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 \( \phi \) is satisfied, QPM provides a quantitative measure of satisfaction by predicting the quantitative (aka robust) STL semantics of \( \phi \). 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.

ACM Reference Format:

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 [28]. 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 [38] would require simulating at runtime a typically large number of Monte-Carlo trajectories to achieve desired error levels.

For this reason, a number of approximate PM techniques based on machine learning have been recently proposed (see e.g. [12, 21, 29, 36]), 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, 35] 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) [14] 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 \( R_\phi \) of an STL formula \( \phi \) (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 meaningfully used to determine the extent of any corrective actions and enable efficient online model predictive control for STL [31]: for instance, depending on the specification \( \phi \), if QPM predicts a high value of \( R_\phi \), then little or no intervention might be needed, while a low \( R_\phi \) 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

\[^{1}\] 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.
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 $\Omega$ is the sample space, $\mathcal{F}$ is the $\sigma$-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 $(S(t, \omega) : t \in T)$, for any point $\omega \in \Omega$, the mapping $S(\cdot, \omega) : T \rightarrow S$, 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) [14, 23], 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:

\[
\phi := \text{true} \mid \mu \mid \neg \phi \mid \phi \land \psi \mid \phi U \psi,
\]

where $I \subseteq T$ is a temporal interval, either bounded, $I = [a, b]$, or unbounded, $I = [a, +\infty)$, for any $0 \leq a < b$. Atomic propositions $\mu$ are (non-linear) inequalities on the states of a signal $\tau$ at a time $t$, $\mu = (g(\tau(t)) > 0)$, where $g : S \to \mathbb{R}$ and $\tau(t)$ 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: $false := \neg \text{true}$, $\phi \land \psi := \neg(\neg \phi \land \neg \psi)$, $F_I := \text{true} U I \phi$ 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.

\[
\begin{align*}
(\bar{\tau}, t) \models \mu & \iff g(\bar{\tau}(t)) > 0 \\
(\bar{\tau}, t) \models \neg \phi & \iff \neg(\bar{\tau}, t) \models \phi \\
(\bar{\tau}, t) \models \phi_1 \land \phi_2 & \iff (\bar{\tau}, t) \models \phi_1 \land (\bar{\tau}, t) \models \phi_2 \\
(\bar{\tau}, t) \models \phi_1 U [a, b] \phi_2 & \iff \exists t' \in [t + a, t + b] \text{ s.t. } (\bar{\tau}, t') \models \phi_2 \land \forall t'' \in [t, t'), (\bar{\tau}, t'') \models \phi_1
\end{align*}
\]

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

\[
R_p(\bar{s}, t) = g(\bar{s}(t)) \\
R_{\neg \phi}(\bar{s}, t) = -R_\phi(\bar{s}, t) \\
R_{\phi_1 \land \phi_2}(\bar{s}, t) = \min(R_{\phi_1}(\bar{s}, t), R_{\phi_2}(\bar{s}, t)) \\
R_{\phi_1 \udots \phi_n}(\bar{s}, t) = \sup_{t' \in [t+\Delta, t+\Theta]} \left( \min \left( R_{\phi_1}(\bar{s}, t'), \inf_{t'' \in [t, t']} R_{\phi_n}(\bar{s}, t'') \right) \right).
\]

The sign of \( R_{\phi} \) indicates the Boolean satisfaction of signal \( \bar{s} \), and in particular, we have:

- \( R_{\phi}(\bar{s}, t) > 0 \Rightarrow (\bar{s}, t) \models \phi \);
- \( R_{\phi}(\bar{s}, t) < 0 \Rightarrow (\bar{s}, t) \nmodels \phi \).

Given a stochastic process \( S = \{S(t, \omega), t \in T\} \), an STL requirement \( \phi \) 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

\[
P \left( R_{\phi}(\bar{s}, 0) \mid \bar{s}(0) = s_0 \right),
\]

where \( \bar{s} \) is a random signal given by the sequence of random variables \( \{S(0, \omega), S(1, \omega), \ldots, S(H-1, \omega)\} \). For time \( k \), \( \bar{s}(k) = S(k, \omega) \) denotes the random variable corresponding to the state at time \( k \) in \( \bar{s} \). 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 \( s_0 \) 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 \( s_0 \). A formal statement of the problem is given below.

**Problem 1 (Quantitative Predictive Monitoring).** Given a discrete-time stochastic process \( S = \{S(t, \omega), t \in T\} \) over a state space \( S \), temporal horizon \( H \), a significance level \( \alpha \in [0, 1] \) and an STL formula \( \phi \), derive a monitoring function \( I(\cdot) \) that maps any state \( s_0 \in S(\cdot, \cdot) \) into an interval \( I(s_0) \) such that

\[
P \left( R_{\phi}(\bar{s}, 0) \in I(s_0) \mid \bar{s}(0) = s_0 \right) \geq 1 - \alpha.
\]

We frame Problem 1 as a conditional quantile regression problem. This boils down to learning for a generic state \( s_0 \) an upper and a lower quantile of the random variable \( R_{\phi}(\bar{s}, 0) \) induced by \( \bar{s} \) conditioned on \( \bar{s}(0) = s_0 \). We then use these two quantiles to build the output of function \( I \) in \( s_0 \). 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 \( x_t \).

For the sake of clarity, we start by showing conformal prediction approaches for deterministic predictors and then move to present Conformalized Quantile Regression (CQR) [33], 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, 35]. 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 \( x_t \), the \( \alpha \)-prediction region for \( x_t \), \( \Gamma^\alpha \subseteq Y \), is a set of target values \( s_t \).

\[
P_{(x_t, y_t)} \left( y_t \in \Gamma^\alpha \right) \geq 1 - \alpha. \tag{2}
\]

The idea of CP is to construct the prediction region by “inverting” a suitable hypothesis test: given a test point \( x_t \) and a tentative target value \( y^t \), we exclude \( y^t \) from the prediction region only if it is unlikely that \( y^t \) is the true value for \( x_t \). 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, \( \delta \) can be viewed as a generalized residual function. In other words, CP builds the prediction region \( \Gamma^\alpha \) for a test point \( x_t \) by excluding all targets \( t' \) whose NCF values are unlikely to follow the NCF distribution of the true targets:

\[
\Gamma^\alpha = \left\{ y^t \in Y \mid \text{Pr}(x, y) \leq Z \left\{ \delta(x, y^t) \geq \delta(x, y^t) > \alpha \right\} \right\}. \tag{3}
\]

This prediction region is guaranteed to contain the true (unknown) value \( y_t \), with confidence \( 1 - \alpha \). The probability term in Eq. (3) is often called the p-value. From a practical viewpoint, the NCF distribution \( \text{Pr}(x, y) \leq Z \left\{ \delta(x, y^t) \geq \delta(x, y^t) > \alpha \right\} \) cannot be derived in an analytical form, and thus we use an empirical approximation derived using a sample \( Z_n \) of \( Z \). This approach is called inductive CP [25] and \( Z_n \) 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 \( Z \) 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 \( \alpha \), 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 \( Y \subseteq \mathbb{R}^d \). The inductive CP algorithm is divided into an offline phase, executed only once, and an online phase, executed for every test point \( x_t \). 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 \( x_t \) using the computed regressor and distribution.

1. Draw sample \( Z' \) of \( Z \). 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 \( \delta \).
3. Construct the calibration distribution by computing, for each \( z_i \in Z_c \), the NCF score \( \hat{\delta}_i = \delta(z_i) \).
4. Identify the critical value \( \hat{\beta}_{(\alpha)} \) of the calibration distribution, i.e., its empirical \( (1 - \alpha) \)-quantile, or the \( \{\alpha : ([|Z_c|] + 1)\} \)-largest calibration score.
5. Return the prediction region

\[
\Gamma^\alpha_s = f(x_s) \pm \hat{\beta}_{(\alpha)}. \tag{4}
\]

Notice that such prediction intervals have the same width (\( \hat{\beta}_{(\alpha)} \)) 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 \( \alpha \). Specifically, from Eq. (4) 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 \( \alpha \) yields a larger critical value \( \hat{\beta}_{(\alpha)} \).

### 3.2 Conformalized Quantile Regression

We now switch to the stochastic setting. Let us consider a probability. The optimization problem can be formally expressed as approximating the conditional quantile function is defined as

\[
\hat{\beta}_{(\alpha)} := \inf \{ y' \in \mathbb{R} \mid F(y' | x = x') \geq \alpha \}. \tag{5}
\]

Given a significance level \( \alpha \), 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 \( PI(x) := [\hat{q}_{\alpha_{lo}}(x), \hat{q}_{\alpha_{hi}}(x)] \). By construction, this interval satisfies

\[
P(y \in PI(x') | x = x') \geq 1 - \alpha. \tag{6}
\]

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 \( \theta \) so that it closely resembles the quantile function. The optimization problem can be formally expressed as finding the optimal parameters \( \hat{\theta} \) such that

\[
\hat{\theta} = \arg \min \theta \left[ \frac{1}{n} \sum L_\alpha(Y_i, f(X_i; \theta)) + G(\theta) \right],
\]

where \( L_\alpha \) is the check or pinball loss, defined as

\[
L_\alpha(y, \hat{y}) = \alpha \max(y - \hat{y}, 0) + (1 - \alpha) \max(\hat{y} - y, 0), \tag{7}
\]

and \( G \) 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. (7) as loss function. Once trained, the output of \( f(x; \theta) \) is a prediction \( \hat{q}_\alpha(x) \) of the conditional \( \alpha \)-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 quantiles 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 \( \hat{q}_{\alpha_{lo}}, \hat{q}_{0.5} \) (the median) and \( \hat{q}_{\alpha_{hi}} \), where for a choice of \( \alpha \in (0, 0.5) \), \( \alpha_{lo} = \alpha \) and \( \alpha_{hi} = 1 - \alpha \). The loss then becomes

\[
L_\alpha(y, \hat{y}) = \frac{1}{2} \cdot (L_{\alpha_{lo}}(y, \hat{y}) + L_{0.5}(y, \hat{y}) + L_{\alpha_{hi}}(y, \hat{y})).
\]

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 (6). 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

\[
E_i := \max\{\hat{q}_{\alpha_{lo}}(x_i) - y_i, y_i - \hat{q}_{\alpha_{hi}}(x_i) \mid (x_i, y_i) \in Z_c \}. \tag{8}
\]

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

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

where \( \tau \) 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 \( \tau \), 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 \( P_x(y \in CPI(x)) \geq 1 - \alpha \).

**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. 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.
We present a method to solve Problem 1. 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

\[
P(\overline{s} | \overline{s}(k) = s)
\]

where \( \overline{s} = (\overline{s}(k), \ldots, \overline{s}(k + H)) \) is the random trajectory of length \( H \) starting at time \( t_0 \), \( \overline{s}(i) = S(i, k) \) for any \( i \in T \).

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

\[
R_\phi(s) \sim P(\overline{s}(k) | \overline{s}(k) = s)
\]

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 \( R_\phi(s) \), 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 \( R_\phi(s) \) 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 \( R_\phi(s) \).

In short, our solution consists of four steps, detailed below:

**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 \( R_\phi(s) \).

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 \( H \), \( \overline{s}_i^1, \ldots, \overline{s}_i^M \) where \( \overline{s}_i^j \) is a realization of \( P(\overline{s} | \overline{s}(k) = s_i) \), and compute the robustness value \( R_\phi(\overline{s}_i^j, 0) \) of each of these trajectories. We note that \( (R_\phi(\overline{s}_i^j, 0))_{j=1}^M \) is an empirical approximation of \( R_\phi(s_i) \).

**QR training and residuals computation.** We divide the dataset \( Z^\phi \) into a training set \( Z^\phi_{\text{train}} \) and a calibration set \( Z^\phi_{\text{cal}} \). We then use \( Z^\phi_{\text{train}} \) to train a QR that learns how to map states \( s \) into three quantiles

\[
f(s; \hat{\theta}) = \{ q_{\alpha/2}(s), \hat{q}_{0.5}(s), \hat{q}_{1-\alpha/2}(s) \},
\]

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

We then apply CQR, i.e. we compute the residuals of the QR over \( Z^\phi_{\text{cal}} \), as described in Equation 8 of the previous section, and find the critical residual value \( \tau \).

**Inference.** For a test state \( s_* \), this step involves predicting the relevant quantiles, \( \hat{q}_{\alpha/2}(s_*) \) and \( \hat{q}_{1-\alpha/2}(s_*) \), using the QR predictor and correct the resulting interval using the critical residual value \( \tau \). The calibrated interval returned at inference time becomes

\[
\text{CPI}(s_*) = [\hat{q}_{\alpha}(s_*) - \tau, \hat{q}_{1-\alpha}(s_*) + \tau] \subseteq \mathbb{R}
\]

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

**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 \( \phi \). On one hand, this is advantageous in that the monitor is tailored, and hence, highly efficient.
accurate at predicting \( R_b \) 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 \( \phi_1 \) and \( \phi_2 \) can be combined in a compositional manner to construct a monitor for a Boolean combination of the two properties, \( \phi_1 \land \phi_2 \) or \( \phi_1 \lor \phi_2 \), 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 \( \phi \) 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,

\[
CP^{\phi_1,\phi_2} := CP^{\phi_1} \cup CP^{\phi_2}.
\]

We now prove that \( CP^{\phi_1,\phi_2} \) has a guaranteed coverage of \((1 - \alpha)\), the same of \( CP^{\phi_1} \) and \( CP^{\phi_2} \), for both \( \phi_1 \land \phi_2 \) and \( \phi_1 \lor \phi_2 \).

**Proposition 1.** Given two properties \( \phi_1 \) and \( \phi_2 \), let \( CP^{\phi_i} \) 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 \( CP^{\phi_1,\phi_2} := CP^{\phi_1} \cup CP^{\phi_2} \) is a valid prediction interval for both \( R_{\phi_1 \land \phi_2} \) and \( R_{\phi_1 \lor \phi_2} \), i.e.,

\[
P_R(s) \land \phi_i(s) \in CP^{\phi_i}(s) \geq 1 - \alpha \]

and

\[
P_R(s) \lor \phi_i(s) \in CP^{\phi_i}(s) \geq 1 - \alpha \]

for every state \( s \in S \).

The proof is provided in the Supplementary Material.

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 \( \phi_1 \land \phi_2 \) we define for a state \( s \in S \) the prediction as

\[
CP^{\phi_1 \land \phi_2}(s) := \left[ \min \left( q_{\phi_1}(s), q_{\phi_2}(s) \right), \min \left( q_{\phi_1}(s), q_{\phi_2}(s) \right) \right].
\]

See Fig. 3 for a schematic visualization. We then compute the residuals of \( CP^{\phi_1 \land \phi_2} \) over the calibration set, obtaining the critical value \( \tau^{\phi_1 \land \phi_2} \) and the calibrated prediction interval

\[
CP^{\phi_1 \land \phi_2}(s) := \left[ \min \left( q_{\phi_1}(s), q_{\phi_2}(s) \right) - \tau^{\phi_1 \land \phi_2}, \min \left( q_{\phi_1}(s), q_{\phi_2}(s) \right) + \tau^{\phi_1 \land \phi_2} \right],
\]

which has a guaranteed coverage of \((1 - \alpha)\).

The procedure is the same for property \( \phi_1 \lor \phi_2 \), with the only difference that the prediction interval is given by

\[
CP^{\phi_1 \lor \phi_2}(s) := \left[ \max \left( q_{\phi_1}(s), q_{\phi_2}(s) \right), \max \left( q_{\phi_1}(s), q_{\phi_2}(s) \right) \right].
\]

Solution (2) is expected to produce tighter intervals compared to solution (1)\(^5\). On the other hand, solution (2) requires the generation

\[\text{Figure 3: Overview of the combination of the conformal quantitative predictive monitors for two different properties } \phi_1 \text{ and } \phi_2.\]

of the respective calibration set, \( Z_{c_1}^{\phi_1 \land \phi_2} \) or \( Z_{c_1}^{\phi_1 \lor \phi_2} \), 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 three-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:

\[
\phi_1 = G_{[0, 60]}(v \in \text{Safe}) \text{ and } \phi_2 = F_{[0, 60]}(v_1 < v_2), \text{ where Safe} = [1, 6; 0, 10], H = 60 \text{min}, \Delta t = 20 \text{s}. \]

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 seven-dimensional, the continuous state space is twodimensional and composed of a state \( v(t) = (v_{\text{height}}(t), v_{\text{temp}}(t)) \) 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 the fluid level is always in the safe range \( \text{Safe} = [H_{\text{dryout}}, H_{\text{overflow}}] \) and the temperature is never too high and it can be expressed by the STL property \( \phi = G_{[t, \infty]}((v_{\text{height}}(t) \land v_{\text{temp}}(t)) \land 100.). \)

**Multi-Room Heating.** (MRH) [2, 15, 24]: 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
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 \( \tau \) is almost negligible (in the order of microseconds on GPU). Once trained, the time needed to have an 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).

The gene-specific requirement to monitor is: the count \( v_i \) of 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, \( Q = \{ \text{on, off} \}^h \), whereas the proteins count is continuous, \( V = \mathbb{R}^h \). 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 \( i \) to stabilize under a certain threshold \( V_{ub}^i \) and it can be expressed by the STL property \( \phi_i = G_\alpha(v_i \leq v_i \leq V_{ub}^i) \).

**Gene Regulatory Network.** (GRN) [4, 19]: 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, \( Q = \{ \text{on, off} \}^h \), whereas the proteins count is continuous, \( V = \mathbb{R}^h \). 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 \( V_{ub}^i \) and it can be expressed by the STL property \( \phi_i = G_\alpha(v_i \leq v_i \leq V_{ub}^i) \).

![Figure 4: Scheme of the heated tank system whose components can undergo exponentially distributed failures.](image)

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

### 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 \( \tau \) 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 \( \epsilon \) 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 \( \epsilon \). 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 \( \phi \). 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 \( \mathcal{D}^\text{test} \) (simulate and compute STL robustness), (3) train the QR, (4) compute the calibration score \( \tau \) (obtaining CQR prediction intervals), (5) evaluate both the QR and the CQR on a test set \( \mathcal{D}^\text{test} \).

**Experimental settings.** The entire pipeline is implemented in Python. The quantitative semantics of pcheck library\(^6\) 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 [26]. The experiments were conducted on a shared virtual

\(^6\)https://github.com/simonesilvetti/pcheck
we train the QR and then recalibrate the predicted interval by means of the latter. The cost of the latter depends on the chosen architecture and on the performance and of the generalization capabilities of our QPM network. 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.

Datasets. For every model, we generate a dataset $Z^\phi$ 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 $N_{\text{train}} = n \cdot 1000$ states for the training set, $N_{\text{cal}} = n \cdot 500$ states for the calibration set and $N_{\text{test}} = n \cdot 100$ states for the test set. From each state, we simulate $M = 50$ trajectories for the training and for the calibration set and $M_{\text{test}} = 50$ 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 a multi-room heating (MRH) system with 4 rooms. 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.

Performance evaluation. We choose in all the experiments $\alpha = 0.1$, so that $\alpha_{\text{cal}} = 0.05$ and $\alpha_{\text{li}} = 0.95$. 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 and Fig. 7-14 in SM show 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 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.

Tables 1-4 and Tables 5-6 in SM provide a detailed summary of the performance and of the generalization capabilities of our QPM 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/aiblab-units/CQR_Quantitative_NPM.

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 GRN2 model is double (1 hour) compared to the MRH2 model (30 minutes), even if they have the same dimensionality. The dataset generation time explodes easily around 20 hour for GRN6 and MRH8.

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.

<table>
<thead>
<tr>
<th>Room 1</th>
<th>MRH2</th>
<th>correct</th>
<th>uncertain</th>
<th>wrong</th>
<th>FP</th>
<th>coverage</th>
<th>efficiency</th>
<th>EQR width</th>
</tr>
</thead>
<tbody>
<tr>
<td>QR</td>
<td>99.50</td>
<td>0.00</td>
<td>0.5</td>
<td>0.00</td>
<td>89.25</td>
<td>0.3221</td>
<td>0.3167</td>
<td></td>
</tr>
<tr>
<td>CQR</td>
<td>99.50</td>
<td>0.00</td>
<td>0.5</td>
<td>0.00</td>
<td>90.19</td>
<td>0.3313</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Room 2</th>
<th>MRH2</th>
<th>correct</th>
<th>uncertain</th>
<th>wrong</th>
<th>FP</th>
<th>coverage</th>
<th>efficiency</th>
<th>EQR width</th>
</tr>
</thead>
<tbody>
<tr>
<td>QR</td>
<td>86.50</td>
<td>11.00</td>
<td>2.50</td>
<td>2.50</td>
<td>90.94</td>
<td>0.2568</td>
<td>0.1644</td>
<td></td>
</tr>
<tr>
<td>CQR</td>
<td>86.50</td>
<td>11.00</td>
<td>2.50</td>
<td>2.50</td>
<td>91.58</td>
<td>0.2591</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 GRN2 model is double (1 hour) compared to the MRH2 model (30 minutes), even if they have the same dimensionality. The dataset generation time explodes easily around 20 hour for GRN6 and MRH8.

Performance evaluation. We choose in all the experiments $\alpha = 0.1$, so that $\alpha_{\text{cal}} = 0.05$ and $\alpha_{\text{li}} = 0.95$. 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 and Fig. 7-14 in SM show 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 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.

Tables 1-4 and Tables 5-6 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

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.

<table>
<thead>
<tr>
<th>Room 1</th>
<th>MRH2</th>
<th>correct</th>
<th>uncertain</th>
<th>wrong</th>
<th>FP</th>
<th>coverage</th>
<th>efficiency</th>
<th>EQR width</th>
</tr>
</thead>
<tbody>
<tr>
<td>QR</td>
<td>99.50</td>
<td>0.00</td>
<td>0.5</td>
<td>0.00</td>
<td>89.25</td>
<td>0.3221</td>
<td>0.3167</td>
<td></td>
</tr>
<tr>
<td>CQR</td>
<td>99.50</td>
<td>0.00</td>
<td>0.5</td>
<td>0.00</td>
<td>90.19</td>
<td>0.3313</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We observe that the percentage of wrong and FP predictions is extremely low in all the experimental configurations (always

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.

<table>
<thead>
<tr>
<th>Room 2</th>
<th>MRH2</th>
<th>correct</th>
<th>uncertain</th>
<th>wrong</th>
<th>FP</th>
<th>coverage</th>
<th>efficiency</th>
<th>EQR width</th>
</tr>
</thead>
<tbody>
<tr>
<td>QR</td>
<td>86.50</td>
<td>11.00</td>
<td>2.50</td>
<td>2.50</td>
<td>90.94</td>
<td>0.2568</td>
<td>0.1644</td>
<td></td>
</tr>
<tr>
<td>CQR</td>
<td>86.50</td>
<td>11.00</td>
<td>2.50</td>
<td>2.50</td>
<td>91.58</td>
<td>0.2591</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 $\phi_2$ (eventually operator).

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

<table>
<thead>
<tr>
<th>Gene 1 $\land$ Gene 2</th>
<th>correct</th>
<th>uncertain</th>
<th>wrong</th>
<th>FP</th>
<th>coverage</th>
<th>efficiency</th>
<th>EQR width</th>
</tr>
</thead>
<tbody>
<tr>
<td>QR</td>
<td>93.25</td>
<td>4.25</td>
<td>2.50</td>
<td>2.50</td>
<td>89.69</td>
<td>0.5713</td>
<td>0.5062</td>
</tr>
<tr>
<td>CQR</td>
<td>93.00</td>
<td>4.50</td>
<td>2.50</td>
<td>2.50</td>
<td>90.00</td>
<td>0.5756</td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>93.88</td>
<td>0.5773</td>
<td>0.5062</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNION</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>91.44</td>
<td>0.5384</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Gene regulatory network results. Correct, uncertain, wrong $\land$ coverage represent percentage values ($\%$). EQR width denotes the average width of the empirical quantile range over the test set.

<table>
<thead>
<tr>
<th>Gene 1 $\land$ Gene 3</th>
<th>correct</th>
<th>uncertain</th>
<th>wrong</th>
<th>FP</th>
<th>coverage</th>
<th>efficiency</th>
<th>EQR width</th>
</tr>
</thead>
<tbody>
<tr>
<td>QR</td>
<td>92.75</td>
<td>4.25</td>
<td>3.00</td>
<td>3.00</td>
<td>89.22</td>
<td>0.5822</td>
<td>0.4763</td>
</tr>
<tr>
<td>CQR</td>
<td>92.75</td>
<td>4.25</td>
<td>3.00</td>
<td>3.00</td>
<td>89.30</td>
<td>0.5439</td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>93.25</td>
<td>4.25</td>
<td>3.00</td>
<td>3.00</td>
<td>89.50</td>
<td>0.5314</td>
<td></td>
</tr>
<tr>
<td>UNION</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>91.40</td>
<td>0.5384</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Gene regulatory network results. Correct, uncertain, wrong $\land$ coverage represent percentage values ($\%$). EQR width denotes the average width of the empirical quantile range over the test set.

<table>
<thead>
<tr>
<th>Gene 1 $\land$ Gene 4</th>
<th>correct</th>
<th>uncertain</th>
<th>wrong</th>
<th>FP</th>
<th>coverage</th>
<th>efficiency</th>
<th>EQR width</th>
</tr>
</thead>
<tbody>
<tr>
<td>QR</td>
<td>85.50</td>
<td>11.00</td>
<td>3.50</td>
<td>3.25</td>
<td>89.43</td>
<td>0.5346</td>
<td></td>
</tr>
<tr>
<td>CQR</td>
<td>85.25</td>
<td>10.75</td>
<td>4.00</td>
<td>2.75</td>
<td>89.06</td>
<td>0.4943</td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>85.50</td>
<td>10.75</td>
<td>3.75</td>
<td>3.25</td>
<td>89.37</td>
<td>0.4708</td>
<td></td>
</tr>
<tr>
<td>UNION</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>90.60</td>
<td>0.5299</td>
<td></td>
</tr>
</tbody>
</table>

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

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 Fig. 14 in the Supplementary Material 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 1-5 we can also experimentally observe that CQR always meets the statistical guarantees: when QR is over-conservative, CQR refines the predicted interval (negative $\tau$), whereas when QR is over-confident CQR enlarges the predicted interval (positive $\tau$). 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, $MRH_2$, $MRH_4$, $MRH_8$, $GRN_2$, $GRN_4$ and $GRN_6$, allow us to analyse the compositionality of QPM over the conjunction of properties (each property is room-specific). Due to space limitations, we focus only on the conjunction $(\land$ operator). The same analysis can be done for the disjunction $(\lor$ operator).
We observe that QPM well captures the evolution of the robustness with sequentially generated data. We plan to further investigate we aim at quantifying the generalization capabilities of our QPM. (CP) are used to provide statistical guarantees. However, NPM is (MIN in tables) of the PI intervals calibrated once again over values but it tends to be either slightly over-confident or slightly support some source of stochasticity in the system: in [10] they restricted to reachability properties. NPM has been extended to the Boolean satisfaction of a property and conformal prediction method [7, 8, 10, 11, 27]. In NPM, neural networks are used to infer Learning-based approaches to runtime PM have been recently presented. 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 [20] and apply the same methodology described here for spatial robustness.

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 [20] 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, 11, 27]. 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 [11] 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 [12], 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, 13, 16, 34, 36]. Of these, [36] develop techniques to detect potential prediction errors using neural network verification methods [17]. 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 [21, 29, 30, 32, 37, 39] have been recently proposed. In particular, [29] provide (like we do) guaranteed prediction intervals, but (unlike our method) they are limited to ARMA/ARIMA models. Ma et al. [21] 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 [18]), 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
Appendices

A. 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.

Automated Anaesthesia Delivery. (AAD) [1] is a three-dimensional discrete-time stochastic process with state $s(t) = (q(t), v(t))$ evolving according to the following dynamics:

$$v(t_{k+1}) = A \cdot v(t_k) + b \cdot q(t_k) + w(t_k),$$

where $A, b$ are patient-specific parameters and there is a Gaussian disturbance $w(t) \sim N(0, 10^{-3}I_3)$. The properties to be monitored are about the safety of the system, $\phi_1 = G_{\{t_0, t_H\}}(v \in \text{Safety})$, $\phi_2 = F_{\{t_0, t_H\}}(v_1 < v_2)$ where Safe = \{1, 6, 10, 15, 16\}, $H = 60\text{min}$, $\Delta t = 20\text{sec}$. 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 system composed of a tank $T$ containing a liquid whose level is influenced by two inflow pumps $P_1$ and $P_2$ and a outflow valve $W$, all managed by a controller $C$. The liquid should absorb and transport the heat from a heat source. The discrete state space is five-dimensional composed of states $q$:

$$q(t) = (q_{\text{height}}(t), q_{\text{temp}}(t), q_{\text{on}}(t), q_{\text{off}}(t), q_{\text{t}}(t)).$$

The general model adopted for this heated tank system is that of piece-wise deterministic Markov processes. The continuous state space is two-dimensional composed of a state $v(t) = (v_{\text{height}}(t), v_{\text{temp}}(t))$ representing the liquid height and the liquid temperature. The continuous part of the system follows the following stochastic difference equations:

$$\delta H(t) = (q_{\text{in}}(t) + q_{\text{out}}(t) - q_{\text{wind}}(t)) \cdot g$$

$$\delta T(t) = \left(\frac{q_{\text{in}}(t) + q_{\text{out}}(t)}{\text{volume}(t)} \cdot \text{temp}(t) \cdot g + E_{\text{in}}\right),$$

where $q_i$ indicates if unit $i \in \{P_1, P_2, W\}$ is working 1 or not 0, $g$ is the flow parameter, $T_{\text{in}} = 15 \text{deg C}$ is the inflow temperature and $E_{\text{in}} = 1 \text{deg Cm/h}$ is the heat source parameter. Each unit can be in one of the four discrete modes $Q_t = \{\text{On, Off, StickOn, StickOff}\}$.

The switching from On to Off and vice-versa is managed by the controller. In addition to this switching, there are two exponentially distributed failure switching possibilities: from On to StickOn and from Off to StickOff with rates $\lambda_{\text{On}} = 1/219$, $\lambda_{\text{Off}} = 1/175$ and $\lambda_{\text{W}} = 1/320$. The controller mode $q_{\text{C}}$ switches between the following configurations $Q_c = \{\text{Normal, Increase, Decrease}\}$. If Normal the controller does not try to influence the pumps and valve. If Increase the controller aims to increase the height of the liquid in the tank, by switching both pumps on, and by switching the valve off. If Decrease the controller aims to decrease the height of the liquid in the tank, by switching both pumps off, and by switching the valve on. The switching by the controller has no effect on unit $i$ if it is in failure mode StickOn or StickOff. If $v_{\text{height}}(t) \leq H_{\text{low}}$ then the controller switches from Normal or Decrease to Increase. If $v_{\text{height}}(t) \geq H_{\text{high}}$ the controller switches from Normal or Increase to Decrease. $H_{\text{low}} = 6\text{m}$, $H_{\text{high}} = 8\text{m}$. The property to monitor is $\phi = G_{\{t_0, t_H\} \{H_{\text{dryout}} \leq v_{\text{height}} \leq H_{\text{overflow}} \land (v_{\text{temp}} \leq T_{\text{overheat}})\}}$.

Multi-Room Heating. (MRH) [2, 15, 24]: 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 hybrid system $S = Q \times V$: the discrete state represents the status of the $h$ heaters, $Q = \{0, 1\}^h$, and the continuous state represents the temperatures in the $h$ rooms, $V = \mathbb{R}^h$. The average temperature evolves according to the following stochastic difference equation:

$$\dot{v}_i(t_{k+1}) = v_i(t_k) + b_i(x_a - v_i(t_k)) + \sum_{j \neq i} a_{ij}(v_j(t_k) - v_i(t_k)) + c_i Q_i(q_i(t_k)) + w_i(t_k),$$

where $x_a$ is the ambient temperature (constant for the entire building) and $Q_i(\cdot)$ is the indicator function of set $Q_i = \{(q_1, \ldots, q_h) \in Q : q_i = 0\}$.

The quantities $b_i, a_{ij}$ and $c_i$ are non-negative constants representing respectively the average heat transfer rate from room $i$ to the ambient ($b_i$) and to room $j \neq i$ ($a_{ij}$) and the heat rate supplied to room $i$ by the heater in room $i$ ($c_i$). $a_{ij}$ also reflects the rooms layout, e.g., $a_{ij} = 0$ if room $i$ and $j$ are not adjacent. The overall complexity of the stochastic hybrid system is determined by the number of rooms. The room-specific requirement to monitor is: the temperature $v_i$ of room $i$ always stays in a desirable range $[v_{i\text{low}}, v_{i\text{high}}]$ and it can be expressed by the STL property $\phi_i = G_{\{0, t_H\}}(v_{i\text{low}} \leq v_i \leq v_{i\text{high}})$.

Gene Regulatory Network. (GRN) [4, 19]: discrete-time stochastic hybrid system modeling a genetic regulatory network composed of $h$ genes that produces respectively $h$ proteins that repress each other in a cyclic fashion. 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, $Q = \{0, 1\}^h$, whereas the proteins count is continuous, $V = \mathbb{R}^h$. Protein counts change deterministically according to a linear differential equation, whereas transition
between modes follow a Markov jump process. The change of protein \(i\) has the form \(a_i - b_iq_i\) if \(q_i = \text{off}\) and \(c_i - d_iq_i\) if \(q_i = \text{on}\), where \(a_i\) and \(c_i\) are the production rates of protein \(i\) if the gene is inactive or active, respectively. The respective degradation rate constants are given by \(b_i\) and \(d_i\). The binding rate between a protein \(i - 1\) and a gene \(i\) is expressed as \(k_b \cdot q_i q_{i-1}\), whereas the unbinding rate is constant \(k_u\). The shooting time for this reactions is exponentially distributed. The complexity of the stochastic hybrid system is determined by the number of genes. The gene-specific requirement to monitor is: the count \(v_{ij}\) of protein \(i\) stabilize under a certain threshold \(V_i^{ub}\) and it can be expressed by the STL property \(\phi_1 = G_{i\in[1,2]}[s_i(t)\leq V_i^{ub}]\).

\[\text{Gene 1} \quad \text{Gene 2} \quad \text{Gene 3} \quad \text{Gene 4}\]

\[\text{protein 1} \quad \text{protein 2} \quad \text{protein 3} \quad \text{protein 4}\]

B PROOF OF PROPOSITION 1

Proposition 2. Given two properties \(\phi_1\) and \(\phi_2\), let CPI\(\phi_i\) 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 \(\text{CPI}_{\phi_1} \cup \text{CPI}_{\phi_2}\) is a valid prediction interval for both \(R_{\phi_1 \land \phi_2}\) and \(R_{\phi_1 \lor \phi_2}\), i.e.,

\[
\mathbb{P}\left(R_{\phi_1 \land \phi_2}(s) \in \text{CPI}_{\phi_1}(s)\right) \geq 1 - \alpha \tag{12}
\]

and

\[
\mathbb{P}\left(R_{\phi_1 \lor \phi_2}(s) \in \text{CPI}_{\phi_2}(s)\right) \geq 1 - \alpha \tag{13}
\]

for every state \(s \in S\).

Proof. We know that, for \(i \in \{1,2\}\) and for each state \(s \in S\),

\[
\mathbb{P}(R_{\phi_i}(s) \in \text{CPI}_{\phi_i}(s)) \geq 1 - \alpha.
\]

We also know that, by the STL quantitative semantics,

\[
R_{\phi_1 \land \phi_2}(s) = \min\left(R_{\phi_1}(s), R_{\phi_2}(s)\right).
\]

The following inequalities hold:

\[
\mathbb{P}\left(\min\left(R_{\phi_1}(s), R_{\phi_2}(s)\right) \in \text{CPI}_{\phi_1}(s)\right) =
\mathbb{P}(R_{\phi_1}(s) \in \text{CPI}_{\phi_1}(s)) + \mathbb{P}(R_{\phi_2}(s) \in \text{CPI}_{\phi_1}(s)) - \mathbb{P}(R_{\phi_1}(s) \in \text{CPI}_{\phi_1}(s), R_{\phi_2}(s) \in \text{CPI}_{\phi_2}(s)).
\]

The first two addends are higher than \(1 - \alpha\) since \(\text{CPI}_{\phi_1}(s) \subseteq \text{CPI}_{\phi_1}(s)\) for every \(i \in \{1,2\}\). The last addend is always lower than \(\mathbb{P}\left(\min\left(R_{\phi_1}(s), R_{\phi_2}(s)\right)\right)\). Thus the inequality becomes:

\[
2\mathbb{P}\left(\min\left(R_{\phi_1}(s), R_{\phi_2}(s)\right) \in \text{CPI}_{\phi_1}(s)\right) \geq 2(1 - \alpha),
\]

which proves the proposition. The proof for the \(\phi_1 \lor \phi_2\) is very similar, the main difference is that

\[
R_{\phi_1 \lor \phi_2}(s) = \max\left(R_{\phi_1}(s), R_{\phi_2}(s)\right).
\]

□
C ADDITIONAL TABLES OF RESULTS

Table 5 and 6 present the performances of our QPM over the high-dimensional modular case studies: GRN6 and MRH8 respectively. They compare the accuracy (rate of correct, uncertain, wrong and false positive predictions), the coverage and the efficiency of the proposed solution over all the different case studies.

<table>
<thead>
<tr>
<th>MRH8 correct</th>
<th>uncertain</th>
<th>wrong</th>
<th>FP</th>
<th>coverage</th>
<th>efficiency</th>
<th>EQR width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QR</td>
<td>67.38</td>
<td>28.75</td>
<td>3.88</td>
<td>3.75</td>
<td>89.04</td>
<td>0.2306</td>
</tr>
<tr>
<td>CQR</td>
<td>67.63</td>
<td>31.13</td>
<td>1.25</td>
<td>1.13</td>
<td>89.87</td>
<td>0.2346</td>
</tr>
<tr>
<td>Room 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QR</td>
<td>95.13</td>
<td>4.38</td>
<td>0.50</td>
<td>0.50</td>
<td>90.37</td>
<td>0.1728</td>
</tr>
<tr>
<td>CQR</td>
<td>95.25</td>
<td>4.25</td>
<td>0.50</td>
<td>0.50</td>
<td>90.31</td>
<td>0.1723</td>
</tr>
<tr>
<td>Room 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QR</td>
<td>95.88</td>
<td>3.50</td>
<td>0.63</td>
<td>0.75</td>
<td>88.55</td>
<td>0.1621</td>
</tr>
<tr>
<td>CQR</td>
<td>95.50</td>
<td>3.88</td>
<td>0.63</td>
<td>0.38</td>
<td>90.35</td>
<td>0.1670</td>
</tr>
<tr>
<td>Room 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QR</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>89.93</td>
<td>0.2593</td>
</tr>
<tr>
<td>CQR</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>89.98</td>
<td>0.2537</td>
</tr>
<tr>
<td>Room 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QR</td>
<td>95.88</td>
<td>3.00</td>
<td>1.13</td>
<td>1.13</td>
<td>89.72</td>
<td>0.1815</td>
</tr>
<tr>
<td>CQR</td>
<td>95.88</td>
<td>3.00</td>
<td>1.13</td>
<td>1.13</td>
<td>89.58</td>
<td>0.1811</td>
</tr>
<tr>
<td>Room 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QR</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>90.11</td>
<td>0.2311</td>
</tr>
<tr>
<td>CQR</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>89.99</td>
<td>0.2311</td>
</tr>
<tr>
<td>Room 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td>0.2085</td>
</tr>
<tr>
<td>CQR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td>0.2085</td>
</tr>
<tr>
<td>Room 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td>0.2085</td>
</tr>
<tr>
<td>CQR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td></td>
</tr>
<tr>
<td>Room 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td>0.2085</td>
</tr>
<tr>
<td>CQR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td>0.2085</td>
</tr>
<tr>
<td>Room 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td>0.2085</td>
</tr>
<tr>
<td>CQR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td>0.2085</td>
</tr>
<tr>
<td>Room 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td>0.2085</td>
</tr>
<tr>
<td>CQR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td>0.2085</td>
</tr>
<tr>
<td>Room 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td>0.2085</td>
</tr>
<tr>
<td>CQR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td></td>
</tr>
<tr>
<td>Room 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td>0.2085</td>
</tr>
<tr>
<td>CQR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td>0.2085</td>
</tr>
<tr>
<td>Room 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td>0.2085</td>
</tr>
<tr>
<td>CQR</td>
<td>69.25</td>
<td>30.75</td>
<td>0.00</td>
<td>0.00</td>
<td>89.77</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: 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.

Table 6: Multi room heating results with 8 rooms. Correct, uncertain, wrong and coverage represent percentage values (%). EQR width denotes the average width of the empirical quantile range over the test set.
D ADDITIONAL VISUALISATIONS
Fig. 7-14 show 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 above zero means that the state is safe, intervals below means that the state is unsafe, whereas intervals that straddle the zero means that the state is risky. Empirical samples that lie outside the EQR are denoted in a lighter orange.

Moreover, we also show the performances of our QPM at runtime for the following case studies: AAD, MRH2 and GRN2. We randomly sample two trajectories of length 20 and apply the QPM to each state in the trajectory. Results are visualized in Fig. 15-17.
Figure 7: 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 anaesthesia delivery system (AAD): property $\phi_1$ (with the globally operator).

Figure 8: Visualization of prediction intervals – cyan for QR and blue for CQR – over 40 randomly selected states for the heated tank system (HT). The central dot denotes the predicted median.

Figure 9: Visualization of prediction intervals – cyan for QR and blue for CQR – over 40 randomly selected states for a multi-room heating system with two rooms: property for room 1 (top) and property for room 2 (bottom) (MRH2). The central dot denotes the predicted median.
Figure 10: Visualization of prediction intervals – cyan for QR and blue for CQR – over 40 randomly selected states for a gene regulatory net system with two genes: property for gene 1 (top) and property for gene 2 (bottom) (GRN2). The central dot denotes the predicted median.

Figure 11: Visualization of prediction intervals – cyan for QR and blue for CQR – over 40 randomly selected states for a multi-room heating system with four rooms (MRH4): property for room 1. The central dot denotes the predicted median.

Figure 12: Visualization of prediction intervals – cyan for QR and blue for CQR – over 40 randomly selected states for a gene regulatory net system with four genes (GRN4): property for room 1. The central dot denotes the predicted median.
Figure 13: Visualization of prediction intervals – cyan for QR and blue for CQR – over 40 randomly selected states for a gene regulatory net system with six genes (GRN6): property for gene 1. The central dot denotes the predicted median.

Figure 14: Visualization of prediction intervals – cyan for QR and blue for CQR – over 40 randomly selected states for a multi-room heating system with eight rooms (MRH8): property for room 1. The central dot denotes the predicted median.

Figure 15: Runtime evaluation of the AAD model w.r.t. property $\phi_1$ (eventually operator) over two randomly selected initial states.
Figure 16: Runtime evaluation of the MRH2 model over two randomly selected initial states.

Figure 17: Runtime evaluation of the GRN2 model over two randomly selected initial states.