Predictive Monitoring with Logic-Calibrated Uncertainty for Cyber-Physical Systems

MEIYI MA and JOHN STANKOVIC, University of Virginia, USA
EZIO BARTOCCI, TU Wien, Austria
LU FENG, University of Virginia, USA

Predictive monitoring—making predictions about future states and monitoring if the predicted states satisfy requirements—offers a promising paradigm in supporting the decision making of Cyber-Physical Systems (CPS). Existing works of predictive monitoring mostly focus on monitoring individual predictions rather than sequential predictions. We develop a novel approach for monitoring sequential predictions generated from Bayesian Recurrent Neural Networks (RNNs) that can capture the inherent uncertainty in CPS, drawing on insights from our study of real-world CPS datasets. We propose a new logic named *Signal Temporal Logic with Uncertainty* (STL-U) to monitor a flowpipe containing an infinite set of uncertain sequences predicted by Bayesian RNNs. We define STL-U strong and weak satisfaction semantics based on whether all or some sequences contained in a flowpipe satisfy the requirement. We also develop methods to compute the range of confidence levels under which a flowpipe is guaranteed to strongly (weakly) satisfy an STL-U formula. Furthermore, we develop novel criteria that leverage STL-U monitoring results to calibrate the uncertainty estimation in Bayesian RNNs. Finally, we evaluate the proposed approach via experiments with real-world CPS datasets and a simulated smart city case study, which show very encouraging results of STL-U based predictive monitoring approach outperforming baselines.

CCS Concepts: • **Theory of computation → Logic**; • **Computing methodologies → Machine learning**; • **Computer systems organization → Embedded and cyber-physical systems**.

Additional Key Words and Phrases: Predictive Monitoring, Uncertainty

ACM Reference Format:
Meiyi Ma, John Stankovic, Ezio Bartocci, and Lu Feng. 2021. Predictive Monitoring with Logic-Calibrated Uncertainty for Cyber-Physical Systems. *ACM Trans. Embedd. Comput. Syst.* 1, 1, Article 1 (January 2021), 25 pages. https://doi.org/10.1145/3477032

1 INTRODUCTION

Predictive monitoring concerns the problem of (continuously) making predictions about future states and monitoring if the predicted states satisfy or violate requirements. Predictive monitoring offers a promising paradigm in supporting the decision making of Cyber-Physical Systems (CPS), for example, reducing an automated insulin delivery system's dosage if a potentially dangerous hypoglycemic condition is predicted, and adapting a traffic control system's signaling if traffic

This article appears as part of the ESWEEK-TECS special issue and was presented in the International Conference on Embedded Software (EMSOFT), 2021.

Authors' addresses: Meiyi Ma, meiyi@virginia.edu; John Stankovic, stankovic@virginia.edu, University of Virginia, 85 Engineer's Way, P.O. Box 400740, Charlottesville, VA, 22903, USA; Ezio Bartocci, ezio.bartocci@tuwien.ac.at, TU Wien, Austria; Lu Feng, lu.feng@virginia.edu, University of Virginia, 85 Engineer's Way, P.O. Box 400740, Charlottesville, VA, 22903, USA.

1539-9087/2021/1-ART1 $15.00
https://doi.org/10.1145/3477032

congestion due to car accidents or inclement weather is forecast. On the one hand, various machine learning and statistical analysis techniques (e.g., neural networks, ARIMA) have been popularly applied to predict future states of CPS across different application domains, such as predicting glucose levels for artificial pancreas systems [32], predicting takeover reaction time for automated vehicles [34], forecasting air quality [25], fire risk [39], and frost damage [43] in smart cities. On the other hand, there have been great efforts over the past decades devoted to develop runtime monitoring techniques and tools. For example, a survey of specification (e.g., Signal Temporal Logic (STL) [31]) based runtime monitoring of CPS is provided in [3]. Nevertheless, research on predictive monitoring that addresses challenges arisen from combining these two aspects has received scant attention until very recently. Existing works of predictive monitoring (e.g., [1, 6]) mostly focus on monitoring individual predictions rather than sequential predictions. A more recent work [35] considers STL-based monitoring for predictions made from statistical time-series analysis, assuming that a joint probability distribution of predictions over multiple time-points can be estimated.

In this paper, we develop a novel approach for monitoring sequential predictions generated from Bayesian Recurrent Neural Networks (RNN) models. RNN-based sequential prediction has been widely used in CPS applications (e.g., [16, 26]). Many commonly used RNN models (e.g., LSTM) are deterministic, which generate the same sequence of predictions given the same set of historical states. A key challenge is how to generate and monitor predictions that can capture the inherent uncertainty in CPS (e.g., due to sensing noise, human interactions). We study two real-world CPS datasets to analyze the uncertainty characteristics and implications on predictive monitoring. Insights from our study show that (i) deterministic RNN models are not suitable for representing the significant uncertainty exhibited in CPS, and (ii) there is a need for developing new monitors for checking sequential predictions with high uncertainty.

To address the first insight, we apply stochastic regularization techniques (SRTs) [12] to cast deterministic RNNs as Bayesian RNNs, which adapt deterministic sequential predictions as a sequence of posterior probability distributions to estimate the uncertainty. We formally define a *flowpipe* signal to represent uncertain sequential predictions generated by Bayesian RNNs. The projection of a flowpipe for a single time-point is a confidence interval induced from a Gaussian distribution, which includes values of all possible sequences predicted by the Bayesian RNN. A larger confidence interval indicates a higher level of uncertainty about the prediction.

To address the second insight, we propose a new logic named *Signal Temporal Logic with Uncertainty* (STL-U). Existing temporal logic based monitors (e.g., STL and its variants) mostly focus on deterministic signals and cannot be directly applied for monitoring an infinite set of sequences contained in a flowpipe. Several recent works (e.g., [18, 23, 24, 38]) extend STL with stochastic predicates to reason about uncertainty. Our approach differs from these previous works fundamentally. Instead of reasoning about the probability of satisfying a predicate, STL-U checks a flowpipe signal containing an infinite set of sequences. For example, consider a STL-U formula $\square_{[0,2]}\mathrm{AQI}_{\varepsilon=95\%} < 50$, which represents the requirement "the predicted Air Quality Index under 95% confidence level should never exceed 50 in the next two hours". We develop a STL-U monitor that checks if all (*resp.* some) sequences contained in the predicated flowpipe satisfy the requirement, which we call STL-U strong (*resp.* weak) satisfaction. In addition, we equip the STL-U monitor with the capability to answer queries such as "Under what confidence level, the predicated flowpipe is guaranteed to strongly (weakly) satisfy the STL-U formula?" It is particularly useful to compute such confidence guarantees when users do not know *a priori* about the level of prediction uncertainty.

Furthermore, the quality of predictive monitoring results depends on the uncertainty estimated from Bayesian RNNs, which varies based on the choice of uncertainty estimation schemas (e.g., SRTs and dropout rates). In the current practice, an uncertainty estimation schema is often selected empirically or guided by traditional deep learning metrics (e.g., mean square error, negative log-likelihood, KL divergence), which tend to over-estimate the uncertainty level [13, 42]. In addition, these metrics treat the uncertainty estimation of each individual value in a predicted sequence separately, and thus lack an integrated view about the uncertainty of the sequence. To address this limitation, we develop novel criteria that leverage STL-U monitoring results to select and tune uncertainty estimation schemas. Such STL-U criteria can help to calibrate the uncertainty estimates for predictive monitoring.

We compare STL-U criteria with state-of-the-art baselines via experimental evaluation on real-world CPS datasets. The results are very promising: STL-U criteria outperform all six baselines in terms of F1-scores comparing STL-U monitoring results for the predicted and target sequences. In addition, experiments also show that STL-U uncertainty calibration is compatible with different types of RNN models.

Finally, we evaluate the STL-U based predictive monitoring approach via a simulated smart city case study with 10 smart services and 390 requirements. Experiment results demonstrate the efficiency of the approach. In one case, it only takes around 281 seconds to monitor 130,000 predicted flowpipes. Moreover, our approach can better support decision making in the simulated smart city. The simulation results show that our approach improves various city performance metrics (e.g., emergency waiting time, vehicle waiting number) significantly when compared with two baselines.

Contributions. We summarize the major contributions of this paper as follows.

- We develop a novel STL-U based predictive monitoring approach for CPS, which continuously monitors uncertain sequential predictions about future states generated by Bayesian RNN models.
- We create novel STL-U criteria for calibrating uncertainty estimation in Bayesian deep learning.
- We evaluate the proposed approach via real-world smart city datasets and a simulated smart city case study, which show encouraging results.

Paper Organization. In the rest of the paper, we describe the motivating study of CPS uncertainty in Section 2, provide an overview of STL-U based predictive monitoring approach in Section 3, propose STL-U logic and monitoring algorithms in Section 4, present STL-U criteria for uncertainty calibration in Section 5, describe the evaluation results in Section 6, discuss the related work in Section 7, and draw conclusions in Section 8.

2 MOTIVATING STUDY

In this section, we study the following real-world smart city datasets as motivating examples to analyze uncertainty characteristics and to discuss implications on predictive monitoring for CPS.

(1) *Air quality dataset* [25] collected by Microsoft Research from 437 air quality monitoring stations in China during the period of 5/1/2014 to 4/30/2015, which includes 2,891,393 records of air quality index (AQI).
(2) *Traffic volume dataset* [33] collected by the NYC Department of Transportation from 1,490 street segments in the New York City during the period of 9/13/2014 to 4/5/2018, which includes 514,776 records of traffic volume count.

Uncertainty characteristics. We made the following observations by analyzing these datasets.

- *Significant uncertainty exists in smart cities and the uncertainty level varies across different locations.* As an illustrative example, Figure 1 shows 10-hour data segments taken from three different stations in the air quality dataset. We preprocessed the raw data by averaging

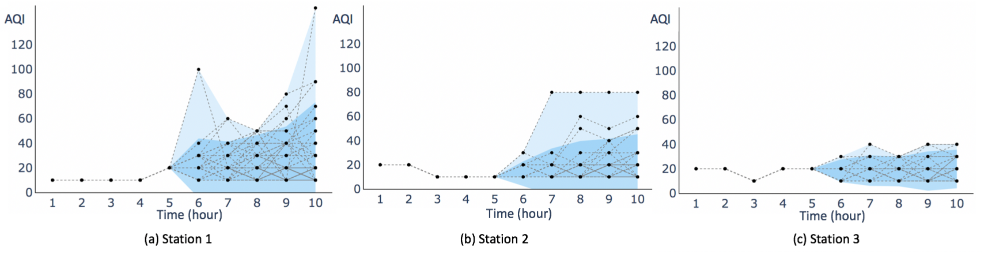


Fig. 1. The uncertainty level varies across different stations in the air quality dataset.

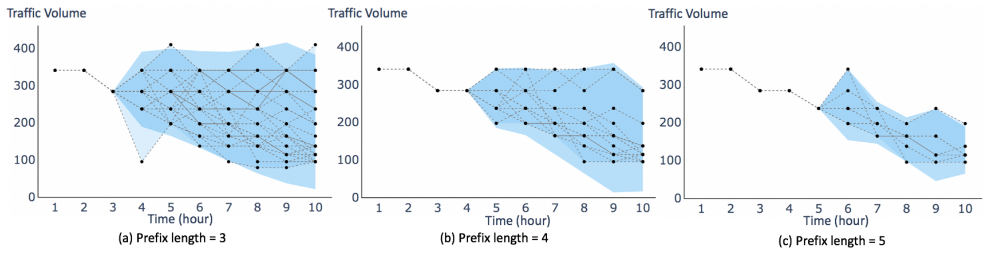


Fig. 2. The uncertainty level varies for different pre-knowledge (prefix lengths) in the traffic volume dataset.

the data within an hour and performing a uniform quantization [20]. Figure 1 plots data segments with the same prefixes (i.e., the same average AQI levels) for the first five hours. However, these data segments show significant uncertainty in the suffixes. The light shadows in the figure cover the entire data range, and the dark shadows represent the range of 95% percentile [1] of the corresponding normal distribution at a time. A larger range of 95% percentile indicates a higher level of data uncertainty. Thus, Figure 1 shows that station 1 has the highest uncertainty level, followed by station 2 and station 3.

- *The data uncertainty level is affected by the pre-knowledge (i.e., the prefix length of data segments).* As an illustrative example, Figure 2 plots 10-hour data segments taken from the same location in the traffic volume dataset. We preprocessed the data by averaging the traffic volume counts within an hour and performing a logarithmic quantization [20]. Figure 2 shows that, as the length of common data segment prefixes increases (i.e., more pre-knowledge about the data), the uncertainty level reduces.

The uncertainty in CPS data could arise from many sources, such as noise from the sensing data (e.g., reading errors, faults, anomalies), the environment (e.g., unexpected weather or events like accidents), and human behaviors (e.g., interventions from human operators), to name a few. Thus, predictive monitoring for CPS should account for the impact of uncertainty.

Implications on predictive monitoring. We discuss how the uncertainty in CPS would affect predictive monitoring from two aspects: (i) prediction, and (ii) monitoring.

First, existing deterministic prediction models (e.g., RNNs) mostly forecast future states based on historical states. Given the same historical data, a deterministic model always yields the same prediction about future states. However, as discussed above, real-world CPS data exhibits significant uncertainty. For example, Figure 1 illustrates that data segments with the same average AQI levels for the first five hours can lead to a diverse range of trends for the following five hours. Thus, deterministic prediction models are not suitable to capture the uncertainty in CPS data. There is

[1]We utilize 95% percentile because it is commonly used to represent the majority of the population distributed [37].

Table 1. Number of satisfying data segments for each STL formula.

STL formulas	$\lambda = 50$	$\lambda = 51$	$\lambda = 70$	$\lambda = 75$	$\lambda = 80$
$\Box_{[0,10]}\text{AQI} < \lambda$	2,807	2,895	3,614	4,011	4,443
$\Diamond_{[0,10]}\text{AQI} < \lambda$	6,670	6,731	7,304	7,408	7,493
$\text{AQI} < 150\ \mathcal{U}_{[0,10]}\ \text{AQI} < \lambda$	5,558	5,613	6,030	6,169	6,230
$\Box_{[0,10]}\text{Traffic} < \lambda$	1,241	1,359	3,090	3,332	3,532
$\Diamond_{[0,10]}\text{Traffic} < \lambda$	4,220	4,283	4,546	4,563	4,598

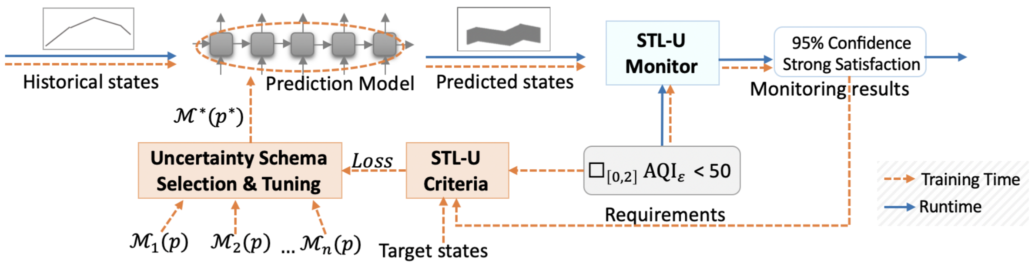


Fig. 3. Overview of STL-U based predictive monitoring approach.

a need for developing new techniques that can predict future states with appropriate levels of uncertainty.

Second, existing works (e.g., [29]) have applied Signal Temporal Logic (STL) to specify and monitor city requirements. For example, a requirement that "the AQI level within 10 hours should always be below certain threshold λ" can be specified with a STL formula $\Box_{[0,10]}(\text{AQI} < \lambda)$, where $\Box$ is the temporal logical operator representing "always" and λ is a parameter (e.g., $\lambda = 50$ for good air quality). Table 1 shows five example STL formulas, with the first three representing city requirements about AQI and the last two representing city requirements about traffic volume. We applied a STL monitor to check how many 10-hour data segments of a selected location in the air quality and traffic volume datasets satisfy these STL formulas with varying parameter values of λ. We observe from Table 1 that the STL monitoring results can be very sensitive to the change of λ values. For example, the number of satisfying data segments for $\Diamond_{[0,10]}(\text{AQI} < \lambda)$ increases by 61 (from 6,670 to 6,731) when the λ value only increases by 1 (from 50 to 51), and goes up 85 (from 7,408 to 7,493) when the λ value increases from 75 to 80. Even though the differences also vary by the type of requirements, the amount is still too large to ignore. From the perspective of the data, it shows that a small difference of the data could completely change the monitoring results. However, it is impossible to predict the data with 100% accuracy due to the existence of uncertainty in CPS, which makes the monitoring results less effective to support decision making. It also indicates that the existing monitors are not suitable for data with high uncertainty. Therefore, there is a need for developing new monitors that can check the prediction results accounting for the uncertainty.

3 APPROACH OVERVIEW

We develop a novel predictive monitoring approach for CPS, as illustrated in Figure 3, to address limitations discussed in the previous section. Our approach adopts Bayesian RNN models to predict future states (e.g., AQI in next 2 hours) based on historical data (e.g., AQI in the past 5 hours). By contrast to deterministic prediction models that output a single sequence of predicted values, Bayesian RNN models generate a sequence of distributions to capture the uncertainty of predicted future states, which are represented by a range of potential values under a given confidence level at each time-point. We propose a new logic named *Signal Temporal Logic with Uncertainty* (STL-U) and develop a STL-U monitor to check such uncertain sequential predictions generated by Bayesian RNN models. STL-U is expressive enough to specify CPS requirements with uncertainty confidence levels. For example, a STL-U formula $\square_{[0,2]}\mathrm{AQI}_{\varepsilon=95\%} < 50$ represents the requirement "the predicted AQI under 95% confidence level should never exceed 50 in the next two hours". The STL-U monitor checks if all or some possible sequences of future states predicted by the Bayesian RNN model satisfy the requirement, which we call strong and weak satisfaction relations. When the confidence level is unspecified in a formula (e.g., $\square_{[0,2]}\mathrm{AQI}_{\varepsilon=?} < 50$), we can also use STL-U monitor to compute the range of confidence levels under which the predicted flowpipe is guaranteed to strongly or weakly satisfy a requirement. In addition, we develop novel criteria (loss functions) based on STL-U monitoring results to calibrate the uncertainty estimation in Bayesian deep learning.

At *training* time (the flow marked by orange dash-lines in Figure 3), the proposed approach automatically selects and tunes an optimal uncertainty estimation schema based on STL-U criteria. As we will discuss in Section 5, such uncertainty calibration is an essential step to guarantee the quality of predictive monitoring, in order to better support decision making of CPS. At *runtime* (the flow marked by the blue lines in Figure 3), the proposed approach runs as a continuous iterative process to monitor the predicted future states. Considering the predictive monitoring of AQI in a smart city, for example, at time t, the proposed approach first predicts the AQI for the future 3 hours from time t and monitors if the predictions satisfy the requirements; after a period Δt (e.g., 30 minutes), it predicts the AQI for the future 3 hours from $t + \Delta t$ and checks if the new predictions satisfy the requirements. In this way, the proposed approach provides continuous predictive monitoring of future states to support decision making of CPS.

4 STL-U MONITOR

As described in the previous section, our approach adopts Bayesian RNN models to make sequential predictions about uncertain future states. We propose a *Signal Temporal Logic with Uncertainty* (STL-U) to monitor such uncertain sequential predictions. We introduce STL-U syntax and semantics in Section 4.1, and present methods to compute STL-U confidence guarantees in Section 4.2.

4.1 STL-U Syntax and Semantics

We formally define a new type of signals called *flowpipes*[2] to represent uncertain sequential predictions generated by Bayesian RNN models. We describe more details about Bayesian RNN models and uncertainty estimation later in Section 5.

Definition 1 (Flowpipe). *A single-variable flowpipe Ω is defined over a finite discrete time domain $\mathbb{T}$ such that $\Omega[t] = \Phi_t$ at any time $t \in \mathbb{T}$ and Φ_t is a Gaussian distribution $\mathcal{N}(\theta_t, \sigma_t^2)$. Let $\omega : \{\Omega\}^n$ be a (multi-variable) flowpipe signal, where $n = |X|$ is the size of a finite set of (independent) real variables X. Each variable $\mathrm{x} \in X$ has a corresponding flowpipe ω_{x} whose value at time t follows a Gaussian distribution Φ_t, denoted by $\omega_{\mathrm{x}}[t] = \Phi_t$.*

Given a confidence level $\varepsilon \in (0, 1) \subseteq \mathbb{R}$, a single-variable flowpipe Ω at time t is bounded by a confidence interval $[\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)]$ with the lower bound $\Phi_t^-(\varepsilon) = \theta_t - \delta \cdot \frac{\sigma_t}{\sqrt{N}}$ and the upper bound $\Phi_t^+(\varepsilon) = \theta_t + \delta \cdot \frac{\sigma_t}{\sqrt{N}}$, where N is the number of samples that the Gaussian distribution is estimated from, and δ is a function $\delta = F^{-1}(\frac{\varepsilon}{2})$ with F denoting the CDF of the standard normal distribution $\mathcal{N}(0, 1)$ [37]. In the special case where the Gaussian distribution's variance is $\sigma_t = 0$, a flowpipe signal becomes a single trace because the lower and upper bounds of the confidence interval coincide (i.e., $\Phi_t^-(\varepsilon) = \Phi_t^+(\varepsilon) = \theta_t$). Given a (multi-variable) trace $\bar{\omega}$ and a flowpipe ω over the same

[2]To be noted, here we use the concept of flowpipes but define it in a new way.

set of real variables X, we say that $\bar{\omega}$ belongs to ω, denoted by $\bar{\omega} \in \omega$, if $\bar{\omega}_{\mathrm{x}}[t] \in [\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)]$ for all $\mathrm{x} \in X$ and $t \in \mathbb{T}$, where $[\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)]$ is the confidence interval of flowpipe ω_{x} under confidence level ε.

Definition 2 (STL-U Syntax). *A STL-U formula φ over a flowpipe signal ω is given by*

$$\varphi := \mu_{\mathrm{x}}(\varepsilon) \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \Box_I \varphi \mid \Diamond_I \varphi \mid \varphi_1 \,\mathcal{U}_I\, \varphi_2$$

where $\mu_{\mathrm{x}}(\varepsilon)$ is an atomic predicate over variable x *with confidence level ε, whose value is determined by $\mu_{\mathrm{x}}(\varepsilon) \equiv f(x) > 0$ with a continuous function $f(x)$ about flowpipe ω_{x} under confidence level ε. Temporal operators $\Box_I$, $\Diamond_I$ and $\mathcal{U}_I$ with a time interval $I \subseteq \mathbb{T}$ represent (bounded) "always", "eventually", and "until", respectively.*

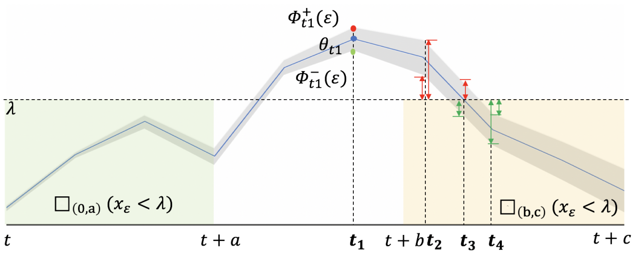


Fig. 4. An example flowpipe under the confidence level ε.

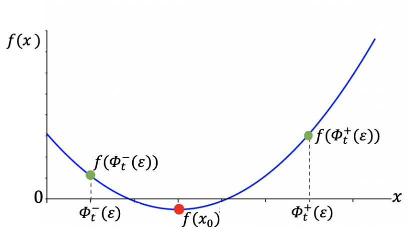


Fig. 5. An example function $f(x)$.

We define the semantics of a flowpipe signal ω satisfying a STL-U formula φ at time t by two indices: *strong satisfaction*, denoted by $(\omega, t) \models_s \varphi$; and *weak satisfaction*, denoted by $(\omega, t) \models_w \varphi$.

Definition 3. *STL-U strong satisfaction semantics.*

$$\begin{aligned}
(\omega,t) \models_s \mu_{\mathrm{x}}(\varepsilon) &\Leftrightarrow \forall x \in [\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)], f(x) > 0\\
(\omega,t) \models_s \neg\varphi &\Leftrightarrow (\omega,t) \not\models_w \varphi\\
(\omega,t) \models_s \varphi_1 \wedge \varphi_2 &\Leftrightarrow (\omega,t) \models_s \varphi_1 \textit{ and } (\omega,t) \models_s \varphi_2\\
(\omega,t) \models_s \Box_I \varphi &\Leftrightarrow \forall t' \in (t+I), (\omega,t') \models_s \varphi\\
(\omega,t) \models_s \Diamond_I \varphi &\Leftrightarrow \exists t' \in (t+I), (\omega,t') \models_s \varphi\\
(\omega,t) \models_s \varphi_1 \mathcal{U}_I \varphi_2 &\Leftrightarrow \exists t' \in (t+I) \cap \mathbb{T}, (\omega,t') \models_s \varphi_2 \textit{ and } \forall t'' \in (t,t'), (\omega,t'') \models_s \varphi_1
\end{aligned}$$

Definition 4. *STL-U weak satisfaction semantics.*

$$\begin{aligned}
(\omega,t) \models_w \mu_{\mathrm{x}}(\varepsilon) &\Leftrightarrow \exists x \in [\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)], f(x) > 0\\
(\omega,t) \models_w \neg\varphi &\Leftrightarrow (\omega,t) \not\models_s \varphi\\
(\omega,t) \models_w \varphi_1 \wedge \varphi_2 &\Leftrightarrow (\omega,t) \models_w \varphi_1 \textit{ and } (\omega,t) \models_w \varphi_2\\
(\omega,t) \models_w \Box_I \varphi &\Leftrightarrow \forall t' \in (t+I), (\omega,t') \models_w \varphi\\
(\omega,t) \models_w \Diamond_I \varphi &\Leftrightarrow \exists t' \in (t+I), (\omega,t') \models_w \varphi\\
(\omega,t) \models_w \varphi_1 \mathcal{U}_I \varphi_2 &\Leftrightarrow \exists t' \in (t+I) \cap \mathbb{T}, (\omega,t') \models_w \varphi_2 \textit{ and } \forall t'' \in (t,t'), (\omega,t'') \models_w \varphi_1
\end{aligned}$$

To be noted, the negation of strong satisfaction is equivalent to weak violation, and the negation of weak satisfaction is equivalent to strong violation. Intuitively, strong satisfaction means that all values bounded within the confidence interval of a flowpipe should satisfy the STL-U formula, while weak satisfaction means that there exist some value within the confidence interval of a flowpipe satisfying the STL-U formula. In CPS applications, strong satisfaction relations can be used for monitoring strict requirements (e.g., safety), while weak satisfaction relations can be used for monitoring soft constraints (e.g., energy consumption).

Figure 4 shows an example flowpipe signal under the confidence level ε, representing predictions from time t to $t + c$. At a time-point t_1, the flowpipe follows a Gaussian distribution Φ_{t_1} with the mean of θ_{t_1} and is bounded by a confidence interval $[\Phi^-_{t_1}(\varepsilon), \Phi^+_{t_1}(\varepsilon)]$. This flowpipe signal strongly satisfies STL-U formula $\square_{(0,a)}(x_\varepsilon < \lambda)$ at time t, because the flowpipe signal values bounded within the confidence interval from time t to $t + a$ are all below the threshold λ (see the left green zone in Figure 4). Consider another STL-U formula $\square_{(b,c)}(x_\varepsilon < \lambda)$. As shown in Figure 4 (yellow zone in the right), the flowpipe's confidence interval is entirely above the threshold λ at time t_2, partially below λ at time t_3, and entirely below λ at time t_4. Therefore, the flowpipe neither strongly nor weakly satisfies the STL-U formula $\square_{(b,c)}(x_\varepsilon < \lambda)$ at time t.

Theorem 1 (Strength relation theorem). *If a flowpipe ω strongly satisfies a STL-U formula φ at time t, then the weak satisfaction relation also holds. On the other hand, if the flowpipe ω does not weakly satisfy a STL-U formula φ at time t, then it would also not strongly satisfy φ. Formally,*

$$\begin{aligned} (\omega, t) \models_s \varphi \quad &\Rightarrow (\omega, t) \models_w \varphi \\ (\omega, t) \not\models_w \varphi \quad &\Rightarrow (\omega, t) \not\models_s \varphi \end{aligned}$$

We include the proof of Theorem 1 and properties of STL-U semantics in the Appendix.

It is challenging to monitor STL-U strong and weak satisfactions for a flowpipe that contains an infinite set of sequences. Take the atomic predicate $\mu_{\mathsf{x}}(\varepsilon)$ as an example. Based on Definition 3, a flowpipe strongly satisfies $\mu_{\mathsf{x}}(\varepsilon)$ iff $f(x) > 0$ for all $x \in [\Phi^-_t(\varepsilon), \Phi^+_t(\varepsilon)]$. It is computationally expensive if not infeasible to exhaustively search through the entire confidence interval. In addition, it does not suffice to only check the lower and upper bounds of the confidence interval when $f(x)$ is a non-monotonic function. Figure 5 shows an example where $f(\Phi^-_t(\varepsilon)) > 0$ and $f(\Phi^+_t(\varepsilon)) > 0$, but there is a $x_0 \in [\Phi^-_t(\varepsilon), \Phi^+_t(\varepsilon)]$ with $f(x_0) < 0$. We tackle this challenge by computing the minimal value f_{min} of $f(x)$ for $x \in [\Phi^-_t(\varepsilon), \Phi^+_t(\varepsilon)]$ (e.g., via minimization algorithms in [8]). If $f_{\mathsf{min}} > 0$, which implies that $f(x) > 0$ for all $x \in [\Phi^-_t(\varepsilon), \Phi^+_t(\varepsilon)]$, then the flowpipe strongly satisfies $\mu_{\mathsf{x}}(\varepsilon)$. Based on Definition 4, a flowpipe weakly satisfies $\mu_{\mathsf{x}}(\varepsilon)$ iff there exist some $x \in [\Phi^-_t(\varepsilon), \Phi^+_t(\varepsilon)]$ such that $f(x) > 0$. For monitoring weak satisfaction, we compute the maximal value f_{max} of $f(x)$ for any $x \in [\Phi^-_t(\varepsilon), \Phi^+_t(\varepsilon)]$ and check if $f_{\mathsf{max}} > 0$. We include pseudo code of monitoring algorithms for STL-U strong and weak satisfactions as Algorithm 1 and Algorithm 2 in the Appendix. Following Definition 1, we use an array of triplets $\langle t, \theta_t, \sigma_t \rangle$, $t \in \mathbb{T}$ to represent a flowpipe in STL-U monitoring algorithms. Given θ, σ, and ε, calculating $\Phi^+_t(\varepsilon)$ and $\Phi^-_t(\varepsilon)$ takes a constant time $O(1)$, which could be further accelerated by caching the intermediate results. The time complexity of calculating the predicate $f(x) > 0$ depends on the complexity of $f(x)$ and the selected minimization algorithms. The time complexity of STL-U monitoring algorithms is similar to STL monitoring algorithms. Thus, STL-U can be used to monitor complex specifications (e.g., with multiple levels of nesting temporal operators) via using Algorithm 1 or Algorithm 2 recursively.

4.2 STL-U Confidence Guarantees

It may not always be possible for users to specify a confidence level for a flowpipe *a priori*. It is therefore useful to query about, under what confidence level, a flowpipe is guaranteed to strongly (weakly) satisfy a STL-U formula. We present methods to compute such confidence guarantees as follows.

Let $\epsilon_s(\varphi, \omega, t)$ and $\epsilon_w(\varphi, \omega, t)$ denote the range of confidence levels that guarantee a flowpipe signal ω strongly and weakly satisfying a STL-U formula φ at time t, respectively. Let ϵ^c_s (*resp.* ϵ^c_w) denotes the complement set of ϵ_s (*resp.* ϵ_w) within the interval $(0, 1)$.

Definition 5. *Confidence guarantees for STL-U strong satisfaction.*

$$\begin{aligned}
\epsilon_s(\mu_{\mathrm{x}}, \omega, t) &= \left(0, \int_{\theta_t-\eta}^{\theta_t+\eta} \Phi_t(x)dx\right), \text{where } \eta = \inf\left\{|x-\theta_t| \,\middle|\, f(x) \le 0\right\} \\
\epsilon_s(\neg\varphi, \omega, t) &= \epsilon_w^c(\varphi, \omega, t) \\
\epsilon_s(\varphi_1 \wedge \varphi_2, \omega, t) &= \epsilon_s(\varphi_1, \omega, t) \cap \epsilon_s(\varphi_2, \omega, t) \\
\epsilon_s(\Box_I\varphi, \omega, t) &= \bigcap_{t' \in (t+I)} \epsilon_s(\varphi, \omega, t') \\
\epsilon_s(\Diamond_I\varphi, \omega, t) &= \bigcup_{t' \in (t+I)} \epsilon_s(\varphi, \omega, t') \\
\epsilon_s(\varphi_1 \mathcal{U}_I \varphi_2, \omega, t) &= \bigcup_{t' \in (t+I)} \left\{\epsilon_s(\varphi_2, \omega, t') \cap (\bigcap_{t'' \in (t,t')} \epsilon_s(\varphi_1, \omega, t''))\right\}
\end{aligned}$$

Definition 6. *Confidence guarantees for STL-U weak satisfaction.*

$$\begin{aligned}
\epsilon_w(\mu_{\mathrm{x}}, \omega, t) &= \left(\int_{\theta_t-\eta}^{\theta_t+\eta} \Phi_t(x)dx, 1\right), \text{ where } \eta = \inf\left\{|x-\theta_t| \,\middle|\, f(x) > 0\right\} \\
\epsilon_w(\neg\varphi, \omega, t) &= \epsilon_s^c(\varphi, \omega, t) \\
\epsilon_w(\varphi_1 \wedge \varphi_2, \omega, t) &= \epsilon_w(\varphi_1, \omega, t) \cap \epsilon_w(\varphi_2, \omega, t) \\
\epsilon_w(\Box_I\varphi, \omega, t) &= \bigcap_{t' \in (t+I)} \epsilon_w(\varphi, \omega, t') \\
\epsilon_w(\Diamond_I\varphi, \omega, t) &= \bigcup_{t' \in (t+I)} \epsilon_w(\varphi, \omega, t') \\
\epsilon_w(\varphi_1 \mathcal{U}_I \varphi_2, \omega, t) &= \bigcup_{t' \in (t+I)} \left\{\epsilon_w(\varphi_2, \omega, t') \cap (\bigcap_{t'' \in (t,t')} \epsilon_w(\varphi_1, \omega, t''))\right\}
\end{aligned}$$

Theorem 2. *Given a flowpipe signal ω and a STL-U formula φ, ω is guaranteed to strongly satisfy φ at time t under a confidence level $\varepsilon \in \epsilon_s(\varphi, \omega, t)$ computed based on Definition 5.*

Theorem 3. *Given a flowpipe signal ω and a STL-U formula φ, ω is guaranteed to weakly satisfy φ at time t under a confidence level $\varepsilon \in \epsilon_w(\varphi, \omega, t)$ computed based on Definition 6.*

We include proofs of the above theorems in the Appendix.

In the following, we explain the intuition behind our methods. Figure 6(a) plots the normal density curve of a Gaussian distribution Φ_t with the mean θ_t. A confidence level ε represents the probability that the corresponding confidence interval $[\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)]$ contains a target value, calculated as the percentage of the area of the normal density curve. When ε approaches 0, the confidence interval shrinks to a single point θ_t; and when ε approaches 1, the confidence interval expands to $(-\infty, \infty)$. In general, the larger the value of ε, the wider the confidence interval range. For example, Figure 6(a) shows confidence intervals for two confidence levels ε_1, ε_2 and $\varepsilon_1 < \varepsilon_2$.

To compute the range of confidence levels that guarantee the strong satisfaction of a STL-U formula φ, we first determine the smallest distance η between the mean θ_t and the set of x values that violate φ. Any x value within the interval $(\theta_t - \eta, \theta_t + \eta)$ should satisfy φ. Thus, we compute the integral $\int_{\theta_t-\eta}^{\theta_t+\eta} \Phi_t(x)dx$ as the upper bound of confidence level guarantee for strong satisfaction, denoted by ϵ_s^+. Under any confidence level $\varepsilon \in (0, \epsilon_s^+)$, the flowpipe is guaranteed to strongly satisfy the STL-U formula φ. In the special case when the mean value θ_t violates φ, we have $\eta = 0$ and $\epsilon_s^+ = 0$; thus, there does not exist a feasible value of ε, under which the flowpipe strongly satisfies φ. Consider a STL-U formula $\varphi_1 : x < \lambda_1$. As shown in Figure 6(a), the flowpipe under ε_1 strongly satisfies φ_1 because $\varepsilon_1 \in (0, \epsilon_s^+)$, while the flowpipe under ε_2 does not strongly satisfy φ_1 because $\varepsilon_2 > \epsilon_s^+$.

To compute the range of confidence levels that guarantee the weak satisfaction of a STL-U formula φ, we find the smallest distance η between the mean θ_t and the set of x values that satisfy φ. We compute the integral $\int_{\theta_t-\eta}^{\theta_t+\eta} \Phi_t(x)dx$ as the lower bound of confidence level guarantee for weak satisfaction, denoted by ϵ_w^-. Under any confidence level $\varepsilon \in (\epsilon_w^-, 1)$, the flowpipe is guaranteed to weakly satisfy the STL-U formula φ. When there does not exist any x value satisfying φ, we have $\eta = \infty$ and $\epsilon_w^- = 1$; thus, there does not exist a feasible value of ε, under which the flowpipe weakly satisfies φ. Figure 6(b) shows the same Gaussian distribution Φ_t as in Figure 6(a). For the STL-U formula $\varphi_2 : x < \lambda_2$, the flowpipe under ε_2 weakly satisfy φ_2 because $\varepsilon_2 \in (\epsilon_w^-, 1)$; however, the flowpipe under ε_1 does not weakly satisfy φ_2, because $\varepsilon_1 < \epsilon_w^-$.

We include pseudo code of algorithms for computing confidence guarantees for STL-U strong and weak satisfaction as Algorithm 3 and Algorithm 4 in the Appendix. Here we use an example to describe the procedure of recursively computing confidence guarantees via parsing the syntax tree of a STL-U formula. Figure 7(a) shows an example flowpipe with values of mean θ and variance σ in each time step t. Consider a STL-U formula $\Box_{[1,3]}(x_\varepsilon > 8) \wedge \Diamond_{[1,3]}(x_\varepsilon < 10)$. Figure 7(b) illustrates how to iterate through the formula's syntax tree and compute confidence guarantees for strong satisfaction. First, at the left bottom of the tree, we compute the range of confidence levels that can guarantee the strong satisfaction of $x_\varepsilon > 8$, and obtain the results of $(0, 0.20)$, $(0, 0.49)$, $(0, 0.68)$ for $t \in [1, 3]$. Then, we move up the tree to compute the confidence guarantee for $\Box_{[1,3]}(x_\varepsilon > 8)$ by taking the intersection of these three ranges, which yields $(0, 0.20)$. Meanwhile, from the right branch of the tree, we obtain the range of confidence levels that guarantee the strong satisfaction of $x_\varepsilon < 10$, and taking a union of these ranges for $\Diamond_{[1,3]}(x_\varepsilon < 10)$, which yields $(0, 0.55)$. At the top of the syntax tree, we take the intersection of $(0, 0.20)$ and $(0, 0.55)$ for $\wedge$ operation, which yields $(0, 0.20)$ as the final result of confidence guarantees for strongly satisfy the STL-U formula. Figure 7(c) shows a similar process of recursively computing confidence guarantees for weak satisfaction of the same STL-U formula.

5 PREDICTION WITH LOGIC-CALIBRATED UNCERTAINTY

Recall from Section 2 that deterministic prediction models are not suitable to capture the uncertainty exhibited in CPS. To address this limitation, we adopt Bayesian RNN models in the proposed predictive monitoring approach. We describe how to build Bayesian RNN models for prediction and motivate the need for uncertainty calibration in Section 5.1. Then, we define STL-U based criteria for uncertainty calibration in Section 5.2.

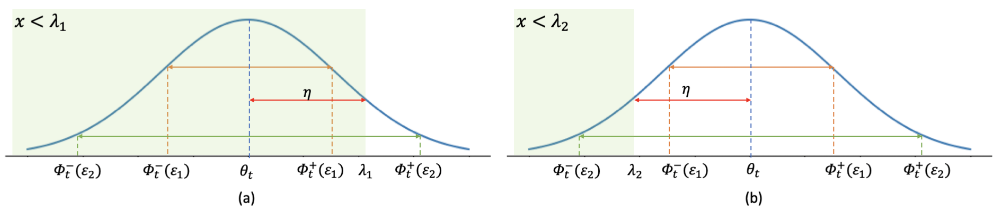

$x < \lambda_1$
η
$\Phi_t^-(\varepsilon_2)$
$\Phi_t^-(\varepsilon_1)$
θ_t
$\Phi_t^+(\varepsilon_1)$
λ_1
$\Phi_t^+(\varepsilon_2)$
(a)
$x < \lambda_2$
η
$\Phi_t^-(\varepsilon_2)$
λ_2
$\Phi_t^-(\varepsilon_1)$
θ_t
$\Phi_t^+(\varepsilon_1)$
$\Phi_t^+(\varepsilon_2)$
(b)

Fig. 6. Computing confidence guarantees for STL-U formulas $x < \lambda_1$ and $x < \lambda_2$.

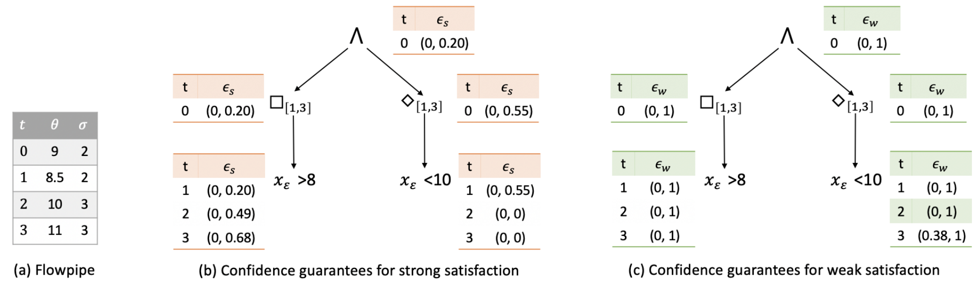

(a) Flowpipe
(b) Confidence guarantees for strong satisfaction
(c) Confidence guarantees for weak satisfaction
x_ε >8
x_ε <10
$\square_{[1,3]}$
$\lozenge_{[1,3]}$

Fig. 7. Computing confidence guarantees for STL-U formula $\square_{[1,3]}(x_\varepsilon > 8) \wedge \lozenge_{[1,3]}(x_\varepsilon < 10)$.

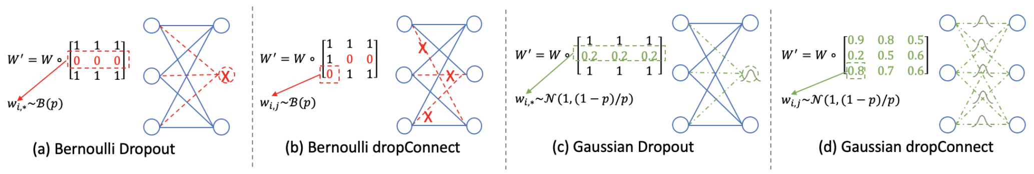

(a) Bernoulli Dropout
(b) Bernoulli dropConnect
(c) Gaussian Dropout
(d) Gaussian dropConnect
$W' = W \circ$
$w_{i,*} \sim \mathcal{B}(p)$
$w_{i,j} \sim \mathcal{B}(p)$
$w_{i,*} \sim \mathcal{N}(1,(1-p)/p)$
$w_{i,j} \sim \mathcal{N}(1,(1-p)/p)$

Fig. 8. Four commonly used SRTs.

5.1 Uncertainty Estimation with Bayesian RNN Models

Stochastic regularization techniques (SRTs) have been popularly used to cast deterministic deep learning models as Bayesian models for uncertainty estimation [11]. Given a well-trained deterministic RNN model with learnable parameters W, we can obtain a Bayesian RNN model with parameters W' via SRTs that transform W to W' by applying a $n \times n$ mask, where n is the number of neurons in each layer. Elements of the mask w are sampled from some probability distribution. The connection from neuron j to neuron i would be dropped if $w_{ij} = 0$, the connection remains the same if $w_{ij} = 1$, and a weight $\beta \in (0, 1)$ would be applied to the connection if $w_{ij} = \beta$. In this work, we consider four commonly used SRTs as illustrated in Figure 8. Let p denote the dropout rate.

- *Bernoulli dropout*: Each row of the mask is sampled from a Bernoulli distribution, denoted by $w_{i,*} \sim \mathcal{B}(p)$.
- *Bernoulli dropConnect*: Each element of the mask is sampled independently as $w_{i,j} \sim \mathcal{B}(p)$.
- *Gaussian dropout*: Each row of the mask is sampled from a Gaussian distribution, denoted by $w_{i,*} \sim \mathcal{N}(1, (1-p)/p)$.
- *Gaussian dropConnect*: Each element of the mask is sampled independently, denoted by $w_{i,j} \sim \mathcal{N}(1, (1-p)/p)$.

Figure 9 shows how to use the obtained Bayesian RNN model to predict future states based on historical states. We apply the Monte Carlo method to repeat the Bayesian RNN prediction for N times, which yield a set of sequential predictions. Thus, we can estimate a Gaussian distribution $\Phi_t \sim \mathcal{N}(\theta_t, \sigma_t^2)$ for each time step t, where the mean θ_t and variance σ_t are computed based on the Monte Carlo samples $\{x_t^{(1)}, \cdots, x_t^{(N)}\}$.

Different uncertainty estimation schemas (i.e., SRTs and dropout rates) can yield different uncertainty estimates for the same model trained with the same data. How to select the best uncertainty estimation schema for an application still remains an open question. Currently, a common practice is to pick a schema empirically, without systematically evaluating how different choices would impact the quality of uncertainty estimates. Furthermore, deep learning methods that seek to optimize the prediction accuracy may overestimate the uncertainty. For example, consider two

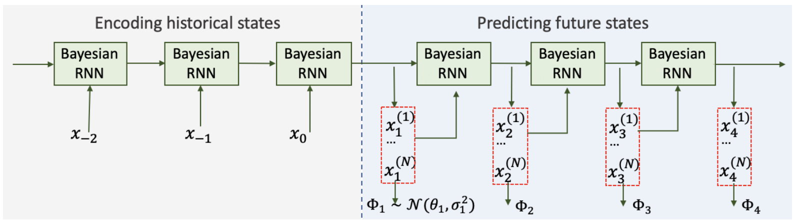

Encoding historical states
Predicting future states
Bayesian RNN
x_{-2}
x_{-1}
x_0
$x_1^{(1)}$ … $x_1^{(N)}$
$x_2^{(1)}$ … $x_2^{(N)}$
$x_3^{(1)}$ … $x_3^{(N)}$
$x_4^{(1)}$ … $x_4^{(N)}$
$\Phi_1 \sim \mathcal{N}(\theta_1, \sigma_1^2)$
Φ_2
Φ_3
Φ_4

Fig. 9. Bayesian RNN-based sequential prediction with uncertainty estimation.

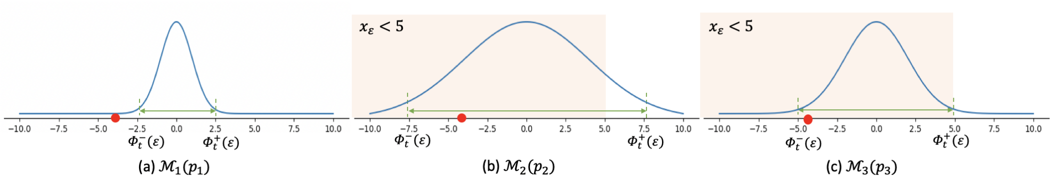


Fig. 10. Comparison of different uncertainty estimation schemas.

distributions predicted with uncertainty estimation schemas $\mathcal{M}_1(p_1)$ and $\mathcal{M}_2(p_2)$, as shown in Figure 10(a) and (b). $\mathcal{M}_2(p_2)$ is a better schema based on the metric of prediction accuracy, because the target value (red dot) falls within the confidence interval $[\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)]$ in Figure 10(b) but not in Figure 10(a). However, $\mathcal{M}_2(p_2)$ yields a higher level of uncertainty, as indicated by the larger confidence interval range.

In this work, we develop novel criteria that leverage STL-U monitoring results to select uncertainty estimation schemas. Here is an example to explain the intuition behind our approach. Consider two distributions predicted with uncertainty estimation schemas $\mathcal{M}_2(p_2)$ and $\mathcal{M}_3(p_3)$, as shown in Figure 10(b) and (c). Both distributions fulfill the accuracy metric, because their confidence intervals contain the target value (red dot). Suppose that the requirement is to check if a flowpipe strongly satisfies a STL-U formula $x_\varepsilon < 5$. As shown in Figure 10(c), the distribution predicted with schema $\mathcal{M}_3(p_3)$ strongly satisfies $x_\varepsilon < 5$, because all values in the confidence interval $[\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)]$ are smaller than 5. By contrast, the resulting distribution of schema $\mathcal{M}_2(p_2)$ does not strongly satisfy $x_\varepsilon < 5$, because some values in the confidence interval are greater than 5. Thus, based on STL-U monitoring results, we would select $\mathcal{M}_3(p_3)$ rather than $\mathcal{M}_2(p_2)$ as the uncertainty estimation schema, which also yields a tighter bound of estimated uncertainty. In the following, we formally define STL-U based criteria for selecting uncertainty estimation schemas.

5.2 STL-U Criteria for Uncertainty Calibration

As shown in Figure 11, given a predicted flowpipe ω and a target trace $\bar{\omega}$, we can calculate the loss based on monitoring results of ω and $\bar{\omega}$ with respect to a STL-U formula φ. We propose two uncertainty calibration criteria as loss functions based on STL-U satisfaction relations and confidence guarantees, denoted by $\mathcal{L}_{\text{sat}}$ and $\mathcal{L}_{\text{cf}}$, respectively.

Criterion based on STL-U satisfaction. We define $\mathcal{L}_{\text{sat}}$ based on the linear combination of three functions: $h_s(\omega, \bar{\omega}, \varphi)$ and $h_w(\omega, \bar{\omega}, \varphi)$ for evaluating if the predicted flowpipe ω and the target trace $\bar{\omega}$ are consistent in terms of strong and weak satisfaction (or violation) of the STL-U formula φ, and $h_b(\omega, \bar{\omega})$ for evaluating the prediction accuracy by checking if the target trace $\bar{\omega}$ belongs to the predicted flowpipe ω. Formally, we define

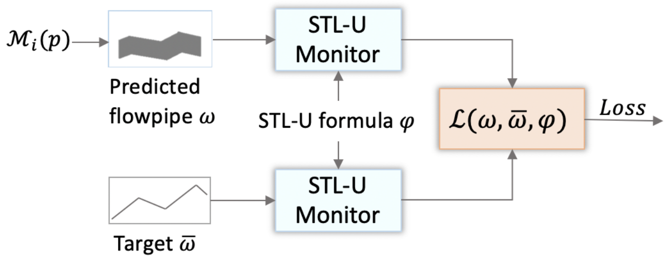


Fig. 11. STL-U criteria for uncertainty calibration computed as loss functions.

$$h_s(\omega, \bar{\omega}, \varphi) = \mathbb{1}\big((\omega \models_s \varphi \wedge \bar{\omega} \models \varphi) \vee (\omega \not\models_s \varphi \wedge \bar{\omega} \not\models \varphi)\big)$$
$$h_w(\omega, \bar{\omega}, \varphi) = \mathbb{1}\big((\omega \models_w \varphi \wedge \bar{\omega} \models \varphi) \vee (\omega \not\models_w \varphi \wedge \bar{\omega} \not\models \varphi)\big)$$
$$h_b(\omega, \bar{\omega}) = \mathbb{1}(\bar{\omega} \in \omega)$$

where $\mathbb{1}(\phi)$ is an indicator function such that $\mathbb{1}(\phi) = 1$ if ϕ = True, and $\mathbb{1}(\phi) = 0$ otherwise. The loss function is then given by

$$\mathcal{L}_{\text{sat}}(\omega, \bar{\omega}, \varphi) = 1 - (\beta_1 \cdot h_s(\omega, \bar{\omega}, \varphi) + \beta_2 \cdot h_w(\omega, \bar{\omega}, \varphi) + (1 - \beta_1 - \beta_2) \cdot h_b(\omega, \bar{\omega}))$$

where $\beta_1, \beta_2 \in (0, 1)$ are real-valued coefficients representing the relative importance of strong/weak satisfaction and prediction accuracy in different domains. The goal is to minimize the loss $\mathcal{L}_{\text{sat}}$, for which we need to maximize the linear combination of $h_s(\omega, \bar{\omega}, \varphi)$, $h_w(\omega, \bar{\omega}, \varphi)$, and $h_b(\omega, \bar{\omega})$. Intuitively, the higher quality of the prediction in terms of the consistency of STL-U monitoring results and the accuracy compared with the target trace, the lower the loss.

Criterion based on STL-U confidence guarantees. Recall from Section 4 that, in addition to checking strong/weak satisfaction relations, the STL-U monitor can also compute a range of confidence levels under which the predicted flowpipe is guaranteed to strongly/weakly satisfy a STL-U formula. Based on STL-U confidence guarantees, we define the following loss function:

$$\mathcal{L}_{\text{cf}}(\omega, \bar{\omega}, \varphi) = 1 - (\beta_1 \cdot g_s(\omega, \bar{\omega}, \varphi) + \beta_2 \cdot g_w(\omega, \bar{\omega}, \varphi) + (1 - \beta_1 - \beta_2) \cdot g_b(\omega, \bar{\omega}))$$

where $\beta_1, \beta_2 \in (0, 1)$ are real-valued coefficients similar to those used for $\mathcal{L}_{\text{sat}}$, and $g_s(\omega, \bar{\omega}, \varphi)$, $g_w(\omega, \bar{\omega}, \varphi)$ and $g_b(\omega, \bar{\omega})$ are functions defined as follows.

$$g_s(\omega, \bar{\omega}, \varphi) = \begin{cases} \epsilon_s^+ & \bar{\omega} \models \varphi \\ 1 - \epsilon_s^+ & \bar{\omega} \not\models \varphi \end{cases}$$
$$g_w(\omega, \bar{\omega}, \varphi) = \begin{cases} 1 - \epsilon_w^- & \bar{\omega} \models \varphi \\ \epsilon_w^- & \bar{\omega} \not\models \varphi \end{cases}$$
$$g_b(\omega, \bar{\omega}) = \inf \big\{\varepsilon \mid \bar{\omega}_x[t] \in [\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)] \text{ for all } x \in X, t \in \mathbb{T}\big\}$$

where ϵ_s^+ is the upper bound of confidence guarantee for strong satisfaction computed based on Definition 5, ϵ_w^- is the lower bound of confidence guarantee for weak satisfaction computed based on Definition 6, and $g_b(\omega, \bar{\omega})$ computes the smallest confidence level under which the predicted flowpipe is guaranteed to contain the target trace. The goal is to minimize the loss $\mathcal{L}_{\text{cf}}$, for which we need to maximize the linear combination of $g_s(\omega, \bar{\omega}, \varphi)$, $g_w(\omega, \bar{\omega}, \varphi)$, and $g_b(\omega, \bar{\omega})$. Intuitively, the lower the loss, the higher quality of predictions in terms of confidence guarantees for strong/weak satisfaction and prediction accuracy.

Uncertainty calibration using STL-U criteria. In order to select the best uncertainty estimation schema, we start with a set of candidate schemas $\mathcal{M}_1(p), \mathcal{M}_2(p), ..., \mathcal{M}_n(p)$. For each schema with SRT $\mathcal{M}_i$, we tune the dropout rate parameter p using loss functions $\mathcal{L}_{\text{sat}}$ or $\mathcal{L}_{\text{cf}}$. Given a dataset with multiple target traces, we average the losses over all traces to obtain the optimal dropout rate p^*. We compare the losses of candidate schemas equipped with their corresponding optimal dropout rates, and select the best schema $\mathcal{M}^*(p^*)$ that yields the lowest loss. Such a process of selecting and turning uncertainty estimation schemas based on STL-U criteria is illustrated as part of Figure 3.

We evaluate and compare the performance of different STL-U criteria in Section 6. Generally speaking, users can choose to use $\mathcal{L}_{\text{sat}}$ or $\mathcal{L}_{\text{cf}}$ depending on their needs and problem domains. For example, we would recommend applications with strict safety requirements (e.g., a fire risk prediction and control service) to adopt $\mathcal{L}_{\text{sat}}$ for checking strong satisfaction relations. By contrast, $\mathcal{L}_{\text{cf}}$ is more flexible and does not require a pre-defined confidence level, which is suitable for applications that do not have a specific confidence level yet try to optimize the uncertainty estimation (e.g., a newly deployed energy control service).

6 EVALUATION

We conducted experiments to evaluate the proposed approach. In Section 6.1, we compare STL-U criteria for uncertainty calibration with state-of-the-art baselines using real-world CPS datasets. In Section 6.2, we demonstrate the performance of our approach on real-time predictive monitoring in a simulated smart city case study. The experiments were run on a machine with 2.2GHz CPU, 32GB memory, and Nvidia GeForce RTX 2080Ti GPU.

6.1 Evaluating STL-U Criteria for Uncertainty Calibration

We use two real-world city datasets (i.e., air quality and traffic volume datasets) described in Section 2. We split each dataset into 80% data for RNN training, 10% data for STL-U based uncertainty estimation (i.e., tuning Bayesian RNN), and 10% data for testing. We trained the model for 30 epochs.

Comparing different STL-U criteria. Figure 12 plots the loss obtained using different STL-U criteria when varying uncertainty estimation schemas (i.e., SRT and dropout rate p) for the air quality dataset. We trained an LSTM as the underlying RNN model. Figure 12(a) shows the results of using STL-U criterion $\mathcal{L}_{\text{sat}}$ for $\varphi_1 = \Box_I(x_\varepsilon < \lambda)$, where the schema of Bernoulli DropConnect with $p = 0.8$ yields the lowest loss. Figure 12(b) shows the results of using STL-U criterion $\mathcal{L}_{\text{cf}}$ for φ_1, where the schema of Gaussian Dropout with $p = 0.9$ yields the lowest loss. Figure 12(c) shows the results of using STL-U criterion $\mathcal{L}_{\text{cf}}$ for $\varphi_2 = \Diamond_I(x_\varepsilon < \lambda)$, where the schema of Bernoulli Dropout with $p = 0.9$ yields the lowest loss. Thus, the optimal uncertainty estimation schema varies based on different STL-U criteria. The experiments demonstrate that the proposed approach is feasible for the automated selection of optimal schemas based on system requirements and user demands (i.e., whether the user is interested in checking requirement satisfaction or computing confidence guarantees).

Comparing STL-U criteria with baselines. Table 2 shows the results of applying uncertainty estimation with STL-U criteria (bottom two rows) and state-of-the-art baselines (top six rows) to the testing data of air quality and traffic volume datasets. We trained an LSTM as the underlying RNN model for each dataset. We consider six *baselines* for comparison. The top four rows of the table are results of using four SRTs with optimal dropout rates p tuned based on the prediction accuracy (i.e., the percentage of target traces covered in the predicted flowpipes). The next two rows are

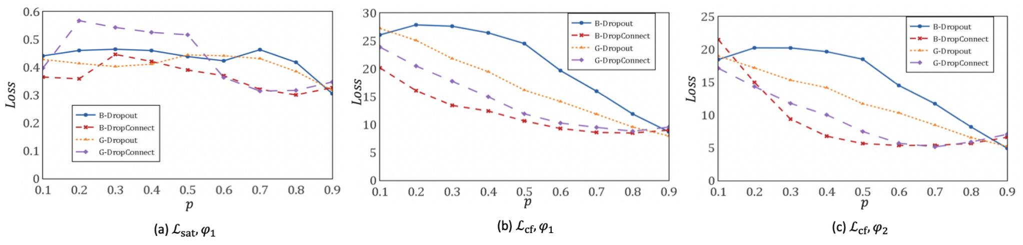


Fig. 12. Selecting uncertainty estimation schemas using different STL-U criteria.

Table 2. Results of comparing STL-U criteria with six baselines.

Criteria	Air Quality SRT	p	HeterLoss	Accuracy	F1-Sat	Traffic Volume SRT	p	HeterLoss	Accuracy	F1-Sat
-	B-Dropout	0.81	183.9	0.67	0.34	B-Dropout	0.50	0.63	0.79	0.17
-	B-DropConnect	0.53	121.0	0.69	0.22	B-DropConnect	0.74	0.23	0.38	0.51
-	G-Dropout	0.45	152.8	0.76	0.10	G-Dropout	0.50	0.66	0.79	0.17
-	G-DropConnect	0.58	129.4	0.78	0.12	G-DropConnect	0.54	0.25	0.56	0.44
$\mathcal{L}_{\text{acc}}$	B-DropConnect	0.53	121.0	0.69	0.22	B-DropConnect	0.74	0.23	0.38	0.51
$\mathcal{L}_{\text{ht}}$	G-Dropout	0.50	119.2	0.81	0.65	G-Dropout	0.50	0.66	0.79	0.17
$\mathcal{L}_{\text{sat}}$	G-DropConnect	0.81	154.1	0.80	**0.81**	B-DropConnect	0.58	0.24	0.51	**0.67**
$\mathcal{L}_{\text{cf}}$	B-DropConnect	0.73	165.4	0.79	**0.76**	B-Dropout	0.90	0.3	0.78	**0.68**

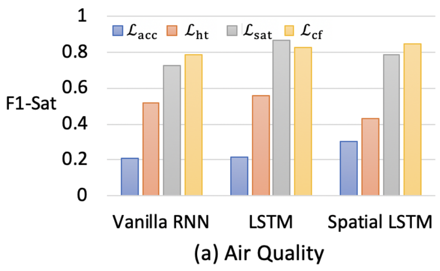
results based on optimizing the uncertainty estimation schema using two commonly used criteria: $\mathcal{L}_{\text{acc}}$ is the loss function concerning the F1-score of prediction accuracy (i.e., if the target trace is covered by the predicted flowpipe), and $\mathcal{L}_{\text{ht}}$ is the loss function approximating the Heteroscedastic aleatoric uncertainty [21]. For the hyperparameters in loss functions, we use $\beta_1 = 0.2$, $\beta_2 = 0.2$ for $\mathcal{L}_{sat}$, and $\beta_1 = 0.3$, $\beta_2 = 0.3$ for $\mathcal{L}_{cf}$.

We compare their performance in terms of three *metrics* shown in columns of the table:

- Heteroscedastic loss: HeterLoss $= \frac{1}{MT}\sum_{i=1}^{M}\sum_{t=1}^{T}(\frac{||y_t^{(i)}-\theta_t^{(i)}||^2}{2(\sigma_t^{(i)})^2} + \frac{1}{2}\log 2\sigma_t^{(i)})$, where M represents the total number of instances in the testing data and T represents the length of the predicted sequence;
- Prediction accuracy (RMSE): Accuracy $= \frac{1}{MT}\sum_{i=1}^{M}\sum_{t=1}^{T}\frac{||y_t^{(i)}-\theta_t^{(i)}||^2}{2(\sigma_t^{(i)})^2}$.
- F1-score comparing the STL-U requirement satisfaction for the predicted and target sequences: F1-Sat $= \frac{TP}{TP+\frac{1}{2}(FP+FN)}$, where TP, FP, FN represents number of true positives, number of false positives, and number of false negatives, respectively.

The results show that both STL-U criteria yield significant higher F1-scores of requirement satisfaction than all six baselines, which having comparable performance with baselines in terms of Heteroscedastic loss and accuracy. Low F1-scores of requirement satisfaction indicate that flowpipes predicted using baselines can be barely used for monitoring city requirements due to the low quality of estimated uncertainty (i.e., the predicted flowpipes may contain too much noise to obtain meaningful results about requirement violations). Thus, using STL-U criteria to calibrate the uncertainty estimation is an essential step for the predictive monitoring.

Comparing different RNN models. Figure 13 compares the F1-score of requirement satisfaction of applying different uncertainty calibration criteria on three types of RNN models: (1) Vanilla RNN, (2) LSTM, and (3) Spatial LSTM [22]. The results show that STL-U criteria $\mathcal{L}_{\text{sat}}$ and $\mathcal{L}_{\text{cf}}$ significantly outperform baseline criteria $\mathcal{L}_{\text{acc}}$ and $\mathcal{L}_{\text{ht}}$ across all three RNN models for both datasets. In addition,



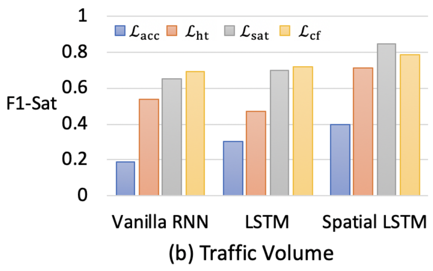


Fig. 13. Results of comparing different RNN models.

both STL-U criteria yield comparable performance across different RNN models. Using $\mathcal{L}_{\text{sat}}$ with an LSTM model and a Spatial LSTM model result in the highest F1-score for the air quality dataset and the traffic volume dataset, respectively. The experiments demonstrate that our proposed approach of uncertainty estimation and calibration is compatible with different underlying RNN models.

6.2 Real-time Predictive Monitoring for a Simulated Smart City

We set up a closed-loop simulated smart city based on the *Simulation of Urban MObility* (SUMO) platform [5] using real-world data of New York City [33]. We implemented ten smart services in the simulated smart city, including S1: Traffic Service, S2: Emergency Service, S3: Accident Service, S4: Infrastructure Service, S5: Pedestrian Service, S6: Air Pollution Control Service, S7: PM2.5/PM10 Service, S8: Parking Service, S9: Noise Control Service, and S10: Event Service. We built a prototype implementation of the STL-U based predictive monitoring and applied it for the predictive monitoring of 390 requirements concerning different city performance metrics (e.g., AQI, traffic volume, noise) in various locations of the simulated smart city. When a smart service requests an action, we predict future city states under the influence of the requested action and monitor if city requirements would be violated. Based on the real-time predictive monitoring results generated by our approach, the control center can decide if the requested action should be accepted or rejected to prevent any potential requirement violation. For the details of the decision making process, we follow the methods in CityResolver [29], which is a decision making system for conflict detection and resolution in smart cities. As an intuitive example, when a smart navigation service requests an action to direct vehicles to a school area to release traffic congestion, the predictive monitoring approach first predicts the future sequences of noise levels and air pollution levels, and verifies them with STL-U specified city requirements on noise and air pollution in the school areas. If the predicted sequences satisfy the requirements, the requested action will be approved; however, if they violate the requirements, the control center will generate a resolution similar to CityResolver. In our experiment, we run the simulated New York city with STL-U predictive monitoring for 30 simulation days and obtain the results regarding the metrics in Table 3. Experimental results show that our approach is efficient in handling a large number of flowpipes and requirements. We did not include the execution time of experiments, because the implementation of our prototype tool is not optimized yet. However, it only takes about 281 seconds to check the satisfaction of 130,000 flowpipes that predict AQI in eight future time units.

Table 3 compares STL-U based predictive monitoring's impact on city performance with two *baselines*: (1) running the simulated city without predictive monitoring, and (2) running the simulated city with a basic predictive monitoring component implemented with a deterministic LSTM predictor and a STL monitor. The results are based on 30-day data in the simulated city. First, we observe that our predictive monitor detects fewer requirement violations for the predicted future city states than the baseline method (2). This is because our approach uses a Bayesian LSTM predictor

Table 3. Results of comparing the impact of STL-U based predictive monitoring with two baselines.

City Performance Metrics	No Monitor	LSTM + STL Monitor	STL-U Predictive Monitor
Number of Violation	-	267	**189**
Air Quality Index	68	57	**43**
Noise (db)	73	49	**48**
Emergency Waiting Time (s)	20	14	**10**
Vehicle Waiting Number	22	18	**15**
Pedestrian Waiting Time (s)	190	148	**121**
Vehicle Waiting Time (s)	112	90	**80**

with calibrated uncertainty, which can generate more accurate predictions about future city states than the deterministic predictor, and thus reducing the number of spurious violations. Furthermore, the results show that our approach has the potential to improve various city performance metrics. For example, compared with the two baselines, our approach reduces the air quality index by 36.8% and 24.6%, and reduces the emergency vehicle waiting time by 50% and 28.6% in the simulated city.

7 RELATED WORK

Predictive monitoring for CPS. The research area of predictive monitoring has been drawing increasing attention in recent years. For example, (Bayesian) Neural Predictive Monitoring [6, 7] checks predictions about neural state classification and uses a principled criteria to reject predictions that are likely to be incorrect; a predictive monitor for rare failures is developed in [1] using Discrete-Time Markov Chains trained with samples of rare events; STLnet [28] incorporates predictive monitoring into the learning process and enhance RNN-based sequential prediction models to follow STL specified model properties in both training and testing processes. Prevent [2] and other runtime verification techniques with state estimation [4, 19, 40] use Hidden Markov Models or Dynamic Bayesian Networks to learn and predict the probability of a hidden state satisfying a safety property. A more recent approach [41] uses instead linear hybrid models of the system under monitoring to bound the uncertainty in the gaps between consecutive samples.

These existing works mostly focus on monitoring individual predictions rather than sequential predictions. A more recent work [35] applies statistical time-series analysis techniques (e.g., ARIMA) to forecast future signal values and computes the satisfaction probability of a STL formula over the prediction horizon; however, the applicability of this approach is limited by the assumption that a joint probability distribution of predictions over multiple time-points can be estimated. By contrast, our approach considers uncertain sequential predictions generated by Bayesian RNN models, which are generally applicable to many CPS domains.

Temporal logic based runtime monitoring. Over the past decades, tremendous progress has been made in developing techniques and tools of runtime monitoring (also called runtime verification) based on rigorous specifications expressed in various temporal logics (e.g., LTL, STL). For example, a survey on STL-based runtime monitoring for CPS is provided in [3], which includes applications such as automotive systems and medical devices; a STL-based framework is developed in [29] for detecting requirement violations in smart cities; SaSTL [27] extends STL for runtime monitoring of spatial-temporal properties in CPS; and another spatial-temporal logic named SpaTeL is applied to monitor the power grid in [15]. However, most of the literature focuses on monitoring deterministic multi-variable signals, which is a limiting factor when we need to monitor predictive models and to reason about uncertainty.

There are some attempts to handle uncertainty by incorporating random variables in predicates. For example, C2TL [18] checks the probability of a deterministic signal satisfying a linear constraint whose coefficients are random variables; PrSTL [38] uses atomic predicates that are parameterized with a time-varying random variable over a deterministic signal; StSTL [24] checks the probability of a real-valued measurable function over stochastic signals; and StTL [23] extends StSTL to reason about the robustness of requirement satisfaction. Our approach differs from these previous works in several aspects. First, the proposed STL-U monitor checks a flowpipe signal that contains an infinite set of uncertain sequences rather than a single sequence. Moreover, instead of computing a single probability value of satisfying a predicate, STL-U reasons about the uncertainty captured by confidence intervals of the flowpipe signal, which is a more suitable representation of uncertainty estimated from Bayesian deep learning.

The problem of monitoring an infinite set of sequences has been studied before in [36] for the reachability analysis of continuous and hybrid system models. This work proposes a Reachset Temporal Logic (RTL), which extends STL and is defined on the reach sequence (i.e., a function mapping time to the set of states reachable from a set of initial states and uncertain inputs). The notion of reachability in RTL (i.e., checking if all the values within the reach sequence satisfies a formula) can be encoded as STL-U strong satisfaction with our approach. In RTL, the formulas are in positive normal form: the negation operator can appear only in basic propositions while it remains undefined for generic formulas that may include also temporal operators. Furthermore, the RTL-based model checking algorithm is limited to a specific fragment of RTL where the only possible temporal operator that can be used is the *next* operator. Similar to RTL [36], the parameter synthesis approach presented in [9] introduces a new semantics for the positive normal form fragment of STL that is defined on sets of traces rather than on a single trace.

In our approach, we introduce the notion of strong/weak semantics to handle the negation operator with more generic formulas: we define the evaluation of strong satisfaction for a formula $\neg\varphi$ (for both atomic predicates and temporal formulas) as equivalent to the violation of the weak satisfaction for the formula φ and vice-versa. Our work takes inspiration from the paper of Eisner et al. [10] where the authors propose a weak/strong semantics to reason with linear temporal logic (LTL) on truncated paths: the weak semantics provides an optimistic view of the satisfaction of an LTL formula on a truncated path, while the strong semantics provides a pessimistic view. In our approach instead, the weak/strong semantics is used to change the existential/universal quantifier when we interpret a proposition over a confidence interval.

Uncertainty estimation in deep learning. While most deep learning models do not offer the uncertainty of their predictions [11], works that capture the uncertainty (or confidence) of the prediction can be dated to the early development of neural networks in the 90s'. Bayesian Neural Network [30] represents a probabilistic model that infers a distribution as output. It is known to be robust and resilient to overfitting. However, the hardness of inference prevents the prevalence of the model in practice. Following these directions, several works [14, 17] use variational inference to perform an approximated inference on Bayesian Neural Networks. Aside from variational inference, Monte Carlo Dropout is another approach to obtain uncertainty estimation of the model [12, 44]. By exploiting the dropout structure in the deep neural network, these approaches turn the original Neural Network model into a simple Bayesian Neural Network without changing the structure and apply approximated inference with the Monte Carlo approach. Existing works [21, 42, 44] mostly focus uncertain estimation on single-time classification or regression tasks. This paper focuses on the case of time series prediction. Moreover, in contrast to previous measures of uncertainty that are rather empirical, our work proposes a formal framework to model and define requirements to the output distribution. Our work can thus be used to provide a confidence guarantee of the model prediction and to evaluate the quality of the uncertainty estimation.

8 CONCLUSION

We developed a novel predictive monitoring approach for CPS, which consists of a logic-calibrated Bayesian RNN prediction model that continuously generates sequential predictions of future states, and a novel STL-U monitor that checks if the generated predictions satisfy CPS requirements. Additionally, we proposed novel criteria based on STL-U monitoring results to calibrate uncertainty estimation in Bayesian deep learning for the predictive monitor. The experimental results show that STL-U criteria leads to improved uncertainty estimation in various Bayesian deep learning models, and STL-U based predictive monitor significantly improves performance metrics in a simulated smart city study.

The proposed STL-U monitor is generally applicable for monitoring an infinite set of sequences beyond those generated by Bayesian deep learning. For example, STL-U monitor can also check trajectories of continuous and hybrid systems (e.g., those considered in [36]). In addition, the proposed STL-U criteria for uncertainty calibration can be used in a broad spectrum of deep learning applications. As demonstrated in Section 6, STL-U criteria can be used for the automated selection of optimal uncertainty estimation schemas and are compatible with different types of RNN models. Applying STL-U criteria for uncertainty calibration does not require knowledge about the inner working of deep learning models and stochastic regularization techniques. Thus, they are amendable for different deep learning applications.

There are several directions to explore for the future work. First, we will extend STL-U logic with quantitative semantics (e.g., robustness of requirement satisfaction). Second, we will explore the theoretical implications of STL-U criteria on uncertainty calibration for Bayesian deep learning. Third, we will investigate the scalability and efficiency of proposed STL-U monitoring algorithms for more complex specifications (e.g., those with multiple layers of nesting temporal operators). Last but not least, we will apply it to a wide range of real-world CPS applications such as autonomous driving, smart health, and smart homes.

ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation grants CCF-1942836, CNS-1952096, NRT Grant 1829004 and by the Austrian FFG-funded IoT4CPS project and the WWTF ICT19-018 grant ProbInG at TU Wien.

REFERENCES

[1] Reza Babaee, Vijay Ganesh, and Sean Sedwards. 2019. Accelerated learning of predictive runtime monitors for rare failure. In *Proc. of RV 2019 (LNCS, Vol. 11757)*. Springer, 111–128. https://doi.org/10.1007/978-3-030-32079-9

[2] Reza Babaee, Arie Gurfinkel, and Sebastian Fischmeister. 2018. Predictive run-time verification of discrete-time reachability properties in black-box systems using trace-level abstraction and statistical learning. In *Proc. RV 2018 (LNCS, Vol. 11237)*. Springer, 187–204. https://doi.org/10.1007/978-3-030-03769-7

[3] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Nickovic, and S. Sankaranarayanan. 2018. Specification-based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications. In *Lectures on Runtime Verification*. LNCS, Vol. 10457. Springer, 135–175. https://doi.org/10.1007/978-3-319-75632-5

[4] Ezio Bartocci, Radu Grosu, Atul Karmarkar, Scott A. Smolka, Scott D. Stoller, Erez Zadok, and Justin Seyster. 2012. Adaptive Runtime Verification. In *Proc. of RV 2012 (LNCS, Vol. 7687)*. Springer, 168–182. https://doi.org/10.1007/978-3-642-35632-2_18

[5] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. 2011. SUMO–simulation of urban mobility: an overview. In *Proc. of Inter. Conference on Advances in System Simulation*. ThinkMind. https://elib.dlr.de/71460/

[6] Luca Bortolussi, Francesca Cairoli, Nicola Paoletti, Scott A. Smolka, and Scott D. Stoller. 2019. Neural predictive monitoring. In *Proc. of RV 2019 (LNCS, Vol. 11757)*. Springer, 129–147. https://doi.org/10.1007/978-3-030-32079-9_8

[7] Luca Bortolussi, Francesca Cairoli, Nicola Paoletti, Scott A. Smolka, and Scott D. Stoller. 2020. Bayesian Neural Predictive Monitoring. In *Proc. of the 2nd Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis (CEUR Workshop Proceedings, Vol. 2785)*. CEUR-WS.org, 95–100. http://ceur-ws.org/Vol-2785/paper16.pdf

[8] Richard P Brent. 2013. *Algorithms for minimization without derivatives.* Courier Corporation.

[9] Thao Dang, Tommaso Dreossi, and Carla Piazza. 2015. Parameter Synthesis Through Temporal Logic Specifications. In *Proc. of FM 2015 (LNCS, Vol. 9109).* Springer, 213–230. https://doi.org/10.1007/978-3-319-19249-9_14

[10] Cindy Eisner, Dana Fisman, John Havlicek, Yoad Lustig, Anthony McIsaac, and David Van Campenhout. 2003. Reasoning with Temporal Logic on Truncated Paths. In *Proc. of CAV 2003 (LNCS, Vol. 2725).* Springer, 27–39. https://doi.org/10.1007/b11831

[11] Yarin Gal. 2016. *Uncertainty in deep learning.* Ph.D. Dissertation. University of Cambridge.

[12] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proc. of ICML 2016*, Vol. 48. JMLR.org, 1050–1059. http://proceedings.mlr.press/v48/

[13] Yarin Gal, Jiri Hron, and Alex Kendall. 2017. Concrete dropout. In *Proc. of NIPS 2017.* 3581–3590. https://proceedings.neurips.cc/paper/2017/hash/84ddfb34126fc3a48ee38d7044e87276-Abstract.html

[14] Alex Graves. 2011. Practical variational inference for neural networks. In *Proc. of NIPS 2011.* 2348–2356. https://proceedings.neurips.cc/paper/2011/hash/7eb3c8be3d411e8ebfab08eba5f49632-Abstract.html

[15] Iman Haghighi, Austin Jones, Zhaodan Kong, Ezio Bartocci, Radu Grosu, and Calin Belta. 2015. SpaTeL: a novel spatial-temporal logic and its applications to networked systems. In *Proc. of HSCC 2015.* IEEE, 189–198. https://doi.org/10.1145/2728606.2728633

[16] Suining He and Kang G Shin. 2020. Towards Fine-grained Flow Forecasting: A Graph Attention Approach for Bike Sharing Systems. In *Proc. of WWW 2020.* 88–98. https://doi.org/10.1145/3366423

[17] Geoffrey E Hinton and Drew Van Camp. 1993. Keeping the neural networks simple by minimizing the description length of the weights. In *Proc. of COLT 1993.* ACM, 5–13. https://doi.org/10.1145/168304.168306

[18] Susmit Jha, Vasumathi Raman, Dorsa Sadigh, and Sanjit A. Seshia. 2018. Safe Autonomy Under Perception Uncertainty Using Chance-Constrained Temporal Logic. *Journal of Automated Reasoning* 60, 1 (2018), 43–62. https://doi.org/10.1007/s10817-017-9413-9

[19] Kenan Kalajdzic, Ezio Bartocci, Scott A. Smolka, Scott D. Stoller, and Radu Grosu. 2013. Runtime Verification with Particle Filtering. In *Proc. of RV 2013 (LNCS, Vol. 8174).* Springer, 149–166. https://doi.org/10.1007/978-3-642-40787-1

[20] Steven M Kay. 1993. *Fundamentals of statistical signal processing.* Prentice Hall PTR.

[21] Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision?. In *Proc. of NIPS 2017.* 5574–5584. https://proceedings.neurips.cc/paper/2017/hash/2650d6089a6d640c5e85b2b88265dc2b-Abstract.html

[22] Dejiang Kong and Fei Wu. 2018. HST-LSTM: a hierarchical spatial-temporal long-short term memory network for location prediction. In *Proc. of IJCAI 2018.* ijcai.org, 2341–2347. https://doi.org/10.24963/ijcai.2018/324

[23] Panagiotis Kyriakis, Jyotirmoy V. Deshmukh, and Paul Bogdan. 2019. Specification Mining and Robust Design under Uncertainty: A Stochastic Temporal Logic Approach. *ACM Trans. Embed. Comput. Syst.* 18, 5s (2019), 96:1–96:21. https://doi.org/10.1145/3358231

[24] Jiwei Li, Pierluigi Nuzzo, Alberto Sangiovanni-Vincentelli, Yugeng Xi, and Dewei Li. 2017. Stochastic Contracts for Cyber-Physical System Design under Probabilistic Requirements. In *Proc. of MEMOCODE 2017.* 5—-14. https://doi.org/10.1145/3127041.3127045

[25] Yuxuan Liang, Songyu Ke, Junbo Zhang, Xiuwen Yi, and Yu Zheng. 2018. GeoMAN: Multi-level Attention Networks for Geo-sensory Time Series Prediction.. In *Proc. of IJCAI 2018.* ijcai.org, 3428–3434. https://doi.org/10.24963/ijcai.2018/476

[26] Lingbo Liu, Zhilin Qiu, Guanbin Li, Qing Wang, Wanli Ouyang, and Liang Lin. 2019. Contextualized spatial–temporal network for taxi origin-destination demand prediction. *IEEE Transactions on Intelligent Transportation Systems* 20, 10 (2019), 3875–3887. https://doi.org/10.1109/TITS.2019.2915525

[27] Meiyi Ma, Ezio Bartocci, Eli Lifland, John Stankovic, and Lu Feng. 2020. SaSTL: Spatial Aggregation Signal Temporal Logic for Runtime Monitoring in Smart Cities. In *Proc. of ICCPS 2020.* IEEE, 51–62. https://doi.org/10.1109/ICCPS48487.2020.00013

[28] Meiyi Ma, Ji Gao, Lu Feng, and John A Stankovic. 2020. STLnet: Signal Temporal Logic Enforced Multivariate Recurrent Neural Networks.. In *NeurIPS 2020.* https://proceedings.neurips.cc/paper/2020/hash/a7da6ba0505a41b98bd85907244c4c30-Abstract.html

[29] Meiyi Ma, John A Stankovic, and Lu Feng. 2018. Cityresolver: a decision support system for conflict resolution in smart cities. In *Proc. of ICCPS 2018.* IEEE Computer Society / ACM, 55–64. https://doi.org/10.1109/ICCPS.2018.00014

[30] David JC MacKay. 1992. A practical Bayesian framework for backpropagation networks. *Neural computation* 4, 3 (1992), 448–472. https://doi.org/10.1162/neco.1992.4.3.448

[31] O. Maler and D. Ničković. 2004. Monitoring Temporal Properties of Continuous Signals. In *Proc. of FORMATS 2004 (LNCS, Vol. 3253).* Springer, 152–166. https://doi.org/10.1007/b100824

[32] Eslam Montaser, José-Luis Díez, and Jorge Bondia. 2017. Stochastic seasonal models for glucose prediction in the artificial pancreas. *Journal of diabetes science and technology* 11, 6 (2017), 1124–1131.

[33] NYC.gov. 2020. *New York City Open Data.* https://nycopendata.socrata.com/.

[34] Erfan Pakdamanian, Shili Sheng, Sonia Baee, Seongkook Heo, Sarit Kraus, and Lu Feng. 2021. DeepTake: Prediction of Driver Takeover Behavior using Multimodal Data. In *Proc. CHI '21*. ACM, 103:1–103:14. https://doi.org/10.1145/3411764.3445563

[35] Xin Qin and Jyotirmoy V Deshmukh. 2020. Clairvoyant Monitoring for Signal Temporal Logic. In *Proc. of FORMATS 2020 (LNCS, Vol. 12288)*. Springer, 178–195. https://doi.org/10.1007/978-3-030-57628-8

[36] Hendrik Roehm, Jens Oehlerking, Thomas Heinz, and Matthias Althoff. 2016. STL model checking of continuous and hybrid systems. In *Proc. of ATVA 2016 (LNCS, Vol. 9938)*. Springer, 412–427. https://doi.org/10.1007/978-3-319-46520-3

[37] Lothar Sachs. 2012. *Applied statistics: a handbook of techniques*. Springer Science & Business Media.

[38] Dorsa Sadigh and Ashish Kapoor. 2016. Safe Control under Uncertainty with Probabilistic Signal Temporal Logic. In *Proc. of the RSS 2016*. https://doi.org/10.15607/RSS.2016.XII.017

[39] Bhavkaran Singh Walia, Qianyi Hu, Jeffrey Chen, Fangyan Chen, Jessica Lee, Nathan Kuo, Palak Narang, Jason Batts, Geoffrey Arnold, and Michael Madaio. 2018. A dynamic pipeline for spatio-temporal fire risk prediction. In *Proc. of KDD 2018*. ACM, 764–773. https://doi.org/10.1145/3219819

[40] Scott D. Stoller, Ezio Bartocci, Justin Seyster, Radu Grosu, Klaus Havelund, Scott A. Smolka, and Erez Zadok. 2012. Runtime Verification with State Estimation. In *Proc. of RV 2011 (LNCS, Vol. 7186)*. Springer, 193–207. https://doi.org/10.1007/978-3-642-29860-8

[41] Masaki Waga, Étienne André, and Ichiro Hasuo. 2021. Model-bounded monitoring of hybrid systems. In *Proc. of ICCPS '21*. ACM, 21–32. https://doi.org/10.1145/3450267.3450531

[42] Yijun Xiao and William Yang Wang. 2019. Quantifying uncertainties in natural language processing tasks. In *Proc. of the AAAI 2019*, Vol. 33. 7322–7329. https://doi.org/10.1609/aaai.v33i01.33017322

[43] Ian Zhou, Justin Lipman, Mehran Abolhasan, Negin Shariati, and David W Lamb. 2020. Frost Monitoring Cyber–Physical System: A Survey on Prediction and Active Protection Methods. *IEEE Internet of Things Journal* 7, 7 (2020), 6514–6527. https://doi.org/10.1109/JIOT.2020.2972936

[44] Lingxue Zhu and Nikolay Laptev. 2017. Deep and Confident Prediction for Time Series at Uber. In *Proc. of ICDM Workshops*. IEEE, 103–110. https://doi.org/10.1109/ICDMW.2017.19

APPENDIX

Proof of Theorem 1. We just need to prove $(\omega, t) \models_s \varphi \Rightarrow (\omega, t) \models_w \varphi$ by structural induction below. By contraposition, we have $((\omega, t) \not\models_w \varphi \Rightarrow (\omega, t) \not\models_s \varphi) \Leftrightarrow ((\omega, t) \models_s \varphi \Rightarrow (\omega, t) \models_w \varphi)$.

- Base case μ_x: by Definition 3, $(\omega, t) \models_s \varphi \Leftrightarrow \forall x \in [\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)], f(x) > 0$, it indicates that $\exists x \in [\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)], f(x) > 0 \Leftrightarrow (\omega, t) \models_w \varphi$.
- Inductive case $\neg\varphi$: from inductive hypothesis $(\omega, t) \models_s \varphi \Rightarrow (\omega, t) \models_w \varphi$ (and consequently $((\omega, t) \not\models_w \varphi \Rightarrow (\omega, t) \not\models_s \varphi)$, we need to prove that $(\omega, t) \models_s \neg\varphi \Rightarrow (\omega, t) \models_w \neg\varphi$.
  We have $(\omega, t) \models_s \neg\varphi \Leftrightarrow (\omega, t) \not\models_w \varphi \Rightarrow (\omega, t) \not\models_s \varphi \Leftrightarrow (\omega, t) \models_w \neg\varphi$.
- Inductive case $\varphi_1 \wedge \varphi_2$: from inductive hypothesis $(\omega, t) \models_s \varphi_1 \Rightarrow (\omega, t) \models_w \varphi_1$ and $(\omega, t) \models_s \varphi_2 \Rightarrow (\omega, t) \models_w \varphi_2$, we need to prove that $(\omega, t) \models_s \varphi_1 \wedge \varphi_2 \Rightarrow (\omega, t) \models_w \varphi_1 \wedge \varphi_2$.
  By Definition 3 and Definition 4, we have $(\omega, t) \models_s \varphi_1 \wedge \varphi_2 \Leftrightarrow (\omega, t) \models_s \varphi_1 \wedge (\omega, t) \models_s \varphi_2 \Rightarrow (\omega, t) \models_w \varphi_1 \wedge (\omega, t) \models_w \varphi_2 \Leftrightarrow (\omega, t) \models_w \varphi_1 \wedge \varphi_2$.
- Inductive case $\Box_I\varphi$: from inductive hypothesis $(\omega, t) \models_s \varphi \Rightarrow (\omega, t) \models_w \varphi$, we need to prove that $(\omega, t) \models_s \Box_I\varphi \Rightarrow (\omega, t) \models_w \Box_I\varphi$.
  By Definition 3 and Definition 4, we have $(\omega, t) \models_s \Box_I\varphi \Leftrightarrow \forall t' \in (t + I), (\omega, t') \models_s \varphi$, which indicates that $\forall t' \in (t + I), (\omega, t') \models_w \varphi \Leftrightarrow (\omega, t) \models_w \Box_I\varphi$.
- Inductive case $\Diamond_I\varphi$: from inductive hypothesis $(\omega, t) \models_s \varphi \Rightarrow (\omega, t) \models_w \varphi$, we need to prove that $(\omega, t) \models_s \Diamond_I\varphi \Rightarrow (\omega, t) \models_w \Diamond_I\varphi$.
  By Definition 3 and Definition 4, we have $(\omega, t) \models_s \Diamond_I\varphi \Leftrightarrow \exists t' \in (t + I), (\omega, t') \models_s \varphi$, which indicates that $\exists t' \in (t + I), (\omega, t') \models_w \varphi \Leftrightarrow (\omega, t) \models_w \Diamond_I\varphi$.
- Inductive case $\varphi_1\mathcal{U}_I\varphi_2$: from inductive hypothesis $(\omega, t) \models_s \varphi_1 \Rightarrow (\omega, t) \models_w \varphi_1$ and $(\omega, t) \models_s \varphi_2 \Rightarrow (\omega, t) \models_w \varphi_2$, we prove that $(\omega, t) \models_s \varphi_1\mathcal{U}_I\varphi_2 \Rightarrow (\omega, t) \models_w \varphi_1\mathcal{U}_I\varphi_2$.
  By Definition 3 and Definition 4, we have $(\omega, t) \models_s \varphi_1\mathcal{U}_I\varphi_2 \Leftrightarrow \exists t' \in (t + I) \cap \mathbb{T}, (\omega, t') \models_s \varphi_2$ and $\forall t'' \in (t, t'), (\omega, t'') \models_s \varphi_1$, which indicates that $\exists t' \in (t + I) \cap \mathbb{T}, (\omega, t') \models_w \varphi_2$ and $\forall t'' \in (t, t'), (\omega, t'') \models_w \varphi_1$, it is equivalent to $(\omega, t) \models_w \varphi_1\mathcal{U}_I\varphi_2$.

□

Additional properties and examples. By applying the rules of the weak/strong semantics for negation we have that the following properties hold:

$$(\omega,t)\models_s \neg\neg\varphi \quad\Leftrightarrow\quad (\omega,t)\models_s \varphi \qquad (\omega,t)\models_w \neg\neg\varphi \quad\Leftrightarrow\quad (\omega,t)\models_w \varphi$$

By applying the De Morgan's laws we have the following:

$$\begin{array}{lcccl}(\omega,t)\not\models_w \neg\varphi_1 \wedge (\omega,t)\not\models_w \neg\varphi_2 & \Leftrightarrow & \neg((\omega,t)\models_w (\neg\varphi_1 \vee \neg\varphi_2)) & \Leftrightarrow & (\omega,t)\not\models_w \neg(\varphi_1\wedge\varphi_2)\\ (\omega,t)\not\models_s \neg\varphi_1 \wedge (\omega,t)\not\models_s \neg\varphi_2 & \Leftrightarrow & \neg((\omega,t)\models_s (\neg\varphi_1 \vee \neg\varphi_2)) & \Leftrightarrow & (\omega,t)\not\models_s \neg(\varphi_1\wedge\varphi_2)\end{array}$$

Furthermore, to clarify the duality of the two semantics, we can interpret the weak semantics using the interval intersection and the strong semantics using the interval inclusion. For example, let us define as basic propositions: μ_x^f s.t. $f(x)=x$ and μ_x^g s.t. $g(x)=-x$. Then we have:

$$\begin{array}{lclcl}(\omega,t)\models_w \mu_x^f(\epsilon) & \Leftrightarrow & \exists x\in[\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)], x>0 & \Leftrightarrow & [\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)]\cap(0,+\infty)\neq\emptyset\\ (\omega,t)\models_w \mu_x^g(\epsilon) & \Leftrightarrow & \exists x\in[\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)], x<0 & \Leftrightarrow & [\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)]\cap(-\infty,0)\neq\emptyset\\ (\omega,t)\models_s \mu_x^f(\epsilon) & \Leftrightarrow & \forall x\in[\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)], x>0 & \Leftrightarrow & [\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)]\subset(0,+\infty)\\ (\omega,t)\models_s \mu_x^g(\epsilon) & \Leftrightarrow & \forall x\in[\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)], x<0 & \Leftrightarrow & [\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)]\subset(+\infty,0)\end{array}$$

$$(\omega,t)\models_w \neg\mu_x^f(\epsilon)\Leftrightarrow \underbrace{\exists x\in[\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)], x\le 0}_{[\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)]\cap(-\infty,0]\neq\emptyset}\Leftrightarrow \underbrace{\neg(\forall x\in[\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)], x>0)}_{[\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)]\not\subset(0,+\infty)}\Leftrightarrow(\omega,t)\not\models_s \mu_x^f(\epsilon)$$

Following the definition of negation for the weak semantics we have that:

$$(\omega,t)\models_w (\mu_x^f(\epsilon)\wedge\mu_x^g(\epsilon)) \quad\Leftrightarrow\quad (\omega,t)\not\models_s \neg(\mu_x^f(\epsilon)\wedge\mu_x^g(\epsilon))$$

We can show the equivalence consistency using the interval intersection/inclusion interpretation:

$$(\omega,t)\models_w (\mu_x^f(\epsilon)\wedge\mu_x^g(\epsilon))\Leftrightarrow(\omega,t)\models_w \mu_x^f(\epsilon)\wedge(\omega,t)\models_w \mu_x^g(\epsilon)$$

$$\underbrace{(\omega,t)\models_w \mu_x^f(\epsilon)}_{[\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)]\cap(0,+\infty)\neq\emptyset} \wedge \underbrace{(\omega,t)\models_w \mu_x^g(\epsilon)}_{[\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)]\cap(-\infty,0)\neq\emptyset} \Leftrightarrow \underbrace{(\omega,t)\not\models_s \neg\mu_x^f(\epsilon)}_{[\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)]\not\subset(-\infty,0]} \wedge \underbrace{(\omega,t)\not\models_s \neg\mu_x^g(\epsilon)}_{[\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)]\not\subset[0,+\infty)}$$

By applying the De Morgan's laws shown before:

$$(\omega,t)\not\models_s \neg\mu_x^f(\epsilon)\wedge(\omega,t)\not\models_s \neg\mu_x^g(\epsilon)\Leftrightarrow(\omega,t)\not\models_s \neg(\mu_x^f(\epsilon)\wedge\mu_x^g(\epsilon))$$

Proof of Theorem 2 and Theorem 3. Mathematically, we want to show that for any ω and t, we have $\forall\varepsilon\in\epsilon_s(\varphi,\omega,t), (\omega,t)\models_s\varphi$ under confidence level ε, and for any ω and t, we have $\forall\varepsilon\in\epsilon_w(\varphi,\omega,t), (\omega,t)\models_w\varphi$ under confidence level ε . We prove the Theorem 2 and Theorem 3 inductively. Since in the definition of the strong definition and weak definitions refer to each other, we prove them together. We study whether Theorem 2 satisfies in every possible case in definition. Then, we finish our proof by the axiom of induction. We omit the cases of $\Box$ and $\Diamond$ since they can be derived from the case of $\mathcal{U}_I$.

- When $\varphi=\mu_x$, we show $\forall\omega,\forall t,\forall\varepsilon\in\epsilon_s(\varphi,\omega,t),(\omega,t)\models_s\varphi$.
  By Definition 5, $\epsilon_s(\varphi,\omega,t)=(0,\int_{\theta_t-\eta}^{\theta_t+\eta}\varphi_t(x)dx)$, where $\eta=\inf\left\{|x-\theta_t|\,\middle|\,f(x)\le 0\right\}$. By the definition of confidence interval, we have for $\varepsilon\in\epsilon_s(\varphi,\omega,t)$, $\Phi_t^+(\varepsilon)\le\theta_t+\eta$ and $\Phi_t^-(\varepsilon)\ge\theta_t-\eta$, which indicates $[\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)]\subseteq[\theta_t-\eta,\theta_t+\eta]$. As $\eta=\inf\left\{|x-\theta_t|\,\middle|\,f(x)\le 0\right\}$, we have $\forall x\in[\Phi_t^-(\varepsilon),\Phi_t^+(\varepsilon)], f(x)>0$. Therefore, $(\omega,t)\models_s\varphi$.

- When $\varphi = \mu_x$, we show $\forall \omega, \forall t, \forall \varepsilon \in \epsilon_w(\varphi, \omega, t), (\omega, t) \models_w \varphi$.
  By Definition 6, $\epsilon_w(\mu_x, \omega, t) = (\int_{\theta_t - \eta}^{\theta_t + \eta} \varphi_t(x)dx, 1)$, where $\eta = \inf \left\{ |x - \theta_t| \;\middle|\; f(x) > 0 \right\}$. By the definition of confidence interval, we have for $\varepsilon \in \epsilon_s(\varphi, \omega, t)$, $\Phi_t^+(\varepsilon) \geq \theta_t + \eta$ and $\Phi_t^-(\varepsilon) \leq \theta_t - \eta$, which indicates $(\theta_t - \eta, \theta_t + \eta) \subseteq [\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)]$. Since $\eta = \inf \left\{ |x - \theta_t| \;\middle|\; f(x) > 0 \right\}$, we have $\exists x \in [\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)], f(x) > 0$. Therefore, $(\omega, t) \models_w \varphi$.
- When $\varphi = \neg\varphi_1$, we show $\forall \omega, \forall t, \forall \varepsilon \in \epsilon_w(\varphi_1, \omega, t), (\omega, t) \models_w \varphi_1 \Rightarrow \forall \omega, \forall t, \forall \varepsilon \in \epsilon_s(\varphi, \omega, t)$, $(\omega, t) \models_s \varphi$.
  By Definition 5, we have $\epsilon_s(\neg\varphi, \omega, t) = \epsilon_w^c(\varphi_1, \omega, t)$. Therefore, $\forall \varepsilon \in \epsilon_s(\neg\varphi, \omega, t)$, we have $\varepsilon \in \epsilon_w^c(\varphi_1, \omega, t)$. Then, by the definition of confidence interval we have $(\omega, t) \not\models_w \varphi_1$ under ε. By the Definition 3, $(\omega, t) \models_s \varphi$.
- When $\varphi = \neg\varphi_1$, we show $\forall \omega, \forall t, \forall \varepsilon \in \epsilon_s(\varphi_1, \omega, t), (\omega, t) \models_s \varphi_1 \Rightarrow \forall \omega, \forall t, \forall \varepsilon \in \epsilon_w(\varphi, \omega, t)$, $(\omega, t) \models_s \varphi$.
  By Definition 6, we have $\epsilon_w(\neg\varphi, \omega, t) = \epsilon_s^c(\varphi_1, \omega, t)$. Therefore, $\forall \varepsilon \in \epsilon_w(\neg\varphi, \omega, t)$, we have $\varepsilon \in \epsilon_s^c(\varphi_1, \omega, t)$. Then, by the definition of confidence interval we have $(\omega, t) \not\models_s \varphi_1$ under ε. By the definition 4, $(\omega, t) \models_w \varphi$.
- $(\varphi = \varphi_1 \wedge \varphi_2) \wedge (\forall \omega, \forall t, \forall \varepsilon \in \epsilon_s(\varphi_1, \omega, t), (\omega, t) \models_s \varphi_1) \wedge (\forall \omega, \forall t, \forall \varepsilon \in \epsilon_s(\varphi_2, \omega, t), (\omega, t) \models_s \varphi_2) \Rightarrow \forall \omega, \forall t, \forall \varepsilon \in \epsilon_w(\varphi, \omega, t), (\omega, t) \models_s \varphi$.
  By Definition 5, $\epsilon_s(\neg\varphi, \omega, t) = \epsilon_s(\varphi_1, \omega, t) \cap \epsilon_s(\varphi_2, \omega, t)$. Therefore, $\forall \varepsilon \in \epsilon_s(\varphi_1 \wedge \varphi_2, \omega, t)$, we have $\varepsilon \in \epsilon_s(\varphi_1, \omega, t)$ and $\varepsilon \in \epsilon_s(\varphi_2, \omega, t)$. Then we have $(\omega, t) \models_s \varphi_1$ under ε and $(\omega, t) \models_s \varphi_2$ under ε, which indicates $(\omega, t) \models_w \varphi$ by Definition 3.
- When $\varphi = \varphi_1 \wedge \varphi_2$, we show $(\forall \omega, \forall t, \forall \varepsilon \in \epsilon_w(\varphi_1, \omega, t), (\omega, t) \models_w \varphi_1) \wedge (\forall \omega, \forall t, \forall \varepsilon \in \epsilon_s(\varphi_2, \omega, t), (\omega, t) \models_w \varphi_2) \Rightarrow \forall \omega, \forall t, \forall \varepsilon \in \epsilon_w(\varphi, \omega, t), (\omega, t) \models_w \varphi$.
  By Definition 6, $\epsilon_w(\neg\varphi, \omega, t) = \epsilon_w(\varphi_1, \omega, t) \cap \epsilon_w(\varphi_2, \omega, t)$. Therefore, $\forall \varepsilon \in \epsilon_w(\varphi_1 \wedge \varphi_2, \omega, t)$, we have $\varepsilon \in \epsilon_w(\varphi_1, \omega, t)$ and $\varepsilon \in \epsilon_w(\varphi_2, \omega, t)$. Then we have $(\omega, t) \models_w \varphi_1$ under ε and $(\omega, t) \models_w \varphi_2$ under ε, which indicates $(\omega, t) \models_w \varphi$ by Definition 4.
- When $\varphi = \varphi_1 \mathcal{U}_I \varphi_2$, we show $(\forall \omega, \forall t, \forall \varepsilon \in \epsilon_s(\varphi_1, \omega, t), (\omega, t) \models_s \varphi_1) \wedge (\forall \omega, \forall t, \forall \varepsilon \in \epsilon_s(\varphi_2, \omega, t)$, $(\omega, t) \models_s \varphi_2) \Rightarrow \forall \omega, \forall t, \forall \varepsilon \in \epsilon_s(\varphi, \omega, t), (\omega, t) \models_s \varphi$.
  By Definition 5, we have $\epsilon_s(\varphi_1 \mathcal{U}_I \varphi_2, \omega, t) = \bigcup_{t' \in (t+I)} \left\{ \epsilon_s(\varphi_2, \omega, t') \cap (\bigcap_{t'' \in (t,t')} \epsilon_s(\varphi_1, \omega, t'')) \right\}$.
  Therefore, for $\varepsilon \in \epsilon_s(\varphi_1 \mathcal{U}_I \varphi_2, \omega, t)$, we have $\exists t' \in (t+I), \varepsilon \in \epsilon_s(\varphi_2, \omega, t') \cap (\bigcap_{t'' \in (t,t')} \epsilon_s(\varphi_1, \omega, t''))$.
  Therefore, for this t' we have $\varepsilon \in \epsilon_s(\varphi_2, \omega, t')$ and $\forall t'' \in (t, t'), \varepsilon \in \epsilon_s(\varphi_1, \omega, t''))$. By the inductive assumption, we have $(\omega, t') \models_s \varphi_2$ and $\forall t'' \in (t, t'), (\omega, t'') \models_s \varphi_1$. Finally, by definition Definition 3, we have $(\omega, t) \models_s \varphi_1 \mathcal{U}_I \varphi_2$.
- When $\varphi = \varphi_1 \mathcal{U}_I \varphi_2$, we show $(\forall \omega, \forall t, \forall \varepsilon \in \epsilon_w(\varphi_1, \omega, t), (\omega, t) \models_w \varphi_1) \wedge (\forall \omega, \forall t, \forall \varepsilon \in \epsilon_w(\varphi_2, \omega, t), (\omega, t) \models_w \varphi_2) \Rightarrow \forall \omega, \forall t, \forall \varepsilon \in \epsilon_w(\varphi, \omega, t), (\omega, t) \models_w \varphi$.
  By Definition 6, we have $\epsilon_w(\varphi_1 \mathcal{U}_I \varphi_2, \omega, t) = \bigcup_{t' \in (t+I)} \left\{ \epsilon_w(\varphi_2, \omega, t') \cap (\bigcap_{t'' \in (t,t')} \epsilon_w(\varphi_1, \omega, t'')) \right\}$.
  Therefore, for $\varepsilon \in \epsilon_w(\varphi_1 \mathcal{U}_I \varphi_2, \omega, t)$, we have $\exists t' \in (t+I), \varepsilon \in \epsilon_w(\varphi_2, \omega, t') \cap (\bigcap_{t'' \in (t,t')} \epsilon_w(\varphi_1, \omega, t''))$.
  Therefore, for this t' we have $\varepsilon \in \epsilon_w(\varphi_2, \omega, t')$ and $\forall t'' \in (t, t'), \varepsilon \in \epsilon_w(\varphi_1, \omega, t''))$. By the inductive assumption, we have $(\omega, t') \models_w \varphi_2$ and $\forall t'' \in (t, t'), (\omega, t'') \models_w \varphi_1$. Finally, by Definition 4, we have $(\omega, t) \models_w \varphi_1 \mathcal{U}_I \varphi_2$.

□

Algorithm 1: STL-U strong satisfaction monitoring algorithm StrongSat(φ, ω, t)

```
Function StrongSat($(\varphi, \omega, t)$):
    begin
        switch $\varphi$ do
            Case $\mu_x(\varepsilon)$
                if
                  minimize$(f(x), \Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)) > 0$
                  then
                    return True ;
                else
                    return False ;
                end
            Case $\neg\varphi$
                return $\neg$WeakSat$(\varphi, \omega, t)$;
            Case $\varphi_1 \wedge \varphi_2$
                return StrongSat$(\varphi, \omega, t) \wedge$
                  StrongSat$(\varphi, \omega, t)$
            Case $\square_I \varphi$
                for $t' \in (t + I)$ do
                    if $\neg$StrongSat$(\varphi, \omega, t')$ then
                        return False;
                    end
                end
                return True ;
            Case $\lozenge_I \varphi$
                for $t' \in (t + I)$ do
                    if StrongSat$(\varphi, \omega, t')$ then
                        return True;
                    end
                end
                return False ;
            Case $\varphi_1 \mathcal{U}_I \varphi_2$
                for $t' \in (t + I)$ do
                    if StrongSat$(\varphi_2, \omega, t')$ then
                        for $t'' \in [t, t']$ do
                            if
                              $\neg$StrongSat$(\varphi_1, \omega, t'')$
                              then
                                return False ;
                            end
                        end
                        return True ;
                    end
                end
                return False ;
        end
    end
```

Algorithm 2: STL-U weak satisfaction monitoring algorithm WeakSat(φ, ω, t)

```
Function WeakSat($(\varphi, \omega, t)$):
    begin
        switch $\varphi$ do
            Case $\mu_x(\varepsilon)$
                if
                  maximize$(f(x), \Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)) > 0$
                  then
                    return True ;
                else
                    return False ;
                end
            Case $\neg\varphi$
                return $\neg$StrongSat$(\varphi, \omega, t)$;
            Case $\varphi_1 \wedge \varphi_2$
                return WeakSat$(\varphi, \omega, t) \wedge$
                  WeakSat$(\varphi, \omega, t)$
            Case $\square_I \varphi$
                for $t' \in (t + I)$ do
                    if $\neg$WeakSat$(\varphi, \omega, t')$ then
                        return False;
                    end
                end
                return True ;
            Case $\lozenge_I \varphi$
                for $t' \in (t + I)$ do
                    if WeakSat$(\varphi, \omega, t')$ then
                        return True;
                    end
                end
                return False ;
            Case $\varphi_1 \mathcal{U}_I \varphi_2$
                for $t' \in (t + I)$ do
                    if WeakSat$(\varphi_2, \omega, t')$ then
                        for $t'' \in [t, t']$ do
                            if
                              $\neg$WeakSat$(\varphi_1, \omega, t'')$
                              then
                                return False ;
                            end
                        end
                        return True ;
                    end
                end
                return False ;
        end
    end
```

Algorithm 3: Confidence Level of Strong Satisfaction StrongConfidenceLevel(φ, ω, t)

```
Function StrongConfidenceLevel($\varphi, \omega, t$):
    begin
        switch $\varphi$ do
            Case $\mu_x$
                $\eta \leftarrow \inf \left\{ |x - \theta_t| \;\middle|\; f(x) \leq 0 \right\}$
                return $(0, \int_{\theta_t-\eta}^{\theta_t+\eta} \Phi_t(x)dx)$;
            Case $\neg\varphi$
                $\epsilon_s \leftarrow$
                  WeakConfidenceLevel($\varphi, \omega, t)^C$
                return $\epsilon_s$;
            Case $\varphi_1 \wedge \varphi_2$
                return
                  StrongConfidenceLevel($\varphi_1, \omega, t$) $\cap$
                  StrongConfidenceLevel($\varphi_2, \omega, t$)
            Case $\Box_I \varphi$
                $\epsilon_s$ = StrongConfidenceLevel($\varphi, \Omega, 0$)
                for $t' \in (t + I)$ do
                    $\epsilon_s \leftarrow \epsilon_s \cap$
                      StrongConfidenceLevel($\varphi, \Omega, t'$)
                end
                return $\epsilon_s$;
            Case $\Diamond_I \varphi$
                $\epsilon_s \leftarrow$
                  StrongConfidenceLevel($\varphi, \Omega, 0$)
                for $t' \in (t + I)$ do
                    $\epsilon_s \leftarrow \epsilon_s \cup$
                      StrongConfidenceLevel($\varphi, \Omega, t'$)
                end
                return $\epsilon_s$;
            Case $\varphi_1 \mathcal{U}_I \varphi_2$
                $\epsilon_s \leftarrow \emptyset$
                for $t' \in (t + I)$ do
                    $\epsilon'_s \leftarrow$
                      StrongConfidenceLevel($\varphi_2, \omega, t'$)
                    for $t'' \in [t, t']$ do
                        $\epsilon'_s \leftarrow \epsilon'_s \cap$
                          StrongConfidenceLevel($\varphi_1, \omega, t''$)
                    end
                    $\epsilon_s \leftarrow \epsilon_s \cup \epsilon'_s$
                end
                return $\epsilon_s$;
        end
    end
```

Algorithm 4: Confidence Level of Weak Satisfaction WeakConfidenceLevel(φ, ω, t)

```
Function WeakConfidenceLevel($\varphi, \omega, t$):
    begin
        switch $\varphi$ do
            Case $\mu_x$
                $\eta \leftarrow \inf \left\{ |x - \theta_t| \;\middle|\; f(x) > 0 \right\}$
                return $(\int_{\theta_t-\eta}^{\theta_t+\eta} \Phi_t(x)dx, 1)$ ;
            Case $\neg\varphi$
                $\epsilon_w \leftarrow$
                  StrongConfidenceLevel($\varphi, \omega, t)^C$
                return $\epsilon_w$;
            Case $\varphi_1 \wedge \varphi_2$
                return
                  WeakConfidenceLevel($\varphi_1, \omega, t$) $\cap$
                  WeakConfidenceLevel($\varphi_2, \omega, t$)
            Case $\Box_I \varphi$
                $\epsilon_w \leftarrow$
                  WeakConfidenceLevel($\varphi, \Omega, 0$)
                for $t' \in (t + I)$ do
                    $\epsilon_w \leftarrow \epsilon_w \cap$
                      WeakConfidenceLevel($\varphi, \Omega, t'$)
                end
                return $\epsilon_w$;
            Case $\Diamond_I \varphi$
                $\epsilon_w \leftarrow$
                  WeakConfidenceLevel($\varphi, \Omega, 0$)
                for $t' \in (t + I)$ do
                    $\epsilon_w \leftarrow \epsilon_w \cup$
                      WeakConfidenceLevel($\varphi, \Omega, t'$)
                end
                return $\epsilon_w$;
            Case $\varphi_1 \mathcal{U}_I \varphi_2$
                $\epsilon_w \leftarrow \emptyset$
                for $t' \in (t + I)$ do
                    $\epsilon'_w \leftarrow$
                      WeakConfidenceLevel($\varphi_2, \omega, t'$)
                    for $t'' \in [t, t']$ do
                        $\epsilon'_w \leftarrow \epsilon'_w \cap$
                          WeakConfidenceLevel($\varphi_1, \omega, t''$)
                    end
                    $\epsilon_w \leftarrow \epsilon_w \cup \epsilon'_w$
                end
                return $\epsilon_w$;
        end
    end
```