Conformal Quantitative Predictive Monitoring of STL Requirements for Stochastic Processes

Francesca Cairoli
University of Trieste
Trieste, Italy
francesca.cairoli@units.it

Nicola Paoletti
King's College London
London, United Kingdom
nicola.paoletti@kcl.ac.uk

Luca Bortolussi
University of Trieste
Trieste, Italy
lbortolussi@units.it

ABSTRACT

We consider the problem of predictive monitoring (PM), i.e., predicting at runtime the satisfaction of a desired property from the current system's state. Due to its relevance for runtime safety assurance and online control, PM methods need to be efficient to enable timely interventions against predicted violations, while providing correctness guarantees. We introduce *quantitative predictive monitoring (QPM)*, the first PM method to support stochastic processes and rich specifications given in Signal Temporal Logic (STL). Unlike most of the existing PM techniques that predict whether or not some property ϕ is satisfied, QPM provides a quantitative measure of satisfaction by predicting the quantitative (aka robust) STL semantics of ϕ. QPM derives prediction intervals that are highly efficient to compute and with probabilistic guarantees, in that the intervals cover with arbitrary probability the STL robustness values relative to the stochastic evolution of the system. To do so, we take a machine-learning approach and leverage recent advances in conformal inference for quantile regression, thereby avoiding expensive Monte Carlo simulations at runtime to estimate the intervals. We also show how our monitors can be combined in a compositional manner to handle composite formulas, without retraining the predictors or sacrificing the guarantees. We demonstrate the effectiveness and scalability of QPM over a benchmark of four discrete-time stochastic processes with varying degrees of complexity.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**.

KEYWORDS

Predictive Monitoring, Stochastic, Conformal Quantile Regression

1 INTRODUCTION

Predictive monitoring (PM) [7, 8, 10] is the problem of predicting at runtime the satisfaction of a certain requirement from the current system's state. Unlike traditional monitoring [5], PM has the potential to detect failures before they occur, thereby enabling preemptive countermeasures, such as switching to a fail-safe mode [29]. To enable effective deployment at runtime, PM methods need to be efficient and respond quickly, so that any system failure can be prevented in time.

Performing model checking 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, unless the model is fully deterministic. In particular, when the system is stochastic, a statistical model checking solution [39] would require simulating at runtime a typically large number of Monte-Carlo trajectories to achieve desired error levels[1].

For this reason, a number of approximate PM techniques based on machine learning have been recently proposed (see e.g. [13, 22, 30, 37]), including the so-called Neural Predictive Monitoring (NPM) method [7, 8, 10]. In NPM in particular, the predictive monitor is a neural network classifier trained using data generated through a model checker to predict for any system state whether or not the state satisfies some reachability property. To improve the reliability of the reachability predictions, NPM relies on conformal prediction [3, 36] to produce prediction regions with guaranteed coverage and derive uncertainty measures used to infer whether a particular prediction can be trusted.

In this paper, we introduce a predictive monitoring method for stochastic processes and Signal Temporal Logic (STL) [15] specifications. We call it *quantitative predictive monitoring (QPM)* as it represents, to the best of our knowledge, the first method that can predict the quantitative semantics R_ϕ of an STL formula ϕ (aka the STL robustness) for a general class of stochastic systems and with probabilistic validity guarantees. QPM is inspired by NPM but it addresses two significant limitations of the latter, which supports only Boolean reachability specifications (as opposed to the full spectrum of STL properties and their quantitative interpretation), and cannot adequately deal with stochastic dynamics.

By supporting arbitrary STL specifications, with QPM we gain expressiveness in the type of requirements that can be actually monitored and thus in the type of violations that can be detected. In particular, by predicting STL robustness values, QPM provides key quantitative information on the degree of property satisfaction, unlike existing predictive monitoring approaches that can only predict Boolean satisfaction. STL robustness predictions can be

---

[1]Numerical/symbolic probabilistic model checking techniques typically are at least as expensive as statistical methods and can be applied only to a restricted class of models.

meaningfully used to determine the extent of any corrective actions and enable efficient online model predictive control for STL [32]: for instance, depending on the specification ϕ, if QPM predicts a high value of R_ϕ, then little or no intervention might be needed, while a low R_ϕ value might require a more substantial intervention to steer the system back to safety.

A major challenge when dealing with stochastic processes is that every state induces a distribution of robustness values (conditional on the state), relative to the future, time-bounded, stochastic evolution of the system. In general, such distribution is analytically intractable[2] and an accurate empirical estimate of such conditional distribution can be very expensive to obtain, potentially requiring a high number of Monte-Carlo simulations.

Our approach overcomes this computational bottleneck by deriving monitors able to directly predict some relevant quantiles of the conditional STL robustness distribution. Such quantiles have a two-fold purpose. First, they provide a *measure of risk* [21, 23]: for instance, if the 10%-quantile of R_ϕ is zero, then "only" 10% of the system's future trajectories will violate ϕ (i.e., lead to a negative R_ϕ), which, depending on the application, can be interpreted as a low-risk scenario. Second, and most crucially, it allows us to derive *prediction intervals* that cover a certain mass of probability. For small $\alpha \in (0, 0.5)$, if the $(\alpha, 1-\alpha)$-quantile interval for R_ϕ is entirely above (below) zero, then we can be fairly confident that the system will evolve into satisfying (violating) ϕ with high probability. On the other hand, intervals straddling the zero denote states whose future trajectories may or may not violate the property, which we, therefore, label as uncertain.

To predict the quantiles in a reliable manner, our approach builds on *Conformalized Quantile Regression (CQR)* [34], a recent conformal inference technique that produces statistically valid prediction intervals on top of quantile regression models. Crucially, this method provides us with probabilistic guarantees, in that, for an arbitrary level α, the resulting interval for R_ϕ is guaranteed to include the true STL robustness value with probability at least $1 - 2\alpha$. Such guarantees are akin to those of statistical model checking, with the key difference that CQR prediction intervals are extremely fast to compute and thus, can be used at runtime.

By default, our QPM approach derives a monitor for a fixed choice of STL property ϕ. On one hand, this is advantageous in that the monitor is tailored, and hence, highly accurate at predicting R_ϕ but on the other hand, it lacks flexibility. To this purpose, we further show how monitors trained for different properties ϕ_1 and ϕ_2 can be combined in a modular way to monitor Boolean combinations of ϕ_1 and ϕ_2, without retraining the underlying models and preserving the desired probabilistic guarantees,

In summary, our contributions are:

- We introduce *quantitative predictive monitoring (QPM)*, the first predictive monitoring method to support stochastic processes and the quantitative semantics of arbitrary STL specifications. QPM builds on recent conformal inference techniques to provide prediction intervals for STL robustness with probabilistic guarantees.

- We demonstrate the compositionality of our approach by showing that composite formulas can be monitored by combining the QPM monitors of the subformulas.
- We evaluate the effectiveness and scalability of QPM on several case studies of stochastic processes.

2 PROBLEM STATEMENT

We illustrate the predictive monitoring problem we target with QPM, after introducing background about stochastic processes and Signal Temporal Logic.

2.1 Stochastic Processes

The systems we consider can be modeled as stochastic processes. A stochastic process is defined as a collection of random variables indexed by some index set T. These random variables are defined on a common probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is the sample space, $\mathcal{F}$ is the σ-algebra and $\mathbb{P}$ is the probability measure. We can denote the stochastic process as $\{S(t, \omega), t \in T\}$. A random variable $S(t, \omega)$ in the collection is thus a function of two variables $t \in T$ and $\omega \in \Omega$. In our application, the index set is countable and represents *discrete time* $T = \{0, 1, \ldots\}$. Each random variable in the collection takes values in a space $S \subseteq \mathbb{R}^n$, the *state space* of dimension n, that should be measurable. A discrete-time step makes the stochastic process move from index i to index $i + 1$. Given a stochastic process $\{S(t, \omega) : t \in T\}$, then for any point $\omega \in \Omega$, the mapping $S(\cdot, \omega) : T \to S$, is called a *realization*, or a *sample trajectory* of the stochastic process $\{S(t, \omega) : t \in T\}$. We assume that the dynamics of the system is Markovian. This assumption is not strict as most systems of interest – Markov chains, stochastic hybrid systems (without non-determinism), and stochastic difference equations – are Markovian or can be made so by augmenting the state space.

2.2 Signal Temporal Logic

System requirements can be expressed via Signal Temporal Logic (STL) [15, 24], which enables the specification of properties of dense-time, real-valued signals, and the automatic generation of monitors for testing properties on individual trajectories. The rationale of STL is to transform real-valued signals into Boolean ones, using formulae built on the following *STL syntax*: $\phi := true \mid \mu \mid \neg\phi \mid \phi \wedge \phi \mid \phi \ U_I \phi$, where $I \subseteq \mathbb{T}$ is a temporal interval, either bounded, $I = [a, b]$, or unbounded, $I = [a, +\infty)$, for any $0 \le a < b$. Atomic propositions μ are (non-linear) inequalities on the states of a signal $\vec{s}$ at a time t, $\mu = (g(\vec{s}(t)) > 0)$, where $g : S \to \mathbb{R}$ and $\vec{s}(t)$ is a state in S. From this essential syntax, it is easy to define other operators, used to abbreviate the syntax in an STL formula: $false := \neg true$, $\phi \vee \psi := \neg(\neg\phi \wedge \neg\psi)$, $F_I := true \ U_I \phi$ and $G_I := \neg F_I \neg\phi$. Monitoring the satisfaction of a formula is done recursively by leveraging the tree structure of the STL formula. The satisfaction relation is defined as follows.

$$
\begin{aligned}
(\vec{s}, t) \models \mu \quad &\Leftrightarrow \quad g(\vec{s}(t)) > 0 \\
(\vec{s}, t) \models \neg\phi \quad &\Leftrightarrow \quad \neg((\vec{s}, t) \models \phi) \\
(\vec{s}, t) \models \phi_1 \wedge \phi_2 \quad &\Leftrightarrow \quad (\vec{s}, t) \models \phi_1 \wedge (\vec{s}, t) \models \phi_2 \\
(\vec{s}, t) \models \phi_1 U_{a,b} \phi_2 \quad &\Leftrightarrow \quad \exists\, t' \in [t+a, t+b] \text{ s.t. } (\vec{s}, t') \models \phi_2 \wedge \\
&\qquad\qquad \forall\, t'' \in [t, t'],\ (\vec{s}, t'') \models \phi_1
\end{aligned}
$$

---

[2]With simple stochastic models and simple properties, it may become tractable but still impractical to analyze at runtime.

Quantitative semantics. The robustness of a trajectory quantifies the level of satisfaction w.r.t. ϕ. Positive robustness means that the property is satisfied, whereas negative robustness means that the property is violated. Robustness is denoted as a function $R_\phi : S^H \times T \to \mathbb{R}$ that maps a given signal $\vec{s}$ of length H, a formula ϕ and a time t to some real value, $R_\phi(\vec{s}, t) \in \mathbb{R}$. It measures the maximum perturbation that can be applied to the signal without changing its truth value w.r.t. ϕ.

Similarly to the Boolean semantics, the quantitative semantics of a formula ϕ over a signal $\vec{s}$ is defined recursively over the tree structure of the STL formula, as described below:

$$R_\mu(\vec{s}, t) = g(\vec{s}(t))$$
$$R_{\neg\phi}(\vec{s}, t) = -R_\phi(\vec{s}, t)$$
$$R_{\phi_1 \wedge \phi_2}(\vec{s}, t) = \min(R_{\phi_1}(\vec{s}, t), R_{\phi_2}(\vec{s}, t))$$
$$R_{\phi_1 U_{[a,b]} \phi_2}(\vec{s}, t) = \sup_{t' \in [t+a,t+b]} \left(\min\left(R_{\phi_2}(\vec{s}, t'), \inf_{t'' \in [t,t']} R_{\phi_1}(\vec{s}, t'') \right) \right).$$

The sign of R_ϕ indicates the Boolean satisfaction of signal $\vec{s}$, and in particular, we have

- $R_\phi(\vec{s}, t) > 0 \Rightarrow (\vec{s}, t) \models \phi$;
- $R_\phi(\vec{s}, t) < 0 \Rightarrow (\vec{s}, t) \not\models \phi$.

Given a stochastic process $\mathbf{S} = \{\mathbf{S}(t, \omega), t \in T\}$, an STL requirement ϕ and a state $s_0 \in S$, the robustness over future evolutions of the system starting from s_0 is stochastically distributed according to the conditional distribution

$$\mathbb{P}\left(R_\phi(\vec{s}, 0) \mid \vec{s}(0) = s_0 \right),$$

where $\vec{s}$ is a random signal given by the sequence of random variables $(\mathbf{S}(0, \cdot), \mathbf{S}(1, \cdot), \ldots, \mathbf{S}(H-1, \cdot))$. For time k, $\vec{s}(k) = \mathbf{S}(k, \cdot)$ denotes the random variable corresponding to the state at time k in $\vec{s}$. This conditional distribution captures the distribution of the STL robustness values for trajectories of length H starting in s_0.

We now formulate the quantitative predictive monitoring problem: from any state s_* of the stochastic process, we aim to construct a prediction interval guaranteed to include, with arbitrary probability, the true STL robustness of any (unknown) stochastic trajectory starting at s. A formal statement of the problem is given below.

PROBLEM 1 (QUANTITATIVE PREDICTIVE MONITORING). *Given a discrete-time stochastic process* $\mathbf{S} = \{\mathbf{S}(t, \omega), t \in T\}$ *over a state space* S, *temporal horizon* H, *a significance level* $\alpha \in [0, 1]$ *and an STL formula* ϕ, *derive a monitoring function* I *that maps any state* $s_* \sim \mathbf{S}(\cdot, \cdot)$ *into an interval* $I(s_*)$ *such that*

$$\mathbb{P}\left(R_\phi(\vec{s}, 0) \in I(s_*) \mid \vec{s}(0) = s_* \right) \geq 1 - \alpha.$$

We frame Problem 1 as a conditional quantile regression problem. This boils down to learning for a generic state s_* an upper and a lower quantile of the random variable $R_\phi(\vec{s}, 0)$ induced by $\vec{s}$ conditioned on $\vec{s}(0) = s_*$. We then use these two quantiles to build the output of function I in s_*. To ensure that such an interval is well-calibrated, meaning that the probabilistic guarantees are satisfied theoretically and empirically, we resort to the framework of conformal prediction. These machine-learning techniques are introduced in the next section.

3 BACKGROUND ON CONFORMAL PREDICTION

Consider a generic supervised learning framework where X denotes the input space, Y the target space, and $Z = X \times Y$. Let $\mathcal{Z}$ be the data-generating distribution, i.e., the distribution of the points $(x, y) \in Z$. We assume that the target y of a point $(x, y) \in Z$ is the result of the application of a function $f^* : X \to Y$, typically unknown or very expensive to evaluate. The goal of a supervised learning algorithm is to find a function $f : X \to Y$ that, from a finite set of observations, learns to behave as similarly as possible to f^* over the entire input space. For an input $x \in X$, we denote with t the true target value of x and with $\hat{y}$ the prediction by f, i.e. $\hat{y} = f(x)$. Test inputs, whose unknown true target values we aim to predict, are denoted by x_*. For the sake of clarity, we start by showing conformal prediction approaches for deterministic predictors and then move to present Conformalized Quantile Regression (CQR) [34], an approach to handle the stochastic case.

Conformal predictions (CP) associate measures of reliability with any traditional supervised learning problem, either regression or classification [3, 36]. CP enriches point-wise predictions with *prediction regions with guaranteed validity*.

DEFINITION 1 (PREDICTION REGION). *For significance level* $\alpha \in (0, 1)$ *and test input* x_*, *the* α-*prediction region for* x_*, $\Gamma_*^\alpha \subseteq Y$, *is a set of target values s.t.*

$$\mathbb{P}_{(x_*, y_*) \sim \mathcal{Z}}(y_* \in \Gamma_*^\alpha) \geq 1 - \alpha. \tag{1}$$

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 y', we *exclude* y' from the prediction region only if it is unlikely that y' 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 $z = (x, y)$, measures the deviation between the true value y 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_*^α for a test point x_* by excluding all targets t' whose NCF values are unlikely to follow the NCF distribution of the true targets:

$$\Gamma_*^\alpha = \left\{ y' \in Y \mid Pr_{(x,y) \sim \mathcal{Z}}\left(\delta(x_*, y') \geq \delta(x, y) \right) > \alpha \right\}. \tag{2}$$

This prediction region is guaranteed to contain the true (unknown) value y_* with confidence $1 - \alpha$. The probability term in Eq. (2) is often called the p-value. From a practical viewpoint, the NCF distribution $Pr_{(x,y) \sim \mathcal{Z}}(\delta(x, y))$ 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 CP* [26] and Z_c is referred to as *calibration set*. CP's theoretical guarantees hold under the *exchangeability* assumption (a "relaxed" version of i.i.d.) by which the joint probability of any sample of $\mathcal{Z}$ is invariant to permutations of the sampled points.

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 α, 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 chosen nonconformity function and on the accuracy of the underlying model.

3.1 Conformal Prediction for Regression

In regression problems, we have a continuous target space $Y \subseteq \mathbb{R}^n$. The inductive CP algorithm 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 regressor and distribution.

(1) Draw sample Z' of $\mathcal{Z}$. Split Z' into training set Z_t and calibration set Z_c.
(2) Train regressor f using Z_t. Use f to define an NCF δ.
(3) Construct the calibration distribution by computing, for each $z_i \in Z_c$, the NCF score $\beta_i = \delta(z_i)$.
(4) Identify the critical value $\beta_{(\alpha)}$ of the calibration distribution, i.e. its empirical $(1 - \alpha)$-quantile, or the $\lfloor \alpha \cdot (|Z_c| + 1) \rfloor$-th largest calibration score.
(5) Return the prediction region

$$\Gamma_*^\alpha = f(x_*) \pm \beta_{(\alpha)}. \tag{3}$$

Notice that such prediction intervals have the same width ($\beta_{(\alpha)}$) for all inputs. A natural NCF in regression is the norm of the difference between the real and the predicted target value, i.e., $\delta(x) = ||y - f(x)||$.

Predictive 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. The size of the prediction region is determined by the chosen significance level α. Specifically, from Eq. (3) we can see that, for levels $\alpha_1 \geq \alpha_2$, the corresponding prediction regions are such that $\Gamma^{\alpha_1} \subseteq \Gamma^{\alpha_2}$, as a smaller α yields a larger critical value $\beta_{(\alpha)}$.

3.2 Conformalized Quantile Regression

We now switch to the stochastic setting. Let us consider a probabilistic function mapping an input $x \in X$ into a distribution over the target space Y.

Quantile Regression. The aim of conditional Quantile Regression (QR) is to estimate a given quantile of such a distribution over Y conditional on an input $x \in X$. Let $F(y'|x = x') := \mathbb{P}(y \leq y'|x = x')$ be the conditional distribution function of y given x. Then, the α-th conditional quantile function is defined as

$$q_\alpha := \inf\{y' \in \mathbb{R} \mid F(y'|x = x') \geq \alpha\}. \tag{4}$$

Given a significance level α, we consider lower and upper quantiles w.r.t. $\alpha_{lo} = \alpha/2$ and $\alpha_{hi} = 1 - \alpha/2$, respectively. We define the desired prediction interval as $PI(x) := [q_{\alpha_{lo}}(x), q_{\alpha_{hi}}(x)]$. By construction, this interval satisfies

$$\mathbb{P}(y \in PI(x')|x = x') \geq 1 - \alpha. \tag{5}$$

Since the prediction interval is conditional on the input, the length of the interval is not fixed in general and changes at different values of x. QR infers such prediction interval from the data. In particular, estimating the quantiles can be expressed as approximating the

quantile function and thus it can be framed as an optimization problem. In a nutshell, the idea is to propose a parametric function $f(x; \theta)$ as a candidate approximator for $q_\alpha(x)$ and then optimize over its parameters θ so that it closely resembles the quantile function. The optimization problem can be formally expressed as finding the optimal parameters $\hat{\theta}$ such that

$$\hat{\theta} = \arg\min_\theta \left[\frac{1}{n} \sum_i \mathcal{L}_\alpha(Y_i, f(X_i; \theta)) + \mathcal{G}(\theta) \right],$$

where $\mathcal{L}_\alpha$ is the check or pinball loss, defined as

$$\mathcal{L}_\alpha(y, \hat{y}) = \alpha \max(y - \hat{y}, 0) + (1 - \alpha) \max(\hat{y} - y, 0), \tag{6}$$

and $\mathcal{G}$ is an optional regularization term. In this work, we use deep neural networks (NN) as candidate parametric approximators $f(\cdot, \theta)$ of the quantile function and train them using Eq. (6) as loss function. Once trained, the output of $f(x; \hat{\theta})$ is a prediction $\hat{q}_\alpha(x)$ of the conditional α-th quantile. In general, each quantile requires the training of a different neural network. However, one could also train a single multi-output NN that learns to approximate multiple quantile functions at the same time. The multi-output objective function is obtained by averaging over the respective losses. In this work, we decide to simultaneously learn $q_{\alpha_{lo}}$, $q_{0.5}$ (the median) and $q_{\alpha_{hi}}$, where for a choice of $\alpha \in (0, 0.5)$, $\alpha_{lo} = \alpha$ and $\alpha_{hi} = 1 - \alpha$. The loss then becomes

$$\mathcal{L}_\alpha(y, \hat{y}) = \frac{1}{3} \cdot (\mathcal{L}_{\alpha_{lo}}(y, \hat{y}) + \mathcal{L}_{0.5}(y, \hat{y}) + \mathcal{L}_{\alpha_{hi}}(y, \hat{y})).$$

In general, a single NN could simultaneously learn even a larger number of quantiles.

Conformalized Quantile Regression. The goal of Conformalized Quantile Regression (CQR) is to adjust the QR prediction interval so that it is guaranteed to contain the $(1 - \alpha)$ mass of probability, i.e. to satisfy (5). As for CP, we divide the dataset Z' in a training set Z_t and a calibration set Z_c. We train the QR $f(\cdot; \theta)$ over Z_t and on Z_c we compute the nonconformity scores as

$$E_i := \max\{\hat{q}_{\alpha_{lo}}(x_i) - y_i, y_i - \hat{q}_{\alpha_{hi}}(x_i) \mid (x_i, y_i) \in Z_c\}. \tag{7}$$

In our notation, $\hat{q}_{\alpha_{lo}}(x)$ and $\hat{q}_{\alpha_{lo}}(x)$ denotes the two outputs of $f(x; \hat{\theta})$. The conformalized prediction interval is thus defined as

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

where τ is the $(1 - \alpha)(1 + 1/|Z_c|)$-th empirical quantile of $\{E_i : i \in Z_c\}$.

In the following, we will abbreviate with *PI* a (non-calibrated) QR prediction interval and with *CPI* a (calibrated) conformalized prediction interval.

REMARK 1. *This nonconformity function, and thus τ, 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), meaning $\mathbb{P}_{(x_*, y_*) \sim \mathcal{Z}}(y_* \in CPI(x_*)) \geq 1 - \alpha$.*

REMARK 2. *In principle, one could use traditional CP for regression (see Section 3.1) to obtain valid prediction intervals for the stochastic case. The main advantage of CQR is that it produces CPIs that are adaptive to heteroscedasticity, i.e., they account for the fact that the variability in the output may be affected by the value of the input.*
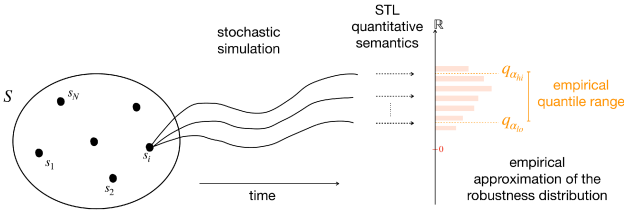
Figure 1: Diagram illustrating the generation of the dataset.

On the other hand, intervals produced by CP for regression have fixed sizes and hence, do not account for heteroscedasticity. Moreover, the PI of CP for regression would be relative to the conditional mean of the STL robustness, not to the conditional quantile range that we are interested in.

4 QUANTITATIVE PREDICTIVE MONITORING

We present a method to solve Problem 1[3]. For a given a discrete-time stochastic process $S = \{S(t, \omega), t \in T\}$ over state space S and a state $s \sim S(k, \cdot)$ at time $k \in T$, the stochastic evolution (bounded by horizon H) of the system starting at s can be described by the conditional distribution

$$\mathbb{P}(\vec{s} \mid \vec{s}(k) = s),$$

where $\vec{s} = (\vec{s}(k), \ldots, \vec{s}(k + H)) \in S^H$ is the random trajectory of length H starting at time t_k, $\vec{s}(i) = S(i, \cdot)$ for any $i \in T$.

The quantitative STL semantics inherits the stochasticity from the dynamics of the system. For an STL property ϕ, we denote with $\mathcal{R}_\phi(s)$ the random variable denoting the STL robustness value relative to ϕ of trajectories starting from $s \in S$ at time t_k, i.e.,

$$\mathcal{R}_\phi(s) \sim \mathbb{P}(R_\phi(\vec{s}, 0) \mid \vec{s}(k) = s).$$

Our conformal solution to the QPM aims at finding, for each state $s \in S$ of the system, a prediction region that covers a certain probability mass of the STL robustness distribution $\mathcal{R}_\phi(s)$, see Problem 1. In simpler terms, we aim at monitoring how safe the system is relative to the unknown and stochastic future evolution from its current state s. In this way, one can intervene preemptively on the system in order to prevent any failures. However, the distribution of $\mathcal{R}_\phi(s)$ is impossible to compute exactly and in an efficient manner. Thus, we resort to Conformalized Quantile Regression (CQR), introduced in Section 3.2, to compute a prediction interval that, for each state s, is guaranteed to cover a desired level $(1 - \alpha)$ of the probability mass for the conditional distribution of robustness values $\mathcal{R}_\phi(s)$.

In short, our solution consists of four steps, detailed below: *dataset generation*, *QR training*, *residuals computation*, and *inference*. Note that only the last step, which is by far the quickest, is performed online, the others are performed offline and hence, do not affect runtime performance.

Dataset generation. In this step, we collect data for training the QR function and constructing the calibration set. To do so, we perform Monte-Carlo simulations of the process in order to obtain an empirical approximation of $\mathcal{R}_\phi(s)$.
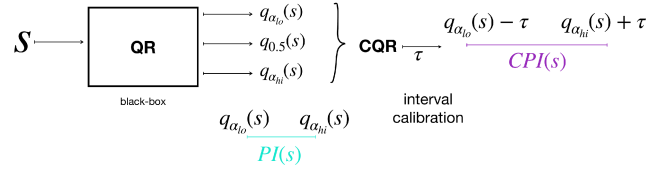

Figure 2: Overview of the conformal quantitative predictive monitoring technique.

In particular, we randomly sample N states $s_1, \ldots, s_N \sim S(\cdot, \cdot)$. Then, for each state s_i, we simulate M trajectories of length H, $\vec{s}_i^1, \ldots, \vec{s}_i^M$ where $\vec{s}_i^j$ is a realization of $\mathbb{P}(\vec{s} \mid \vec{s}(k) = s_i)$, and compute the robustness value $R_\phi(\vec{s}_i^j, 0)$ of each of these trajectories. We note that $\{R_\phi(\vec{s}_i^j)\}_{j=1}^M$ is an empirical approximation of $\mathcal{R}_\phi(s_i)$.

Moreover, for a choice of $\alpha \in (0, 1)$, we derive empirical quantiles $q_{\alpha/2}^i$ and $q_{1-\alpha/2}^i$ from samples $R_\phi(\vec{s}_i^1), \ldots, R_\phi(\vec{s}_i^M)$ and use these to label the state into one of *safe*, *unsafe*, or *risky*: if the trajectories starting from s_i satisfy ϕ with probability above $1 - \alpha/2$ (i.e., $q_{\alpha/2}^i > 0$), then we label s_i as $\ell_i = +1$ (safe); if the probability of satisfaction rather than violation here is below $\alpha/2$ (i.e., $q_{1-\alpha/2}^i < 0$), then we label s_i as $\ell_i = -1$ (unsafe); otherwise, we use label $\ell_i = 0$ (risky).

The dataset is thus defined as

$$Z^\phi = \left\{ \left(s_i, (R_\phi(\vec{s}_i^1), \ldots, R_\phi(\vec{s}_i^M)), \ell_i \right), i = 1, \ldots, N \right\}. \quad (8)$$

Fig. 1 shows an overview of the steps needed to generate the dataset.

The generation of the test set Z_{test}^ϕ is very similar to that of Z^ϕ. 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 of $\mathcal{R}_\phi(s)$, which we use as the ground-truth baseline in our experimental evaluation[4].

QR training and residuals computation. We divide the dataset Z^ϕ into a training set Z_t^ϕ and a calibration set Z_c^ϕ. We then use Z_t^ϕ to train a QR that learns how to map states s into three quantiles

$$f(s; \hat{\theta}) = \{\hat{q}_{\alpha_{lo}}(s), \hat{q}_{0.5}(s), \hat{q}_{\alpha_{hi}}(s)\},$$

where $\alpha_{lo} = \alpha/2$ and $\alpha_{hi} = 1 - \alpha/2$. In order to better reconstruct the shape of the target distribution, we also predict the median quantile, $q_{0.5}(s)$.

We then apply CQR, i.e. we compute the residuals of the QR over Z_c^ϕ, as described in Equation 7 of the previous section, and find the critical residual value τ.

Inference. For a test state s_*, this step involves predicting the relevant quantiles, $\hat{q}_{\alpha_{lo}}(s_*)$ and $\hat{q}_{\alpha_{hi}}(s_*)$, using the QR predictor and correct the resulting interval using the critical residual value τ. The calibrated interval returned at inference time becomes

$$CPI(s_*) := [\hat{q}_{\alpha_{lo}}(s_*) - \tau, \hat{q}_{\alpha_{hi}}(s_*) + \tau] \subseteq \mathbb{R}$$

and it is guaranteed to contain the true unknown robustness value with probability $(1 - \alpha)$.

---

[3]When the context is clear, we'll use the term "Quantitative Predictive Monitoring" to refer to both the problem and the solution method.

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

4.1 Composing monitors for composite properties

The above-described QPM approach is end-to-end, in that it directly predicts the robustness of a fixed property ϕ. On one hand, this is advantageous in that the monitor is tailored, and hence, highly accurate at predicting R_ϕ but on the other hand, it lacks flexibility. Thus, it becomes natural to ask oneself whether and how two predictive monitors trained for distinct properties ϕ_1 and ϕ_2 can be combined in a compositional manner to construct a monitor for a Boolean combination of the two properties, $\phi_1 \wedge \phi_2$ or $\phi_1 \vee \phi_2$, in such a way that the resulting monitor meets the desired probabilistic guarantees and doesn't require additional data collection or training.

Here we propose two solutions to combine monitors for logical conjunction and disjunction. Handling the negation of a property, $\neg \phi$, is rather simple as we can take the interval with opposite signs. Therefore, we cover the basic Boolean logic. In what follows we will use a superscript ϕ_i with $i = 1, 2$ to distinguish predictors, datasets, critical values, and prediction regions of the two properties.

1. Union of intervals. The first, simple, solution considers the *union* of the two property-specific prediction intervals. Mathematically,

$$CPI^{\phi_{1,2}} := CPI^{\phi_1} \cup CPI^{\phi_2}.$$

We now prove that $CPI^{\phi_{1,2}}$ has a guaranteed coverage of $(1 - \alpha)$, the same of CPI^{ϕ_1} and CPI^{ϕ_2}, for both $\phi_1 \wedge \phi_2$ and $\phi_1 \vee \phi_2$.

PROPOSITION 1. *Given two properties ϕ_1 and ϕ_2, let CPI^{ϕ_i} be the monitor that maps each state $s \in S$ into a prediction interval for $R_{\phi_i}(s)$ with coverage $1 - \alpha$. Then, the interval $CPI^{\phi_{12}} := CPI^{\phi_1} \cup CPI^{\phi_2}$ is a valid prediction interval for both $R_{\phi_1 \wedge \phi_2}$ and $R_{\phi_1 \vee \phi_2}$, i.e.,*

$$\mathbb{P}\left(R_{\phi_1 \wedge \phi_2}(s) \in CPI^{\phi_{12}}(s)\right) \geq 1 - \alpha \tag{9}$$

and

$$\mathbb{P}\left(R_{\phi_1 \vee \phi_2}(s) \in CPI^{\phi_{12}}(s)\right) \geq 1 - \alpha \tag{10}$$

for every state $s \in S$.

The proof is provided in the Supplementary Material[5].

2. Calibrated interval arithmetic. The second solution leverages the fact that CP (and CQR) can provide valid intervals on top of any predictor. Hence, we build a monitor for the composite property by first combining the predictors of the individual properties and then re-calibrating such obtained predictor. In particular, for $\phi_1 \wedge \phi_2$ we define for a state $s \in S$ the prediction interval as

$$PI^{\phi_1 \wedge \phi_2}(s) := \left[\min\left(q_{\alpha_{lo}}^{\phi_1}(s), q_{\alpha_{lo}}^{\phi_2}(s)\right), \min\left(q_{\alpha_{hi}}^{\phi_1}(s), q_{\alpha_{hi}}^{\phi_2}(s)\right) \right].$$

See Fig. 3 for a schematic visualization. We then compute the residuals of $PI^{\phi_1 \wedge \phi_2}$ over the calibration set, obtaining the critical value $\tau^{\phi_1 \wedge \phi_2}$ and the conformalized prediction interval

$$CPI^{\phi_1 \wedge \phi_2}(s) := \Big[\min\left(q_{\alpha_{lo}}^{\phi_1}(s), q_{\alpha_{lo}}^{\phi_2}(s)\right) - \tau^{\phi_1 \wedge \phi_2},$$
$$\min\left(q_{\alpha_{hi}}^{\phi_1}(s), q_{\alpha_{hi}}^{\phi_2}(s)\right) + \tau^{\phi_1 \wedge \phi_2} \Big],$$

which has a guaranteed coverage of $1 - \alpha$.

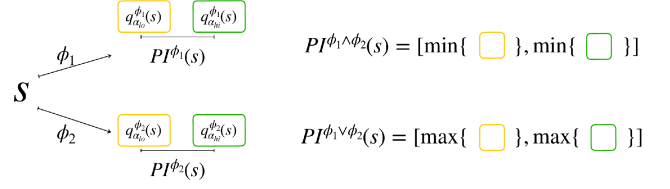[5]Supplementary Material is available online in [11].

$$PI^{\phi_1 \wedge \phi_2}(s) = [\min\{\ \square\ \}, \min\{\ \square\ \}]$$

$$PI^{\phi_1 \vee \phi_2}(s) = [\max\{\ \square\ \}, \max\{\ \square\ \}]$$

Figure 3: Overview of the combination of the conformal quantitative predictive monitors for two different properties ϕ_1 and ϕ_2.

The procedure is the same for property $\phi_1 \vee \phi_2$, with the only difference that the prediction interval is given by

$$PI^{\phi_1 \vee \phi_2}(s) := \left[\max\left(q_{\alpha_{lo}}^{\phi_1}(s), q_{\alpha_{lo}}^{\phi_2}(s)\right), \max\left(q_{\alpha_{hi}}^{\phi_1}(s), q_{\alpha_{hi}}^{\phi_2}(s)\right) \right].$$

Solution (2) is expected to produce tighter intervals compared to solution (1)[6]. On the other hand, solution (2) requires the generation of the respective calibration set, $Z_c^{\phi_1 \wedge \phi_2}$ or $Z_c^{\phi_1 \vee \phi_2}$, which solution (1) does not need.

We remark that such compositionality is defined only for Boolean connectives and not for temporal operators.

5 EXPERIMENTAL RESULTS

We experimentally evaluate the proposed QPM over a benchmark of four discrete-time stochastic processes with varying degrees of complexity.

5.1 Case Studies

The first two case studies are selected from the stochastic hybrid benchmarks presented in the ARCH-COMP competitions [1]. We then consider two stochastic hybrid systems with modular nature where the complexity can arbitrarily grow, making them suitable to test the scalability of the proposed solution. See the Supplementary Material (SM) for more details about the case studies presented below.

Automated Anaesthesia Delivery. (AAD) [1] is a 3-dimensional discrete-time stochastic process with state $s(t) = (q(t), v(t))$ evolving according to linear dynamics with a controller and a Gaussian disturbance. The properties to be monitored are: $\phi_1 = G_{[0,t_H]}(v \in Safe)$ and $\phi_2 = F_{[0,t_H]}(v_1 < v_2)$, where $Safe = [1, 6; 0, 10; 0, 10]$, $H = 60min$, $\Delta t = 20s$. The controller is defined as follows: $q = 7$ if $v_1 < 3.5$ and $q = 3.5$ if $v_1 \geq 3.5$.

Heated Tank. (HT) is a stochastic hybrid system composed of a tank containing a liquid whose level is influenced by two inflow pumps P_1 and P_2 and an outflow valve W, all managed by a controller C (see Fig. 4). The liquid should absorb and transport the heat from a heat source. The discrete state space is five-dimensional. The continuous state space is two-dimensional and composed of a state $v(t) = (v_{height}(t), v_{temp}(t))$ representing the liquid height and the liquid temperature. The general model adopted for this heated tank system is that of piece-wise deterministic Markov processes, where the continuous part evolves according to switching differential equations. The switching is managed by the controller and by two exponentially distributed failure events. The property to be monitored is:

[6]In particular, the PI of solution (2) is strictly tighter than the CPI of solution (1).
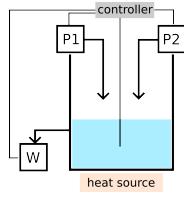
Figure 4: Scheme of the heated tank system whose components can undergo exponentially distributed failures.

the liquid level is always in the safe range $Safe = [H_{dryout}, H_{overflow}]$ and the temperature is never too high and it can be expressed by the STL property $\phi = G_{[t_0, t_H]}\big((v_{height} \in Safe) \wedge (v_{temp} \leq T_{overheat})\big)$.

Multi-Room Heating. (MRH) [2, 16, 25]: discrete-time stochastic hybrid system modeling the temperature evolution in a building with h rooms. In each room is located a heater that switches between *on* and *off* depending on the current temperature in the room. The state of the system is hybrid $S = Q \times V$: the discrete state represents the status of the h heaters, $Q = \{on, off\}^h$, and the continuous state represents the temperatures in the h rooms, $V = \mathbb{R}^h$. The average temperature evolves according to a stochastic difference equation that depends on the rooms' layout (see Fig. 5 (left)). The room-specific requirement to be monitored is: the temperature v_i of room i always stays in a desirable range $[V_i^{lb}, V_i^{ub}]$ and it can be expressed by the STL property $\phi_i = G_{[0, t_H]}(V_i^{lb} \leq v_i \leq V_i^{ub})$.

Gene Regulatory Network. (GRN) [4, 20]: discrete-time stochastic hybrid system modeling a genetic regulatory network composed of h genes that produce respectively h proteins that repress each other in a cyclic fashion (see Fig. 5 (right)). Each gene could either be *on*, actively producing its protein, or *off*, expression is deactivated when a repressing protein binds to the gene. The state is thus hybrid $S = Q \times V$: the state of the genes is discrete, $Q = \{on, off\}^h$, whereas the proteins count is continuous, $V = \mathbb{N}^h$. Protein counts change deterministically according to a linear differential equation, whereas transitions between modes follow a Markov jump process. The gene-specific requirement to monitor is: the count v_i of protein i stabilize under a certain threshold V_i^{ub} and it can be expressed by the STL property $\phi_i = G_{[t_{H/2}, t_H]}(v_i \leq V_i^{ub})$.
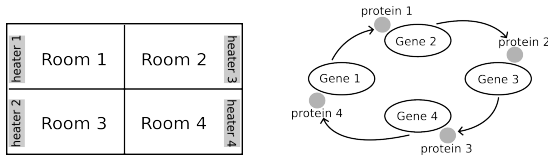

Figure 5: Schematic visualization of the layout of the two modular case studies: multi-room heating (left) and gene regulatory network (right).

5.2 Evaluation metrics

We want our method to be capable of working at runtime in safety-critical applications, which translates into the need for high reliability and high computational efficiency in producing the predictive intervals and in calibrating them. We emphasize that the time required to train the quantile regressor and to compute the calibration

score τ does not affect its runtime efficiency, as it is performed in advance (offline) only once. Once trained, the time needed to have a prediction interval for the current state is simply the time needed to evaluate a neural network with a rather simple architecture. This time is almost negligible (in the order of microseconds on GPU). In addition, we do not want an over-conservative predictor as an unnecessarily large interval would reduce the effectiveness of our QPM. Keeping that in mind, we introduce some relevant metrics to evaluate the performances of our QPM.

Accuracy. We compare the sign of the predicted interval with the sign ℓ of the empirical quantile range over the test set. In simpler words, the latter can be interpreted as a classification accuracy over the sign of the prediction intervals and the dataset labels ℓ. If both the real and the estimated interval are risky, i.e. if they both straddle zero, then the prediction is correct. If the signs of the two intervals are opposite the prediction is *wrong*. On the other hand, if the state is either safe or unsafe and it is predicted as risky, we label the prediction as *uncertain*. Moreover, we analyze the percentage of *false positive* (FP) errors, the most dangerous ones as they compromise the safety of the system. A false positive occurs when the state is either unsafe or risky but it is predicted to be safe.

Coverage and efficiency. We experimentally check that the guaranteed validity of CQR is empirically met in the test evaluation. The efficiency represents the average width of the prediction intervals over the test set. In general, the larger the prediction interval the more conservative the CQR predictions. If the prediction intervals over the robustness values are always very large we have little information about the satisfaction of the property ϕ. However, the predictive efficiency must be compared with the width of the empirical quantile range (EQR), i.e. the interval that contains $(1 - \alpha)$ of the simulated robustness values. We can thus measure the conservativeness as the difference in width between the predicted efficiency and the EQR width. We also compare the coverage and the efficiency of QR against those of CQR.

Overall, we aim at reaching high accuracy in the reconstruction of the robustness distribution, which translates into a high percentage of correct predictions, i.e. intervals with agreeing signs. However, it is even more important to reduce the number of wrong predictions, especially false positives as they can compromise the safety of the system. We rather have more uncertain predictions but avoid erroneous ones. We check if the statistical guarantees about the coverage are met empirically by measuring the robustness of how many trajectories fall inside the predicted interval. Let us remark that the coverage and the percentage of correct predictions denote two different quantities and that the efficiency is not a percentage but the average width of the prediction interval measuring how conservative the predictor is compared to the empirical efficiency (EQR width).

5.3 Experiments

The workflow can be divided into steps: (1) define the model of the stochastic process, (2) generate the synthetic datasets Z^ϕ (simulate and compute STL robustness), (3) train the QR, (4) compute the calibration score τ (obtaining CQR prediction intervals), (5) evaluate both the QR and the CQR on a test set Z_{test}^ϕ.

Experimental settings. The entire pipeline is implemented in Python. The quantitative semantics of pcheck library[7] is used to quantify the satisfaction of a certain formula for a specific trajectory. The neural networks of the quantile regressors are trained with PyTorch [27]. The experiments were conducted on a shared virtual machine with a 32-Core Processor, 64GB of RAM and an NVidia A100 GPU with 20GB, and 8 VCPU. Our implementation is available at https://github.com/ailab-units/CQR_Quantitative_NPM.

Datasets. For every model, we generate a dataset Z^ϕ with size that increases with the dimensionality of the model. In general, the chosen datasets are not too large. In particular, being n the model's dimension, we sample $N_{train} = n \cdot 1000$ states for the training set, $N_{cal} = n \cdot 500$ states for the calibration set and $N_{test} = n \cdot 100$ states for the test set. From each state, we simulate $M = 50$ trajectories for the training and for the calibration set and $M_{test} = 500$ trajectories for the test set. Data are scaled to the interval $[-1, 1]$ to avoid sensitivity to different scales.

Training details. In all the experiments we use the same neural network architecture. The network is composed of 3 hidden layers with 20 neurons each. We use the Adam optimizer with a learning rate of 0.0005 and a dropout rate of 0.1 applied on each layer. We use a LeakyReLU activation function with slope 0.01 and train the network for 500 epochs over batches of size 512.

Offline costs. The offline overhead consists of the time needed to generate the dataset and the time needed to train the neural network. The cost of the latter depends on the chosen architecture and on the dataset size. In our experiments, it ranges from 30 minutes to 6 hours for larger models. On the other hand, the time needed to generate the dataset depends on the time needed to simulate the trajectories and label them. These quantities are influenced by many factors: the chosen horizon H, the complexity of the STL property, and the complexity of the stochastic dynamics. For instance, the time needed to generate a dataset of $2000 \times 50 = 100K =$ observations for the *GRN*2 model is double (1 hour) compared to the *MRH*2 model (30 minutes), even if they have the same dimensionality. The dataset generation time explodes easily around 20 hour for *GRN*6 and *MRH*8.

Performance evaluation. We choose in all the experiments $\alpha = 0.1$, so that $\alpha_{lo} = 0.05$ and $\alpha_{hi} = 0.95$. The results are presented with the following structure. For each case study and for each property we train the QR and then recalibrate the predicted interval by means of CQR. For the modular case studies, i.e. MRH and GRN, we have a property regarding each individual component, i.e. for each room and for each gene respectively. To analyze how the performance scales with the dimensionality of the underlying system, we test our QPM over different configurations of the modular system. In particular, we consider a multi-room heating (MRH) system with respectively 2, 4, and 8 rooms and a gene regulatory net (GRN) system with 2, 4, and 6 genes.

5.3.1 Results. Fig. 6 (additional figures in SM) shows a comparison of the predicted intervals, both PI (cyan) and CPI (purple), over 40 states randomly sampled from the test set and compare them with the empirical samples (orange) collected in the test set. Intervals

---

[7]https://github.com/simonesilvetti/pcheck

AAD	correct	uncertain	wrong	FP	coverage	efficiency	EQR width
QR	83.00	14.50	2.50	2.50	87.86	0.1342	0.0497
CQR	83.00	14.50	2.50	2.50	87.98	0.1349	

AAD-F	correct	uncertain	wrong	FP	coverage	efficiency	EQR width
QR	95.50	3.50	1.00	1.00	89.92	0.4333	0.1942
CQR	95.50	3.50	1.00	1.00	91.28	0.4495	

HT	correct	uncertain	wrong	FP	coverage	efficiency	EQR width
QR	99.00	1.00	0.00	0.00	88.70	0.2778	0.2583
CQR	99.00	1.00	0.00	0.00	90.47	0.2812	

Table 1: Correct, uncertain, wrong and coverage represent percentage values (%). *EQR width* denotes the average width of the empirical quantile range over the test set.

MRH2	correct	uncertain	wrong	FP	coverage	efficiency	EQR width
Room 1							
QR	99.50	0.00	0.50	0.00	89.25	0.3221	0.3167
CQR	99.50	0.00	0.50	0.00	90.19	0.3313	
Room 2							
QR	86.50	11.00	2.50	2.50	90.94	0.2568	0.1644
CQR	86.50	11.00	2.50	2.50	91.58	0.2591	
Room 1 ∧ Room 2							
QR	99.50	0.00	0.50	0.00	89.40	0.3137	0.3026
CQR	99.50	0.00	0.50	0.00	90.33	0.3225	
MIN	99.50	0.00	0.50	0.0	90.64	0.3230	
UNION	-	-	-	-	93.17	0.3169	

MRH4	correct	uncertain	wrong	FP	coverage	efficiency	EQR width
Room 1							
QR	100.00	0.00	0.00	0.00	89.71	0.2945	0.2788
CQR	100.00	0.00	0.00	0.00	90.03	0.2967	
Room 2							
QR	91.00	6.75	2.25	2.25	90.10	0.2129	0.1540
CQR	91.00	6.75	2.25	2.25	90.31	0.2151	
Room 3							
QR	94.50	4.75	0.75	0.75	91.21	0.2170	0.1707
CQR	94.50	4.25	1.25	1.00	90.40	0.2117	
Room 4							
QR	100.00	0.00	0.00	0.00	90.70	0.3019	0.2766
CQR	100.00	0.00	0.00	0.00	90.14	0.2970	
Room 1 ∧ Room 2							
QR	100.00	0.00	0.00	0.00	90.19	0.2820	0.2547
CQR	100.00	0.00	0.00	0.00	90.70	0.2861	
MIN	100.00	0.00	0.00	0.00	90.13	0.2747	
UNION	-	-	-	-	93.50	0.2815	
Room 1 ∧ Room 3							
QR	100.00	0.00	0.00	0.00	89.57	0.2818	0.2584
CQR	100.00	0.00	0.00	0.00	90.04	0.2860	
MIN	100.00	0.00	0.00	0.00	90.17	0.2764	
UNION	-	-	-	-	93.96	0.2850	
Room 1 ∧ Room 4							
QR	100.00	0.00	0.00	0.00	90.73	0.3016	0.2760
CQR	100.00	0.00	0.00	0.00	90.16	0.2965	
MIN	100.00	0.00	0.00	0.00	89.59	0.2938	
UNION	-	-	-	-	91.79	0.3336	

Table 2: Multi room heating results with either two or four rooms. Correct, uncertain, wrong and coverage represent percentage values (%). *EQR width* denotes the average width of the empirical quantile range over the test set.

above zero mean that the state is safe, intervals below mean that the state is unsafe, whereas intervals that straddle zero mean that the state is risky. Empirical samples that lie outside the EQR are denoted in a lighter orange.
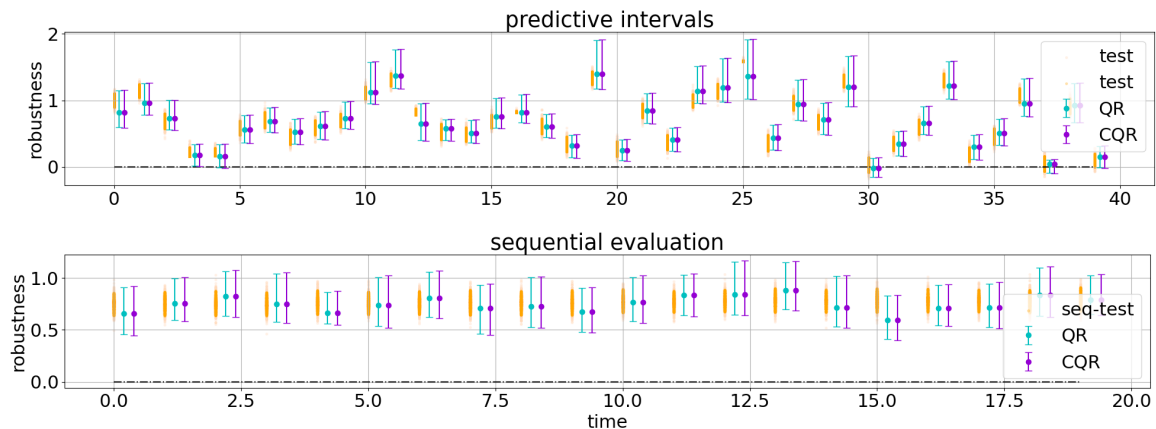
Figure 6: Visualization of prediction intervals – cyan for QR and blue for CQR – over 40 randomly selected states (top) and for a runtime evaluation (bottom) for the automated anesthesia delivery system (AAD-F) w.r.t. property ϕ_2 (eventually operator).

GRN2	correct	uncertain	wrong	FP	coverage	efficiency	EQR width
Gene 1							
QR	90.00	6.50	3.50	3.50	88.95	0.8551	0.7998
CQR	89.50	7.00	3.50	3.50	89.94	0.8744	
Gene 2							
QR	86.00	11.50	2.50	1.00	89.55	0.5773	0.4229
CQR	86.00	11.50	2.50	1.00	89.59	0.5780	
Gene 1 ∧ Gene 2							
QR	81.50	16.50	2.00	0.00	88.18	0.6332	0.5266
CQR	81.50	16.50	2.00	0.00	88.25	0.6345	
MIN	81.50	16.00	2.50	2.00	89.02	0.6432	
UNION	-	-	-	-	93.88	0.6308	

Table 3: Gene regulatory network results. Correct, uncertain, wrong ∧ coverage represent percentage values (%). *EQR width* denotes the average width of the empirical quantile range over the test set.

GRN4	correct	uncertain	wrong	FP	coverage	efficiency	EQR width
Gene 1							
QR	93.25	4.25	2.50	2.50	89.69	0.5713	0.5062
CQR	93.00	4.50	2.50	2.50	90.00	0.5756	
Gene 2							
QR	90.75	6.50	2.75	2.75	89.49	0.5232	0.4359
CQR	90.75	6.50	2.75	2.75	89.09	0.5187	
Gene 3							
QR	100.00	0.00	0.00	0.00	89.04	0.3759	0.2620
CQR	100.00	0.00	0.00	0.00	90.74	0.3942	
Gene 4							
QR	91.75	6.25	2.00	1.25	90.49	0.3877	0.2820
CQR	91.75	6.25	2.00	1.25	90.46	0.3873	
Gene 1 ∧ Gene 2							
QR	85.50	10.75	3.75	3.50	89.13	0.5299	0.4723
CQR	85.50	11.00	3.50	3.25	89.43	0.5346	
MIN	86.50	9.00	4.50	4.00	89.23	0.5196	
UNION	-	-	-	-	91.44	0.5884	
Gene 1 ∧ Gene 3							
QR	92.75	4.25	3.00	3.00	89.22	0.5482	0.4763
CQR	92.75	4.25	3.00	3.00	89.30	0.5439	
MIN	93.25	4.25	2.50	2.50	89.30	0.5334	
UNION	-	-	-	-	91.40	0.5384	
Gene 1 ∧ Gene 4							
QR	86.00	10.00	4.00	2.50	88.22	0.4857	0.4133
CQR	85.25	10.75	4.00	2.75	89.06	0.4943	
MIN	85.50	10.75	3.75	3.25	89.37	0.4708	
UNION	-	-	-	-	90.60	0.5289	

Table 4: Caption as in Table 3.

Tables 1-4 (additional tables in SM) provide a detailed summary of the performance and of the generalization capabilities of our QPM method. In particular, they compare the accuracy (rate of correct, uncertain, wrong and false positive predictions), the coverage and the efficiency of our solution over all the different case studies.

We observe that the percentage of wrong and FP predictions is extremely low in all the experimental configurations (always lower than 4%). The percentage of uncertain prediction interval is, in general, very low (lower than 10%). Some case studies make an exception and present a high percentage of uncertain predictions (around 30% of the test predictions). This happens because most of the empirical quantile ranges in the test set have one of the two extremes very close to zero (see SM for an intuitive visualization of the problem). Therefore, even if numerical results may seem to suggest that we have a poor reconstruction, in reality, we are successfully capturing the fact that most of the states are on the verge of violating the property.

From Tables 1- 4 (and SM) we can also experimentally observe that CQR always meets the statistical guarantees: when QR is over-conservative CQR refines the predicted interval (negative τ), whereas when QR is over-confident CQR enlarge the predicted interval (positive τ). That is, the conformal recalibration adjusts the width of the interval w.r.t. desired confidence level, reducing the number of over-conservative predictions. The efficiencies of the PI and CPI predicted intervals are compared to the width of the empirical quantile range computed over the test set (*EQR width* column). CQR produces intervals that are on average 34% more conservative than the EQR, with a minimum of 0.3% and a maximum of 170% (for the *AAD*). Recall that the efficiency is the width of the interval and so lower values are in general more desirable.

The modular experiments, $MRH2, MRH4, MRH8, GRN2, GRN4$ and $GRN6$, allow us to analyse the compositionality of QPM over the conjunction of properties (each property is room- or gene-specific). Due to space limitations, we focus only on the conjunction (∧ operator). The same analysis can be done for the disjunction (∨

operator). In Tables 1-4 (and SM), we compare the performances of four different approaches: QR and CQR trained over the training set $Z_t^{\phi_1 \wedge \phi_2}$ and calibrated over $Z_c^{\phi_1 \wedge \phi_2}$, the minimum w.r.t.interval arithmetic (MIN in tables) of the PI intervals calibrated once again over $Z_c^{\phi_1 \wedge \phi_2}$ (solution (2) presented in Sec. 4.1), and finally the union (UNION in tables) of the CPI intervals (solution (1) presented in Sec. 4.1). Results confirm our intuition that the UNION approach produces over-conservative predictions, i.e. intervals with high coverage and high efficiency. The MIN approach, on the other hand, provides better coverage (less conservative) and better efficiency (comparable to that of CQR). In particular, the CQR approach produces intervals that are on average 12.50% more conservative than the EQR, MIN's intervals are on average 10% more conservative than the EQR, and UNION's intervals are on average 19% more conservative, i.e. the intervals are wider.

Sequential data. All the results presented so far consider a dataset of states independently sampled from a distribution $\mathcal{S}$. By doing so, we aim at quantifying the generalization capabilities of our QPM. However, we are also interested in applying our QPM at runtime to systems that are evolving in time. States will thus have a temporal dependency, meaning that we lose the exchangeability requirement behind the theoretical validity of the CPI. Fig. 6 (bottom) – additional figures in SM –shows the performance of QPM applied to such sequential data. In particular, we randomly sampled a trajectory of length 20 and applied the QPM to each state in the trajectory. We observe that QPM well captures the evolution of the robustness values but it tends to be either slightly over-confident or slightly over-conservative. This behaviour is typical of CP when the data-generating distribution at test time differs from the one used during the training and the calibration phase.

5.4 Discussion

In summary, we can see that the proposed QPM is effective in tackling the scalability issues surrounding the predictive problem for stochastic processes. The solution is inevitably approximate, as an exact solution is unfeasible even for extremely simple stochastic processes. In this sense, the statistical guarantees our method provides play a crucial role when dealing with safety-critical applications. We show that QPM performs reasonably well also at runtime with sequentially generated data. We plan to further investigate the theoretical guarantees of QPM when applied to sequential and non-exchangeable settings. We stress that the proposed methodology is not strictly related to the chosen measure of robustness. For instance, one could decide to focus on a notion of time robustness [21] and apply the same methodology described here for spatial robustness.

6 RELATED WORK

Learning-based approaches to runtime PM have been recently proposed, including the so-called Neural Predictive Monitoring (NPM) method [7, 8, 10, 12, 28]. In NPM, neural networks are used to infer the Boolean satisfaction of a property and conformal prediction (CP) are used to provide statistical guarantees. However, NPM is restricted to reachability properties. NPM has been extended to support some source of stochasticity in the system: in [10] they allow partial observability and noisy observations, whereas in [12]

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. Predictive monitoring under partial observability is also analysed in [13], 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.

Other learning-based approaches for reachability prediction of stochastic systems include [6, 9, 14, 17, 35, 37]. Of these, [37] develop techniques to detect potential prediction errors using neural network verification methods [18]. These verification methods, however, do not scale well on large models and support only specific classes of neural networks. In [6], the authors introduce an efficient approximate Bayesian monitor for the satisfaction of STL properties for stochastic systems. However, this work focuses, once again, on the Boolean satisfaction.

Various learning-based PM approaches for temporal logic properties [22, 30, 31, 33, 38, 40] have been recently proposed. In particular, [30] provide (like we do) guaranteed prediction intervals, but (unlike our method) they are limited to ARMA/ARIMA models. Ma et al. [22] 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 [19]), making the resulting guarantees not theoretically valid.

We contribute to the state of the art by developing a quantitative predictive monitoring method that offers good scalability, provides statistical guarantees, and supports stochastic systems with rich STL-based requirements. Another important novelty is to monitor and predict the quantitative STL satisfaction of a requirement rather than the Boolean one.

7 CONCLUSIONS

We presented quantitative predictive monitoring (QPM), a technique to reliably monitor the evolution of a stochastic system at runtime. In particular, given a requirement expressed as an STL formula, QPM quantifies how robustly this requirement is satisfied by means of a range of STL robustness values. This interval undergoes a principled recalibration that guarantees a desired level of coverage, i.e. the interval covers the exact STL robustness values with a given confidence. The proposed technique avoids expensive Monte-Carlo simulations at runtime by leveraging conformalized quantile regression. The resulting method has very little overhead during runtime execution.

Our experimental evaluation demonstrates that we overall reach a high accuracy and low rate of prediction errors, and the statistical guarantees are always met. The conformal approach also adjusts the width of the interval w.r.t. desired confidence level, reducing the number of over-conservative predictions. These results support the claim of having an efficient and reliable QPM.

In future work, we intend to explore the effects of our QPM method over temporal, rather than spatial, STL robustness. We will investigate a possible dynamics-aware approach to inference. The latter should aim at limiting the 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] Alessandro Abate, Henk Blom, Marc Bouissou, Nathalie Cauchi, Hassane Chraibi, Joanna Delicaris, Sofie Haesaert, Arnd Hartmanns, Mahmoud Khaled, Abolfazl Lavaei, et al. 2021. ARCH-COMP21 Category Report: Stochastic Models. In *8th International Workshop on Applied Verification of Continuous and Hybrid Systems, ARCH 2021*. EasyChair, 55–89.

[2] Alessandro Abate, Joost-Pieter Katoen, John Lygeros, and Maria Prandini. 2010. Approximate model checking of stochastic hybrid systems. *European Journal of Control* 16, 6 (2010), 624–641.

[3] Vineeth Balasubramanian, Shen-Shyang Ho, and Vladimir Vovk. 2014. *Conformal prediction for reliable machine learning: theory, adaptations and applications.* Newnes.

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

[5] Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donzé, Georgios Fainekos, Oded Maler, Dejan Ničković, and Sriram Sankaranarayanan. 2018. Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. In *Lectures on Runtime Verification*. Springer, 135–175.

[6] Luca Bortolussi, Francesca Cairoli, Ginevra Carbone, and Paolo Pulcini. 2022. Stochastic Variational Smoothed Model Checking. *arXiv preprint arXiv:2205.05398* (2022).

[7] 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.

[8] 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.

[9] Luca Bortolussi, Dimitrios Milios, and Guido Sanguinetti. 2016. Smoothed model checking for uncertain continuous-time Markov chains. *Information and Computation* 247 (2016), 235–253.

[10] Francesca Cairoli, Luca Bortolussi, and Nicola Paoletti. 2021. Neural Predictive Monitoring under Partial Observability. arXiv:2108.07134 [cs.LG]

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

[12] Francesca Cairoli, Nicola Paoletti, and Luca Bortolussi. 2022. Neural Predictive Monitoring for Collective Adaptive Systems. In *ISoLA 2022 Symposium, to appear*.

[13] Yi Chou, Hansol Yoon, and Sriram Sankaranarayanan. 2020. Predictive Runtime Monitoring of Vehicle Models Using Bayesian Estimation and Reachability Analysis. In *Intl. Conference on Intelligent Robots and Systems (IROS)*.

[14] Badis Djeridane and John Lygeros. 2006. Neural approximation of PDE solutions: An application to reachability computations. In *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 3034–3039.

[15] Alexandre Donzé and Oded Maler. 2010. Robust Satisfaction of Temporal Logic over Real-Valued Signals. In *Proceedings of International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, Klosterneuburg, Austria, 92–106. https://doi.org/10.1007/978-3-642-15297-9_9

[16] Ansgar Fehnker and Franjo Ivančić. 2004. Benchmarks for hybrid systems verification. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 326–341.

[17] Wolfgang Granig, Stefan Jakšić, Horst Lewitschnig, Cristinel Mateis, and Dejan Ničković. 2020. Weakness Monitors for Fail-Aware Systems. In *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 283–299.

[18] Radoslav Ivanov, James Weimer, Rajeev Alur, George J Pappas, and Insup Lee. 2019. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. 169–178.

[19] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. 2018. Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning*. PMLR, 2796–2804.

[20] Pavel Kurasov, Alexander Lück, Delio Mugnolo, and Verena Wolf. 2018. Stochastic hybrid models of gene regulatory networks–a pde approach. *Mathematical biosciences* 305 (2018), 170–177.

[21] Lars Lindemann, Alena Rodionova, and George Pappas. 2022. Temporal Robustness of Stochastic Signals. In *25th ACM International Conference on Hybrid Systems: Computation and Control*. 1–11.

[22] Meiyi Ma, John Stankovic, Ezio Bartocci, and Lu Feng. 2021. Predictive monitoring with logic-calibrated uncertainty for cyber-physical systems. *ACM Transactions on Embedded Computing Systems (TECS)* 20, 5s (2021), 1–25.

[23] Anirudha Majumdar and Marco Pavone. 2020. How should a robot assess risk? towards an axiomatic theory of risk in robotics. In *Robotics Research*. Springer, 75–84.

[24] Oded Maler and Dejan Nickovic. 2004. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 152–166.

[25] Roland Malhame and Chee-Yee Chong. 1985. Electric load model synthesis by diffusion approximation of a high-order hybrid-state stochastic system. *IEEE Trans. Automat. Control* 30, 9 (1985), 854–860.

[26] Harris Papadopoulos. 2008. Inductive conformal prediction: Theory and application to neural networks. In *Tools in artificial intelligence*. InTech.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*. 8024–8035.

[28] Dung Phan, Nicola Paoletti, Timothy Zhang, Radu Grosu, Scott A Smolka, and Scott D Stoller. 2018. Neural state classification for hybrid systems. In *International Symposium on Automated Technology for Verification and Analysis*. Springer, 422–440.

[29] Dung T Phan, Radu Grosu, Nils Jansen, Nicola Paoletti, Scott A Smolka, and Scott D Stoller. 2020. Neural simplex architecture. In *NASA Formal Methods Symposium*. Springer, 97–114.

[30] Xin Qin and Jyotirmoy V Deshmukh. 2019. Predictive monitoring for signal temporal logic with probabilistic guarantees. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. ACM, 266–267.

[31] Xin Qin and Jyotirmoy V Deshmukh. 2020. Clairvoyant Monitoring for Signal Temporal Logic. In *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 178–195.

[32] Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M. Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A. Seshia. 2014. Model predictive control with signal temporal logic specifications. In *Proceedings of the IEEE Conference on Decision and Control*. IEEE, Los Angeles, USA, 81–87. https://doi.org/10.1109/CDC.2014.7039363

[33] Alëna Rodionova, Lars Lindemann, Manfred Morari, and George J Pappas. 2021. Time-robust control for STL specifications. In *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 572–579.

[34] Yaniv Romano, Evan Patterson, and Emmanuel J Candès. 2019. Conformalized quantile regression. *arXiv preprint arXiv:1905.03222* (2019).

[35] Vicenc Rubies Royo, David Fridovich-Keil, Sylvia Herbert, and Claire J Tomlin. 2018. Classification-based Approximate Reachability with Guarantees Applied to Safe Trajectory Tracking. *arXiv preprint arXiv:1803.03237* (2018).

[36] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. 2005. *Algorithmic learning in a random world*. Springer Science & Business Media.

[37] E. Yel, T. J. Carpenter, C. Di Franco, R. Ivanov, Y. Kantaros, I. Lee, J. Weimer, and N. Bezzo. 2020. Assured Runtime Monitoring and Planning: Toward Verification of Neural Networks for Safe Autonomous Operations. *IEEE Robotics & Automation Magazine* 27, 2 (2020), 102–116.

[38] Hansol Yoon and Sriram Sankaranarayanan. 2021. Predictive runtime monitoring for mobile robots using logic-based bayesian intent inference. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 8565–8571.

[39] Håkan LS Younes, Marta Kwiatkowska, Gethin Norman, and David Parker. 2006. Numerical vs. statistical probabilistic model checking. *International Journal on Software Tools for Technology Transfer* 8, 3 (2006), 216–228.

[40] Xinyi Yu, Weijie Dong, Xiang Yin, and Shaoyuan Li. 2022. Model Predictive Monitoring of Dynamic Systems for Signal Temporal Logic Specifications. *arXiv preprint arXiv:2209.12493* (2022).