s.t.
  $(\vec{s},t') \models \varphi_2 \wedge \forall t'' \in [t,t'), (\vec{s},t'') \models \varphi_1$.
- Eventually: $(\vec{s},t) \models \Diamond_{[a,b]}\varphi \iff \exists t' \in [t+a, t+b]$ s.t. $(\vec{s},t') \models \varphi$;
- Always: $(\vec{s},t) \models \Box_{[a,b]}\varphi \iff \forall t' \in [t+a, t+b] \quad (\vec{s},t') \models \varphi$.

Given formula φ and a signal $\vec{s}$ over a bounded time interval, we can define the Boolean satisfaction signal as $\chi^\varphi(\vec{s},t) = 1$ if $(\vec{s},t) \models \varphi$ and $\chi^\varphi(\vec{s},t) = 0$ otherwise. Monitoring the satisfaction of a formula is done recursively, by computing $\chi^{\varphi_i}(\vec{s}, \cdot)$ for each sub-formula φ_i of φ. The recursion is performed by leveraging the tree structure of the STL formula, where each node represents a sub-formula, in an incremental fashion, so that the leaves are the atomic propositions and the root represents the whole formula. Thus the procedure goes bottom-up from atomic predicated to the top formula.
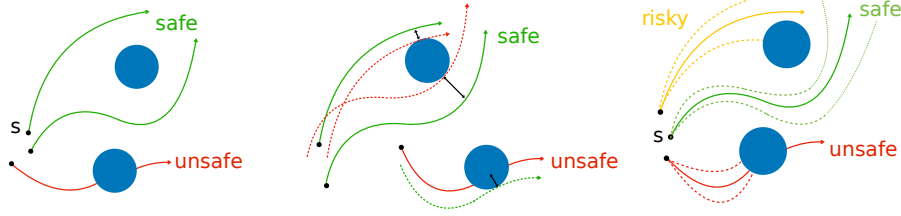
Fig. 1. Example of predictive monitoring of a safety property for a deterministic system (left) and a stochastic system (right). Blue circles denote the obstacles to avoid. (middle) shows the space robustness over the deterministic system.

Quantitative semantics. The main kind of quantitative STL semantics is *space robustness*, which quantifies how much a signal can be perturbed with additive noise before changing the truth value of a given property φ [29]. It is defined as a function R_φ such that:

- $R_{\mu_g}(\vec{s},t) = g(\vec{v}[t])$;
- $R_{\neg\varphi}(\vec{s},t) = -R_\varphi(\vec{s},t)$;
- $R_{\varphi_1 \wedge \varphi_2}(\vec{s},t) = \min(R_{\varphi_1}(\vec{s},t), R_{\varphi_2}(\vec{s},t))$;
- $R_{\varphi_1 \mathcal{U}_{[a,b]} \varphi_2}(\vec{s},t) = \sup_{t' \in [t+a,t+b]} \left(\min\left(R_{\varphi_2}(\vec{s},t'), \inf_{t'' \in [t,t']} R_{\varphi_1}(\vec{s},t'') \right) \right)$.

The sign of R_φ indicates the satisfaction status: $R_\varphi(\vec{s},t) > 0 \Rightarrow (\vec{s},t) \models \varphi$ and $R_\varphi(\vec{s},t) < 0 \Rightarrow (\vec{s},t) \not\models \varphi$. The definition of R_{μ_ℓ}, i.e., the robustness of discrete atoms, is arbitrary as long as it returns a non-negative value when μ_ℓ is true and non-positive when μ_ℓ is false (a common choice is returning $+\infty$ and $-\infty$, respectively). As for the Boolean semantics, it is possible to automatically generate monitors for the quantitative semantics as well. The algorithm follows a similar bottom-up approach over the syntax tree of the formula.

Similarly, *time robustness* capture the effect on the satisfaction of shifting events in time. The left and right time robustness of an STL formula φ with respect to a trace $\vec{s}$ at time t are defined inductively by letting:

$$Q_\varphi^-(\vec{s},t) = \chi^\varphi(\vec{s},t) \cdot \max\{d \geq 0 \ s.t. \ \forall t' \in [t-dt], \ \chi^\varphi(\vec{s},t') = \chi^\varphi(\vec{s},t)\}$$

$$Q_\varphi^+(\vec{s},t) = \chi^\varphi(\vec{s},t) \cdot \max\{d \geq 0 \ s.t. \ \forall t' \in [t,t+d], \ \chi^\varphi(\vec{s},t') = \chi^\varphi(\vec{s},t)\}.$$

While space and robustness are most common, our PM approach can support any other kind of STL quantitative semantics, e.g., based on a combined space-time robustness [29] or resiliency [22]. Hereafter, we represent a generic STL monitor, encompassing either Boolean or quantitative (spatial or temporal) satisfaction, as $C_\varphi \in \{\chi_\varphi, R_\varphi, Q_\varphi\}$.

Running example. Let's consider, as a running example, a point moving at a constant velocity on a two-dimensional plane (see Fig. 1). Given the system's current state s, a controller regulates the yawn angle to avoid obstacles (D_1 and D_2 in our example). The avoid property can be easily expressed as an STL formula: $\varphi := G(\ (d(s,o_1) > r_1) \wedge (d(s,o_2) > r_2)\ )$, where o_i, r_i denote respectively the centre and the radius of obstacle $i \in \{1,2\}$. Fig. 1 (left) shows the deterministic evolution for three randomly chosen initial states. Fig. 1 (middle) shows, for the same deterministic

scenario, an intuition of the concept of spatial STL robustness, i.e. how much we can perturb a trajectory with additive noise before changing its truth value. Fig. 1 (right) shows the evolution of a stochastic dynamics for three randomly initial states. The dashed lines denote the upper and lower quantiles of the distribution over the trajectory space.

3 Predictive Monitoring

Predictive monitoring of an HA is concerned with establishing whether given an initial state s and a desired property φ – e.g. always avoid a set of unsafe states D – the HA admits a trajectory starting from s that violates φ. We express such properties by means of time-bounded STL formulas, e.g. $\varphi_D := G_{[0,H_f]} \, (s \notin D)$. STL monitors automatically check whether an HA signal satisfies an STL property φ over a bounded temporal horizon (Boolean semantics) and possibly how robust is the satisfaction (quantitative semantics).

SAT Oracles. Given an HA $\mathcal{M}$ with state space $S(\mathcal{M})$, and an STL requirement φ over a time bound H_f, SAT oracles decide whether a state $s \in S(\mathcal{M})$ satisfies φ. This means that, when the system is *deterministic*, the SAT oracle decides whether the unique trajectory $\vec{s}$ starting from $s \in S(\mathcal{M})$ satisfies φ, i.e. decide whether $(\vec{s},0) \models \varphi$. This information can be retrieved from STL monitors by checking whether $\chi^\varphi(\vec{s},0) = 1$ or equivalently whether $R_\varphi(\vec{s},0) > 0$. On the other hand, when the system is *nondeterministic*, a state s satisfies φ if all trajectories starting from s satisfy φ. Similarly, a quantitative SAT oracle returns the minimal STL robustness value of all trajectories starting from s. For non-stochastic systems, the SAT oracle can thus be represented as a map $\mathsf{Sat} : S(\mathcal{M}) \to \mathsf{B}$, where the output space B is either $\mathbb{B}$ in the Boolean scenario or $\mathbb{R}$ in the quantitative scenario. On the other hand, oracles for stochastic systems require a different treatment and we define them later in this section.

SAT Tools. Several tools have been developed for the automated verification of CPS properties and can thus be used as SAT oracles. The choice of the best tool depends on the problem at hand. STL monitors such as Breach [28] and RTAMT [44] allow to automatically check whether realizations of the system satisfy an STL property. When the system is nondeterministic we need tools that perform reachability analysis or falsification. Due to the well-known undecidability of HA model checking problem [35,16], none of existing tools are both sound and complete. Falsification tools like Breach [28], S-Taliro [2], C2E2 [30], and HyLAA [6] search for counter-example trajectories, i.e., for violations to the property of interest. A failure in finding a counter-example does not imply that the property is satisfied (i.e., the outcome is unknown). Conversely, HA reachability tools like PHAVer [31], SpaceEx [32], Flow*, HyPro/HyDra [56], Ariadne [10] and JuliaReach [11] rely on computing an over-approximation of the reachable set, meaning that the outcome is unknown when the computed reachable set intersects the target set. In order to be conservative, we treat unknown verdicts in a pessimistic way.

On the other hand, *stochastic* systems require the use of probabilistic model checking techniques implemented in tools like PRISM [39] or STORM [34]. Such tools provide precise numerical/symbolic techniques to determine the satisfaction probability of a formula, but only for a restricted class of systems and with significant scalability issues. Statistical model checking (SMC) techniques overcomes these limitations by solving the

problem as one of hypothesis testing given a sample of system trajectories (at the cost of admitting some a priori statistical errors).

The above list of tools is far from being exhaustive, and we refer the interested reader to the ARCH-COMP competitions [4], where state-of-the-art verification tools are compared on a set of well-known benchmarks.

In the following, we formulate the predictive monitoring problem for non-stochastic systems (Problem 1), for partially observable systems (Problem 2), and for stochastic systems (Problem 3). We conclude by characterizing the probabilistic guarantees sought for our learning-based monitors (Problem 4).

We aim at deriving a predictive monitor for HA time-bounded satisfaction, i.e., a function that can predict whether or not the property φ is satisfied by the future evolutions of the system (bounded by time H_f) starting from the system's current state. In solving this problem, we assume a distribution $\mathcal{S}$ of HA states and seek the monitor that predicts HA reachability with minimal error probability w.r.t. $\mathcal{S}$. The choice of $\mathcal{S}$ depends on the application at hand and can include a uniform distribution on a bounded state space or a distribution reflecting the density of visited states in some HA executions [48].

Problem 1 (Predictive monitoring for HA) *Given an HA $\mathcal{M}$ with state space $S(\mathcal{M})$, a distribution $\mathcal{S}$ over $S(\mathcal{M})$, a time bound H_f and STL property φ, inducing the satisfaction function* Sat, *find a function $h^* : S(\mathcal{M}) \to \mathsf{B}$ that minimizes the probability*
$$Pr_{s \sim \mathcal{S}}(h^*(s) \neq \mathsf{Sat}(s)).$$
A state $s \in S(\mathcal{M})$ is called positive *w.r.t. a predictor $h : S(\mathcal{M}) \to \mathsf{B}$ if $h(s) > 0$. Otherwise, s is called* negative.

Any practical solution to the above PM problem must also assume a space of functions within which to restrict the search for the optimal predictive monitor h^*, for instance, one can consider functions described by deep neural networks (DNNs). Finding h^*, i.e., finding a function approximation with minimal error probability, is indeed a classical *supervised learning* problem. In particular, in the Boolean scenario, h^* is a classifier, i.e., a function mapping HA state inputs s into one of two classes: 1 (x is positive, property φ is satisfied) and 0 (s is negative, property φ is violated). On the other hand, in the quantitative scenario, h^* is a regressor aiming at reconstructing the robustness of satisfaction for a state s.

Dataset generation. In supervised learning, one minimizes a measure of the empirical prediction error w.r.t. a *training set*. In our case, the training set Z' is obtained from a finite sample S' of $\mathcal{S}$ by labelling the training inputs $s \in S'$ using some SAT oracle, that is computing the true value for $\mathsf{Sat}(s)$. Hence, given a sample S' of $\mathcal{S}$, the training set is defined by $Z' = \{(s, \mathsf{Sat}(s)) \mid s \in S'\}$ (see Fig. 2).

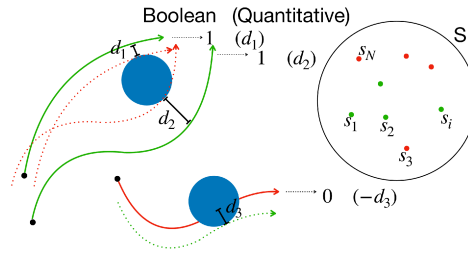**Fig. 2.** Generation of the dataset to learn a PM (Boolean or quantitative) for deterministic HS.

---

[4] https://cps-vo.org/group/ARCH/FriendlyCompetition

Partial Observability Problem 1
relies on the full observability (FO) assumption, i.e. the assumption of possessing full
knowledge about the system's state. However, in most practical applications, state
information is partial and noisy. Consider a discrete-time deterministic HS[5] modeled as
a HA $\mathcal{M}$. The discrete-time deterministic dynamics of the system can be expressed by
$v_{i+1} = Flow(l_i)(v_i)$, where $s_i = (l_i, v_i) = (l(t_i), v(t_i))$ and $t_i = t_0 + i \cdot \Delta t$. The measurement
process can be modeled as

$$y_i = \pi(s_i) + w_i, \tag{2}$$

which produces partial and noisy observations $y_i \in Y$ by means of an *observation
function* $\pi : S(\mathcal{M}) \to Y$ and additive noise $w_i \sim \mathcal{W}$. Under *partial observability* (PO), we
only have access to a sequence of past observations $\mathbf{y}_t = (y_{t-H_p}, ..., y_t)$ of the *unknown*
state sequence $\mathbf{s}_t = (s_{t-H_p}, ..., s_t)$ (as per (2)). Let $\mathcal{Y}$ denote the distribution over Y^{H_p},
the space of sequences of observations $\mathbf{y}_t$ induced by the sequence of states $\mathbf{s}_t \sim \mathcal{S}^{H_p}$
and a sequence of i.i.d. noise vectors $\mathbf{w}_t = (w_{t-H_p}, ..., w_t) \sim \mathcal{W}^{H_p}$.

Problem 2 (PM for HS under noise and partial observability) *Given the
HA and reachability specification of Problem 1, find a function* $h_{po}^* : Y^{H_p} \to \mathsf{B}$ *that
minimizes*

$$Pr_{s_t \sim \mathcal{S}, \mathbf{y}_t \sim \mathcal{Y}} \left(h_{po}^* (\mathbf{y}_t) \neq \mathsf{Sat}(s_t) \right).$$

In other words, h_{po}^* should predict the satisfaction values given in input only a sequence
of past observations, instead of the true HA state. In particular, we require a sequence of
observations (as opposed to one observation only) for the sake of identifiability. Indeed,
for general non-linear systems, a single observation does not contain enough information
to infer the HS state[6]. Problem 2 considers only deterministic systems. Dealing with
partial observability and noise in nondeterministic systems remains an open problem as
state identifiability is a non-trivial issue.

There are two natural learning-based approaches to tackle Problem 2 (Fig. 3):

1. an **end-to-end** solution that learns a direct mapping from the sequence of past
   measurements $\mathbf{y}_t$ to the satisfaction value in B.
2. a **two-step** solution that combines steps (a) and (b) below:
   (a) learns a *state estimator* able to reconstruct the history of full states $\mathbf{s}_t = (s_{t-H_p}, ..., s_t)$ from the sequence of measurements $\mathbf{y}_t = (y_{t-H_p}, ..., y_t)$;
   (b) learns a *state classifier/regressor* mapping the sequence of states $\mathbf{s}_t$ to the
   satisfaction value in B;

Dataset generation. Given that we consider deterministic dynamics, we can use
simulation, rather than model checking, to label the states as safe (positive), if $\mathsf{Sat}(s) > 0$,
or unsafe (negative) otherwise. Because of the deterministic and Markovian (see
Remark 2) nature of the system, one could retrieve the future satisfaction of a property
at time t from the state of the system at time t alone. However, one can decide
to exploit more information and make a prediction based on the previous H_p states.
Formally, the generated dataset under FO can be expressed as $Z' = \{(\mathbf{s}_t^i, \mathsf{Sat}(s_t^i))\}_{i=1}^N$,

---

[5] In case of partial observability we restrict our analysis to deterministic systems.

[6] Feasibility of state reconstruction is affected by the time lag and the sequence length. Our
focus is to derive the best predictions for fixed lag and sequence length, not to fine-tune
these to improve identifiability.

where $\mathbf{s}_t^i = (s_{t-H_p}^i, s_{t-H_p+1}^i, ..., s_t^i)$. Under PO, we use the (known) observation function $\pi : S(\mathcal{M}) \to Y$ to build a dataset Z'' made of tuples $(\mathbf{y}_t, \mathbf{s}_t, l_t)$, where $\mathbf{y}_t$ is a sequence of noisy observations for $\mathbf{s}_t$, i.e., such that $\forall j \in \{t-H_p, ..., t\}$ $y_j = \pi(s_j) + w_j$ and $w_j \sim \mathcal{W}$.

The distribution of $\mathbf{s}_t$ and $\mathbf{y}_t$ is determined by the distribution $\mathcal{S}$ of the initial state of the sequences, s_{t-H_p}. We consider two different distributions: *independent*, where the initial states s_{t-H_p} are sampled independently, thus resulting in independent state/observation sequences; and *sequential*, where states come from tem-

Fig. 3. Diagram of the learning steps under noise and partial observability.

porally correlated trajectories in a sliding-window fashion. The latter is more suitable for real-world runtime applications, where observations are received in a sequential manner. On the other hand, temporal dependency violates the exchangeability property, which affects the theoretical validity guarantees of CP, as we will soon discuss.
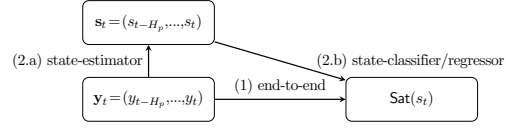
Stochastic dynamics If the system evolves stochastically, we have a distribution over the trajectory space rather than a single trajectory that either satisfies of violates the property. Some realizations will satisfy the property, some others will not. Therefore, reasoning about satisfaction gets more complicated. Let $\mathbb{T} = \{0, 1, ...\}$ denote a discrete set of time instants and let $\mathcal{M}$ be a discrete-time stochastic HA over state space $S(\mathcal{M})$ and $\mathbb{T}$. Given that the system is in state $s \sim \mathcal{S}$ at time $t \in \mathbb{T}$, the stochastic evolution (bounded by horizon H_f) of the system starting at s can be described by the conditional distribution $p(\vec{s} \,|\, \vec{s}(t) = s)$, where $\vec{s} = (\vec{s}(t), ..., \vec{s}(t+H_f)) \in S^H$ is the random trajectory of length H_f starting at time t. We thus introduce a satisfaction function SSat that inherits the stochasticity of the system's dynamics. For an STL property φ, we define SSat as a function mapping a state $s \in S(\mathcal{M})$ into a random variable $\mathsf{SSat}(s)$ denoting the distribution of satisfaction values over $\mathbb{B}$. In other words, the satisfaction function transforms the distribution over trajectories into the distribution over satisfaction values.

The predictive monitoring problem under stochastic dynamics can be framed as estimating one or more functionals of $\mathsf{SSat}(s)$ (e.g., mean, variance, quantiles). A formal statement of the problem is given below.

Problem 3 (PM for Stochastic HS) *Given a discrete-time stochastic HA $\mathcal{M}$ over a state space $S(\mathcal{M})$, temporal horizon H_f, and an STL formula φ, we aim at approximating a functional q of the distributions induced by SSat. We thus aim at deriving a monitoring function h_q^* that maps any state $s \sim \mathcal{S}$ into the functional $\mathsf{q}[\mathsf{SSat}(s)]$ such that*

$$Pr_{s \sim \mathcal{S}} \Big(h_\mathsf{q}^*(s) \neq \mathsf{q}\big[\mathsf{SSat}(s)\big] \Big). \tag{3}$$

We will focus on the case where q is a quantile function, making Problem 3 equivalent to a conditional quantile regression (QR) problem. This boils down to learning for a generic state s a quantile of the random variable $\mathsf{SSat}(s)$.

Dataset generation. We perform Monte-Carlo simulations of the process in order to obtain empirical approximations of SSat. In particular, we randomly sample N states

$s_1,...,s_N \sim \mathcal{S}$. Then, for each state s_i, we simulate M trajectories of length H_f, $\vec{s}_i^1,...,\vec{s}_i^M$ where $\vec{s}_i^j$ is a realization of $p(\vec{s} \mid \vec{s}(t)=s_i)$, and compute the satisfaction value $C_\varphi(\vec{s}_i^j)$ of each of these trajectories ($C_\varphi \in \{\chi_\varphi, R_\varphi, Q_\varphi\}$). Note how $\{C_\varphi(\vec{s}_i^j)\}_{j=1}^M$ is an empirical approximation of $\mathsf{SSat}(s_i)$. The dataset is thus defined as

$$Z' = \left\{ \Big(s_i, \big(C_\varphi(\vec{s}_i^1),...,C_\varphi(\vec{s}_i^M)\big)\Big), i=1,...,N \right\}. \tag{4}$$

Fig. 4 shows an overview of the steps needed to generate the dataset. The generation of the test set Z'_{test} is very similar to that of Z'. The main difference is in that the number of trajectories that we simulate from each state s is much larger than M. This allows us to obtain a highly accurate empirical approximation of the distribution induced by SSat, which we use as the ground-truth baseline in our experimental evaluation[7]. Moreover, since func-

Fig. 4. Generation of the dataset to learn a PM for stochastic HS.

tionals of $\mathsf{SSat}(s)$ can not in general be computed exactly, for a choice of $\epsilon \in (0,1)$, we derive the empirical quantile $\hat{q}_\epsilon^{s_i}$ from samples $C_\varphi(\vec{s}_i^1),...,C_\varphi(\vec{s}_i^M)$ and use the generated training set Z' to train the QR h_{q} that learns how to map states s into $\hat{q}_\epsilon^s$.
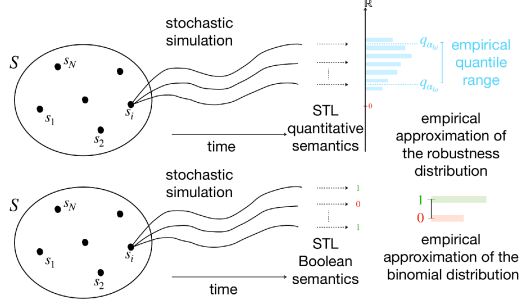
The predictors, either h, h_{po} or h_{q}, are approximate solutions and, as such, they can commit safety-critical prediction errors. The general goal of Problems 1, 2 and 3 is to minimize the risk of making mistakes in predicting the satisfaction of a property. We are also interested in establishing probabilistic guarantees on the expected error rate of an *unseen (test) state*, in the form of prediction regions guaranteed to include the true satisfaction value with arbitrary probability. We now introduce some notation to capture all three previously stated scenarios. Let f be the predictor (either h of Problem 1, h_{po} of Problem 2 or h_{q} of Problem 3) and let $x \in X$ be the input of predictor f (either a state s or a sequence of past measurements $\mathbf{y}$). The distribution over the generic input space X is denoted by $\mathcal{X}$.

Problem 4 (Probabilistic guarantees) *Given a system and property φ as in Problems 1, 2 and 3, find a function $\Gamma^\varepsilon : X \to 2^{\mathsf{B}}$, mapping every input x into a prediction region for the corresponding satisfaction value, i.e., a region that satisfies, for any error probability level $\varepsilon \in (0,1)$, the validity property below*

$$Pr_{x \sim \mathcal{X}}\Big(\mathsf{SAT}(x) \in \Gamma^\varepsilon(x)\Big) \geq 1-\varepsilon,$$

where $\mathsf{SAT}(\cdot)$ corresponds to $\mathsf{Sat}(\cdot)$ in Problems 1 and 2, and to $\mathsf{q}[\mathsf{SSat}(\cdot)]$ in Problem 3.

Among the maps that satisfy validity, we seek the most *efficient* one, meaning the one with the smallest, i.e. less conservative, prediction regions.

---

[7] In the limit of infinite sample size, the empirical approximation approaches the true distribution.

4   Uncertainty Estimation and Statistical Guarantees

The learning-based solutions of Problems 1, 2 and 3 are approximate and, even when extremely high accuracies are reached, offer no guarantees over the reliability of the learned predictor, and thus are not applicable in safety-critical scenarios. In this section, we present techniques for uncertainty estimation, techniques that overcome the above limitation by providing point-wise information about the reliability of the predictions. In particular, we examine two uncertainty quantification techniques, based on conformal prediction (CP) and Bayesian inference, respectively. We focus more on CP as, unlike Bayesian inference, can provide the desired statistical guarantees stated in Problem 4.

To simplify the presentation, we illustrate the techniques by considering a generic supervised learning model, as follows. Let X be the input space, T be the target (output) space, and define $Z = X \times T$. Let $\mathcal{Z}$ be the data-generating distribution, i.e., the distribution of the points $(x,t) \in Z$. We assume that the target t of a point $(x,t) \in Z$ is the result of the application of a function $f^* : X \to T$, typically unknown or very expensive to evaluate. Using a finite set of observations, the goal of a supervised learning algorithm is to find a function $f : X \to T$ that accurately approximates f^* over the entire input space. For a generic input $x \in X$, we denote with t the true target value of x and with $\hat{t}$ the prediction by f, i.e. $\hat{t} = f(x)$. Test inputs, whose unknown true target values we aim to predict, are denoted by x_*.

4.1   Conformal Inference

Conformal Prediction (CP) associates measures of reliability to any traditional supervised learning problem. It is a very general approach that can be applied across all existing deterministic classifiers and regressors [8,61]. CP produces *prediction regions with guaranteed validity.*

Definition 2 (Prediction region). *For significance level $\varepsilon \in (0,1)$ and test input x_*, the ε-prediction region for x_*, $\Gamma_*^\varepsilon \subseteq T$, is a set of target values s.t.*

$$\Pr_{(x_*,t_*) \sim \mathcal{Z}}(t_* \in \Gamma_*^\varepsilon) = 1 - \varepsilon. \tag{5}$$

The idea of CP is to construct the prediction region by "inverting" a suitable hypothesis test: given a test point x_* and a tentative target value t', we *exclude* t' from the prediction region only if it is unlikely that t' is the true value for x_*. The test statistic is given by a so-called *nonconformity function (NCF)* $\delta : Z \to \mathbb{R}$, which, given a predictor f and a point $z = (x,t)$, measures the deviation between the true value t and the corresponding prediction $f(x)$. In this sense, δ can be viewed as a generalized residual function. In other words, CP builds the prediction region Γ_*^ε for a test point x_* by excluding all targets t' whose NCF values are unlikely to follow the NCF distribution of the true targets:

$$\Gamma_*^\varepsilon = \left\{ t' \in T \mid Pr_{(x,t) \sim \mathcal{Z}}(\delta(x_*,t') \geq \delta(x,t)) > \varepsilon \right\}. \tag{6}$$

The probability term in Eq. 6 is often called the p-value. From a practical viewpoint, the NCF distribution $Pr_{(x,t) \sim \mathcal{Z}}(\delta(x,t))$ cannot be derived in an analytical form, and thus we use an empirical approximation derived using a sample Z_c of $\mathcal{Z}$. This approach is called *inductive* (or split) CP [45] and Z_c is referred to as *calibration set.*

Validity and Efficiency. CP performance is measured via two quantities: 1) *validity* (or *coverage*), i.e. the empirical error rate observed on a test sample, which should be as close as possible to the significance level ε, and 2) *efficiency*, i.e. the size of the prediction regions, which should be small. CP-based prediction regions are automatically valid, whereas the efficiency depends on the size of the calibration set (leading to high uncertainty when data is scarce), the quality of the underlying model and the chosen nonconformity function.

Remark 3 (Assumptions and guarantees of inductive CP). Importantly, CP prediction regions have *finite-sample validity* [8], i.e., they satisfy (5) for any sample of $\mathcal{Z}$ (of reasonable size), and not just asymptotically. On the other hand, CP's theoretical guarantees hold under the *exchangeability* assumption (a "relaxed" version of iid) by which the joint probability of any sample of $\mathcal{Z}$ is invariant to permutations of the sampled points. Independent observations are exchangeable but sequential ones are not (due to the temporal dependency). In such scenarios, some adaptations to conformal inference (see [58,66]) are needed to recover and preserve validity guarantees.

CP for classification In classification, the target space is a discrete set of possible labels (or classes) $T = \{t^1,...,t^c\}$. We represent the classification model as a function $f_d : X \to [0,1]^c$ mapping inputs into a vector of class likelihoods, such that the predicted class is the one with the highest likelihood[8].

The inductive CP algorithm for classification is divided into an offline phase, executed only once, and an online phase, executed for every test point x_*. In the offline phase (steps 1–3 below), we train the classifier f and construct the calibration distribution, i.e., the empirical approximation of the NCF distribution. In the online phase (steps 4–5), we derive the prediction region for x_* using the computed classifier and distribution.

1. Draw sample Z' of $\mathcal{Z}$. Split Z' into training set Z_t and calibration set Z_c.
2. Train classifier f using Z_t. Use f_d to define an NCF δ.
3. Construct the calibration distribution by computing, for each $z_i \in Z_c$, the NCF score $\alpha_i = \delta(z_i)$.
4. For each label $t^j \in T$, compute $\alpha_*^j = \delta(x_*, t^j)$, i.e., the NCF score for x_* and t^j, and the associated p-value p_*^j:
$$p_*^j = \frac{|\{z_i \in Z_c \,|\, \alpha_i > \alpha_*^j\}|}{|Z_c|+1} + \theta \frac{|\{z_i \in Z_c \,|\, \alpha_i = \alpha_*^j\}|+1}{|Z_c|+1}, \qquad (7)$$
where $\theta \in \mathcal{U}[0,1]$ is a tie-breaking random variable.
5. Return the prediction region
$$\Gamma_*^\varepsilon = \{t^j \in T \,|\, p_*^j > \varepsilon\}. \qquad (8)$$

In defining the NCF δ, we should aim to obtain high δ values for wrong predictions and low δ values for correct ones. Thus, a natural choice in classification is to define
$$\delta(x, t^j) = 1 - f_d^j(x), \qquad (9)$$
where $f_d^j(x)$ is the likelihood predicted by f_d for class t_j. Indeed, if t^j is the true target for x and f correctly predicts t^j, then $f_d^j(x)$ is high (the highest among all classes) and $\delta(x, t^j)$ is low; the opposite holds if f does not predict t^j.

---

[8] Ties can be resolved by imposing an ordering over the classes.

Prediction uncertainty. A CP-based prediction region provides a set of plausible predictions with statistical guarantees, and as such, also captures the uncertainty about the prediction. Indeed, if CP produces a region Γ_*^ε with more than one class, then the prediction for x_* is *ambiguous* (i.e., multiple predictions are plausible), and thus, potentially erroneous. Similarly, if Γ_*^ε is empty, then there are no plausible predictions at all, and thus, none can be trusted. The only reliable prediction is the one where Γ_*^ε contains only one class. In this case, $\Gamma_*^\varepsilon = \{\hat{t}_*\}$, i.e., the region only contains the predicted class, as stated in the following proposition.

Proposition 1. *For the NCF function* (9)*, if $\Gamma_*^\varepsilon = \{t^{j_1}\}$, then $t^{j_1} = f(x_*)$.*

The size of the prediction region is determined by the chosen significance level ε and by the p-values derived via CP. Specifically, from Equation (8) we can see that, for levels $\varepsilon_1 \geq \varepsilon_2$, the corresponding prediction regions are such that $\Gamma^{\varepsilon_1} \subseteq \Gamma^{\varepsilon_2}$. It follows that, given a test input x_*, if ε is lower than all its p-values, i.e. if $\varepsilon < \min_{j=1,\ldots,c} p_*^j$, then the region Γ_*^ε contains all the classes, and Γ_*^ε shrinks as ε increases. In particular, Γ_*^ε is empty when $\varepsilon \geq \max_{j=1,\ldots,c} p_*^j$.

In the classification scenario, CP introduces two additional point-wise measures of uncertainty, called confidence and credibility, defined in terms of two p-values, independently of the significance level ε. The intuition is that these two p-values identify the range of ε values for which the prediction is reliable, i.e., $|\Gamma_*^\varepsilon| = 1$.

Definition 3 (Confidence and credibility). *Given a predictor F, the confidence of a point $x_* \in X$, denoted by $1 - \gamma_*$, is defined as $1 - \gamma_* = \sup\{1 - \varepsilon : |\Gamma_*^\varepsilon| = 1\}$, and the credibility of x_*, denoted by κ_*, is defined as $\kappa_* = \inf\{\varepsilon : |\Gamma_*^\varepsilon| = 0\}$. The so-called confidence-credibility interval $[\gamma_*, \kappa_*)$ contains all the values of ε such that $|\Gamma_*^\varepsilon| = 1$.*

The confidence $1 - \gamma_*$ is the highest probability value for which the corresponding prediction region contains only $\hat{t}_*$, and thus it measures how likely (according to the calibration set) our prediction for x_* is. In particular, γ_* corresponds to the second largest p-value. The credibility κ_* is the smallest level for which the prediction region is empty, i.e., no plausible prediction is found by CP. It corresponds to the highest p-value, i.e., the p-value of the predicted class.

Fig. 5 illustrates CP p-values and corresponding prediction region sizes. In binary classification problems, each point x_* has only two p-values: κ_* (p-value of the predicted class) and γ_* (p-value of the other class). It follows that the higher $1 - \gamma_*$ and κ_* are, the more reliable the prediction $\hat{t}_*$ is, because we have an expanded range $[\gamma_*, \kappa_*)$ of ε values by which $|\Gamma_*^\varepsilon| = 1$. Indeed, in the degenerate case where $\kappa_* = 1$ and $\gamma_* = 0$, then $|\Gamma_*^\varepsilon| = 1$ for any value of $\varepsilon < 1$. This

Fig. 5. CP p-values and corresponding sizes of prediction interval. $\tilde{y}^i$ is the class with the i-th largest p-value, so $p_*^{\tilde{\ell}^1} = \kappa_*$ and $p_*^{\tilde{\ell}^2} = \gamma_*$.

is why, as we will explain in the next section, we do not trust predictions with low values of $1 - \gamma_*$ and κ_*. Hence, our CP-based uncertainty measure associates with each input its confidence and credibility values.
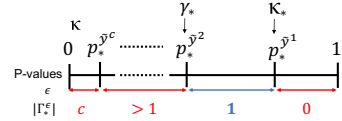
Label-conditional approach. The validity property, as stated above, guarantees an error rate over all possible labels, not on a per-label basis. The latter can be achieved with a

CP variant, called *label-conditional CP* [33,57,60]. In this variant, the p-value associated with class t^j on a test point x_* is defined in a conditional manner as follows:

$$p_*^j = \frac{|\{z_i \in Z_c : t_i = t^j, \alpha_i > \alpha_*^j\}|}{|\{z_i \in Z_c : t_i = t^j\}| + 1} + \theta \frac{|\{z_i \in Z_c : t_i = t^j, \alpha_i = \alpha_*^j\}| + 1}{|\{z_i \in Z_c : t_i = t^j\}| + 1}. \tag{10}$$

In other words, we consider only the α_i corresponding to examples with the same label t^j as the hypothetical label that we are assigning at the test point.

Label-conditional validity is very important when CP is applied to an unbalanced dataset, whereby CP regions tend to have larger error rates with the minority class than with the majority one. The label-conditional approach ensures that, even for the minority class, the expected error rate will tend to the chosen significance level ε.

CP for Regression In regression, we have a continuous target space $T \subseteq \mathbb{R}^n$. The CP algorithm for regression is similar to the classification one. In particular, the offline phase of steps 1–3, i.e., training of regression model f and definition of NCF δ, is the same (with obviously a different kind of f and δ).

The online phase changes though, because T is a continuous space and thus, it is not possible to enumerate the target values and compute for each a p-value. Instead, we proceed in an equivalent manner, that is, identify the critical value $\alpha_{(\varepsilon)}$ of the calibration distribution, i.e., the NCF score corresponding to a p-value of ε. The resulting ε-prediction region is given by $\Gamma_*^\varepsilon = f(x_*) \pm \alpha_{(\varepsilon)}$, where $\alpha_{(\varepsilon)}$ is the $(1-\varepsilon)$-quantile of the calibration distribution, i.e., the $\lfloor \varepsilon \cdot (|Z_c| + 1) \rfloor$-th largest calibration score. A natural NCF in regression, and the one used in our experiments, is the norm of the difference between the real and the predicted target value, i.e., $\delta(x,t) = ||t - f(x)||$.

Normalized CP. The main limitation of CP for regression, presented above, is that the size of prediction intervals is identical ($2\alpha_{(\varepsilon)}$) for all test inputs, making CP non-informative to check how the uncertainty distributes over X. Normalized Conformal Predictions (NCP) [46,47] tackle this limitation. In order to get individual, input-conditional bounds for each point x_i, we can define normalized nonconformity scores as follows

$$\tilde{\alpha}_c = \left\{ \frac{\delta(x_i, t_i)}{u(x_i)} \;\middle|\; (x_i, t_i) \in Z_c \right\}, \tag{11}$$

where $u(x_i)$ estimates the difficulty of predicting $f(x_i)$. The rationale is that if two points have the same nonconformity scores using δ, the one expected to be more accurate, should be stranger (more nonconforming) than the other one. Hence, we aim at error bounds that are tighter for inputs x that are deemed easy to predict and vice-versa. Even for locally-weighted residuals, as in (11), the validity of the conformal methods carries over. As before we compute $\tilde{\alpha}_{(\varepsilon)}$ as the $(1-\varepsilon)$-quantile of the scores $\tilde{\alpha}_c$ and the coverage guarantees over the error become:

$$Pr_{(x,t) \sim \mathcal{Z}} \left(\delta(x,t) \leq \tilde{\alpha}_{(\varepsilon)} \cdot u(x) \right) \geq 1 - \varepsilon. \tag{12}$$

Conformalized Quantile Regression. The goal of conformalized quantile regression (CQR) [53] is to adjust the QR prediction interval (i.e. the interval obtained by the prediction of two quantiles as in Problem 3) so that it is guaranteed to contain the $(1-\varepsilon)$ mass of probability. As for CP, we divide the dataset Z' in a training set Z_t and a calibration set Z_c. We train the QR f over Z_t and on Z_c we compute the nonconformity scores as

$$\alpha_c := \max\{\hat{q}_{\varepsilon_{lo}}(x_i) - t_i, t_i - \hat{q}_{\varepsilon_{hi}}(x_i) \mid (x_i, t_i) \in Z_c\}. \tag{13}$$

In our notation, $\hat{q}_{\varepsilon_{lo}}(x)$ and $\hat{q}_{\varepsilon_{hi}}(x)$ denotes the two outputs of the pretrained predictor f evaluated over x[9]. The conformalized prediction interval is thus defined as

$$CPI(x_*) = [\hat{q}_{\varepsilon_{lo}}(x_*) - \alpha_{(\varepsilon)}, \hat{q}_{\varepsilon_{hi}}(x_*) + \alpha_{(\varepsilon)}],$$

where $\alpha_{(\varepsilon)}$ is the $\lfloor(1-\varepsilon)(1+1/|Z_c|)\rfloor$-th empirical quantile of α_c. In the following, we will abbreviate with *PI* a (non-calibrated) QR prediction interval and with *CPI* a (calibrated) conformalized prediction interval.

Similarly to normalized CP, the above-defined CPI provides individual uncertainty estimates as the size of the interval changes according to the (input-conditional) quantile predictions.

Remark 4. This nonconformity function, and thus $\alpha_{(\varepsilon)}$, can be negative and thus the conformalized prediction interval can be tighter than the original prediction interval. This means that the CPI can be more efficient than the PI, where the efficiency is the average width of the prediction intervals over a test set. The CPI has guaranteed coverage (the PI does not), i.e. $\mathbb{P}_{(x_*,t_*)\sim\mathcal{Z}}(t_* \in CPI(x_*)) \geq 1-\varepsilon$.

CP under Covariate Shift. CP guarantees hold under the assumption that training, calibration and test data come from the same data distribution $\mathcal{Z}$. However, there exist CP extensions [59,20] that provide statistical guarantees even in the presence of covariate shift at test time, meaning that the distribution $\mathcal{X}$ over inputs changes. The core concept is to reweight the nonconformity scores of the calibration set to account for the distribution shift. Such weights are defined using the density ratio between the shifted and original distribution, to quantify the probability of observing a particular calibration input relative to the shifted distribution.

4.2 Bayesian Inference

In general, a Bayesian inference problem aims at inferring an accurate probabilistic estimate of the unknown function from X to T (as before). In the following, let $f: X \to T$. The main ingredients of a Bayesian approach are the following:

1. Choose a *prior* distribution, $p(f)$, over a suitable function space, encapsulating the beliefs about function f prior to any observations being taken.
2. Determine the functional form of the observation process by defining a suitable *likelihood* function $p(Z'|f)$ that effectively models how the observations depend on the input.
3. Leverage Bayes' theorem to define the *posterior* distribution over functions given the observations $p(f|Z') = p(Z'|f)p(f)/p(Z')$. Computing $p(Z') = \int p(Z'|f)p(f)df$ is almost always computationally intractable as we have non-conjugate prior-likelihood distributions. Therefore, we need algorithms to accurately approximate such posterior distribution.
4. Evaluate such posterior at points x_*, resulting in a predictive distribution $p(f_*|x_*,Z')$, whose statistics are used to obtain the desired estimate of the satisfaction probability together with the respective credible interval.

---

[9] If f outputs more than two quantiles, $\hat{q}_{\varepsilon_{lo}}(x)$ and $\hat{q}_{\varepsilon_{hi}}(x)$ denote the predicted quantiles associated respectively with the lowest and highest associated significance level.

Predictive uncertainty. Once the empirical approximation of the predictive distribution $p(f_*|x_*, Z')$ is derived, one can extract statistics from it to characterize predictive uncertainty. We stress that the predictive distribution, and hence its statistics, effectively capture prediction uncertainty. For instance, the empirical mean and variance of the predictive distribution can be used as measures for Bayesian predictive uncertainty.

Remark 5. The Bayesian quantification of uncertainty, despite being based on statistically sound operations, offers no guarantees per se as it strongly depends on the chosen prior and typically relies on approximate inference. However, we can make predictions based on a functional of the predictive distribution and exploit the provided quantification of uncertainty as the normalizing function of an NCP approach, that in turn will provide us with point-specific statistical guarantees over the error coverage.

In a Bayesian framework two main ingredients are essential to define the solution strategy to a Bayesian inference problem define above: (i) the probabilistic model chosen to describe the distribution over functions f and (ii) the approximate inference strategy. We refer to Appendix A of [12] for details on the possible approaches to Bayesian inference. In particular, we present Gaussian Processes and Bayesian Neural Nets, as alternatives for ingredient (i), and Variational Inference (VI) and Hamilton Monte Carlo (HMC), as alternatives for ingredient (ii).

5 Learning-based PM with Statistical Guarantees

5.1 Monitoring under Full Observability

Given a fully observable Markovian system with a known SAT oracle, the system's current state s_t at time t is sufficient information to predict the future satisfaction of a requirement φ. The input space of the Sat function is thus $S(\mathcal{M})$.

When the system evolves *deterministically* each state $s \in S(\mathcal{M})$ is associated with a unique satisfaction value (as in Problem 1). If we are interested in the Boolean satisfaction the output space B of the Sat function is $\{0,1\}$ and the learning problem is a classical binary classification problem, i.e. inferring a function $h: S(\mathcal{M}) \to \{0,1\}$ that classify a state as positive (if it satisfies the requirement) or negative (if it violates the requirement). Analogously, if we want to better quantify how robust is the satisfaction we can leverage the quantitative STL semantics, either spatial or temporal. In this scenario, the output space B of the Sat function is $\mathbb{R}$ and the learning problem becomes a regression task, i.e. inferring a function $h: S(\mathcal{M}) \to \mathbb{R}$ that estimates the level of satisfaction of φ for each state in $S(\mathcal{M})$.

The function h, introduced in its general form in Problem 1, can be inferred either using a deterministic neural network or one of the proposed Bayesian approaches. CP can be used on top of both approaches to meet the validity guarantees of Problem 4. In the CP-based version, we apply either CP for classification or CP for regression to obtain prediction regions with guaranteed coverage over the entire state space (see [13] for details). On the other, we can design either a Bayesian classifier (with Bernoulli likelihood for the Boolean semantics) or a Bayesian regressor (Gaussian likelihood for quantitative semantics). See [14] for details. In order to meet the desired statistical guarantees we could use CP. Since Bayesian predictions are probabilistic, whereas CP is defined for deterministic predictors, we apply CP to the expectation over the predictive distribution. Nonetheless, the variance of the latter can be exploited as normalizing

constant in a NCP framework so to obtain state-specific prediction intervals while preserving the statistical guarantees. Similar reasoning is applied to *nondeterministic systems.*

Otherwise, when the system evolves *stochastically*, each state $s \in S(\mathcal{M})$ is associated with a distribution over the satisfaction values $\mathsf{SSat}(s)$ (as discussed in Problem 3 in Section 3). We are not able to extract an analytic expression for this distribution but we can empirically approximate it via sampling. If we consider the Boolean semantics, $\mathsf{SSat}(s)$ is a Binomial distribution centred around the satisfaction probability in the interval [0,1]. In such a scenario, we could either train a deterministic neural regressor that infers the satisfaction probability in [0,1] or design a Bayesian framework with Binomial likelihood (see [12] for details). In order to meet the desired statistical guarantees we could use CP for regression. Once again, in the Bayesian scenario, CP is applied to the expectation over the predictive distribution. However, the variance of the latter can be used as normalizing constant in a NCP framework so to obtain state-specific prediction intervals. On the other hand, the quantitative STL semantics, either spatial or temporal, results in a distribution over $\mathbb{R}$. We can train neural quantile regression (QR) that, given a desired confidence level ε, infers some quantiles of this distribution (e.g. $q_{\frac{\varepsilon}{2}}$, $q_{0.5}$ and $q_{\frac{\varepsilon}{2}}$). A typical loss for the regression of a quantile q_α is the pinball loss $\mathcal{L}_\alpha(t,\hat{q}_\alpha) = \alpha \cdot \max(t - \hat{q}_\alpha, 0) + (1 - \alpha) \cdot \max(\hat{q}_\alpha - t, 0)$, where $\hat{q}_\alpha$ is the predicted quantile and $t \in \mathbb{R}$ denotes an observed output. Once the QR is trained we can resort to CQR (see Fig. 6) to meet the probabilistic guarantees, in that the conformal intervals cover with probability at least $1 - \varepsilon$ the STL robustness values relative to the stochastic evolution of the system. The rationale is to evaluate the nonconformity scores of the interval $\left[\hat{q}_{\frac{\varepsilon}{2}}(s), \hat{q}_{1 - \frac{\varepsilon}{2}}(s) \right]$ over the calibration set and extract τ, the $\lfloor (1 - \varepsilon)(1 + 1/|Z_c|) \rfloor$-th empirical quantile of α_c, to recalibrate the prediction interval (see [19] for details).

5.2 Monitoring under Partial Observability

For ease of discussion, in the PO scenario (outlined in Problem 2), we discuss only the CP-based setting and not the Bayesian one (see [17] for details). The *end-to-end* approach is very similar to the FO deterministic one. The main difference is that instead of the state at time t we map the history of past measurements $\mathbf{y}_t$ to the satisfaction value in B, i.e. we infer a function $h_{po} : Y^{H_p} \to \mathsf{B}$. As before, the output can be either Boolean $\mathsf{B} = \{0,1\}$ (binary classification task), or quantitative $\mathsf{B} = \mathbb{R}$ (regression task). The sequence of past observations is mapped to a unique satisfaction value in B and CP can be used to enrich the predictions with guaranteed validity. On the other hand, if we consider a *two-step* approach we first estimate the sequence of states $\mathbf{s}_t$ (regression task) and then we estimate the satisfaction value associated with each sequence which is either a classification or a regression task (as in the end-to-end approach). The two
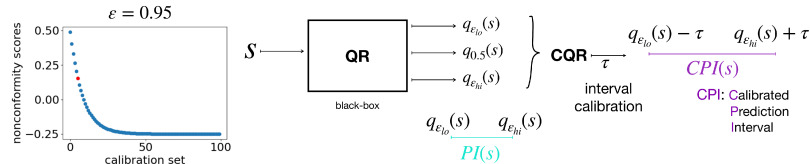

Fig. 6. Overview of conformalized quantile regression.

steps can be fine-tuned together and conformal inference can be applied to both steps to obtain statistical guarantees.

5.3 Uncertainty-aware Error Detection and Active Learning

It is well known that neural networks are universal approximators. However, such methods cannot completely avoid prediction errors (no supervised learning method can). Therefore, we have to deal with predictive monitors f that are prone to prediction errors: when, for a state $s \in S(\mathcal{M})$, $f(x) \neq \mathsf{Sat}(x)$. These errors are respectively denoted by predicates $pe(s)$.

Problem 5 (Uncertainty-based error detection) *Given a reachability predictor f, a distribution $\mathcal{X}$ over HA states X, a predictive uncertainty measure $u_f : X \to U$ over some uncertainty domain U, and a kind of error pe find an optimal error detection rule $G^*_{f,pe} : U \to \{0,1\}$, i.e., a function that minimizes the probability*

$$Pr_{x \sim \mathcal{X}} \big(pe(x) \neq G^*_{f,pe}(u_f(x)) \big).$$

In the above problem, we consider all kinds of prediction errors, but the definition and approach could be easily adapted to focus on the detection of only e.g., false positives (the most problematic errors from a safety-critical viewpoint).

In the CP-based setting, a meaningful measure of predictive uncertainty is given by confidence and credibility. In the Bayesian framework, we can consider the mean and the variance of the predictive distribution.

As for Problem 1, 2 and 3, we can obtain a sub-optimal solution $G_{f,pe}$ to Problem 5 by expressing the latter as a supervised learning problem, where the inputs are, once again, sampled according to $\mathcal{X}$ and labelled using a SAT oracle. We call *validation set* the set of labelled observations used to learn $G_{f,pe}$. These observation need to be independent from the above introduced training set Z', i.e., those used to learn the reachability predictor f. The final rejection rule $\mathsf{Rej}_{f,pe}$ for detecting HA states where the satisfaction prediction (given by f) should not be trusted, and thus rejected, is readily obtained by the composition of the uncertainty measure and the error detection rule $\mathsf{Rej}_{f,pe} = G_{f,e} \circ u_f : X \to \{0,1\}$, where $\mathsf{Rej}_{f,pe}(x) = 1$ if the prediction on x is rejected and $\mathsf{Rej}_{f,pe}(x) = 0$ otherwise.

This error-detection criterion can be also used as a query strategy in an uncertainty-aware *active learning* setting. Active learning should reduce the overall number of erroneous predictions because it improves the predictor on the inputs where it is most uncertain.

6 Related Work

A number of methods have been proposed for online reachability analysis that rely on separating the reachability computation into distinct offline and online phases. However, these methods are limited to restricted classes of models [23,63], or require handcrafted optimization of the HA's derivatives [7], or are efficient only for low-dimensional systems and simple dynamics [55]. In contrast, the approaches presented in this paper are based on learning DNN-based predictors, are fully automated and have negligible computational cost at runtime. In [26,54], similar techniques are introduced for neural approximation of Hamilton-Jacobi (HJ) reachability. However, our methods for prediction rejection and active learning are independent of the class of systems and

the machine-learning approximation of reachability, and thus can also be applied to neural approximations of HJ reachability. In [62], Yel and others present a runtime monitoring framework that has similarities with our approach, in that they also learn neural network-based reachability monitors (for UAV planning applications), but instead of using, like we do, uncertainty measures to pin down potentially erroneous predictions, they apply NN verification techniques [36] to identify input regions that might produce false negatives. Thus, their approach is complementary to our uncertainty-based error detection, but, due to the limitations of the underlying verification algorithms, they can only support deterministic neural networks with sigmoid activations. On the contrary, our techniques support any kind of ML-based monitors, including probabilistic ones. The work of [4,3] addresses the predictive monitoring problem for stochastic black-box systems, where a Markov model is inferred offline from observed traces and used to construct a predictive runtime monitor for probabilistic reachability checking. In contrast to our method, this method focuses on discrete-space models, which allows the predictor to be represented as a look-up table, as opposed to a neural network. In [49], a method is presented for predictive monitoring of STL specifications with probabilistic guarantees. These guarantees derive from computing prediction intervals of ARMA/ARIMA models learned from observed traces. Similarly, we use CP which also can derive prediction intervals with probabilistic guarantees, with the difference that CP supports any class of prediction models (including auto-regressive ones). In [27], model predictions are used to forecast future robustness values of MTL specifications for runtime monitoring. However, no guarantee, statistical or otherwise, is provided for the predicted robustness. Deshmukh and others [25] have proposed an interval semantics for STL over partial traces, where such intervals are guaranteed to include the true STL robustness value for any bounded continuation of the trace. This approach can be used in the context of predictive monitoring but tends to produce over-conservative intervals. Another related approach is smoothed model checking [15], where Gaussian processes [51] are used to approximate the satisfaction function of stochastic models, i.e., mapping model parameters into the satisfaction probability of a specification. Smoothed model checking leverages Bayesian statistics to quantify prediction uncertainty, but faces scalability issues as the dimension of the system increases. These scalability issues are alleviated in [12] using stochastic variational inference. In contrast, computing our conformal measure of prediction reliability is very efficient, because it is nearly equivalent to executing the underlying predictor.

This tutorial builds on the methods presented in [48,13,14,17,18,19,12]. In NPM [13,14], neural networks are used to infer the Boolean satisfaction of a reachability property and conformal prediction (CP) are used to provide statistical guarantees. NPM has been extended to support some source of stochasticity in the system: in [17] they allow partial observability and noisy observations, in [18] the system dynamics are stochastic but the monitor only evaluates the Boolean satisfaction of some quantile trajectories, providing a limited understanding of the safety level of the current state. Finally in [19] a conformal quantitative predictive monitor to reliably check the satisfaction of STL requirements over evolutions of a stochastic system at runtime is presented. Predictive monitoring under partial observability is also analysed in [24], where the authors combine Bayesian state estimation with pre-computed reach sets to reduce the runtime overhead. While their reachability bounds are certified, no correctness guarantees can be established for the estimation step.

Various learning-based PM approaches for temporal logic proper-
ties [50,41,64,65,52,43] have been recently proposed. In particular, Ma et al. [41] use
uncertainty quantification with Bayesian RNNs to provide confidence guarantees.
However, these models are, by nature, not well-calibrated (i.e., the model uncertainty
does not reflect the observed one [38]), making the resulting guarantees not theoretically
valid. In [5] the parameter space of a parametric CTMC is explicitly explored, while [21]
assumes a probability distribution over the parameters and proposes a sampling-based
approach. In [40] conformal predictions are used over the expected value of the
stochastic process rather than its distribution.

We contribute to the state of the art by presenting a wide variety of learning-based
predictive monitors that offer good scalability, provide statistical guarantees, and support
partial observability, stochasticity and rich STL-based requirements.

7 Conclusions

We have presented an overview of various learning-based approaches to reliably monitor
the evolution of a CPS at runtime. The proposed methods complement predictions
over the satisfaction of an STL specification with principled estimates of the prediction
uncertainty. These estimates can be used to derive optimal rejection criteria that identify
potentially erroneous predictions without knowing the true satisfaction values. The
latter can be exploited as an active learning strategy increasing the accuracy of the
satisfaction predictor. The strength is given by high-reliability and high computational
efficiency of our predicitons. The efficiency is not directly affected by the complexity
of the system under analysis but only by the complexity of the learned predictor.
Our approach overcomes the computational footprint of model checking (infeasible at
runtime) while improving on traditional runtime verification by being able to detect
future violations in a preemptive way. We have devised two alternative solution methods:
a frequentist and a Bayesian approach. Conformal predictions are used on top of both
methods to obtain statistical guarantees.

In future work, we will investigate dynamics-aware approaches to inference. The aim
is to improve the performances by limiting inference only to an estimate of the system
manifold, i.e. the region of the state space that is likely to be visited by the evolving
stochastic process.

References

1. Alur, R.: Principles of cyber-physical systems. MIT Press (2015)
2. Annpureddy, Y., Liu, C., Fainekos, G., Sankaranarayanan, S.: S-taliro: A tool for temporal
logic falsification for hybrid systems. In: International Conference on Tools and Algorithms
for the Construction and Analysis of Systems. pp. 254–257. Springer (2011)

3. Babaee, R., Ganesh, V., Sedwards, S.: Accelerated learning of predictive runtime monitors for rare failure. In: International Conference on Runtime Verification. pp. 111–128. Springer (2019)

4. Babaee, R., Gurfinkel, A., Fischmeister, S.: Predictive run-time verification of discrete-time reachability properties in black-box systems using trace-level abstraction and statistical learning. In: International Conference on Runtime Verification. pp. 187–204. Springer (2018)

5. Badings, T.S., Jansen, N., Junges, S., Stoelinga, M., Volk, M.: Sampling-based verification of ctmcs with uncertain rates. In: Computer Aided Verification: 34th International Conference, CAV 2022, Haifa, Israel, August 7–10, 2022, Proceedings, Part II. pp. 26–47. Springer (2022)

6. Bak, S., Duggirala, P.S.: Hylaa: A tool for computing simulation-equivalent reachability for linear systems. In: Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control. pp. 173–178 (2017)

7. Bak, S., Johnson, T.T., Caccamo, M., Sha, L.: Real-time reachability for verified simplex design. In: Real-Time Systems Symposium (RTSS), 2014 IEEE. pp. 138–148. IEEE (2014)

8. Balasubramanian, V., Ho, S.S., Vovk, V.: Conformal prediction for reliable machine learning: theory, adaptations and applications. Newnes (2014)

9. Bartocci, E., Deshmukh, J., Donzé, A., Fainekos, G., Maler, O., Ničković, D., Sankaranarayanan, S.: Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. In: Lectures on Runtime Verification, pp. 135–175. Springer (2018)

10. Benvenuti, L., Bresolin, D., Casagrande, A., Collins, P., Ferrari, A., Mazzi, E., Sangiovanni-Vincentelli, A., Villa, T.: Reachability computation for hybrid systems with ariadne. IFAC Proceedings Volumes **41**(2), 8960–8965 (2008)

11. Bogomolov, S., Forets, M., Frehse, G., Potomkin, K., Schilling, C.: JuliaReach: a toolbox for set-based reachability. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control. pp. 39–44 (2019)

12. Bortolussi, L., Cairoli, F., Carbone, G., Pulcini, P.: Stochastic variational smoothed model checking. arXiv preprint arXiv:2205.05398 (2022)

13. Bortolussi, L., Cairoli, F., Paoletti, N., Smolka, S.A., Stoller, S.D.: Neural predictive monitoring. In: Runtime Verification: 19th International Conference, RV 2019, Porto, Portugal, October 8–11, 2019, Proceedings 19. pp. 129–147. Springer (2019)

14. Bortolussi, L., Cairoli, F., Paoletti, N., Smolka, S.A., Stoller, S.D.: Neural predictive monitoring and a comparison of frequentist and bayesian approaches. International Journal on Software Tools for Technology Transfer **23**(4), 615–640 (2021)

15. Bortolussi, L., Milios, D., Sanguinetti, G.: Smoothed model checking for uncertain continuous-time Markov chains. Information and Computation **247**, 235–253 (2016)

16. Brihaye, T., Doyen, L., Geeraerts, G., Ouaknine, J., Raskin, J.F., Worrell, J.: On reachability for hybrid automata over bounded time. In: International Colloquium on Automata, Languages, and Programming. pp. 416–427. Springer (2011)

17. Cairoli, F., Bortolussi, L., Paoletti, N.: Neural predictive monitoring under partial observability. In: Runtime Verification: 21st International Conference, RV 2021, Virtual Event, October 11–14, 2021, Proceedings 21. pp. 121–141. Springer (2021)

18. Cairoli, F., Paoletti, N., Bortolussi, L.: Neural predictive monitoring for collective adaptive systems. In: International Symposium on Leveraging Applications of Formal Methods. pp. 30–46. Springer (2022)

19. Cairoli, F., Paoletti, N., Bortolussi, L.: Conformal quantitative predictive monitoring of stl requirements for stochastic processes. In: Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control. pp. 1–11 (2023)

20. Cauchois, M., Gupta, S., Ali, A., Duchi, J.C.: Robust validation: Confident predictions even when distributions shift. arXiv preprint arXiv:2008.04267 (2020)

22

21. Češka, M., Dannenberg, F., Paoletti, N., Kwiatkowska, M., Brim, L.: Precise parameter synthesis for stochastic biochemical systems. Acta Informatica **54**, 589–623 (2017)
22. Chen, H., Lin, S., Smolka, S.A., Paoletti, N.: An STL-based formulation of resilience in cyber-physical systems. In: International Conference on Formal Modeling and Analysis of Timed Systems. pp. 117–135. Springer (2022)
23. Chen, X., Sankaranarayanan, S.: Model predictive real-time monitoring of linear systems. In: Real-Time Systems Symposium (RTSS), 2017 IEEE. pp. 297–306. IEEE (2017)
24. Chou, Y., Yoon, H., Sankaranarayanan, S.: Predictive runtime monitoring of vehicle models using bayesian estimation and reachability analysis. In: Intl. Conference on Intelligent Robots and Systems (IROS) (2020)
25. Deshmukh, J.V., Donzé, A., Ghosh, S., Jin, X., Juniwal, G., Seshia, S.A.: Robust online monitoring of signal temporal logic. Formal Methods in System Design **51**(1), 5–30 (2017)
26. Djeridane, B., Lygeros, J.: Neural approximation of PDE solutions: An application to reachability computations. In: Proceedings of the 45th IEEE Conference on Decision and Control. pp. 3034–3039. IEEE (2006)
27. Dokhanchi, A., Hoxha, B., Fainekos, G.: On-line monitoring for temporal logic robustness. In: International Conference on Runtime Verification. pp. 231–246. Springer (2014)
28. Donzé, A.: Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: International Conference on Computer Aided Verification. pp. 167–170. Springer (2010)
29. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: International Conference on Formal Modeling and Analysis of Timed Systems. pp. 92–106. Springer (2010)
30. Duggirala, P.S., Mitra, S., Viswanathan, M., Potok, M.: C2e2: A verification tool for stateflow models. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 68–82. Springer (2015)
31. Frehse, G.: Phaver: Algorithmic verification of hybrid systems past hytech. In: International workshop on hybrid systems: computation and control. pp. 258–273. Springer (2005)
32. Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: Spaceex: Scalable verification of hybrid systems. In: International Conference on Computer Aided Verification. pp. 379–395. Springer (2011)
33. Gammerman, A., Vovk, V.: Hedging predictions in machine learning. The Computer Journal **50**(2), 151–163 (2007)
34. Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.: The probabilistic model checker storm. International Journal on Software Tools for Technology Transfer pp. 1–22 (2021)
35. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? In: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing. pp. 373–382 (1995)
36. Ivanov, R., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verisig: verifying safety properties of hybrid systems with neural network controllers. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control. pp. 169–178 (2019)
37. Johnson, T.T., Bak, S., Caccamo, M., Sha, L.: Real-time reachability for verified simplex design. ACM Transactions on Embedded Computing Systems (TECS) **15**(2), 1–27 (2016)
38. Kuleshov, V., Fenner, N., Ermon, S.: Accurate uncertainties for deep learning using calibrated regression. In: International Conference on Machine Learning. pp. 2796–2804. PMLR (2018)
39. Kwiatkowska, M., Norman, G., Parker, D.: Prism 4.0: Verification of probabilistic real-time systems. In: Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings 23. pp. 585–591. Springer (2011)
40. Lindemann, L., Qin, X., Deshmukh, J.V., Pappas, G.J.: Conformal prediction for stl runtime verification. In: Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023). pp. 142–153 (2023)

41. Ma, M., Stankovic, J., Bartocci, E., Feng, L.: Predictive monitoring with logic-calibrated uncertainty for cyber-physical systems. ACM Transactions on Embedded Computing Systems (TECS) **20**(5s), 1–25 (2021)

42. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, pp. 152–166. Springer (2004)

43. Muthali, A., Shen, H., Deglurkar, S., Lim, M.H., Roelofs, R., Faust, A., Tomlin, C.: Multi-agent reachability calibration with conformal prediction. arXiv preprint arXiv:2304.00432 (2023)

44. Ničković, D., Yamaguchi, T.: Rtamt: Online robustness monitors from stl. In: International Symposium on Automated Technology for Verification and Analysis. pp. 564–571. Springer (2020)

45. Papadopoulos, H.: Inductive conformal prediction: Theory and application to neural networks. In: Tools in artificial intelligence. InTech (2008)

46. Papadopoulos, H., Haralambous, H.: Reliable prediction intervals with regression neural networks. Neural networks : the official journal of the International Neural Network Society **24 8**, 842–51 (2011)

47. Papadopoulos, H., Vovk, V., Gammerman, A.: Regression conformal prediction with nearest neighbours. J. Artif. Intell. Res. **40**, 815–840 (2014)

48. Phan, D., Paoletti, N., Zhang, T., Grosu, R., Smolka, S.A., Stoller, S.D.: Neural state classification for hybrid systems. In: International Symposium on Automated Technology for Verification and Analysis. pp. 422–440. Springer (2018)

49. Qin, X., Deshmukh, J.V.: Predictive monitoring for signal temporal logic with probabilistic guarantees. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control. pp. 266–267. ACM (2019)

50. Qin, X., Deshmukh, J.V.: Clairvoyant monitoring for signal temporal logic. In: International Conference on Formal Modeling and Analysis of Timed Systems. pp. 178–195. Springer (2020)

51. Rasmussen, C.E., Williams, C.K.: Gaussian processes for machine learning, vol. 1. MIT press Cambridge (2006)

52. Rodionova, A., Lindemann, L., Morari, M., Pappas, G.J.: Time-robust control for STL specifications. In: 2021 60th IEEE Conference on Decision and Control (CDC). pp. 572–579. IEEE (2021)

53. Romano, Y., Patterson, E., Candes, E.: Conformalized quantile regression. Advances in neural information processing systems **32** (2019)

54. Royo, V.R., Fridovich-Keil, D., Herbert, S., Tomlin, C.J.: Classification-based approximate reachability with guarantees applied to safe trajectory tracking. arXiv preprint arXiv:1803.03237 (2018)

55. Sauter, G., Dierks, H., Fränzle, M., Hansen, M.R.: Lightweight hybrid model checking facilitating online prediction of temporal properties. In: Proceedings of the 21st Nordic Workshop on Programming Theory. pp. 20–22 (2009)

56. Schupp, S., Abraham, E., Makhlouf, I.B., Kowalewski, S.: H y p ro: A c++ library of state set representations for hybrid systems reachability analysis. In: NASA Formal Methods Symposium. pp. 288–294. Springer (2017)

57. Shafer, G., Vovk, V.: A tutorial on conformal prediction. Journal of Machine Learning Research **9**(Mar), 371–421 (2008)

58. Stankeviciute, K., M Alaa, A., van der Schaar, M.: Conformal time-series forecasting. Advances in neural information processing systems **34**, 6216–6228 (2021)

59. Tibshirani, R.J., Foygel Barber, R., Candes, E., Ramdas, A.: Conformal prediction under covariate shift. Advances in neural information processing systems **32** (2019)

60. Toccaceli, P., Gammerman, A.: Combination of inductive mondrian conformal predictors. Machine Learning **108**(3), 489–510 (2019)

61. Vovk, V., Gammerman, A., Shafer, G.: Algorithmic learning in a random world. Springer Science & Business Media (2005)
62. Yel, E., Carpenter, T.J., Di Franco, C., Ivanov, R., Kantaros, Y., Lee, I., Weimer, J., Bezzo, N.: Assured runtime monitoring and planning: Toward verification of neural networks for safe autonomous operations. IEEE Robotics & Automation Magazine **27**(2), 102–116 (2020)
63. Yoon, H., Chou, Y., Chen, X., Frew, E., Sankaranarayanan, S.: Predictive runtime monitoring for linear stochastic systems and applications to geofence enforcement for uavs. In: International Conference on Runtime Verification. pp. 349–367. Springer (2019)
64. Yoon, H., Sankaranarayanan, S.: Predictive runtime monitoring for mobile robots using logic-based bayesian intent inference. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 8565–8571. IEEE (2021)
65. Yu, X., Dong, W., Yin, X., Li, S.: Model predictive monitoring of dynamic systems for signal temporal logic specifications. arXiv preprint arXiv:2209.12493 (2022)
66. Zaffran, M., Féron, O., Goude, Y., Josse, J., Dieuleveut, A.: Adaptive conformal predictions for time series. In: International Conference on Machine Learning. pp. 25834–25866. PMLR (2022)