# Model Predictive Online Monitoring of Dynamical Systems for Nested Signal Temporal Logic Specifications

Tao Han, Shaoyuan Li and Xiang Yin

*Abstract*— **This paper investigates the online monitoring problem for cyber-physical systems under signal temporal logic (STL) specifications. The objective is to design an online monitor that evaluates system correctness at runtime based on partial signal observations up to the current time so that alarms can be issued whenever the specification is violated or will inevitably be violated in the future. We consider a model-predictive setting where the system's dynamic model is available and can be leveraged to enhance monitoring accuracy. However, existing approaches are limited to a restricted class of STL formulae, permitting only a single application of temporal operators. This work addresses the challenge of nested temporal operators in the design of model-predictive monitors. Our method utilizes syntax tree structures to resolve dependencies between temporal operators and introduces the concept of basic satisfaction vectors. A new model-predictive monitoring algorithm is proposed by recursively updating these vectors online while incorporating pre-computed satisfaction regions derived from offline model analysis. We prove that the proposed approach is both sound and complete, ensuring no false alarms or missed alarms. Case studies are provided to demonstrate the effectiveness of our method.**

## I. INTRODUCTION

Specification-based monitoring has emerged as a popular approach for evaluating the safety and correctness of complex engineering cyber-physical systems, such as smart cities [16], autonomous vehicles [21], power plants [3], and industrial IoTs [6]. In this context, a monitor observes the state trajectory generated by the system and evaluates the correctness of the trajectory based on a given formal specification [2], [23], [4]. Compared with formal verification techniques such as model checking, the key advantage of monitoring is that it is more lightweight, as it only assesses the correctness of the system along a single observed trajectory without explicitly enumerating the entire reachable state space. Therefore, monitoring can be designed independently and implemented as an add-on module for arbitrary systems—even those treated as black boxes.

In recent years, significant advancements have been made in designing monitoring algorithms for various types of formal specifications, such as Linear Temporal Logic (LTL) [11], Metric Temporal Logic (MTL) [9], [10], and Signal Temporal Logic (STL) [8], [27]. Among these, STL has become one of the most widely used formal specifications for CPS due to its ability to characterize complex spatio-temporal constraints in real-valued, real-time physical signals. Monitoring algorithms can be further categorized into offline monitoring [12], [10] and online monitoring [9], [8], depending on the information available to the monitor. In the offline setting, the monitor evaluates the correctness of a system using the entire trajectory. However, this approach is unsuitable for systems operating in real time, where only the signal generated up to the current moment is accessible. In contrast, online monitoring algorithms must account for all possible future behaviors to issue early warnings or terminate processes before a specification violation occurs. As a result, online monitoring is widely adopted in safety-critical systems as a predictive add-on component to ensure operational safety.

Due to the physical dynamics of the system, online signals are not arbitrary in general but follow underlying rules. By leveraging information about system behavior, we can better reason about the feasibility or likelihood of future signals, enabling more precise monitoring decisions [1]. To this end, *predictive monitoring* has recently emerged for systems with either an explicit model or a large dataset of operational history. For example, in [20], [17], [24], [19], neural networks trained on historical data are used to predict future signals in order to improve monitoring accuracy. However, since these approaches adopt a data-driven methodology, their predictions may be unreliable. To address this, uncertainty quantification techniques, such as conformal predictions, are often employed to ensure prediction confidence [5], [15], [28]. While effective, these methods may introduce additional conservatism into the final monitoring results.

More recently, by leveraging the explicit system model of the underlying dynamical system, a new framework called *model-predictive monitoring* has been proposed in [26]. This approach further improves monitoring accuracy when a precise system model is available. By integrating offline reachability computations with online satisfaction evaluation, it achieves an effective balance between computational complexity and accuracy. In [22], a self-triggered information acquisition mechanism is introduced for model-predictive monitoring to reduce state-sampling overhead. However, existing model-predictive monitoring algorithms remain limited to a simple fragment of STL specifications and nested temporal operators, e.g., "eventually always stays in a region", are currently unsupported.

In this work, we investigate the design of model-predictive online monitors for signal temporal logic specifications. Un-

This work was supported by the National Natural Science Foundation of China (62173226, 62061136004).

T. Han, S. Li and X. Yin are with the School of Automation and Intelligent Sensing, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: {coolhantao,syli,yinxiang}@sjtu.edu.cn.

like existing approaches, we consider a general fragment of STL that permits nested temporal operators. This nested setting introduces fundamental challenges beyond prior works, which track STL formula progress using an index set of remaining formulae. However, this approach fails under nested temporal constraints due to dependencies between operators. To address this, we propose to use syntax trees [13], [25], [7] to resolve temporal operator dependencies, and use basic satisfaction vectors as dynamically updated key information during online monitoring. Our algorithm leverages these vectors, updated recursively in real time, along with pre-computed safety regions derived from offline model analysis. This framework successfully extends model-predictive monitoring to general STL specifications. Case studies are also provided to demonstrate the effectiveness of our approach.

The remainder of the paper is organized as follows. In Section II, we first present some basic preliminaries. The model-predictive monitoring problem is then formulated in Section III. In Section IV, we present the syntax tree structure and show how to store the progress of nested STL formulae using basic satisfaction vectors. The main online monitoring algorithm is provided in Section V, where the offline pre-computation of satisfaction regions is also discussed. The proposed results are illustrated by case studied in Section VI. Finally, we conclude the paper in Section VII.

## II. Preliminary

### A. System Model

We consider a discrete-time control system of form

$$x_{k+1} = f(x_k, u_k), \tag{1}$$

where $x_k \in \mathcal{X} \subset \mathbb{R}^n$ denotes the system state at time instant $k \in \mathbb{Z}_{\geq 0}$, $u_k \in \mathcal{U} \subset \mathbb{R}^m$ represents the control input at time instant $k$ and $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is the dynamic function of the system, assumed to be continuous in $\mathcal{X} \times \mathcal{U}$. We assume that the state space $\mathcal{X}$ and input space $\mathcal{U}$ are both bounded due to physical constraints.

Suppose that the system state is $x_k$ at time instant $k$. Then given a sequence of control inputs $\mathbf{u}_{k:T-1} = u_k u_{k+1} \dots u_{T-1} \in \mathcal{U}^{T-k}$, the corresponding state trajectory generated by the system is defined as $\xi_f(x_k, \mathbf{u}_{k:T-1}) = \mathbf{x}_{k+1:T} = x_{k+1} \dots x_T \in \mathcal{X}^{T-k}$, where each subsequent state satisfies the recursive relation $x_{i+1} = f(x_i, u_i)$ for all $i = k, \dots, T-1$.

### B. Signal Temporal Logic

We adopt Signal Temporal Logic (STL) as the formal specification language to evaluate trajectory correctness. The syntax of STL formulae is recursively defined as:

$$\Phi ::= \top \mid \pi^\mu \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \mathbf{U}_{[a,b]} \Phi_2,$$

where $\top$ is the *true* predicate and $\pi^\mu$ is an atomic predicate whose truth value is determined by the sign of its underlying predicate function $\mu : \mathbb{R}^n \to \mathbb{R}$. Specifically, an atomic predicate $\pi^\mu$ is true at state $x_k$ when $\mu(x_k) \geq 0$; otherwise it is false. Operators $\neg$ and $\wedge$ are the standard Boolean operators "negation" and "conjunction", respectively. One can further use them to induce other operators such as "disjunction" $\Phi_1 \vee \Phi_2 := \neg(\neg\Phi_1 \wedge \neg\Phi_2)$ and "implication" $\Phi_1 \to \Phi_2 := \neg\Phi_1 \vee \Phi_2$. $\mathbf{U}_{[a,b]}$ is the temporal operator "until", where $a, b \in \mathbb{Z}_{\geq 0}$ are two integers with $a \leq b$. Note that, since we consider discrete-time setting, $[a, b]$ is the set of all integers between $a$ and $b$ including themselves.

Let $\mathbf{x} = x_0 x_1 \dots$ be a state sequence, $k \in \mathbb{Z}_{\geq 0}$ be a time instant and $\Phi$ be an STL formula. We denote by $(\mathbf{x}, k) \models \Phi$ if sequence $\mathbf{x}$ satisfies STL formula $\Phi$ at time instant $k$. The reader is referred to [18] for more details on the semantics of STL formulae. Particularly, for atomic predicates,, we have $(\mathbf{x}, k) \models \pi^\mu$ iff $\mu(x_k) \geq 0$, i.e., $\mu(x_k)$ is non-negative for the current state $x_k$, and for temporal operators, we have $(\mathbf{x}, k) \models \Phi_1 \mathbf{U}_{[a,b]} \Phi_2$ iff there exists $k' \in [k+a, k+b]$ such that

$$(\mathbf{x}, k') \models \Phi_2 \wedge (\forall k'' \in [k, k'])[(\mathbf{x}, k'') \models \Phi_1],$$

i.e., $\Phi_2$ will eventually be satisfied at some instant between $[k+a, k+b]$ and $\Phi_1$ holds consistently before then. Furthermore, we can also induce temporal operators:

- "*eventually*" $\mathbf{F}_{[a,b]}\Phi := \top \mathbf{U}_{[a,b]} \Phi$ such that it holds when $(\mathbf{x}, k) \models \Phi$ for some $k' \in [k+a, k+b]$;
- "*always*" $\mathbf{G}_{[a,b]}\Phi := \neg \mathbf{F}_{[a,b]} \neg\Phi$ such that it holds when $(\mathbf{x}, k) \models \Phi$ for any $k' \in [k+a, k+b]$.

We write $\mathbf{x} \models \Phi$ whenever $(\mathbf{x}, 0) \models \Phi$. We assume that the operation horizon $T$ is sufficiently long to evaluate the satisfaction of $\Phi$.

## III. Problem Formulation

### A. Fragment of STL Formulae

In this paper, we consider the following slightly restricted fragments of STL formulae

$$\varphi ::= \top \mid \pi^\mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \tag{2a}$$

$$\Phi ::= \mathbf{F}_{[a,b]}\Phi \mid \mathbf{G}_{[a,b]}\Phi \mid \Phi_1 \mathbf{U}_{[a,b]} \Phi_2 \mid \Phi_1 \wedge \Phi_2 \mid \varphi. \tag{2b}$$

Note that, compared to the full fragment of STL, we require that negation can only be applied to Boolean operators. Therefore, the overall STL formula considered is the conjunction of a set of sub-formulae in which nested temporal operators can be applied to an arbitrary Boolean formula.

For technical purposes, we introduce a new temporal operator $\mathbf{U}'$, defined by: $(\mathbf{x}, k) \models \Phi_1 \mathbf{U}'_{[a,b]} \Phi_2$ iff there exists $k' \in [k+a, k+b]$ such that

$$(\mathbf{x}, k') \models \Phi_2 \wedge (\forall k'' \in [k+a, k'])[(\mathbf{x}, k'') \models \Phi_1].$$

Compared to the original definition of "until," the key difference is that the effective horizon of $\mathbf{U}_{[a,b]}$ is $[0, b]$, while the effective horizon of $\mathbf{U}'_{[a,b]}$ is $[a, b]$. Throughout this paper, we will refer to $\mathbf{U}'_{[a,b]}$ as the "until" operator. This replacement is primarily technical and does not lose generality, as the standard $\mathbf{U}$ can always be expressed as

$$(\mathbf{x}, k) \models \Phi_1 \mathbf{U}_{[a,b]} \Phi_2 \Leftrightarrow (\mathbf{x}, k) \models (\Phi_1 \mathbf{U}'_{[a,b]} \Phi_2) \wedge (\mathbf{G}_{[0,a]} \Phi_1).$$

Recall that for the predicate $\pi^\mu$ in (2a), its satisfaction region, denoted by $\mathcal{H}^\mu$, is the solution to the inequality

$\mu(x) \geq 0$, i.e., $\mathcal{H}^\mu = \{x \in \mathcal{X} \mid \mu(x) \geq 0\}$. For other Boolean operators, we have $\mathcal{H}^{\neg\varphi} = \mathcal{X} \setminus \mathcal{H}^\varphi$ and $\mathcal{H}^{\varphi_1 \wedge \varphi_2} = \mathcal{H}^{\varphi_1} \cap \mathcal{H}^{\varphi_2}$. Therefore, instead of writing $\varphi$, we will hereafter simply denote it as $x \in \mathcal{H}^\mu$ or $x \in \mathcal{H}$, using its satisfaction region.

Additionally, while we consider the temporal operator "eventually" ($\mathbf{F}$) in the semantics, it is subsumed by "until" ($\mathbf{U}'$) since $\mathbf{F}_{[a,b]}\Phi$ can be expressed as $x \in \mathcal{X}\mathbf{U}'_{[a,b]}\Phi$. Thus, we only need to handle the temporal operators $\mathbf{G}$ and $\mathbf{U}'$ from a technical standpoint.

Based on the above discussion, the STL formula $\Phi$ in (2) can be equivalently expressed as:

$$\Phi ::= \mathbf{G}_{[a,b]}\Phi \mid \Phi_1 \mathbf{U}'_{[a,b]}\Phi_2 \mid \Phi_1 \wedge \Phi_2 \mid x \in \mathcal{H} \tag{3}$$

### B. Online Monitoring of STL

At time instant $k \leq T$, the system only generates a partial signal $\mathbf{x}_{0:k} = x_0 x_1 \ldots x_k$ (called a prefix) and the remaining signals $\mathbf{x}_{k+1:T}$ (called suffix) will only be available in the future. Therefore, we a partial signal $\mathbf{x}_{0:k}$, we denote by

- $\mathbf{x}_{0:k} \models \Phi$ if $\forall \mathbf{x}_{k+1:T} \in \mathcal{X}^{T-k} : \mathbf{x}_{0:k}\mathbf{x}_{k+1:T} \models \Phi$;
- $\mathbf{x}_{0:k} \not\models \Phi$ if $\forall \mathbf{x}_{k+1:T} \in \mathcal{X}^{T-k} : \mathbf{x}_{0:k}\mathbf{x}_{k+1:T} \not\models \Phi$;
- $\mathbf{x}_{0:k} \models_? \Phi$ otherwise.

Note that the above partial signal evaluation does not account for system dynamics. In other words, some suffixes in $\mathbf{x}_{k+1:T} \in \mathcal{X}^{T-k}$ may be dynamically infeasible. Thus, in the context of *model-predictive monitoring*, we further classify a prefix signal $\mathbf{x}_{0:k}$ as

- **violated** if, for any control input $\mathrm{u}_{k:T-1}$, we have $\mathbf{x}_{0:k}\xi_f(x_k, \mathrm{u}_{k:T-1}) \not\models \Phi$;
- **feasible** if, for some control input $\mathrm{u}_{k:T-1}$, we have $\mathbf{x}_{0:k}\xi_f(x_k, \mathrm{u}_{k:T-1}) \models \Phi$.

Intuitively, a prefix signal is violated when either the current state already violates the specification or when all possible future trajectories will violate it inevitably. For instance, consider a safety specification $\mathbf{G}_{[0,T]} x \in \mathcal{H}$. If the system reaches any state $x_k \notin \mathcal{H}$ for $k < T$, this immediately constitutes a violation. More subtly, even when the current state satisfies the predicate, the prefix is violated if from state $x_k$ there exists no control sequence $\mathrm{u}_{k:T-1}$ that can generate a trajectory $\xi_f(x_k, \mathrm{u}_{k:T-1})$ remaining entirely within $\mathcal{H}$ throughout the remaining horizon. This mirrors real-world scenarios such as a vehicle approaching an obstacle at excessive speed. Even before physical collision occurs, the system is already violated because no braking action can prevent the impending violation given the current velocity and distance.

**Problem 1** (**Model-Predictive Online Monitoring).** Given a system with dynamic in (1) and an STL formula $\Phi$, design an online monitor

$$\mathcal{M} : \mathcal{X}^* \to \{\texttt{vio}, \texttt{feas}\} \tag{4}$$

such that, for any prefix $\mathbf{x}_{0:k} \in \mathcal{X}^*$, we have

- $\mathcal{M}(\mathbf{x}_{0:k}) = \texttt{vio}$ iff $\mathbf{x}_{0:k}$ is violated;
- $\mathcal{M}(\mathbf{x}_{0:k}) = \texttt{feas}$ iff $\mathbf{x}_{0:k}$ is feasible.

## IV. Tree of Nested STL Formulae

The main challenge in handling STL formulae with nested temporal operators lies in the dependency between inner and outer operators. For instance, consider the STL formula $\mathbf{G}_{[a_1,b_1]}\mathbf{F}_{[a_2,b_2]}\varphi$. This formula requires that the inner formula $\mathbf{F}_{[a_2,b_2]}\varphi$ must be satisfied for all time instants between $a_1$ and $b_1$. Specifically, starting from any time instant within $[a_1, b_1]$, the system should satisfy $\varphi$ within the interval $[a_2, b_2]$ from *that point onward*. Consequently, evaluating the correctness of a trajectory requires a total horizon of $b_1 + b_2$. To address this dependency clearly, we introduce the concept of *syntax tree* and the notion of *satisfaction vector*. These tools help systematically resolve the dependencies between nested temporal operators.

### A. Syntax Trees for STL formulae

Hereafter, we will equivalently represent an STL formula by a *rooted tree*, which is a 4-tuple

$$\mathcal{T} = (V, L, E, v_{\text{root}}),$$

where:

- $V = \{v_1, v_2, \ldots, v_m\}$ is the set of nodes;
- $L = \{l, r, \perp\}$ is a label set, where $l$ and $r$ denote "left" and "right", respectively, and $\perp$ denotes "no order";
- $E \subseteq V \times L \times V$ is the set of edges, where each $(v, \ell, v') \in E$ indicates that $v'$ is a child node of $v$, and is specifically a left or right child if $\ell = l$ or $\ell = r$, respectively;
- $v_{\text{root}} \in V$ is the unique root node with no parent.

Clearly, a rooted tree induces a partial order $\prec$ on $V$, where $v' \prec v$ iff $v$ is an ancestor of $v'$. For each node $v$, we denote by $\mathsf{child}(v) = \{v' \mid \exists \ell \in L : (v, \ell, v') \in E\}$ the set of its children. Additionally, we denote by $\mathsf{child\text{-}l}(v)$ and $\mathsf{child\text{-}r}(v)$ the left and right child of $v$, respectively, if they exist.

In order to represent an STL formula as a tree, we introduce the following four types of nodes:

- $\mathcal{H}$-nodes: Each node represents a satisfaction region of a Boolean formula. Such nodes have no children, as Boolean formulae are always at the innermost level of the entire formula.
- $\wedge$-nodes: Each node represents the Boolean operator "conjunction" and has two or more unordered children. These nodes are evaluated instantly, meaning there is no time interval associated with them.
- $\mathbf{G}$-nodes: Each node represents the temporal operator "always" and has exactly one child. Such nodes are associated with a time interval that determines the evaluation period of their descendants.
- $\mathbf{U}'$-nodes: Each node represents the temporal operator "until" and has both a left and a right child. These nodes are also associated with a time interval that determines the evaluation period of their descendants.

Now, we are ready to formally define the syntax tree.

**Definition 1** (**Syntax Trees).** Let $\Phi$ be an STL formula defined by syntax in (3). The syntax tree of STL formula $\Phi$, denoted by $\mathcal{T}_\Phi = (V_\Phi, L, E_\Phi, v_{\text{root},\Phi})$, is defined recursively as follows:

- If $\Phi = \Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_n$, then $V_\Phi = (\cup_{i=1}^n V_{\Phi_i}) \cup \{v_\wedge\}$, $E_\Phi = (\cup_{i=1}^n E_{\Phi_i}) \cup \{(v_\wedge, \perp, v_{\text{root},\Phi_i}) \mid i = 1, \ldots, n\}$, and $v_{\text{root},\Phi} = v_\wedge$ , where $v_\wedge$ is a new $\wedge$-node that does not exist in each $\mathcal{T}_{\Phi_i}$;
- If $\Phi = \mathbf{G}_{[a,b]}\Phi'$, then $V_\Phi = V_{\Phi'} \cup \{v_\mathbf{G}\}$, $E_\Phi = E_{\Phi'} \cup \{(v_\mathbf{G}, \perp, v_{\text{root},\Phi'})\}$, and $v_{\text{root},\Phi} = v_\mathbf{G}$, where $v_\mathbf{G}$ is a new $\mathbf{G}$-node does not exist in $\mathcal{T}_{\Phi'}$ and it is associated with time interval $[a, b]$;
- If $\Phi = \Phi_1 \mathbf{U'}_{[a,b]} \Phi_2$, then $V_\Phi = V_{\Phi_1} \cup V_{\Phi_2} \cup \{v_{\mathbf{U'}}\}$, $E_\Phi = E_{\Phi_1} \cup E_{\Phi_2} \cup \{(v_\mathbf{G}, l, v_{\text{root},\Phi_1}), (v_\mathbf{G}, r, v_{\text{root},\Phi_2})\}$, and $v_{\text{root},\Phi} = v_{\mathbf{U'}}$ , where $v_{\mathbf{U'}}$ is a new $\mathbf{U'}$-node does not exist in $\mathcal{T}_{\Phi_1}$ or $\mathcal{T}_{\Phi_2}$, and it is also associated with time interval $[a, b]$;
- If $\Phi = x \in \mathcal{H}$, then $\mathcal{T}_\Phi$ only contains a single $\mathcal{H}$-node associated with predicate $x \in \mathcal{H}$.

For each node $v \in V_\Phi$, we denote by $[a^v, b^v]$ its **associated time interval**, with $a^v = b^v = 0$ when $v$ is a $\wedge$-node or an $\mathcal{H}$-node. Also, for each $\mathcal{H}$-node $v \in V_\mathcal{H}$, we denote by $\mathcal{H}^v$ the associated predicate region. Hereafter, we will omit the subscript $\Phi$ and refer to $\mathcal{T} = (V, L, E, v_{\text{root}})$ as the syntax tree of the formula $\Phi$ when the context is clear. For $\star \in \{\wedge, \mathbf{G}, \mathbf{U'}, \mathcal{H}\}$, we denote by $V_\star \subseteq V$ the set of all $\star$-nodes in $\mathcal{T}$. Furthermore, let

$$V = \{v_1, v_2, \ldots, v_m\},$$

and we assume that all $\mathcal{H}$-nodes are ordered as the first $h \geq 1$ elements, i.e., $V_\mathcal{H} = \{v_1, \ldots, v_h\}$.

### *B. Satisfaction Vector*

As discussed, for each node $v \in V$ in the syntax tree, it evaluates its descendants nodes within the time interval $[a^v, b^v]$. Note that this interval is relative to the local perspective, as $v$ may have ancestor nodes in the tree corresponding to outer operators whose time intervals also apply to $v$. Therefore, to determine the absolute time interval for evaluating node $v$ from a global perspective, one must consider the local time intervals of all its ancestors.

Formally, for each node $v \in V$, we denote by $\mathsf{ances}(v) = \{v' \in V \mid v \prec v'\}$ the set of all its ancestors. This set includes all nodes on the unique path from the root node to $v$ in $\mathcal{T}$, excluding $v$ itself. Based on this, we introduce the following definition of *evaluation horizon*.

**Definition 2 (Evaluation Horizons).** Let $\mathcal{T}$ be the syntax tree of STL formula $\Phi$ and $v \in V$ be a node. Then the *evaluation horizon* of node $v$ is defined by

$$[\mathsf{int}^v, \mathsf{end}^v] = [\textstyle\sum_{v' \in \mathsf{ances}(v)} a^{v'}, \sum_{v' \in \mathsf{ances}(v)} b^{v'}].$$

For example, consider the following STL formula

$$\Phi = ((\mathbf{G}_{[1,3]} x \in \mathcal{H}_1) \mathbf{U'}_{[2,5]} x \in \mathcal{H}_2) \wedge (\mathbf{G}_{[3,7]} x \in \mathcal{H}_3). \quad (5)$$

Its syntax tree $\mathcal{T}$ is shown in Fig. 1. For $\mathcal{H}$ node $v_1$, its evaluation horizon is $[\mathsf{int}^{v_1}, \mathsf{end}^{v_1}] = [2 + 1 = 3, 5 + 3 = 8]$.

Therefore, for each time instant $t \in [\mathsf{int}^v, \mathsf{end}^v]$, it is meaningful to discuss the satisfaction status of node $v$. In

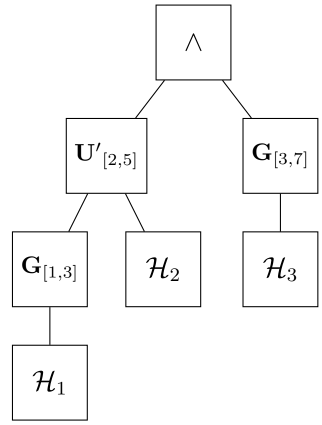


Fig. 1: Syntax Tree of STL formula (5)

general, there are three possible satisfaction statuses for a node: (i) **satisfied**, denoted by "1"; (ii) **violated**, denoted by "0"; and (iii) **uncertain**, denoted by "?". The last case arises when there is insufficient information to evaluate the node. For example, consider the formula $\mathbf{G}_{[10,12]} x \in \mathcal{H}$. If the current time instant is $k = 8$, then the satisfaction status of the $\mathcal{H}$-node for $t = 10, 11, 12$ is uncertain because the evaluation horizon of the $\mathcal{H}$-node has not yet been reached. However, if at time instant $k = 10$, the current state $x_k$ falls within the region $\mathcal{H}$, then the satisfaction status of the $\mathcal{H}$-node at $t = 10$ becomes 1. However, the satisfaction status for future instants $t = 11, 12$ remains uncertain.

To capture this, we define the satisfaction status for each time instant within the evaluation horizon of a node as the *satisfaction vector*, as follows.

**Definition 3 (Satisfaction Vectors).** Let $\mathcal{T}$ be the syntax tree of STL formula $\Phi$ and $v \in V$ be a node with evaluation horizon $[\mathsf{int}^v, \mathsf{end}^v]$. Then a *satisfaction vector* of node $v$ is a vector of form

$$\imath = (\imath[\mathsf{int}^v], \imath[\mathsf{int}^v + 1], \ldots, \imath[\mathsf{end}^v]) \in \{0, 1, ?\}^H,$$

where $H = \mathsf{end}^v - \mathsf{int}^v + 1$ is the length of its evaluation horizon and $\imath[t], t = \mathsf{int}^v, \ldots, \mathsf{end}^v$ denotes its $t$-th element.

In the above definition, we count the first element of $\imath$ starting from $\imath[\mathsf{int}^v]$ rather than $\imath[1]$ to align with the absolute time instant from the perspective of the root node. This ensures consistency between the indexing of the satisfaction vector and the actual time instants being evaluated.

For each node, its satisfaction vector is essentially determined by the vectors of its children nodes. Through an inductive argument based on the tree structure, once the satisfaction vector of each $\mathcal{H}$-node is known, we can compute the satisfaction vectors of all non-leaf nodes. Therefore, we refer to the satisfaction vector of an $\mathcal{H}$-node as a *basic vector*, as it serves as the foundation for *inducing vectors* for other nodes. This leads to the following definitions.

**Definition 4 (Basic Vectors).** A *basic set* of satisfaction vectors is a tuple of satisfaction vectors of the form

$$I = (\imath^{v_1}, \ldots, \imath^{v_h}) \in \{0, 1, ?\}^{H_1} \times \cdots \times \{0, 1, ?\}^{H_h},$$

where $V_{\mathcal{H}} = \{v_1, \dots, v_h\}$ is the set of all $\mathcal{H}$-nodes and for each $i = 1, \dots, h$, $H_i = \mathsf{end}^{v_i} - \mathsf{int}^{v_i} + 1$ represents the length of the evaluation horizon of $v_i$.

Hereafter, we will maintain basic vectors as time-updated information. For each time instant $k = 0, 1, \dots, T$, a basic set $I_k = (\imath_k^{v_1}, \dots, \imath_k^{v_h})$ must satisfy the following condition for each $v \in V_{\mathcal{H}}$ and $t = \mathsf{int}^v, \dots, \mathsf{end}^v$:

- If $t \geq k$, then $\imath_k^v[t] = ?$ (unknown status for futures); and
- If $t < k$, then $\imath_k^v[t] \neq ?$ (determined status for pasts).

This reflects the constraint that satisfaction status cannot be evaluated for future time instants, while for past instants the status must be resolved to either 0 (violated) or 1 (satisfied). We denote by $\mathcal{I}_k$ the set of all potential basic vectors at instant $k$ satisfying the above two constraints.

**Definition 5** (**Induced Vectors).** Given a basic set of satisfaction vectors $I = (\imath^{v_1}, \dots, \imath^{v_h})$, the *induced set of satisfaction vectors* for the remaining nodes, denoted by $I^+ = (\imath^{v_{h+1}}, \dots, \imath^{v_m})$, is defined recursively by: for each $v \in \{v_{h+1}, \dots, v_m\}$, we have

- If $v \in V_\wedge$, then for each $t = \mathsf{int}^v, \dots, \mathsf{end}^v$, we have
$$\imath^v[t] = \begin{cases} 0 & \text{if } \exists v' \in \mathsf{child}(v) : \imath^{v'}[t] = 0 \\ 1 & \text{if } \forall v' \in \mathsf{child}(v) : \imath^{v'}[t] = 1 \\ ? & \text{otherwise} \end{cases} .$$
- If $v \in V_{\mathbf{G}}$, then for each $t = \mathsf{int}^v, \dots, \mathsf{end}^v$, we have
$$\imath^v[t] = \begin{cases} 0 & \text{if } \exists t' \in [a^v, b^v] : \imath^{v'}[t + t'] = 0 \\ 1 & \text{if } \forall t' \in [a^v, b^v] : \imath^{v'}[t + t'] = 1 \\ ? & \text{otherwise} \end{cases} .$$
where $v'$ is the unique child node of $v$.
- If $v \in V_{\mathbf{U}'}$, then for each $t = \mathsf{int}^v, \dots, \mathsf{end}^v$, we have
$$\imath^v[t] = \begin{cases} 0 & \text{if } \begin{matrix} (\forall t' \in [a^v, b^v])(\imath^{v_r}[t+t'] = 0) \\ \textbf{or} \\ \begin{bmatrix} (\forall t' \in [a^v, b^v] : \imath^{v_r}[t+t'] = 1) \\ (\exists t'' \in [a^v, t'])(\imath^{v_l}[t+t''] = 0) \end{bmatrix} \end{matrix} \\ 1 & \text{if } \begin{bmatrix} (\exists t' \in [a^v, b^v] : \imath^{v_r}[t+t'] = 1) \\ (\forall t'' \in [a^v, t'])(\imath^{v_l}[t+t''] = 1) \end{bmatrix} \\ ? & \text{otherwise} \end{cases} ,$$
where $v_l$ and $v_r$ denotes the left and right child nodes of $v$, respectively.

The intuition behind the above definition is essentially a reinterpretation of the semantics of STL formulae. Note that the induced vectors are well-defined because all nodes are ordered in the tree structure, where the basic vectors representing $\mathcal{H}$-nodes are leaves in the tree with no children. Therefore, given a basic set of satisfaction vectors, one can compute its induced vectors in a bottom-up manner, starting from the leaf nodes and progressing to the root node.

## V. Online Monitoring Algorithms

### A. Evolution of Basic Vectors

In order to track the progress of the STL formulae without storing the entire state trajectory, our approach is to maintain only the basic satisfaction vectors and update them recursively upon observing the new system state at each time instant. This recursive computation proceeds as follows:

- **Initialization:** At time instant $k = 0$, prior to observing any system state, the initial basic set is defined as:
$$I_0 = (\imath_0^{v_1}, \imath_0^{v_2}, \dots, \imath_0^{v_h}),$$
where each $\imath_0^{v_i} = (?, ?, \dots, ?)$ is a vector with all entries initialized to the uncertain status "?". This reflects complete uncertainty about future satisfaction before any state observations are made.
- **Online Update:** At time instant $k$, given the current basic satisfaction vector set $I_k = (\imath_k^{v_1}, \dots, \imath_k^{v_h})$, the update upon observing a new state $x_k$ is defined as:
$$I_{k+1} = (\imath_{k+1}^{v_1}, \dots, \imath_{k+1}^{v_h}) = \mathtt{update}(I_k, x_k),$$
where for each $v_i \in \{v_1, \dots, v_h\}$ and $t \in [\mathsf{int}^{v_i}, \mathsf{end}^{v_i}]$, the updated satisfaction vector entry is:
$$\imath_{k+1}^{v_i}[t] = \begin{cases} \imath_k^{v_i}[t] & \text{if } t \neq k, \\ 0 & \text{if } t = k \wedge x_k \notin \mathcal{H}^{v_i}, \\ 1 & \text{if } t = k \wedge x_k \in \mathcal{H}^{v_i}. \end{cases} \tag{6}$$
Therefore, the uncertain status "?" in the basic vectors is resolved at time $k$ based on the observed state $x_k$.

Let $\mathbf{x}_{0:k-1} = x_0 x_1 \dots x_{k-1}$ be a state sequence of length $k$. We denote by $I(\mathbf{x}_{0:k-1})$ the basic set of satisfaction vectors reached recursively by the state sequence $\mathbf{x}_{0:k-1}$ from $I_0$. On the other hand, for any basic set $I$, we define the set of state sequences of length $k$ consistent with $I$ as
$$\mathbf{x}_{0:k-1}^I = \{\mathbf{x}_{0:k-1} \in \mathcal{X}^k \mid I(\mathbf{x}_{0:k-1}) = I\}.$$

The following result demonstrates that the recursive computation of the basic set above indeed captures all task-relevant information from the complete state trajectory.

**Proposition 1.** Let $I$ be a basic set and $k$ be a time instant. For any two sequences $\mathbf{x}'_{0:k-1}, \mathbf{x}''_{0:k-1} \in \mathbf{x}^I_{0:k-1}$ consistent with $I$, and any future sequence $\mathbf{x}_{k:T} = x_k x_{k+1} \cdots x_T$, we have
$$\mathbf{x}'_{0:k-1}\mathbf{x}_{k:T} \models \Phi \quad \Leftrightarrow \quad \mathbf{x}''_{0:k-1}\mathbf{x}_{k:T} \models \Phi. \tag{7}$$

*Proof:* Due to space constraint, all proofs in the paper are omitted and they are available in the supplementary materials `https://xiangyin.sjtu.edu.cn/25CDC-STL.pdf`

Based on the above result, it is meaningful to write
$$\mathbf{x}^I_{0:k-1}\mathbf{x}_{k:T} \models \Phi \tag{8}$$
whenever $\mathbf{x}'_{0:k-1}\mathbf{x}_{k:T} \models \Phi$ holds for some (or equivalently, for all, due to Proposition 1) $\mathbf{x}'_{0:k-1} \in \mathbf{x}^I_{0:k-1}$.

Given a basic set of satisfaction vectors $I$, its induced set of satisfaction vectors is defined as $I^+ = (\imath^{v_{h+1}}, \dots, \imath^{v_m})$. We call the satisfaction vector corresponding to the root node $v_{\text{root}}$ in $I^+$ the *root satisfaction vector*, denoted by $\imath_I^{\text{root}} \in I^+$. With lose of generality, we assume that the root node is always a $\wedge$-node, whose evaluation horizon is $[0, 0]$.

The following result demonstrates that this root vector fully captures the satisfaction status of the entire STL formula.

**Proposition 2.** For any state sequence $\mathbf{x}_{0:k}$, we have

$$\mathbf{x}_{0:k} \models \Phi \quad \Leftrightarrow \quad \imath^{\text{root}}_{I(\mathbf{x}_{0:k})}[0] = 1. \tag{9}$$

For simplicity, we will write $\imath^{\text{root}}_{I(\mathbf{x}_{0:k})} = 0, 1, ?$ whenever $\imath^{\text{root}}_{I(\mathbf{x}_{0:k})}[0] = 0, 1, ?$ as the root vector only has one element.

### *B. Model-Predictive Monitor*

By recursively maintaining the basic set $I_k = I(\mathbf{x}_{0:k-1})$, we can draw the following conclusions about the current status of the entire specification formula:

- If $\imath^{\text{root}}_{I_k} = 0$, the specification has been violated, and the monitor should output $\mathcal{M}(\mathbf{x}_{0:k-1}) = \texttt{vio}$;
- If $\imath^{\text{root}}_{I_k} = 1$, the specification has been satisfied, and no further monitoring is required.

These conclusions are independent of the system dynamics, as the satisfaction status is fully determined in these cases. However, when $\imath^{\text{root}}_{I_k} =?$, both $\texttt{vio}$ and $\texttt{feas}$ remain possible outcomes, requiring analysis of the system dynamics. This leads to the following key definition.

**Definition 6** (**$I$-Determined Feasible Sets).** Let $\Phi$ be an STL formula, $k \in [0, T]$ be a time instant and $I$ be a basic set. Then the *$I$-determined feasible set* at instant $k$, denoted by $X^I_k \subseteq \mathcal{X}$, is the set of states from which there exists a solution $\mathbf{u}_{k:T-1}$ that satisfies $\Phi$ given the current basic set $I$, i.e.,

$$X^I_k = \left\{ x_k \in \mathcal{X} \;\middle|\; \begin{array}{c} \exists \mathbf{u}_{k:T-1} \in \mathcal{U}^{T-k} \\ \text{s.t. } \mathbf{x}^I_{0:k-1} x_k \xi_f(x_k, \mathbf{u}_{k:T-1}) \models \Phi \end{array} \right\} \tag{10}$$

Based on the above notion, now we present our main online monitoring algorithm as shown in Algorithm 1. The algorithm initializes with the time instant $k = 0$ and the initial basic set $I = I_0$ (line 1). The monitoring process iterates as long as the STL formula remains unresolved (line 2). At each iteration, the current state $x_k$ is read (line 3) and checked against the feasible set $X^I_k$. If $x_k \notin X^I_k$, the algorithm immediately terminates with a violation decision (lines 4-6). Otherwise, the monitoring decision is set to $\mathcal{M} = \texttt{feas}$, and the basic set $I$ is recursively updated by $\texttt{update}(I, x_k)$ (lines 7-9). The time instant is then increased (line 10), and the loop continues until either a violation is detected or the STL formula is satisfied.

The correctness of Algorithm is established as follows.

**Theorem 1.** The online monitor $\mathcal{M}$ defined by Algorithm 1 indeed solves Problem 1.

### *C. Offline Computation of Feasible Sets*

The proposed online monitoring algorithm utilizes pre-computed all feasible sets $X^I_k$ at each time instant. This subsection details the offline computation of these sets.

Note that, at each time instant $k$, we only need to consider those basic vectors for which the entire STL formula $\Phi$ has not been violated. Therefore, we define

$$\mathbb{I}_k = \{ I \in \mathcal{I}_k \mid \imath^{\text{root}}_I \neq 0 \}$$

---

**Algorithm 1:** Online Monitoring algorithm

**Input:** Feasible sets $X^I_k$ computed offline
**Output:** Online monitoring decision $\mathcal{M}(\mathbf{x}_{0:k})$

1 $k \leftarrow 0$, $I \leftarrow I_0$
2 **while** $\imath^{\text{root}}_I \neq 1$ **do**
3   observe a new current state $x_k$
4   **if** $x_k \notin X^I_k$ **then**
5     $\mathcal{M}$ issues "$\texttt{vio}$"
6     **return** *"$\Phi$ is violated"*
7   **else**
8     $\mathcal{M}$ issues "$\texttt{feas}$"
9     $I \leftarrow \texttt{update}(I, x_k)$
10   $k \leftarrow k + 1$

---

the set of **feasible basic vectors** at time instant $k$.

Furthermore, the basic vector set cannot be updated arbitrarily in the next time instant as it should be consistent with the existing history. Therefore, let $I = (\imath^{v_1}, \ldots, \imath^{v_h}) \in \mathbb{I}_k$ be a basic vector set for time instant $k$ and $I_s = (\imath^{v_1}_s, \ldots, \imath^{v_h}_s) \in \mathbb{I}_{k+1}$ be a basic vector set for time instant $k+1$. We say $I_s \in \mathbb{I}_{k+1}$ is a **successor basic set** of $I \in \mathbb{I}_k$ at instant $k$ if

$$\forall v \in V_{\mathcal{H}}, \forall t \in [\mathsf{int}^v, \min(k-1, \mathsf{end}^v)] : \imath^v[t] = \imath^v_s[t].$$

We denote by $\mathrm{succ}(I, k) \subseteq \mathbb{I}_{k+1}$ the set of all successor basic sets of $I$ at instant $k$. Moreover, when the basic set evolves from $I$ to $I_s$ at instant $k$, it must reach a state $x_k$ consistent to the update rule. We define the region that $x_k$ needs to be in as *consistent region*.

**Definition 7** (**Consistent Regions).** Let $I = (\imath^{v_1}, \ldots, \imath^{v_h}) \in \mathbb{I}_k$ be a basic vector at instant $k$, and $I_s = (\imath^{v_1}_s, \ldots, \imath^{v_h}_s) \in \mathrm{succ}(I, k)$ be a successor basic set of $I$. In order to trigger the evolution of the basic set from $I$ to $I_s$ at instant $k$, the system should be in the consistent region $H_k(I, I_s)$, which is defined by

$$H_k(I, I_s) = \bigcap_{i=1}^{h} H_{ik}, \tag{11}$$

where

$$H_{ik} = \begin{cases} \mathcal{X} \setminus \mathcal{H}^{v_i} & \text{if } \imath^{v_i}_s[k] = 0 \\ \mathcal{H}^{v_i} & \text{if } \imath^{v_i}_s[k] = 1 \\ \mathcal{X} & \text{if } k > \mathsf{end}^{v_i} \text{ or } k < \mathsf{int}^{v_i} \end{cases}$$

Finally, in order to ensure that the prefix signal is always feasible, the feasible set $X^I_k$ needs to be able to reach region $X^{I_s}_{k+1}$ in one step. This is formalized by the *one-step feasible set* defined as follows.

**Definition 8** (**One-Step Feasible Set).** Let $\mathcal{S} \subseteq \mathcal{X}$ be a set of states representing the "target region". Then the one-step feasible set of $\mathcal{S}$ is defined by

$$\Upsilon(\mathcal{S}) = \{ x \in \mathcal{X} \mid \exists u \in \mathcal{U} \text{ s.t. } f(x, u) \in \mathcal{S} \}. \tag{12}$$

We define $\mathbb{X}_k = \{ X^I_k \mid I \in \mathbb{I}_k \}$ as the set of all feasible sets for instant $k$. Then our offline objective is to compute all possible $\mathbb{X}_0, \mathbb{X}_1, \ldots, \mathbb{X}_T$, which are used as a look-up

**Algorithm 2:** Offline Computations of Feasible Sets

**Input:** STL formula $\Phi$

**Output:** All potential sets $\{\mathbb{X}_k : k \in [0,T]\}$

1 **for** $k \in [0,T]$ **do**

2 $\quad \mathbb{X}_k \leftarrow \emptyset$

3 **for** $I \in \mathbb{I}_{T+1}$ **do**

4 $\quad X^I_{T+1} \leftarrow \mathbb{R}^n$

5 $k \leftarrow T$

6 **while** $k \geq 0$ **do**

7 $\quad$ **for** $I \in \mathbb{I}_k$ **do**

8 $\quad\quad$ **if** $\imath^{\text{root}}_I = 1$ **then**

9 $\quad\quad\quad X^I_k \leftarrow \mathbb{R}^n$

10 $\quad\quad$ **else**

11 $\quad\quad\quad X^I_k \leftarrow \bigcup_{I_s \in \text{succ}(I,k)} \Big( H_k(I,I_s) \cap \Upsilon(X^{I_s}_{k+1}) \Big)$

12 $\quad\quad \mathbb{X}_k \leftarrow \mathbb{X}_k \cup \{X^I_k\}$

13 $\quad k \leftarrow k-1$

table during the online monitoring process. In terms of our computation of feasible regions, if the system is evolving from $I$ to $I_s$ and maintains the satisfiability of $I_s$ from instant $k+1$, then we know that the system should be in region $H_k(I,I_s) \cap \Upsilon(X^{I_s}_{k+1})$ at instant $k$. However, the basic set $I_s$ for the next instant depends on the current state of the system. Therefore, to compute $X^I_k$, we need to consider all possible successor sets $I_s \in \text{succ}(I,k)$, and take the union of these regions. This is formalized by the following equation

$$X^I_k = \bigcup_{I_s \in \text{succ}(I,k)} \Big( H_k(I,I_s) \cap \Upsilon(X^{I_s}_{k+1}) \Big). \tag{13}$$

Based on the above equation, we compute all feasible sets across the entire horizon using a backward recursion procedure, as formalized in Algorithm 2. The algorithm initializes by setting the terminal feasible set $X^I_{T+1} \leftarrow \mathbb{R}^n$ for all $I \in \mathbb{I}_{T+1}$ (lines 3–4). The recursion proceeds backward in time from $k = T$ to $k = 0$. If the STL formula is already satisfied (i.e., $\imath^{\text{root}}_I = 1$), then set $X^I_k \leftarrow \mathbb{R}^n$. At each time step $k$, the feasible set $X^I_k$ is computed for every basic set $I \in \mathbb{I}_k$ through two operations:

- Intersection with $H_k(I,I_s)$ and $\Upsilon(X^{I_s}_{k+1})$, and
- Union over all successor sets $I_s \in \text{succ}(I,k)$ to ensure completeness.

The resulting sets are aggregated into $\mathbb{X}_k$, which captures all feasible states at time $k$ (lines 5–13).

## VI. Case Studies for Online Monitoring

In this section, we illustrate our online monitoring algorithm with two cases. We implemented the above methods in Python language.[1]

[1] Our codes are available at `https://github.com/sjtu-hantao/MPM4STLnested`, where more details on the computations can be found.

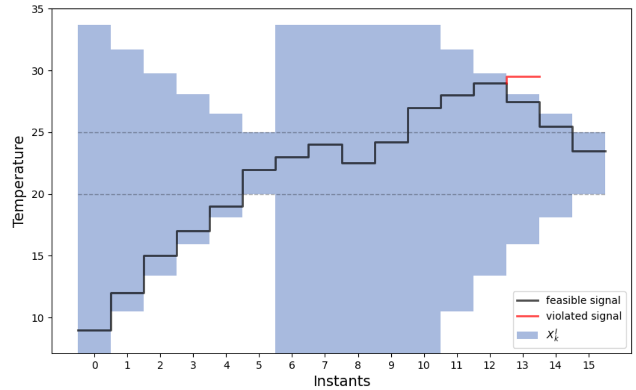


Fig. 2: Two trajectories temperature control system.

### A. Building Temperatures

We consider the problem of monitoring the temperature of a single zone building whose dynamic is [14]

$$x_{k+1} = x_k + \tau_s(\alpha_e(T_e - x_k) + \alpha_H(T_h - x_k)u_k),$$

where $x_k \in \mathcal{X} = [0,45]$ is the zone temperature (°C) at step $k$, $u_k \in \mathcal{U} = [0,1]$ is the heater valve's normalized position, and $\tau_s = 1$ min is the sampling interval. The model parameters include the heater temperature $T_h = 55°\text{C}$, ambient temperature $T_e = 0°\text{C}$, and heat transfer coefficients $\alpha_e = 0.06$ (environmental) and $\alpha_H = 0.08$ (heater).

The temperature control system aims to maintain the zone temperature within the comfortable range of 20°C–25°C at all times, ensuring that any temperature deviation is corrected within 5 minutes during any 10-minute window. This requirement can be specified using the STL formula:

$$\Phi = \mathbf{G}_{[0,10]}(\mathbf{F}_{[0,5]} x_k \in [20,25]).$$

Fig. 2 presents two temperature trajectories that are identical until time step $k = 13$. For the black signal, the condition $x_{13} \in X^I_{13}$ holds, indicating that the control task remains feasible. However, for the red signal, $x_{13} \notin X^I_{13}$, which means the specification will inevitably be violated. That is, no control input exists that could bring the temperature within the required range within the specified time window.

### B. Autonomous Robots

We consider a simple autonomous robot whose dynamic model is given as follows

$$x_{k+1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} \tau_s & 0 \\ 0 & \tau_s \end{bmatrix} u_k,$$

where state $x_k \in \mathcal{X} = [0,12] \times [0,12]$ denotes the position of the robot at instant $k$, control input $u_k \in \mathcal{U} = [-1,1] \times [-1,1]$ is the speed of the robot, and $\tau_s = 1s$ is the sampling time.

The objective of the robot is to patrol both region $A_1$ and region $A_2$ in 6 seconds and stay in $A_2$ for at least 2 seconds. This task can be described by the following STL formula

$$\Phi = \mathbf{F}_{[0,6]} A_1 \wedge \mathbf{F}_{[0,6]}(\mathbf{G}_{[0,2]} A_2),$$

where $A_1 = (x{\in}[3,5]) \wedge (y{\in}[3,5])$ and $A_2 = (x{\in}[6,8]) \wedge (y{\in}[6,8])$.

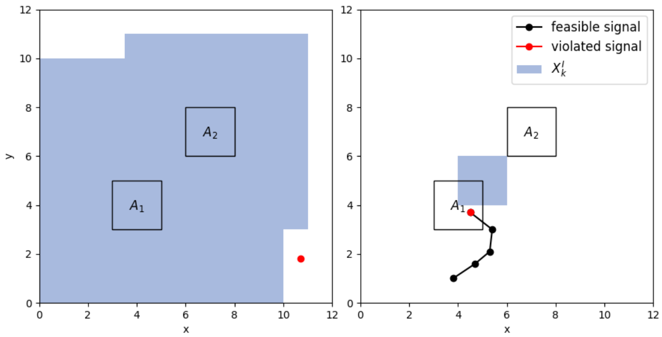


Fig. 3: Two trajectories that violate at different instants.

Let us consider two trajectories of the robot shown in Fig. 3. For the sake of clarity, in each figure, we only draw one feasible set at a certain instant. In the left figure, since $x_0 \notin X_0^I$, the monitor can claim initially that the task cannot be satisfied as it is too far away from the target regions. In the right figure, for $k = 0, 1, 2, 3$, the trajectory stays within the feasible region and the monitor issues "$\texttt{feas}$". Yet, we have $x_4 \notin X_4^I$. Therefore, the monitor issues "$\texttt{vio}$" at $k = 4$.

## VII. Conclusions

In this paper, we present a new model-based online monitoring algorithm for signal temporal logic specifications with nested temporal operators. Our approach utilizes the syntax tree as the fundamental information structure to characterize and dynamically update all relevant information for STL task progression. The online monitoring process is highly efficient as it requires only basic set updates and membership checks. While implemented in an offline manner, the feasible set computation remains challenging for highly nonlinear systems with long time horizons. In the future, we plan to enhance the framework by developing more efficient set representation techniques to optimize the offline computation of feasible sets.

## References


[1] Houssam Abbas and Borzoo Bonakdarpour. Leveraging system dynamics in runtime verification of cyber-physical systems. In *International Symposium on Leveraging Applications of Formal Methods*, pages 264–278. Springer, 2022.

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

[3] Omar Ali Beg, Luan V Nguyen, Taylor T Johnson, and Ali Davoudi. Signal temporal logic-based attack detection in dc microgrids. *IEEE Transactions on Smart Grid*, 10(4):3585–3595, 2018.

[4] Ernest Bonnah and Khaza Anuarul Hoque. Runtime monitoring of time window temporal logic. *IEEE Robotics and Automation Letters*, 7(3):5888–5895, 2022.

[5] Francesca Cairoli, Luca Bortolussi, and Nicola Paoletti. Learning-based approaches to predictive monitoring with conformal statistical guarantees. In *International Conference on Runtime Verification*, pages 461–487. Springer, 2023.

[6] G. Chen, M. Liu, and Z. Kong. Temporal-logic-based semantic fault diagnosis with time-series data from industrial internet of things. *IEEE Transactions on Industrial Electronics*, 68(5):4393–4403, 2020.

[7] T. Claudet, D. Martire, D. Losa, F. Sanfedino, and D. Alazard. A novel graph-based theory for convexification of mission-planning constraints and generative pre-trained trajectory optimization. *IFAC Journal of Systems and Control*, 30:100286, 2024.

[8] Jyotirmoy V Deshmukh, Alexandre Donzé, Shromona Ghosh, Xiaoqing Jin, Garvit Juniwal, and Sanjit A Seshia. Robust online monitoring of signal temporal logic. *Formal Methods in System Design*, 51:5–30, 2017.

[9] Adel Dokhanchi, Bardh Hoxha, and Georgios Fainekos. On-line monitoring for temporal logic robustness. In *International Conference on Runtime Verification*, pages 231–246. Springer, 2014.

[10] Alexandre Donzé, Thomas Ferrere, and Oded Maler. Efficient robust monitoring for stl. In *International conference on computer aided verification*, pages 264–279. Springer, 2013.

[11] Cindy Eisner, Dana Fisman, John Havlicek, Yoad Lustig, Anthony McIsaac, and David Van Campenhout. Reasoning with temporal logic on truncated paths. In *International Conference on Computer Aided Verification*, pages 27–39. Springer, 2003.

[12] Georgios E Fainekos and George J Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.

[13] Qi Heng Ho, Roland B Ilyes, Zachary N Sunberg, and Morteza Lahijanian. Automaton-guided control synthesis for signal temporal logic specifications. In *IEEE Conference on Decision and Control*, pages 3243–3249. IEEE, 2022.

[14] Pushpak Jagtap, Sadegh Soudjani, and Majid Zamani. Formal synthesis of stochastic systems via control barrier certificates. *IEEE Transactions on Automatic Control*, 66(7):3097–3110, 2020.

[15] L. Lindemann, X. Qin, J. V. Deshmukh, and G. J. Pappas. Conformal prediction for stl runtime verification. In *ACM/IEEE International Conference on Cyber-Physical Systems*, pages 142–153, 2023.

[16] M. Ma, E. Bartocci, E. Lifland, J. A. Stankovic, and L. Feng. A novel spatial–temporal specification-based monitoring system for smart cities. *IEEE Internet of Things Journal*, 8(15):11793–11806, 2021.

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

[18] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In Yassine Lakhnech and Sergio Yovine, editors, *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, volume 3253, pages 152–166. Series Title: Lecture Notes in Computer Science.

[19] Anik Momtaz, Niraj Basnet, Houssam Abbas, and Borzoo Bonakdarpour. Predicate monitoring in distributed cyber-physical systems. *International Journal on Software Tools for Technology Transfer*, 25(4):541–556, 2023.

[20] Xin Qin and Jyotirmoy V Deshmukh. Clairvoyant monitoring for signal temporal logic. In *International Conference Formal Modeling and Analysis of Timed Systems*, pages 178–195. Springer, 2020.

[21] Yunus Emre Sahin, Rien Quirynen, and Stefano Di Cairano. Autonomous vehicle decision-making and monitoring based on signal temporal logic and mixed-integer programming. In *2020 American Control Conference (ACC)*, pages 454–459. IEEE, 2020.

[22] Chuwei Wang, Xinyi Yu, Jianing Zhao, Lars Lindemann, and Xiang Yin. Sleep when everything looks fine: Self-triggered monitoring for signal temporal logic tasks. *IEEE Robotics and Automation Letters*, 2024.

[23] Ruixuan Yan and Agung Julius. Distributed consensus-based online monitoring of robot swarms with temporal logic specifications. *IEEE Robotics and Automation Letters*, 7(4):9413–9420, 2022.

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

[25] P. Yu, X. Tan, and D.V. Dimarogonas. Continuous-time control synthesis under nested signal temporal logic specifications. *IEEE Transactions on Robotics*, 40:2272–2286, 2024.

[26] Xinyi Yu, Weijie Dong, Shaoyuan Li, and Xiang Yin. Model predictive monitoring of dynamical systems for signal temporal logic specifications. *Automatica*, 160:111445, 2024.

[27] Z. Zhang, J. An, P. Arcaini, and I. Hasuo. Online causation monitoring of signal temporal logic. In *International Conference on Computer Aided Verification*, pages 62–84. Springer, 2023.

[28] Yiqi Zhao, Bardh Hoxha, Georgios Fainekos, Jyotirmoy V Deshmukh, and Lars Lindemann. Robust conformal prediction for stl runtime verification under distribution shift. In *ACM/IEEE International Conference on Cyber-Physical Systems*, pages 169–179. IEEE, 2024.

Let $v \in V$ be a node in tree $\mathcal{T}$. The *sub-tree* induced from node $v$ is a new rooted tree $\mathcal{T}^v = (V', L, E', v'_{\text{root}})$, where $V' = \{v' \in V \mid v' \preceq v\}$ is the set of descendants of $v$ including itself, $E' \subseteq V' \times L \times V'$ is the restriction of $E$ onto $V'$, and $v'_{\text{root}} = v$ is the root of the sub-tree.

Let $\Phi$ be an STL formula and $\mathcal{T}_\Phi$ be its syntax tree. The induced sub-tree from any node $v \in V_\Phi$ is also a syntax tree We use $\mathcal{T}^v$ to denote the subtree with $v$ as the root node. The subtree $\mathcal{T}^v$ represents a subformula $\Phi^v$ of the original STL formula $\Phi$.

To prove Proposition 1 and Proposition 2, we prove Proposition 3 at first. Then we use Proposition 3 to prove Proposition 2 and prove Proposition 1 at last.

**Proposition 3.** For any state sequence $\mathbf{x}_{0:k}$ and subformula $\Phi^v$ we have

$$(\mathbf{x}_{0:k}, t) \models \Phi^v \quad \Leftrightarrow \quad \imath^v_{I(\mathbf{x}_{0:k})}[t] = 1.$$

*Proof:* Prove this proposition recursively by mathematical induction. For the case of leaf node $\Phi^v = x \in \mathcal{H}$:
$\Rightarrow$: When $(\mathbf{x}_{0:k}, t) \models \Phi^v$, there must be $x_t \in \mathcal{H}$. According to (6), $\imath^v_{I(\mathbf{x}_{0:k})}[t] = 1$.
$\Leftarrow$: When $\imath^v_{I(\mathbf{x}_{0:k})}[t] = 1$, which also means $x_t \in \mathcal{H}$. So $(\mathbf{x}_{0:k}, t) \models \Phi^v$.
The proposition holds for leaf nodes. Assuming it always holds for child nodes, now prove the case of other nodes.

1) $\Phi^v = \Phi^{v_1^c} \wedge \Phi^{v_2^c} \wedge \cdots \wedge \Phi^{v_n^c}$:
   $\Rightarrow$: When $(\mathbf{x}_{0:k}, t) \models \Phi^v$, there must be $(\mathbf{x}_{0:k}, t) \models \Phi^{v_1^c}, (\mathbf{x}_{0:k}, t) \models \Phi^{v_2^c}, \ldots, (\mathbf{x}_{0:k}, t) \models \Phi^{v_n^c}$. So $\forall v_i^c \in \mathsf{child}(v)$, $\imath^{v_i^c}_{I(\mathbf{x}_{0:k})}[t] = 1$. Then $\imath^v_{I(\mathbf{x}_{0:k})}[t] = 1$ according to Definiton 5.
   $\Leftarrow$: Reverse the proof of $\Rightarrow$.
2) $\Phi^v = \mathbf{G}_{[a,b]}\Phi^{v^c}$:
   $\Rightarrow$: When $(\mathbf{x}_{0:k}, t) \models \Phi^v$, there must be $\forall t' \in [t+a, t+b], (\mathbf{x}_{0:k}, t') \models \Phi^{v^c}$. So $\forall t' \in [t+a, t+b], \imath^{v^c}_{I(\mathbf{x}_{0:k})}[t'] = 1$. So $\forall v_i^c \in \mathsf{child}(v)$, $\imath^{v_i^c}_{I(\mathbf{x}_{0:k})}[t] = 1$. Then $\imath^v_{I(\mathbf{x}_{0:k})}[t] = 1$ according to Definiton 5.
   $\Leftarrow$: Reverse the proof of $\Rightarrow$.
3) $\Phi^v = \Phi^{v^{lc}}\mathbf{U}'_{[a,b]}\Phi^{v^{rc}}$:
   $\Rightarrow$: When $(\mathbf{x}_{0:k}, t) \models \Phi^v$, there must be $\exists t' \in [t+a, t+b], (\mathbf{x}_{0:k}, t') \models \Phi^{v^{rc}}$ i.e. $\imath^{v^{rc}}_{I(\mathbf{x}_{0:k})}[t'] = 1$ and $\forall t'' \in [t+a, t']$ such that $(\mathbf{x}_{0:k}, t'') \models \Phi^{v^{lc}}$ i.e. $\imath^{v^{lc}}_{I(\mathbf{x}_{0:k})}[t''] = 1$. So $\imath^v_{I(\mathbf{x}_{0:k})}[t] = 1$ according to Definiton 5.
   $\Leftarrow$: Reverse the proof of $\Rightarrow$.

According to the above, we prove $(\mathbf{x}_{0:k}, t) \models \Phi^v \Leftrightarrow \imath^v_{I(\mathbf{x}_{0:k})}[t] = 1$.

**Proposition 2.** For any state sequence $\mathbf{x}_{0:k}$, we have

$$\mathbf{x}_{0:k} \models \Phi \quad \Leftrightarrow \quad \imath^{\text{root}}_{I(\mathbf{x}_{0:k})}[0] = 1.$$

*Proof:* As a special case of Proposition 3, when $v = v_{\text{root}}$ and $t = 0$, $\mathbf{x}_{0:k} \models \Phi \Leftrightarrow \imath^{\text{root}}_{I(\mathbf{x}_{0:k})}[0] = 1$, which means the root vector fully captures the satisfaction status of the entire STL formula.

**Proposition 1.** Let $I$ be a basic set and $k$ be a time instant. For any two sequences $\mathbf{x}'_{0:k-1}, \mathbf{x}''_{0:k-1} \in \mathbf{x}^I_{0:k-1}$ consistent with $I$, and any future sequence $\mathbf{x}_{k:T} = x_k x_{k+1} \cdots x_T$, we have

$$\mathbf{x}'_{0:k-1}\mathbf{x}_{k:T} \models \Phi \quad \Leftrightarrow \quad \mathbf{x}''_{0:k-1}\mathbf{x}_{k:T} \models \Phi.$$

*Proof:* According to Proposition 2, we know that $\mathbf{x}'_{0:k-1}\mathbf{x}_{k:T} \models \Phi \Leftrightarrow \imath^{v_{\text{root}}}_{I(\mathbf{x}'_{0:k-1}\mathbf{x}_{k:T})}[0] = 1$. For $\mathbf{x}'_{0:k-1}, \mathbf{x}''_{0:k-1} \in \mathbf{x}^I_{0:k-1}$, we have $I(\mathbf{x}'_{0:k-1}\mathbf{x}_{k:T}) = I(\mathbf{x}''_{0:k-1}\mathbf{x}_{k:T})$. So there must be $\imath^{v_{\text{root}}}_{I(\mathbf{x}'_{0:k-1}\mathbf{x}_{k:T})}[0] = \imath^{v_{\text{root}}}_{I(\mathbf{x}''_{0:k-1}\mathbf{x}_{k:T})}[0] = 1$. Then due to Proposition 2, we have $\mathbf{x}''_{0:k-1}\mathbf{x}_{k:T} \models \Phi$. The proof above also holds in reverse. So there is $\mathbf{x}'_{0:k-1}\mathbf{x}_{k:T} \models \Phi \Leftrightarrow \mathbf{x}''_{0:k-1}\mathbf{x}_{k:T} \models \Phi$.