Model predictive monitoring of dynamical systems for signal temporal logic specifications☆

Xinyi Yu, Weijie Dong, Shaoyuan Li *, Xiang Yin *

Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China
Key Laboratory of System Control and Information Processing, Ministry of Education of China, Shanghai 200240, China

ARTICLE INFO

ABSTRACT

Online monitoring aims to evaluate or to predict, at runtime, whether or not the behaviors of a system satisfy some desired specification. It plays a key role in safety-critical cyber–physical systems. In this work, we propose a new monitoring approach, called *model predictive monitoring*, for specifications described by Signal Temporal Logic (STL) formulae. Specifically, we assume that the observed state traces are generated by an underlying dynamical system whose model is known but the control law is unknown. The main idea is to use the dynamic of the system to predict future states when evaluating the satisfaction of the STL formulae. To this end, effective approaches for the computation of feasible sets of STL formulae are provided. We show that, by explicitly utilizing the model information of the dynamical system, the proposed online monitoring algorithm can falsify or certify of the specification in advance compared with existing algorithms, where no model information is used. We also demonstrate the proposed monitoring algorithm by several real world case studies.

1. Introduction

1.1. Background

Cyber–Physical Systems (CPS) are man-made modern engineering systems involving both computational devices and physical dynamics. Safety is one of the major considerations in the designs of many CPS such as intelligent transportation systems, smart manufacturing systems and medical devices. For those safety-critical systems, it is crucial to determine whether or not the behaviors of the system satisfy some desired high-level specifications. For example, once we detect that the system has violated or will inevitably violate some desired specifications, additional corrective actions can be taken to ensure safety.

Specification-based monitoring is one of the major techniques in evaluating behavior correctness of CPS (Bartocci et al., 2018). It is well-known as a light-weight alternative to formal verification and is particularly applicable to block-box systems with unknown inputs for which formal verification cannot be applied. In this context, it is usually assumed that the desired behavior of the system is described by a specification formula and the state traces (a.k.a. signals) generated by the system are observed by a *monitor* that can issue alarms when the specification is violated. In the past years, numerous algorithms have been developed for monitoring specifications described by, e.g., Linear Temporal Logic (LTL) (Eisner et al., 2003), Metric Temporal Logic (MTL) (Dokhanchi, Hoxha, & Fainekos, 2014; Thati & Roşu, 2005) and Signal Temporal Logic (STL) (Deshmukh et al., 2017; Donzé, Ferrere, & Maler, 2013). Recent applications of specification-based monitoring techniques include, e.g., robot systems (Bonnah & Hoque, 2022), autonomous vehicles (Sahin, Quirynen, & Di Cairano, 2020), fuel control system (Jakšić, Bartocci, Grosu, Nguyen, & Ničković, 2018), smart cities (Ma, Bartocci, Lifland, Stankovic and Feng, 2021), Internet of Things (Zhao, Zhou, Cai, Yangui, & Xue, 2022) and intelligent medicines (Qin & Deshmukh, 2020).

Depending on what information can be utilized by the monitor, the monitoring problem can be categorized as *offline* (e.g., Donzé et al., 2013; Donzé & Maler, 2010; Fainekos & Pappas, 2009) and *online* (e.g., Deshmukh et al., 2017; Dokhanchi et al., 2014; Ho, Ouaknine, & Worrell, 2014). In offline monitoring, it is assumed that the complete signal to evaluate has already been generated and the monitor needs to determine either the Boolean satisfaction or the quantitative satisfaction degree of the complete signal. Such offline technique is usually used in the design phase to evaluate the simulated traces of the system prototype.

On the other hand, when the CPS is operating online, the monitor only observes *partial* state traces that have been generated so far. Therefore, *online monitoring* focuses on evaluating signals in real time during the operation of the system in order to, e.g., issue alarms or trigger corrective actions.

In the context of qualitative online monitoring, monitor may make the following evaluations on the observed partial signals: (i) the specification cannot be satisfied, i.e., there is no future possibility to correct the signal; (ii) the specification has already been satisfied, i.e., the future signal does not matter anymore; or (iii) inconclusive, i.e., the signal can be either satisfied or not depending on what will happen in the future. Furthermore, in the quantitative setting, the monitor may also estimate the possible robustness interval based on the observed partial signals. In the past years, numerous algorithms have been developed for online monitoring for specifications described by temporal logic formulae. For example, the basic setting is to consider monitoring the Boolean satisfaction of LTL formulae (Abate, Feron, & Coogan, 2019; Bauer, Leucker, & Schallhart, 2011; Mascle et al., 2020) or MTL formulae (Ho et al., 2014). In Deshmukh et al. (2017) and Dokhanchi et al. (2014), algorithms have been developed for quantitatively monitoring the satisfaction of specifications by using robust semantics of STL formulae.

Most of the aforementioned online monitoring techniques are *model-free* in the sense that the satisfaction of the specification is only evaluated based on the observed signal without considering the dynamic of the system or without predicting future states. In some cases, however, the model of the underlying system, can provide additional information to accelerate the monitoring process. Let us consider a scenario, where for an observed signal, a model-free monitor may provide inconclusive evaluation since the partial signal can be extended to either satisfiable or unsatisfiable signals. However, those satisfiable continuations may not be feasible physically in the dynamical system. In this scenario, by leveraging the model information of the dynamical system and predicting future states, the monitor can better assert that the specification cannot be satisfied before it is actually violated. We refer such type of online monitoring process to as the *model predictive monitoring*, which uses model information to predict future states so that the specifications can be better evaluated.

1.2. Our contributions

In this paper, we propose a new model predictive monitoring approach of dynamical systems for STL. STL formulae are interpreted over continuous time signals and have the advantage of quantitatively evaluating the degree of the satisfaction or violation using robust semantics (Gilpin, Kurtz, & Lin, 2020; Hashimoto, Hashimoto, & Takai, 2022; Lindemann & Dimarogonas, 2018, 2019; Maler & Nickovic, 2004; Salamati, Soudjani, & Zamani, 2021). The monitor aims to issue alarms when the specification has already or will inevitably be violated. However, different from existing approaches, here we explicitly consider the model information of underlying dynamical system. Specifically, we consider a discrete-time nonlinear system. In order to incorporate the model information into the evaluation of STL formulae, we propose the notion of *feasible sets*, which are the regions of states from which STL formulae can potentially be satisfied considering the system dynamic. Effective algorithms in a dynamic programming manner have been developed for computing feasible sets offline. To monitor the specification in real-time, we propose online monitoring algorithms that correctly combine both the online observed partial signals and the offline computed feasible sets. We show that the proposed model predictive monitoring algorithm may predict the violation of the specification in advance compared with existing model-free approaches. Hence, it may leave more time for the system to take corrective actions to ensure safety.

1.3. Related works

Model-Based Monitoring for State-Discrete Systems: There are already many works along the line of leveraging model information for the purpose of online monitoring in the state-discrete domain during the last decade. For example, Leucker (2012) and Zhang, Leucker, and Dong (2012) introduce predictive semantics for monitoring of untimed LTL specifications for systems that are not black boxes. In Pinisetty et al. (2017), the authors proposed a predictive runtime verification framework for systems with timing requirements. Recently, Ferrando et al. (2022) extend the single-model predictive monitoring approach to the multi-model case for both centralized and compositional settings. In Yoon and Sankaranarayanan (2021), the authors introduce a Bayesian intent inference framework leveraging the robot's intent information to predict future positions.

Model-Based Monitoring for State-Continuous Systems: However, there are only a few existing works along this line in the state-continuous domain these years. For example, Ghosh and André (2022) and Waga, André, and Hasuo (2021) use prior knowledge about the over-approximations of target systems represented by linear hybrid automata and linear dynamical systems, respectively, to tackle the problem of scattered sampling uncertainties during the monitoring process. In Momtaz, Basnet, Abbas, and Bonakdarpour (2021), the authors adopt a lightweight mechanism for incorporating bounds on system dynamics to reduce monitoring overhead, and Abbas and Bonakdarpour (2022) exploit the rough knowledge of dynamics for STL specifications. For data-driven STL predictive monitoring, Qin and Deshmukh (2020) propose to use statistical time-series analysis techniques to predicate future states; Ma, Stankovic, Bartocci and Feng (2021) use Bayesian recurrent neural networks learned from data to predict future states with uncertainties; Lindemann, Qin, Deshmukh, and Pappas (2023) apply conformal prediction to provide probability guarantees. However, these approaches either use some rough information of the model or consider a purely unknown system dynamic.

Model Checking for STL Specifications: Our pre-computation of feasible sets is closely related to the model checking problem for STL specifications (Bae & Lee, 2019; Lee, Yu, & Bae, 2021; Roehm, Oehlerking, Heinz, & Althoff, 2016; Yu, Lee and Bae, 2022). Specifically, in the model checking problem, one is given a system model with an initial state and the objective is to determine whether or not all possible traces generated by the system model from the initial state satisfy the STL formula. Strategies for model checking usually involve computations of reachable sets, which are related to our approach. However, our problem essentially requires to compute the satisfaction region of an STL formula taking a given prefix trace into account. This particular requirement cannot be handled by existing model checking strategies due to the lack of automata representation of STL formulae. Finally, we note that the satisfaction of an STL formula can also be characterized by control barrier functions (CBFs); see, e.g., Buyukkocak, Aksaray, and Yazıcıoğlu (2022) and Lindemann and Dimarogonas (2018). However, CBF only provides a sufficient condition for the satisfaction of the task, and it may be overly conservative for the purpose of online monitoring.

The preliminary version of some results in this paper is presented in Yu, Dong, Yin and Li (2022). Compared with Yu, Dong et al. (2022), the present work has the following differences. The major difference is that Yu, Dong et al. (2022) only consider a restrictive fragment of STL formulae in which no overlap of time horizons is allowed for temporal operators. In contrast, this work handles a more general case that supports arbitrarily overlapped temporal operators. Due to this more general setting, our approaches for both the online monitoring algorithm

and the offline pre-computation of feasible sets are also very different. Furthermore, our generalized approach also supports the monitoring problem for the satisfaction of the STL task for all control input sequences, while Yu, Dong et al. (2022) can only handle the existential case. Finally, the present work provides four detailed case studies of real-world systems to illustrate the proposed algorithm, which are not provided in Yu, Dong et al. (2022).

1.4. Organization

The rest of the paper is organized as follows. We present some basic preliminaries in Section 2 and formulate the problem in Section 3. Section 4 presents the main body of the online monitoring algorithm, which uses feasible sets that are computed offline in Section 5. Section 6 is the extension of monitoring satisfaction of STL formula. The overall framework is demonstrated by several real world case studies in Section 7 and finally, we conclude this work in Section 8.

2. Preliminary

2.1. System model

We consider a discrete-time control system of form

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

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

Suppose that the system is in state $x_k \in \mathcal{X}$ at time instant $k \in \mathbb{Z}_{\geq 0}$. Then given a sequence of control inputs $\mathbf{u}_{k:T-1} = u_k u_{k+1} \ldots u_{T-1} \in \mathcal{U}^{T-k}$, the solution of the system is a sequence of states $\xi_f(x_k, \mathbf{u}_{k:T-1}) = \mathbf{x}_{k+1:T} = x_{k+1} \ldots x_T \in \mathcal{X}^{T-k}$ such that $x_{i+1} = f(x_i, u_i), i = k, \ldots, T-1$.

2.2. Signal temporal logic

We use Signal Temporal Logic (STL) formulae with bounded-time temporal operators (Maler & Nickovic, 2004) to describe whether or not the trajectory of the system satisfies some desired high-level properties. Formally, the syntax of STL formulae is as follows

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

where $\top$ is the true predicate, π^μ is an atomic predicate whose truth value is determined by the sign of its underlying predicate function $\mu : \mathbb{R}^n \to \mathbb{R}$ and it is true at state x_k when $\mu(x_k) \geq 0$; otherwise it is false. Notations $\neg$ and $\wedge$ are the standard Boolean operators "negation" and "conjunction", respectively, which can further induce "disjunction" by $\Phi_1 \vee \Phi_2 := \neg(\neg\Phi_1 \wedge \neg\Phi_2)$ and "implication" by $\Phi_1 \to \Phi_2 := \neg\Phi_1 \vee \Phi_2$. $\mathbf{U}_{[a,b]}$ is the temporal operator "*until*", where $a, b \in \mathbb{Z}_{\geq 0}$ are two integer instants with $a \leq b$. As we consider a discrete-time setting, we use $[a, b]$ as a shorthand notation for the discrete-time interval $[a, b] \cap \mathbb{Z}$ which is the set of all integers between a and b including a and b; this set is non-empty when $a, b \in \mathbb{Z}_{\geq 0}$ and $a \leq b$. Also we note that, since we consider a discrete-time setting, time instant $k \in \mathbb{Z}_{\geq 0}$ does not necessarily represent the real time; the specific real time also depends on the sampling rate when discretizing the system. STL formulae are evaluated on state sequence $\mathbf{x} = x_0 x_1 \cdots$. We use notation $(\mathbf{x}, k) \models \Phi$ to denote that sequence $\mathbf{x}$ satisfies STL formula Φ at time instant k. The reader is referred to Maler and Nickovic (2004) for more details on the semantics of STL

formulae. Particularly, 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 $(\mathbf{x}, k) \models \Phi_1 \mathbf{U}_{[a,b]} \Phi_2$ iff $\exists k' \in [k+a, k+b]$ such that $(\mathbf{x}, k') \models \Phi_2$ and $\forall k'' \in [k, k']$, we have $(\mathbf{x}, k'') \models \Phi_1$, i.e., Φ_2 will hold at some instant between $[k+a, k+b]$ in the future and before that Φ_1 always holds. 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 $k' \in [k+a, k+b]$; and
- "*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 $(\mathbf{x}, 0) \models \Phi$.

Given an STL formula Φ, in fact, it is well-known that the satisfaction of Φ can be completely determined only by those states within its *horizon*. Specifically, we will use notation $\Phi^{[S,T]}$ to emphasize that the satisfaction of formula Φ only depends on time horizon $[S, T]$, where S is the starting instant of Φ which is the minimum time instant that appears in the formula and T is the terminal instant of Φ which is the maximum sum of all nested upper bounds. For example, for $\Phi = \mathbf{F}_{[2,7]} \pi^{\mu_1} \wedge \mathbf{G}_{[3,12]} \pi^{\mu_2}$, we have $T = \max\{7, 12\} = 12$ and $S = \min\{2, 3\} = 2$.

3. Problem formulation

3.1. Fragment of STL formulae

In this paper, we consider the following restricted but still expressive enough fragments of STL formulae:

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

$$\Phi ::= \mathbf{F}_{[a,b]} \varphi \mid \mathbf{G}_{[a,b]} \varphi \mid \varphi_1 \mathbf{U}_{[a,b]} \varphi_2 \mid \Phi_1 \wedge \Phi_2, \tag{2b}$$

where φ_1, φ_2 are formulae of class φ, and Φ_1, Φ_2 are formulae of class Φ. Specifically, we only allow the temporal operators to be applied once for Boolean formulae.

Note that, for the standard "until" operator, $\varphi_1 \mathbf{U}_{[a,b]} \varphi_2$ requires that φ_1 holds *from the initial instant* before φ_2 holds. In order to facilitate subsequent expression, we introduce a new temporal operator $\mathbf{U}'$ defined by $(\mathbf{x}, k) \models \Phi_1 \mathbf{U}'_{[a,b]} \Phi_2$ iff $\exists k' \in [k+a, k+b]$ such that $(\mathbf{x}, k') \models \Phi_2$ and $\forall k'' \in [k+a, k']$, we have $(\mathbf{x}, k'') \models \Phi_1$. Compared with $\mathbf{U}$, the new operator $\mathbf{U}'$ only required that Φ_1 holds *from instant a* before Φ_2 holds. Throughout this paper, we will refer "$\mathbf{U}'$" to as the "until" operator. As illustrated by Fig. 1, our setting is without loss of generality since we can express the standard $\mathbf{U}$ using $\mathbf{U}'$ by:

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

Intuitively, the effective horizon of $\mathbf{U}_{[a,b]}$ is $[0, b]$, while the effective horizon of $\mathbf{U}'_{[a,b]}$ is $[a, b]$. This separation is mainly technical and for the sake of efficiency. As we will show later, handling "until" is more complicated than handling "always", and therefore, it is beneficial to have "until" with short effective horizon.

Furthermore, we can always rewrite Boolean formula φ in Eq. (2a) in terms of the region of states satisfying the formula. Specifically, for predicate π^μ, its satisfaction region is the solution of inequality $\mu(x) \geq 0$; we denote it by set $\mathcal{H}^\mu$, i.e., $\mathcal{H}^\mu = \{x \in \mathcal{X} \mid \mu(x) \geq 0\}$. Similarly, we have $\mathcal{H}^{\neg\varphi} = \mathcal{X} \setminus \mathcal{H}^\varphi$ and $\mathcal{H}^{\varphi_1 \wedge \varphi_2} = \mathcal{H}^{\varphi_1} \cap \mathcal{H}^{\varphi_2}$. Hereafter, instead of using φ, we will only write it as $x \in \mathcal{H}^\varphi$ or simply $x \in \mathcal{H}$ using its satisfaction region.

Based on the above discussion, STL formulae Φ in Eq. (2) can be expressed equivalently by:

$$\Phi ::= \mathbf{F}_{[a,b]} x \in \mathcal{H} \mid \mathbf{G}_{[a,b]} x \in \mathcal{H} \mid x \in \mathcal{H}^1 \mathbf{U}'_{[a,b]} x \in \mathcal{H}^2 \mid \Phi_1 \wedge \Phi_2, \tag{3}$$

where $\mathcal{H} \subseteq \mathbb{R}^n$ is a set of states representing the satisfaction region of a Boolean formula.
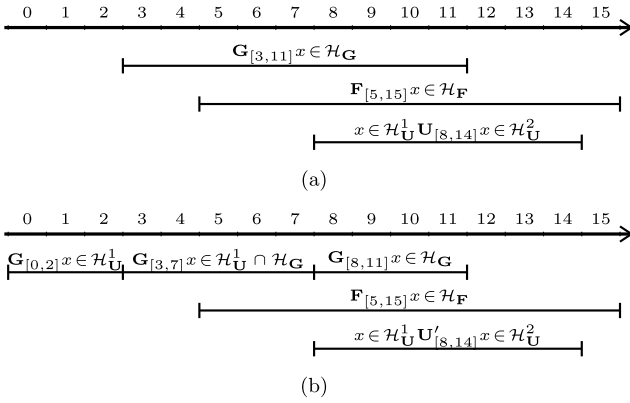
Fig. 1. Illustration of formulae equivalence in Example 1.

In summary, we consider STL formulae of form

$$\Phi = \bigwedge_{i=1}^{N} \Phi_i^{[a_i, b_i]}, \tag{4}$$

where $\Phi_i^{[a_i, b_i]}$ is a sub-formula that applies within time interval $[a_i, b_i]$ in the form of $\mathbf{G}_{[a_i, b_i]} x \in \mathcal{H}_i$, $\mathbf{F}_{[a_i, b_i]} x \in \mathcal{H}_i$ or $x \in \mathcal{H}_i^1 \mathbf{U}_{[a_i, b_i]}' x \in \mathcal{H}_i^2$, and N denotes the total number of sub-formulae.

We denote by $\mathcal{I} = \{1, \dots, N\}$ the index set of all sub-formulae. We assume that the indices are ordered according to the starting instants of the sub-formulae, i.e., $a_1 \leq a_2 \leq \cdots \leq a_N$. For each sub-formula $i \in \mathcal{I}$, we denote by $O_i \in \{\mathbf{G}, \mathbf{F}, \mathbf{U}'\}$ the unique temporal operator in Φ_i. Note that, for each time instant k, there may have multiple sub-formulae applied, and we denote by $\mathcal{I}_k = \{i \in \mathcal{I} \mid a_i \leq k \leq b_i\}$ the index set of sub-formulae that are effective at instant k. Similarly, we denote by $\mathcal{I}_{<k} = \{i \in \mathcal{I} \mid b_i < k\}$ and $\mathcal{I}_{>k} = \{i \in \mathcal{I} \mid k < a_i\}$ the index sets of sub-formulae that are effective strictly before and after instant k, respectively.

Example 1. Let us consider the following STL formula of the original form (2) with horizon $T = 15$

$$\Phi = \mathbf{G}_{[3,11]} x \in \mathcal{H}_{\mathbf{G}} \wedge \mathbf{F}_{[5,15]} x \in \mathcal{H}_{\mathbf{F}} \wedge x \in \mathcal{H}_{\mathbf{U}}^1 \mathbf{U}_{[8,14]} x \in \mathcal{H}_{\mathbf{U}}^2. \tag{5}$$

The effective horizon of each sub-formula is shown in Fig. 1(a). Equivalently, this formula can also be written in the form of (3) as

$$\Phi = (\mathbf{G}_{[0,2]} x \in \mathcal{H}_{\mathbf{U}}^1) \wedge (\mathbf{G}_{[3,7]} x \in \mathcal{H}_{\mathbf{U}}^1 \cap \mathcal{H}_{\mathbf{G}}) \wedge \tag{6}$$

$$(\mathbf{F}_{[5,15]} x \in \mathcal{H}_{\mathbf{F}}) \wedge (\mathbf{G}_{[8,11]} x \in \mathcal{H}_{\mathbf{G}}) \wedge (x \in \mathcal{H}_{\mathbf{U}}^1 \mathbf{U}_{[8,14]}' x \in \mathcal{H}_{\mathbf{U}}^2)$$

which is shown in Fig. 1(b). In this case, we have $\mathcal{I} = \{1, 2, 3, 4, 5\}$ and $O_1, O_2, O_4 = \mathbf{G}$, $O_3 = \mathbf{F}$ and $O_5 = \mathbf{U}'$. Also, for time instant $k = 9$, we have $\mathcal{I}_k = \{3, 4, 5\}$.

3.2. Online monitoring of STL

Given a state sequence $\mathbf{x}$, whose length is equal to or longer than the horizon of Φ, we can always completely determine whether or not $\mathbf{x} \models \Phi$. However, during the operation of the system, at each time k, we can observe the current state x_k, and therefore, only the partial signal $\mathbf{x}_{0:k} = x_0 x_1 \cdots x_k$ (called *prefix*) is available at time instant k, and the remaining signals $\mathbf{x}_{k+1:T}$ (called *suffix*) will only be available in the future. We say a prefix signal $\mathbf{x}_{0:k}$ is

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

Intuitively, a prefix signal is violated if we know for sure in advance that the formula will be violated inevitably. For example, for safety specification $\mathbf{G}_{[0,T]} x \in \mathcal{H}$, once the system reaches a state $x_k \notin \mathcal{H}$ for $k < T$, we know immediately that the formula is violated. Also, if the system is in state x_k from which no solution $\xi_f(x_k, \mathbf{u}_{k:T-1})$ can be found such that each state is in region $\mathcal{H}$, then we can also claim the formula cannot be satisfied anymore, i.e., it is violated.

Therefore, an online *monitor* is a function

$$\mathcal{M} : \mathcal{X}^* \rightarrow \{0, 1\}$$

that determines the satisfaction of formula based on the partial signal, where $\mathcal{X}^*$ denotes the set of all finite sequences over $\mathcal{X}$, "0" denotes "feasible" and "1" denotes "violated". Then the online monitoring problem is formulated as follows.

Problem 1. Given a dynamical system of form (1) and an STL formula Φ as in (4), design an online monitor $\mathcal{M} : \mathcal{X}^* \rightarrow \{0, 1\}$ such that for any prefix signal $\mathbf{x}_{0:k}$ where $k \leq T$, we have $\mathcal{M}(\mathbf{x}_{0:k}) = 1$ iff $\mathbf{x}_{0:k}$ is a violated prefix.

Remark 1. We note that, for any prefix signal $\mathbf{x}_{0:k}$, it is a violated prefix iff we cannot find a sequence of control inputs $\mathbf{u}_{k:T-1}$ such that $\mathbf{x}_{0:k} \xi_f(x_k, \mathbf{u}_{k:T-1}) \models \Phi$. The existence of such a control sequence can be determined by the binary encoding technique proposed in Raman et al. (2014). Therefore, a naive approach for designing an online monitor is to solve the above constrained satisfaction problem based on $\mathbf{x}_{0:k}$. However, such a direct approach has the following issues

- First, the computations are performed purely online by solving a satisfaction problem, which is computationally very challenging especially for nonlinear systems with long horizon STL formulae. Hence, the monitor may not be able to provide evaluations in time.
- Second, this requires to store the entire state sequence up to now. It is more desirable if the monitor can just store the satisfaction status of the formula by "forgetting" those irrelevant information.

Compared with the direct approach, in this paper, we will present an alternative approach by *pre-computing* the set of feasible regions in an offline fashion. Then the pre-computed information will be used online, which ensures timely online evaluations.

Remark 2. Problem 1 is formulated by assuming that the control inputs are unknown. There are several motivations for this problem setting. Note that, an online monitor is usually implemented as an *additional component* at the top of a control system, which can be either semi-autonomous in the sense that it is controlled by an experienced human-operator or a black-box controller in the sense that the control law is claimed to be "correct" but cannot be formally verified. For both cases, we do not know precisely the control inputs of the system. For the purpose of safety, one still wants to add a monitor at the top of the system. When the control law is known, the system dynamic is already autonomous. For this case, however, the unknown control inputs here can be considered as disturbances from environments. Therefore, our problem formulation can also be applied to the case of known control law under unknown disturbances.

4. Set-based online monitoring

4.1. Remaining formulae and feasible set

As we mentioned above, our objective is to evaluate the satisfaction of STL formulae of the form (4) which is the conjunction of several sub-formulae. Specifically, at each instant k, the monitor needs to determine the following two issues:

- for sub-formulae effective currently, check whether or not each of them has been achieved; and
- for those sub-formulae (either effective currently or in the future) that have not been achieved, check whether or not the system is still able to fulfill them in the future.

To formalize the above issues, we use $I \subseteq \mathcal{I}$ to denote the index set of the remaining sub-formulae (which will be referred to as the *remaining set* latter), i.e., sub-formulae that have not been achieved yet. Then we introduce the notion of *I-remaining formulae* as follows.

Definition 1 (*I-Remaining Formula*). Given an STL formula Φ of form (4), a subset of indices $I \subseteq \mathcal{I}$ and a time instant $k \in [0, T]$, I-remaining formula at instant k is defined by

$$\hat{\Phi}_k^I = \bigwedge_{i \in I \cap \mathcal{I}_k} \Phi_i^{[k,b_i]} \wedge \bigwedge_{i \in I \cap \mathcal{I}_{>k}} \Phi_i^{[a_i,b_i]}, \tag{7}$$

where $\Phi_i^{[k,b_i]}$ is obtained from $\Phi_i^{[a_i,b_i]}$ by replacing the start instant of the temporal operator from a_i to k.

Intuitively, $\hat{\Phi}_k^I$ denotes the conjunction of all sub-formulae that have not been achieved. Clearly, sub-formulae with index in $\mathcal{I}_{>k}$ effective in the future are naturally not achieved. For sub-formulae with index in $\mathcal{I}_k$ effective currently, we only consider those in I. Furthermore, since we are only interested in the satisfiability in the future, the formulae are truncated from the given instant k.

Example 2 (*Cont.*). Let us consider the STL formula Φ in Eq. (6). For time instant $k = 7$, we have $\mathcal{I}_{<7} = \{1\}, \mathcal{I}_7 = \{2, 3\}$ and $\mathcal{I}_{>7} = \{4, 5\}$. Now, suppose that the remaining index set at $k = 7$ is $I = \{3, 4, 5\}$, which is determined by the sequence $\mathbf{x}_{0:6}$. Then I-remaining formula at $k = 7$ is $\hat{\Phi}_7^I = (\mathbf{F}_{[7,15]}x \in \mathcal{H}_\mathbf{F}) \wedge (\mathbf{G}_{[8,11]}x \in \mathcal{H}_\mathbf{G}) \wedge (x \in \mathcal{H}_\mathbf{U}^1\mathbf{U}'_{[8,14]}x \in \mathcal{H}_\mathbf{U}^2)$.

In order to capture whether or not the I-remaining formulae can possibly be fulfilled in the future under the constraint of the system dynamic, we introduce the notion of *I-remaining feasible set*.

Definition 2 (*I-Remaining Feasible Set*). Given an STL formula Φ of form (4), a subset of indices $I \subseteq \mathcal{I}$ and a time instant $k \in [0, T]$, the I-remaining feasible set at instant k, denoted by $X_k^I \subseteq \mathcal{X}$, is the set of states from which there exists a solution that satisfies the I-remaining formula at k, i.e.,

$$X_k^I = \left\{ x_k \in \mathcal{X} \,\middle|\, \exists \mathbf{u}_{k:T-1} \in \mathcal{U}^{T-k}, \text{ s.t., } x_k \xi_f(x_k, \mathbf{u}_{k:T-1}) \models \hat{\Phi}_k^I \right\}. \tag{8}$$

In what follows, we will present the main online monitoring algorithm by using the I-remaining feasible sets. The computation of set X_k^I will be detailed in Section 5.

4.2. Online monitoring algorithm

Note that, although we consider temporal operator "Eventually" in the semantics, it is subsumed by operator "Until" since $\mathbf{F}_{[a_i,b_i]}x \in \mathcal{H}_i$ can be expressed as $x \in \mathcal{X}\mathbf{U}'_{[a_i,b_i]}x \in \mathcal{H}_i$. Therefore, technically, we only need to handle temporal operators $\mathbf{G}$ and $\mathbf{U}'$. Specifically, in terms of the satisfaction:

- For sub-formula of form $\Phi_i^{[a_i,b_i]} = \mathbf{G}_{[a_i,b_i]}x \in \mathcal{H}_i$ with operator $\mathbf{G}$, it is satisfied only when state x_k is still in the region $\mathcal{H}_i$ at the last instant b_i.
- For sub-formula of form $\Phi_i^{[a_i,b_i]} = x \in \mathcal{H}_i^1\mathbf{U}'_{[a_i,b_i]}x \in \mathcal{H}_i^2$, however, its satisfaction can be determined at any instant $k \in [a_i, b_i]$ if state x_k is in region $\mathcal{H}_i^1 \cap \mathcal{H}_i^2$.

Based on the above discussion, now we present the complete online monitoring algorithm, which is shown in Algorithm 1. We use a global variable I to record the indices of sub-formulae that have not been satisfied. We start from the initial instant $k = 0$ (line 1) and I is set as the indices of all formulae $\mathcal{I}$ (line 2). The monitor decision for each instant k is computed in the while-loop. Specifically, the monitor first reads the current state x_k (line 4) and uses its I-remaining feasible set X_k^I to issue a monitoring decision. Specifically, if x_k is not in X_k^I, we know that entire formula cannot be satisfied anymore (lines 5–7). If it is in X_k^I, then we use this state information to determine whether or not some sub-formulae in set I are achieved based on rules discussed above (lines 8–12). If sub-formulae Φ_i are satisfied, then we delete its index i from the remaining set I (line 13). Note that we only need to check the satisfaction of remaining sub-formulae with indices in $I \cap \mathcal{I}_k$ since sub-formulae with indices in $\mathcal{I}_{>k}$ cannot be satisfied at instant k. This process is repeated until remaining index set I is empty, i.e., the entire formula is satisfied.

Algorithm 1: Online Monitoring Algorithm

    Input: feasible sets
    Output: monitoring decision $\mathcal{M}_k$
1   $k \leftarrow 0$
2   $I \leftarrow \mathcal{I}$
3   **while** $I \neq \emptyset$ **do**
4   read new current state x_k
5   **if** $x_k \notin X_k^I$ **then**
6         $\mathcal{M}_k = 1$
7   **return** "*prefix is violated*"
8   **else**
9         $\mathcal{M}_k = 0$
10   **forall** $i \in I \cap \mathcal{I}_k$ **do**
11   **if** $[O_i = \mathbf{G} \wedge k = b_i \wedge x_k \in \mathcal{H}_i]$ **or**
12          $[O_i = \mathbf{U}' \wedge x_k \in \mathcal{H}_i^1 \cap \mathcal{H}_i^2]$ **then**
13           $I \leftarrow I \setminus \{i\}$
14      $k \leftarrow k + 1$
15 **return** "*Φ has been satisfied*"

The following theorem establishes the correctness of Algorithm 1 for Problem 1.

Theorem 1. *Given a dynamical system of form (1) and an STL formula Φ as in (4), Algorithm 1 correctly solves Problem 1 if all feasible sets in (8) are computed exactly, i.e., it is both sound and complete.*

Proof. If the conclusion of Algorithm 1 is $\mathcal{M}_k = 1$ "violated prefix", i.e., $x_k \notin X_k^I$, then according to Definition 2, starting from x_k, for any control input $\mathbf{u}_{k:T-1}$, we have $x_k \xi_f(x_k, \mathbf{u}_{k:T-1}) \not\models \hat{\Phi}_k^I$. Therefore, we know that $\forall \mathbf{u}_{k:T-1} : \mathbf{x}_{0:k}\xi_f(x_k, \mathbf{u}_{k:T-1}) \not\models \Phi$, which is the same as the definition of violated prefix in the beginning of Section 3.2. On the other hand, if the conclusion is $\mathcal{M}_k = 0$ "feasible prefix", i.e., $x_k \in X_k^I$, then starting from x_k, there exists a control sequence $\mathbf{u}_{k:T-1}$ such that $x_k \xi_f(x_k, \mathbf{u}_{k:T-1}) \models \hat{\Phi}_k^I$ by Definition 2. Furthermore, all sub-formulae with index in $\mathcal{I} \setminus I$ have already been satisfied by $\mathbf{x}_{0:k-1}$ for the following reasons. From Algorithm 1, we have $\mathcal{M}_{k'} = 0$ and $x_{k'} \in X_{k'}^{I'}$ for all $k' \in [0, k-1]$, where I' is the remaining index set at instant k'. Then for any $i \in \mathcal{I} \setminus I$, we consider the following two cases:

- If $O_i = \mathbf{G}$, then we have $X_{k'}^{I'} \subseteq \mathcal{H}_i$ for $k' \in [a_i, b_i]$ according to Definition 2 without specific computation. Then, $x_{k'} \in X_{k'}^{I'}$

for all $k' \in [0, k-1]$ implies that $\mathbf{G}_{[a_i,b_i]}x \in \mathcal{H}_i$ is satisfied by $\mathbf{x}_{0:k-1}$;

- If $O_i = \mathbf{U}'$, assume Line 12 in Algorithm 1 is satisfied and i is removed from I at instant k', then $X_{k''}^{I''} \subseteq \mathcal{H}_i^1$ for $k'' \in [a_i, k']$ where I'' is the remaining index set at instant k'' and $i \in I''$, which implies that $x \in \mathcal{H}_i^1 \mathbf{U}'_{[a_i,b_i]} x \in \mathcal{H}_i^2$ is satisfied by $\mathbf{x}_{0:k-1}$.

Also, we can prove that sub-formulae $\mathbf{G}_{[a_i,k-1]}x \in \mathcal{H}_i, i \in I \cap \mathcal{I}_k$ have been satisfied in the same way. Hence, the prefix $\mathbf{x}_{0:k}$ is indeed feasible for the entire Φ, since $\hat{\Phi}_k^I$ can be satisfied, and sub-formulae with index $\mathcal{I} \setminus I$ and "Always sub-formulae" with index $I \cap \mathcal{I}_k$ have been satisfied.

Remark 3. Theorem 1 states the properties of Algorithm 1 for the ideal case where all feasible sets can be precisely computed. In general, over-/inner-approximation techniques are needed for the computation of feasible sets. If we compute feasible sets X_k^I by over-approximations, then miss-alarms may be possible since we allow states that are not actually feasible. On the other hand, if we compute feasible sets by inner-approximations, then false-alarms may be possible. For safety-critical systems, however, it is more meaningful to use inner-approximations in order to avoid miss-alarms. In terms of our implementation in Section 7, we adopt inner-approximations for Cases 1, 3 and 4, and for Case 2, the feasible sets are computed exactly due to its simple system dynamic.

Remark 4. Compared with the direct approach discussed in Remark 1, the major advantage of the proposed online monitoring algorithm is that the online computation burden is very low. At each time instant, instead of solving a complicated satisfaction problem on-the-fly, our approach just needs to check a set membership. Particularly, the I-remaining feasible sets can be computed in an offline fashion and stored in the monitor. Furthermore, our algorithm is only based on the current state x_k and do not need to remember the entire trajectory generated by the system.

5. Pre-computations of feasible sets

In this section, we present methods for the computation of I-remaining feasible sets X_k^I at time instant k.

5.1. Computation list

Recall that during online monitoring process, the monitor uses I-remaining feasible sets X_k^I at each instant k, where I is the set of remaining unsatisfied sub-formula index which is determined by state trajectory $\mathbf{x}_{0:k-1}$. Since the state trajectory $\mathbf{x}_{0:k-1}$ is unknown a priori at the starting instant of the system, the remaining set I also has multiple possibilities for each instant k. It seems that we need to compute feasible sets for all subsets in $2^{\mathcal{I}}$ for each instant $k \in [0, T]$. However, the number of remaining sets that are actually possible at each instant is much smaller than the exponential worst-case due to the following observations. First, at instant k, the remaining set I cannot contain any indices of sub-formulae in $\mathcal{I}_{<k}$ and that in $\mathcal{I}_{>k}$ are all contained in I since they have not been evaluated. Second, for each index in $\mathcal{I}_k$, if the temporal operator is "Always" or k is the first instant of "Until", then their index must be in set I since they cannot be satisfied based on the past information. Formally, we define the *potential index set* for instant k as follows.

Definition 3 (*Potential Index Set*). For each instant $k \in [0, T]$, we say subset $I \subseteq \mathcal{I}$ is a *potential index set* for instant k if

(i) $\mathcal{I}_{<k} \cap I = \emptyset$; and

(ii) $\mathcal{I}_{>k} \subseteq I$; and
(iii) $\{i \in \mathcal{I}_k \mid [O_i = \mathbf{G}] \vee [O_i = \mathbf{U}' \wedge k = a_i]\} \subseteq I$.

We denote by $\mathbb{I}_k$ the set of all potential index sets for instant k. Similarly, we define $\mathbb{X}_k = \{X_k^I \mid I \in \mathbb{I}_k\}$ as the set of all *potential feasible sets* for instant k. Then our offline objective is to compute all elements in $\{\mathbb{X}_k : k \in [0, T]\}$.

Example 3 (*Cont.*). Let us consider the STL formula Φ in Eq. (6). For instant $k = 7$, the set of all potential index sets and all potential feasible sets are $\mathbb{I}_7 = \{\{2, 3, 4, 5\}\}$ and $\mathbb{X}_7 = \{X_7^{\{2,3,4,5\}}\}$, respectively. For instant $k = 8$, we have $\mathbb{I}_8 = \{\{4, 5\}, \{3, 4, 5\}\}$ and $\mathbb{X}_8 = \{X_8^{\{4,5\}}, X_8^{\{3,4,5\}}\}$.

5.2. Backwards computation of feasible regions

Now we present our approach for computing all potential feasible sets $\mathbb{X}_k$ for each instant k. The basis idea is to compute $\mathbb{X}_k$ recursively in a backwards manner. Specifically, suppose that we have already known the all potential feasible set in $\mathbb{X}_{k+1}$, and then we can use elements in $\mathbb{X}_{k+1}$ to compute each element in $\mathbb{X}_k$.

Note that for each remaining set $I \in \mathbb{I}_k$ for instant k, there are multiple choices $I' \in \mathbb{I}_{k+1}$ for the next instant depending on which currently effective sub-formulae are satisfied. Clearly, given the current remaining set $I \in \mathbb{I}_k$ not arbitrary $I' \in \mathbb{I}_{k+1}$ can be the remaining set for the next instant. To this end, we introduce the notion of *successor set* as follows.

Definition 4 (*Successor Sets*). Let $I \in \mathbb{I}_k$ be a remaining set at instant k, we say that $I' \in \mathbb{I}_{k+1}$ is a successor set of I for the next instant if

$$\forall i \in \mathcal{I}_{k+1} : [O_i = \mathbf{U}' \wedge i \notin I] \Rightarrow i \notin I'. \tag{9}$$

We denote by $\mathrm{succ}(I, k) \subseteq \mathbb{I}_{k+1}$ as the set of all successor sets of I from instant k.

Intuitively, I' is a successor set of I says that, for any sub-formula with "Until" operator, if it has been satisfied in I, then it should also be satisfied in I'. For the purpose of backwards computation, we define

- $\mathbb{I}_{T+1} = \{\emptyset\}$; and
- $\forall I \in \mathbb{I}_T : \mathrm{succ}(I, T) = \mathbb{I}_{T+1} = \{\emptyset\}$.

Note that, the successor set of I may not be unique in general since for those sub-formulae that have not yet been satisfied in I and are still effective at the next instant, they can be either in I' or not depending on the current state of the system. To capture this issue, we define the *satisfaction sets* and *consistent regions* as follows.

Definition 5 (*Satisfaction Sets and Regions*). Let $I \in \mathbb{I}_k$ be a remaining set for instant k and $I' \in \mathbb{I}_{k+1}$ be a successor set of I.

- The *satisfaction set* w.r.t. pair (I, I') is defined by

$$\mathrm{sat}_\mathsf{U}(I, I') = \{i \in I : O_i = \mathbf{U}' \wedge i \notin I'\}, \tag{10}$$

which is the set of indices of sub-formulae with "Until" operator that are in I but not in I'.

- The *consistent region* w.r.t. pair (I, I') is defined by

$$H_k(I, I') = \bigcap_{i \in I \cap \mathcal{I}_k} H_i, \tag{11}$$

where

$$H_i = \begin{cases} \mathcal{H}_i^1 \cap \mathcal{H}_i^2 & \text{if } i \in \mathrm{sat}_\mathsf{U}(I, I') \\ \mathcal{H}_i^1 \setminus \mathcal{H}_i^2 & \text{if } O_i = \mathbf{U}' \wedge i \notin \mathrm{sat}_\mathsf{U}(I, I') \\ \mathcal{H}_i & \text{if } O_i = \mathbf{G}. \end{cases} \tag{12}$$

The intuitions of the above definitions are as follows. Suppose that the current remaining set is I at instant k and becomes I' at instant $k+1$. Then satisfaction set $\mathsf{sat}_\mathsf{U}(I, I')$ captures the index set of "Until" formulae that are satisfied at instant k. Since I is considered as an element in $\mathbb{I}_k$, we naturally have $\mathsf{sat}_\mathsf{U}(I, I') \subseteq \mathcal{I}_k$. In order to trigger the evolution of the remaining set from I to I', at instant k, the system should be in the consistent region $H_k(I, I')$. Specifically, we consider each sub-formula that is remaining and effective at instant k, i.e., $i \in I \cap \mathcal{I}_k$. Then we have the following three cases as shown in Eq. (12):

- If $i \in \mathsf{sat}_\mathsf{U}(I, I')$, then it means that the system must be in region $\mathcal{H}_i^1 \cap \mathcal{H}_i^2$ in order to satisfy sub-formula i;
- If $O_i = \mathbf{U}'$ but $i \notin \mathsf{sat}_\mathsf{U}(I, I')$, then it means that the sub-formula i is not yet satisfied, and, therefore, the system should be in region $\mathcal{H}_i^1 \setminus \mathcal{H}_i^2$;
- If $O_i = \mathbf{G}$, then the system should stay in region $\mathcal{H}_i$.

Since each sub-formula $i \in I \cap \mathcal{I}_k$ should satisfy the above requirements, $H_k(I, I')$ is taken as the intersection of the region of each sub-formula.

Now, for some $I \in \mathbb{I}_k$ and its successor set $I' \in \mathbb{I}_{k+1}$, the computation of feasible set X_k^I can be divided into two parts:

- First, it should stay in the consistent region $H_k(I, I')$ at instant k;
- Also, it needs to be able to reach region $X_{k+1}^{I'}$ in one step to satisfy the subsequent requirements.

This observation is formalized with the help of *one-step feasible set* defined as follows.

Definition 6 (*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

$$\Upsilon(\mathcal{S}) = \{x \in \mathcal{X} \mid \exists u \in \mathcal{U}, \text{s.t.}, f(x, u) \in \mathcal{S}\}. \quad (13)$$

In terms of our computation of feasible regions, if the system is evolving from I to I' and maintains the satisfiability of I' from instant $k+1$, then we know that the system should be in region $H(I, I') \cap \Upsilon(X_{k+1}^{I'})$ at instant k. However, the remaining set I' for the next instant depends on the current-state of the system. Therefore, to compute X_k^I, we need to consider all possible successor sets $I' \in \mathsf{succ}(I, k)$, and take the union of these regions. This is formalized by the following theorem.

Theorem 2. *Suppose that I is the remaining index set and $\mathbb{X}_{k+1}$ is the set of all potential feasible sets at next time instant. Then I-remaining feasible set X_k^I defined in Definition 2 for the time instant k can be computed as follows*

$$X_k^I = \bigcup_{I' \in \mathsf{succ}(I, k)} \left(H_k(I, I') \cap \Upsilon(X_{k+1}^{I'}) \right). \quad (14)$$

Proof. When $k = T$, since $\mathsf{succ}(I, k) = \{\emptyset\}$ and $\mathsf{sat}_\mathsf{U}(I, I') = \mathsf{sat}_\mathsf{U}(I, \emptyset) = \{i \in I : O_i = \mathbf{U}'\}$, from Eq. (14), we have

$$X_T^I = \bigcap_{i \in \mathsf{sat}_\mathsf{U}(I, I')} (\mathcal{H}_i^1 \cap \mathcal{H}_i^2) \cap \bigcap_{i \in I \setminus \mathsf{sat}_\mathsf{U}(I, I')} \mathcal{H}_i,$$

which is clearly the I-remaining feasible set of $\hat{\Phi}_k^I = \bigwedge_{i \in I} \Phi_i^{[T, T]}$.

For the case of $k \neq T$, we can write $\hat{\Phi}_k^I$ in the form of Eq. (15) in Box I, where I_k is the abbreviation of $I \cap \mathcal{I}_k$, and $I_k^\mathbf{G}, I_k^\mathbf{U}$ are the sets of "Always" and "Until" indices contained in I_k, respectively. Intuitively, Eq. (15) divides I_k into two parts, $I_k \setminus \mathcal{I}_{k+1}$ and $I_k \cap$

$\mathcal{I}_{k+1}$, which are the sets of indices of the last instants and the non-last instants, respectively. We can further split $I_k \cap \mathcal{I}_{k+1}$ according to different temporal operators as shown in the second line of Eq. (15). For "Until" sub-formulae with indices in $I_k^\mathbf{U} \cap \mathcal{I}_{k+1}$, it can be either satisfied currently or postponed to next instant. We denote by $\hat{I}$ the index set for those sub-formulae that are not satisfied currently, which can be any subset of $I_k^\mathbf{U} \cap \mathcal{I}_{k+1}$. Then $\hat{\Phi}_k^I$ can be further rewritten as the third line of Eq. (15), where for each possible $\hat{I}$, we divide the corresponding formula into two parts: one only related to the current state (denoted by $\psi_1(\hat{I})$) and the other related to the future requirements (denoted by $\psi_2(\hat{I})$).

We observe that, for any $\hat{I}$, we have $x_k \xi(x_k, \mathbf{u}_{k:T-1}) \models \psi_1(\hat{I}) \wedge \psi_2(\hat{I})$ if the following holds:

- $x_k \models \psi_1(\hat{I})$,
- $x_{k+1} \xi(x_{k+1}, \mathbf{u}_{k+1:T-1}) \models \psi_2(\hat{I})$,
- x_{k+1} is reachable from x_k under some u_k.

The first condition holds iff x_k stays in region $H_k(I, I')$ with $I' = \mathcal{I}_{>k} \cup (I_k^\mathbf{G} \cap \mathcal{I}_{k+1}) \cup \hat{I}$ and $\mathsf{sat}_\mathsf{U}(I, I') = (I_k^\mathbf{U} \setminus \mathcal{I}_{k+1}) \cup (I_k^\mathbf{U} \cap \mathcal{I}_{k+1} \setminus \hat{I})$. The second condition holds iff x_{k+1} is in I'-remaining feasible set $X_{k+1}^{I'}$. The third condition holds iff $x_k \in \Upsilon(X_{k+1}^{I'})$. Therefore, $\psi(\hat{I}) := \psi_1(\hat{I}) \wedge \psi_2(\hat{I})$ holds iff x_k is in $H_k(I, I') \cap \Upsilon(X_{k+1}^{I'})$. Finally, recall that $\hat{\Phi}_k^I$ is the disjunction of all possible $\psi(\hat{I})$. This suffices to consider all possible $I' \in \mathsf{succ}(I, k)$. Therefore, we have $X_k^I = \bigcup_{I' \in \mathsf{succ}(I, k)} \left(H_k(I, I') \cap \Upsilon(X_{k+1}^{I'}) \right)$ which is the same as Eq. (14), i.e., the theorem is proved.

Example 4 (*Cont.*). Let us consider the STL formula Φ in Eq. (6). For instant $k = 11$, assume that remaining set is $I = \{3, 4, 5\} \in \mathbb{I}_{11}$. In this case, any set in $\mathbb{I}_{12} = \{\emptyset, \{3\}, \{5\}, \{3, 5\}\}$ could be a possible successor set as defined in Definition 4, i.e., $\mathsf{succ}(I, 11) = \mathbb{I}_{12}$, and we denote by $I_1' = \emptyset, I_2' = \{3\}, I_3' = \{5\}$ and $I_4' = \{3, 5\}$. Note that for sub-formula $\Phi_3 = \mathbf{F}_{[5,15]} x \in \mathcal{H}_\mathbf{F}$, we write it as $x \in \mathcal{X} \mathbf{U}_{[5,15]}' x \in \mathcal{H}_\mathbf{F}$ here. For four possible successor sets, their satisfaction sets are

$$\mathsf{sat}_\mathsf{U}(I, I_1') = \{3, 5\}, \quad \mathsf{sat}_\mathsf{U}(I, I_2') = \{5\},$$
$$\mathsf{sat}_\mathsf{U}(I, I_3') = \{3\} \text{ and } \mathsf{sat}_\mathsf{U}(I, I_4') = \emptyset$$

respectively, and their consistent regions are

$$H_k(I, I_1') = (\mathcal{H}_\mathbf{U}^1 \cap \mathcal{H}_\mathbf{U}^2) \cap (\mathcal{X} \cap \mathcal{H}_\mathbf{F}) \cap \mathcal{H}_\mathbf{G},$$
$$H_k(I, I_2') = (\mathcal{H}_\mathbf{U}^1 \cap \mathcal{H}_\mathbf{U}^2) \cap (\mathcal{X} \setminus \mathcal{H}_\mathbf{F}) \cap \mathcal{H}_\mathbf{G},$$
$$H_k(I, I_3') = (\mathcal{X} \cap \mathcal{H}_\mathbf{F}) \cap (\mathcal{H}_\mathbf{U}^1 \setminus \mathcal{H}_\mathbf{U}^2) \cap \mathcal{H}_\mathbf{G},$$
$$H_k(I, I_4') = (\mathcal{H}_\mathbf{U}^1 \setminus \mathcal{H}_\mathbf{U}^2) \cap (\mathcal{X} \setminus \mathcal{H}_\mathbf{F}) \cap \mathcal{H}_\mathbf{G},$$

respectively. Then, we have I-remaining feasible set at instant 11 is

$$X_{11}^I = \left(H_k(I, I_1') \cap \Upsilon(X_{12}^{I_1'}) \right) \cup \left(H_k(I, I_2') \cap \Upsilon(X_{12}^{I_2'}) \right)$$
$$\cup \left(H_k(I, I_3') \cap \Upsilon(X_{12}^{I_3'}) \right) \cup \left(H_k(I, I_4') \cap \Upsilon(X_{12}^{I_4'}) \right).$$

5.3. Offline computation algorithm

Now, we present the complete procedure for offline computations of feasible sets in Algorithm 2. Initially, for each instant $k \in [0, T]$, we set $\mathbb{X}_k$ as the empty set since no X_k^I is computed for some $I \in \mathbb{I}_k$ (line 1). Then for the unique element $X_{T+1}^\emptyset$ in $\mathbb{X}_{T+1}$, we set it as $\mathbb{R}^n$ since the formula has already been finished at instant T and there is no requirement at instant $T + 1$. To proceed the backwards induction, at each instant $k \in [0, T]$, for each $I \in \mathbb{I}_k$, we compute X_k^I according to Eq. (14) by considering its all possible successor sets at instant $k + 1$ (lines 4–8).

$$\hat{\Phi}_k^I = \bigwedge_{i \in I \cap \mathcal{I}_k} \Phi_i^{[k,b_i]} \wedge \bigwedge_{i \in \mathcal{I}_{>k}} \Phi_i^{[a_i,b_i]} = \bigwedge_{i \in I_k \backslash \mathcal{I}_{k+1}} \Phi_i^{[k,k]} \wedge \underbrace{\bigwedge_{i \in I_k \cap \mathcal{I}_{k+1}} \Phi_i^{[k,b_i]}} \wedge \bigwedge_{i \in \mathcal{I}_{>k}} \Phi_i^{[a_i,b_i]} \tag{15}$$

$$= \bigwedge_{i \in I_k \backslash \mathcal{I}_{k+1}} \Phi_i^{[k,k]} \wedge \underbrace{\bigwedge_{i \in I_k^{\mathbf{U}} \cap \mathcal{I}_{k+1}} \Phi_i^{[k,b_i]}} \wedge \bigwedge_{i \in I_k^{\mathbf{G}} \cap \mathcal{I}_{k+1}} \Phi_i^{[k,b_i]} \wedge \bigwedge_{i \in \mathcal{I}_{>k}} \Phi_i^{[a_i,b_i]}$$

$$= \bigvee_{\hat{I} \subseteq I_k^{\mathbf{U}} \cap \mathcal{I}_{k+1}} \Big(\underbrace{\bigwedge_{i \in I_k \backslash \mathcal{I}_{k+1}} \Phi_i^{[k,k]} \wedge \bigwedge_{i \in I_k^{\mathbf{U}} \cap \mathcal{I}_{k+1} \backslash \hat{I}} \Phi_i^{[k,k]} \wedge \bigwedge_{i \in \hat{I}} \mathbf{G}_{[k,k]} x \in \mathcal{H}_i^1 \wedge \bigwedge_{i \in I_k^{\mathbf{G}} \cap \mathcal{I}_{k+1}} (\Phi_i^{[k,k]} \wedge \Phi_i^{[k+1,b_i]})}_{\psi_1(\hat{I})} \wedge \underbrace{\bigwedge_{i \in \hat{I}} \Phi_i^{[k+1,b_i]} \wedge \bigwedge_{i \in \mathcal{I}_{>k}} \Phi_i^{[a_i,b_i]}}_{\psi_2(\hat{I})} \Big)$$

Box I.

Algorithm 2: Offline Computations of Feasible Sets

Input: STL formula $\Phi = \bigwedge_{i=1}^N \Phi^{[a_i,b_i]}$
Output: All potential sets $\{\mathbb{X}_k : k \in [0, T]\}$

1   for each $k \in [0, T]$, $\mathbb{X}_k \leftarrow \emptyset$
2   $X_{T+1}^\emptyset \leftarrow \mathbb{R}^n$
3   $k \leftarrow T$
4   **while** $k \geq 0$ **do**
5   **forall** $I \in \mathbb{I}_k$ **do**
6       $X_k^I \leftarrow \bigcup_{I' \in \text{succ}(I,k)} \left(H_k(I, I') \cap \Upsilon(X_{k+1}^{I'}) \right)$
7       $\mathbb{X}_k \leftarrow \mathbb{X}_k \cup \{X_k^I\}$
8     $k \leftarrow k - 1$

5.4. Numerical computation considerations

Finally, we conclude this section by discussing some considerations in the numerical computation of above feasible sets.

Computations of One-Step Sets: In order to realize Algorithm 2, the key is to compute one-step feasible set $\Upsilon(\cdot)$. In general, there is no close-form expression for such sets and the computation highly depends on the dynamic of the system. Particularly, (inner or outer) approximation methodologies have been widely used in practice to achieve the trade off between the computational accuracy and complexity. For example, for linear systems, computation methods for one-step set have been presented subject to polytopic constraints described by linear differential inclusions or for piece-wise affine systems; see, e.g., Blanchini (1994), Kerrigan (2001) and Mayne (2001). For general nonlinear systems, however, computing the one-step set precisely is much more challenging (Bravo, Limón, Alamo, & Camacho, 2005; Mitchell & Tomlin, 2003; Stipanović, Hwang, & Tomlin, 2003). For example, Bravo et al. (2005) proposed a branch and bound algorithm with interval arithmetic approach which provides an inner approximation with a given bound of the error. Furthermore, we can also use some data-driven methods to compute it, and similar framework can be found in Devonport and Arcak (2020).

In our implementation in Section 7, we adopt the generic method in Bravo et al. (2005). The main idea of Bravo et al. (2005) is to start with a large interval box that contains all feasible (satisfiable) sets for sure and then to iteratively divide the box into smaller boxes and track which boxes are still within the feasible (satisfiable) sets by some forward reachable analysis tools; the readers are referred to Bravo et al. (2005) for more details. Note that the original algorithm in Bravo et al.

(2005) supports inner-approximation and can be easily extended to over-approximation.

Complexity of Pre-Computations: To perform the precomputations, first, we need to compute region $\mathcal{H}^\mu$ for each π^μ. This complexity generally depends on the nonlinearity degree of the predication function (Evtushenko, Posypkin, Rybak, & Turkin, 2018). The complexity for computing all feasible sets is linear in the horizon of the entire formula. Also, since we just need to consider sets in $\mathbb{I}_k$ for each instant rather than the entire power set of the index set, the total number of feasible sets to be computed generally will not grow exponentially as the horizon increases. However, the complexity of computing feasible sets at instant k is exponential in terms of the cardinality of index set $\mathcal{I}_k$, especially the cardinality of index of "Until" operators in $\mathcal{I}_k$, since the cardinality of $\mathbb{I}_k$ will explode exponentially as the number of "Until" operators in $\mathcal{I}_k$ grows. For each step in the iteration, the complexity for computing the one-step sets for constrained systems largely depends on the system model and may increase exponentially with the order of the system. Finally, it is worth mentioning again that the computations of feasible sets are purely offline, which do not affect the complexity of the online execution of the monitoring algorithm.

6. Extension to the case of satisfied prefixes

In the previous sections, we have formulated and solved the problem to detect violated prefixes. In some scenarios, however, *satisfied prefixes* are also of interest since the monitor may claim the satisfaction of task in advance and save resources for future processes. Formally, we say a prefix signal $\mathbf{x}_{0:k}$ is *satisfied* if for any control input $\mathbf{u}_{k:T-1}$, we have $\mathbf{x}_{0:k}\xi_f(x_k, \mathbf{u}_{k:T-1}) \models \Phi$. Our framework can be easily extended to the case where one is also interested in detecting satisfied prefix before the task is actually satisfied. Here, we introduce this extension briefly.

Online Monitoring: To detect satisfied prefixes online, we just need to check whether or not the system can always fulfill those I-remaining sub-formulae that have not been achieved up to now in the future. To capture this information, we introduce notion similar to Definition 2, called I-remaining satisfiable set as follows:

$$Y_k^I = \left\{ x_k \in \mathcal{X} \,\middle|\, \forall \mathbf{u}_{k:T-1} \in \mathcal{U}^{T-k}, \text{ s.t., } x_k \xi_f(x_k, \mathbf{u}_{k:T-1}) \models \hat{\Phi}_k^I \right\}.$$

The only difference between the I-remaining feasible set and the satisfiable set is their quantifiers "$\exists$" and "$\forall$". Clearly, we have $Y_k^I \subseteq X_k^I$. Then for online monitoring Algorithm 1, to detect satisfied prefixes, one can just add a new testing condition after line 9: if $x_k \in Y_k^I$, then return "*prefix is satisfied*". The soundness and completeness analysis of this case is consistent with that of monitoring algorithm for only feasible prefixes in Theorem 1.

Offline Computation: The computation method for satisfiable set Y_k^I is also similar to the case of feasible set and can be done with the help of one-step satisfiable set as follows:

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

Correspondingly, the I-remaining satisfiable set Y_k^I can be computed with Eq. (14) by replacing one-step feasible set $\Upsilon(\cdot)$ to satisfiable set $\hat{\Upsilon}(\cdot)$ as follows:

$$Y_k^I = \bigcup_{I' \in \text{succ}(I,k)} \left(H_k(I, I') \cap \hat{\Upsilon}(X_{k+1}^{I'}) \right),$$

and for offline computation we just need to repeat lines 6-7 again after it in the form of satisfiable set. The correctness can be also established in the same way which are omitted here.

7. Case studies for online monitoring

We have implemented our proposed computation methods for feasible sets as well as satisfiable sets in Julia language (Bezanson, Edelman, Karpinski, & Shah, 2017) with the help of existing package JuliaReach (Benet, Forets, Sanders, & Schilling, 2019; Bogomolov, Forets, Frehse, Potomkin, & Schilling, 2019; Forets & Schilling, 2021). All algorithms are carried out by a computer with i9-9900K CPU 3.60 GHz and 32 GB of RAM. Our codes are available at https://github.com/Xinyi-Yu/MPM4STL, where more details on the computations can be found.

Then we illustrate our online monitoring algorithm by applying it to four different case studies: building temperature control, double integrator, nonholonomic mobile robot and spacecraft rendezvous. Specifically, the offline computations are performed with one-step set computation method in Bravo et al. (2005) with some approximations except the second case study, whose feasible sets are computed analytically and exactly due to its simple linear model structure. Furthermore, for the first and the third case studies, we detect both violated and satisfied prefixes, while for the second and the fourth case studies, we only detect violated prefixed since most of their I-remaining satisfiable sets are empty. The total execution times for pre-computations of feasible sets in the four case studies are 1 min, 55 min, 3.3 h, and 2.1 h, respectively. The shorter execution time for the first case study can be attributed to its simpler system dynamics compared to the other case studies. We show that, by leveraging the model information of the dynamical system, our model-based approach may provide better monitoring evaluations compared with purely model-free approaches.
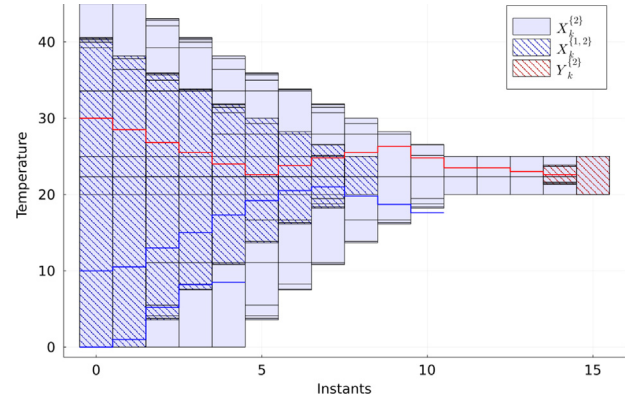
7.1. Building temperature control

We consider the problem of monitoring the temperature of a single zone building whose dynamic is given by the following difference equation

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

where state $x_k \in \mathcal{X} = [0, 45]$ denotes the zone temperature of building the at instant k, control input $u_k \in \mathcal{U} = [0, 1]$ is the ratio of the heater valve, $\tau_s = 1$ min is the sampling time, $T_h = 55\ °C$ is the heater temperature, $T_e = 0\ °C$ is outside temperature, and $\alpha_e = 0.06$ and $\alpha_H = 0.08$ are the heat exchange coefficients. The model is adopted from Jagtap, Soudjani, and Zamani (2020).

The objective of the temperature control system is to warm the single room to specific comfortable environment between 20 °C–25 °C in 8 min, and then keep the temperature in this interval from 10 to 15 min, which can be described by the following STL formula

$$\Phi := \mathbf{F}_{[0,8]} x_k \in [20, 25] \wedge \mathbf{G}_{[10,15]} x_k \in [20, 25].$$

Fig. 2. Three signals for the building temperature control system. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Before starting the online monitoring process, we first compute the I-remaining feasible and satisfiable sets of STL formula Φ by Algorithm 2 for each time instant k and for all $I \in \mathbb{I}_k$. The offline computation results are shown in Fig. 2. Specifically, areas filled with black dotted lines and blue colors are $\{1, 2\}$-remaining feasible set and $\{2\}$-remaining feasible set respectively, and red dotted lines denote $\{2\}$-remaining satisfiable sets. Some black horizontal lines in the figure were caused by one-step set computation since the computation needs to divide boxes into smaller ones. Note that satisfied sets $Y_k^{\{1,2\}}$ for all instants and $Y_k^{\{2\}}$ for $k \in [1, 13]$ are all empty.

During the online monitoring process, the monitor observes the current state at each time and makes evaluations. For example, let us consider three possible state traces generated by the system shown as the two blue lines and a red line in Fig. 2. At instant $k = 4$ and $k = 10$ respectively, using the model-free approach, one can only make the inconclusive evaluations for the two blue lines since the future signals can either satisfy remaining tasks or not without any constraints. However, using our model-based approach, since $x_4 \notin X_4^{\{1,2\}}$ for the below blue line and $x_{10} \notin X_{10}^{\{2\}}$ for the above, we can conclude immediately that the formula will be violated inevitably since there exists no controller under which the STL formula can be satisfied. Therefore, compared with existing model-free algorithms (Deshmukh et al., 2017; Ho et al., 2014), our method can claim the violation of specification in advance at instant 4 and 10 respectively, while existing algorithms cannot provide a clear violation conclusion. Also, for the red line, we can claim at instant 14 that the task will be satisfied definitely in the future since $x_{14} \in Y_{14}^{\{2\}}$ which implies that the next state x_{15} will be in [20, 25] no matter what controller u_{14} is, although it has not been satisfied currently according to model-free approaches.

7.2. Double integrator

For the second case study, we consider the planar motion of a single robot with double integrator dynamic, where the system model and temporal logic task are similar to those studied in Lindemann and Dimarogonas (2017). The system model with a sampling period of 0.5 s is as follows

$$x_{k+1} = \begin{bmatrix} 1 & 0.5 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0.125 & 0 \\ 0.5 & 0 \\ 0 & 0.125 \\ 0 & 0.5 \end{bmatrix} u_k,$$

where state $x_k = [x\ v_x\ y\ v_y]^T$ denotes x-position, x-velocity, y-position and y-velocity, and control input $u_k = [u_x\ u_y]^T$ denotes
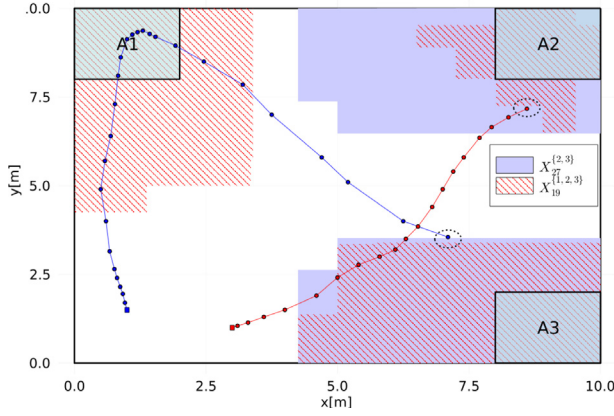
Fig. 3. Two trajectories for the double integrator system. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 4. Two trajectories for the nonholonomic mobile robot. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

x-acceleration and y-acceleration, respectively. The physical constraints are $x \in \mathcal{X} = [0, 10] \times [-1.5, 1.5] \times [0, 10] \times [-1.5, 1.5]$ and $u \in \mathcal{U} = [-1, 1] \times [-1, 1]$.

The objective of the robot is to visit all three regions A1, A2 and A3 shown in Fig. 3 within time interval between 10 to 40 instants (i.e., 5 to 20 s) in any order, which can be described by the following STL formula

$$\Phi := \Phi_1 \wedge \Phi_2 \wedge \Phi_3,$$

where $\Phi_1 := \mathbf{F}_{[10,40]}(x \in [0, 2] \wedge y \in [8, 10])$, $\Phi_2 := \mathbf{F}_{[10,40]}(x \in [8, 10] \wedge y \in [8, 10])$ and $\Phi_3 := \mathbf{F}_{[10,40]}(x \in [8, 10] \wedge y \in [0, 2])$.

We consider two trajectories of the robot starting from two rectangle points up to instant $k = 27$ and $k = 19$ (i.e., 13.5 s and 9.5 s) shown in Fig. 3, respectively, where the feasible sets $X_{27}^{\{2,3\}}$ and $X_{19}^{\{1,2,3\}}$ computed offline are also depicted. For the blue trajectory, at instant $k = 12$ the first time robot comes to A1, the remaining index set turns from $I = \{1, 2, 3\}$ to $\{2, 3\}$ and then we use $X_k^{\{2,3\}}$ to monitor. At instant 27, we observe that $x_{27} \notin X_{27}^{\{2,3\}}$ and the monitor can claim that the robot will never satisfy the task. Similarly, for the red trajectory, it does not visit any region of interest before $k = 19$ and also $x_{19} \notin X_{19}^{\{1,2,3\}}$. Then we can claim in advance that it will not complete the task. Note that the offline results shown in the figure are the projection to the first and third dimensions but the set membership is still checked in the complete 4-dimensional state space (also for later case studies).
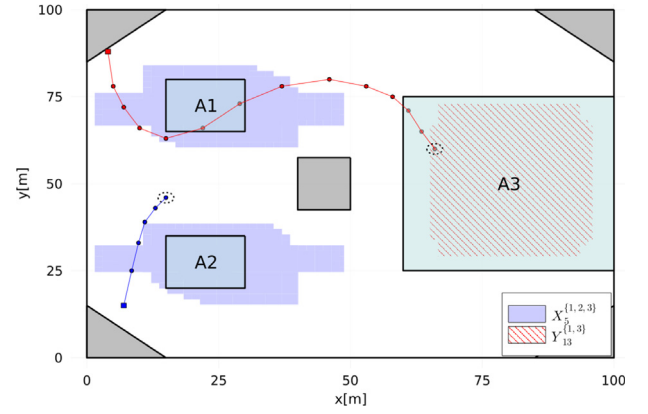
7.3. Nonholonomic mobile robot

Consider a nonholonomic mobile robots modeled by kinematic unicycles (Yu & Su, 2023) in the form of

$$\dot{x} = v \cos \theta, \ \dot{y} = v \sin \theta, \ \dot{\theta} = \omega,$$

and we discretize it with sampling time 0.5 s where state $x_k = [x \ y \ \theta]^T$ denotes x-position, y-position and angle, and control input $u_k = [v \ \omega]^T$ denotes speed and angular velocity, respectively. The physical constraints are $x \in \mathcal{X} = [0, 100] \times [0, 100] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$ and $u \in \mathcal{U} = [-10, 10] \times [-0.3, 0.3]$ if $k \leq 10$ and $\mathcal{U} = [-3, 3] \times [-0.3, 0.3]$ otherwise.

The workspace of the mobile robot is shown in Fig. 4. The objective is to first visit regions A1 or A2 (green) with a specific angle before instant 8 (i.e., 4 s) and then to stay in A3 (green) between instants 10 and 15 (i.e., 5 s and 7.5 s). Meanwhile, the robot should always avoid the obstacles (gray) in the map. The task can be described by the following STL formula

$$\Phi := \Phi_1 \wedge \Phi_2 \wedge \Phi_3,$$

where $\Phi_1 := \mathbf{G}_{[0,15]}(y < x + 85 \wedge y > -x + 15 \wedge y < -x + 185 \wedge y > x - 85 \wedge \neg(x \in [40, 50] \wedge y \in [42.5, 47.5]))$, $\Phi_2 := \mathbf{F}_{[0,8]}((x \in [15, 30] \wedge y \in [20, 35] \wedge \theta \in [-1.4, -0.2]) \vee (x \in [15, 30] \wedge y \in [65, 80] \wedge \theta \in [-1.4, -0.2]))$ and $\Phi_3 := \mathbf{G}_{[10,15]}(x \in [60, 100] \wedge y \in [25, 75])$.

We consider two trajectories of the robot starting from two rectangle points up to $k = 5$ in blue and up to $k = 13$ in red as shown in Fig. 4, respectively. The monitor can claim that the blue trajectory will violate the task at $k = 5$ since $x_5^{blue} \notin X_5^{\{1,2,3\}}$. Also, we can claim that red trajectory will satisfy the specification for sure at $k = 13$ since $x_{13}^{red} \in Y_{13}^{\{1,3\}}$.

7.4. Spacecraft rendezvous

Finally, we consider a spacecraft rendezvous example adopted from Chan and Mitra (2017). The model is the so-called Hill's relative coordinate frame centered on the *target* spacecraft, and the following nonlinear dynamic equations describe the two-dimensional, planar motion of *chaser* spacecraft on an orbital plane towards a target spacecraft

$$\dot{x} = v_x,$$
$$\dot{y} = v_y,$$
$$\dot{v}_x = n^2 x + 2n v_y + \frac{\mu}{r^2} - \frac{\mu}{r_c^3}(r + x) + \frac{u_x}{m_c},$$
$$\dot{v}_y = n^2 y - 2n v_x - \frac{\mu}{r_c^3} y + \frac{u_y}{m_c},$$

where state is $x_k = [x \ y \ v_x \ v_y]^T$, control input is chaser's thrusters $u_k = [u_x \ u_y]^T$ and we discretize the model with sampling time 0.5 min. The parameters are $\mu = 3.986 \times 10^{14} \times 60^2$ [m^3/min^2], $r = 42164 \times 10^3$ [m], $m_c = 500$ [kg], $n = \sqrt{\frac{\mu}{r^3}}$ and $r_c = \sqrt{(r + x)^2 + y^2}$. When the target and the chaser spacecrafts' separation distance is less than 100 m, the chaser needs to continue to rendezvous and be docked to the target spacecraft with specific angle of approach and closing velocity. Therefore, we focus on monitoring this critical rendezvous period. Specifically, the x-velocity and y-velocity should be less than 3.5 [m/min] and the thrusters cannot provide more than 10 N of force in any single direction in this period, i.e., the physical constraints are $x \in \mathcal{X} = [-100, 0] \times [-100, 0] \times [0, 3.5] \times [0, 3.5]$ and $u \in \mathcal{U} = [0, 10] \times [0, 10]$.

In terms of the temporal logic task, we require that the chaser will arrive a closer place to the target with a lower speed in 60 instants (i.e., 30 min), i.e., $P = \{(x, y, v_x, v_y) \mid x \in [-5, 0] \wedge y \in$
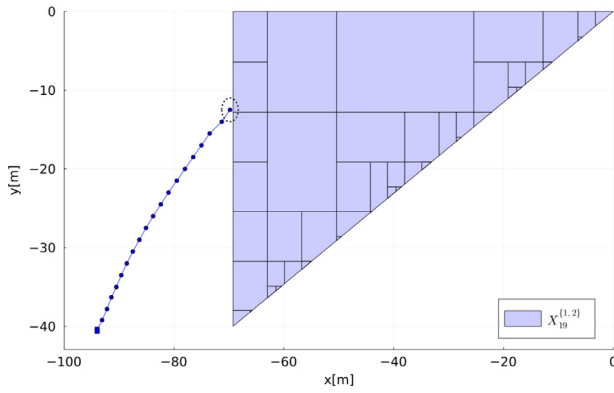
Fig. 5. One possible signal for spacecraft rendezvous case.

$[-5, 0] \wedge v_x \in [0, 2.5] \wedge v_y \in [0, 2.5]\}$, and once in P, we can use high precision camera with more state information to monitor, not just position and velocity. Also, the chaser must always remain with a line-of-sight cone $L = \{(x, y) \mid (y \geq x \tan(30°)) \wedge (-y \geq x \tan(30°))\}$ to keep a good position for docking preparation. Such a task can be described by the following formula

$$\Phi := \Phi_1 \wedge \Phi_2,$$

where $\Phi_1 := \mathbf{F}_{[0,60]}(x_k \in P)$ and $\Phi_2 := \mathbf{G}_{[0,60]}((x, y) \in L)$.

Considering a trajectory of a spacecraft starting from a rectangle points up to $k = 19$ instants (i.e., 9.5 min) as shown in Fig. 5, the monitor can make sure that the spacecraft will not be able to reach the area that the camera can monitor on time since $x_{19} \notin X_{19}^{\{1,2\}}$. This will provide engineers more time to reschedule the subsequent mission.

8. Conclusion

In this paper, we proposed a new model-based approach for online monitoring of tasks described by signal temporal logic formulae called model predictive monitoring, where we assume the underlying system model is known. Our algorithm consists of both offline pre-computation and online monitoring. Most of the computation efforts are made for the pre-computation characterized by the notion of feasible and satisfiable sets. The pre-computation information is used during the online monitoring to provide evaluations in real-time. We showed that the proposed method can evaluate the violation and satisfaction earlier than existing model-free approaches. Simulation results were provided to illustrate our results. Note that, in this work, we only consider the STL formula in the form of conjunction of subformulae in which the temporal operator will only be applied once for Boolean formulae. In the future, we would like to consider more general formula, including nested temporal operators, to further generalize our results. Also, we would like to extend our approach to the quantitative setting by taking robustness degrees into account.

Acknowledgment

References

Abate, M., Feron, E., & Coogan, S. (2019). Monitor-based runtime assurance for temporal logic specifications. In *IEEE conference on decision and control* (pp. 1997–2002).

Abbas, H., & Bonakdarpour, B. (2022). Leveraging system dynamics in runtime verification of cyber-physical systems. In *Leveraging applications of formal methods, verification and validation. verification principles: 11th international symposium, ISoLA 2022, Rhodes, Greece, October 22–30, 2022, proceedings, Vol. Part I* (pp. 264–278). Springer.

Bae, K., & Lee, J. (2019). Bounded model checking of signal temporal logic properties using syntactic separation. *Proceedings of the ACM on Programming Languages*, 3(POPL), 1–30.

Bartocci, E., Deshmukh, J., Donzé, A., Fainekos, G., Maler, O., Ničković, D., et al. (2018). Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. In *Lectures on runtime verification* (pp. 135–175).

Bauer, A., Leucker, M., & Schallhart, C. (2011). Runtime verification for LTL and TLTL. *ACM Transactions on Software Engineering and Methodology*, 20(4), 1–64.

Benet, L., Forets, M., Sanders, D., & Schilling, C. (2019). TaylorModels. jl: Taylor models in julia and their application to validated solutions of ODEs. In *SWIM*.

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1), 65–98.

Blanchini, F. (1994). Ultimate boundedness control for uncertain discrete-time systems via set-induced Lyapunov functions. *IEEE Transactions on Automatic Control*, 39(2), 428–433.

Bogomolov, S., Forets, M., Frehse, G., Potomkin, K., & Schilling, C. (2019). JuliaReach: a toolbox for set-based reachability. In *ACM international conference on hybrid systems: computation and control* (pp. 39–44).

Bonnah, E., & Hoque, K. (2022). Runtime monitoring of time window temporal logic. *IEEE Robotics and Automation Letters*, 7(3), 5888–5895.

Bravo, J., Limón, D., Alamo, T., & Camacho, E. (2005). On the computation of invariant sets for constrained nonlinear systems: An interval arithmetic approach. *Automatica*, 41(9), 1583–1589.

Buyukkocak, A., Aksaray, D., & Yazıcıoğlu, Y. (2022). Control barrier functions with actuation constraints under signal temporal logic specifications. In *European control conference* (pp. 162–168). IEEE.

Chan, N., & Mitra, S. (2017). Verifying safety of an autonomous spacecraft rendezvous mission. *EPiC Series in Computing*, 48, 20–32.

Deshmukh, J., Donzé, A., Ghosh, S., Jin, X., Juniwal, G., & Seshia, S. (2017). Robust online monitoring of signal temporal logic. *Formal Methods in System Design*, 51(1), 5–30.

Devonport, A., & Arcak, M. (2020). Data-driven reachable set computation using adaptive Gaussian process classification and Monte Carlo methods. In *American control conference* (pp. 2629–2634). IEEE.

Dokhanchi, A., Hoxha, B., & Fainekos, G. (2014). On-line monitoring for temporal logic robustness. In *International conference on runtime verification* (pp. 231–246). Springer.

Donzé, A., Ferrere, T., & Maler, O. (2013). Efficient robust monitoring for STL. In *International conference on computer aided verification* (pp. 264–279).

Donzé, A., & Maler, O. (2010). Robust satisfaction of temporal logic over real-valued signals. In *International conference on formal modeling and analysis of timed systems* (pp. 92–106). Springer.

Eisner, C., Fisman, D., Havlicek, J., Lustig, Y., McIsaac, A., & Campenhout, D. (2003). Reasoning with temporal logic on truncated paths. In *International conference on computer aided verification* (pp. 27–39).

Evtushenko, Y., Posypkin, M., Rybak, L., & Turkin, A. (2018). Approximating a solution set of nonlinear inequalities. *Journal of Global Optimization*, 71, 129–145.

Fainekos, G., & Pappas, G. (2009). Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42), 4262–4291.

Ferrando, A., Cardoso, R., Farrell, M., Luckcuck, M., Papacchini, F., Fisher, M., et al. (2022). Bridging the gap between single-and multi-model predictive runtime verification. *Formal Methods in System Design*, 1–33.

Forets, M., & Schilling, C. (2021). LazySets.jl: Scalable Symbolic-Numeric Set Computations. *Proceedings of the JuliaCon Conferences*, 1(1), 11.

Ghosh, B., & André, É. (2022). Offline and online monitoring of scattered uncertain logs using uncertain linear dynamical systems. In *International Conference on Formal Techniques for Distributed Objects, Components, and Systems* (pp. 67–87). Springer.

Gilpin, Y., Kurtz, V., & Lin, H. (2020). A smooth robustness measure of signal temporal logic for symbolic control. *IEEE Control Systems Letters*, 5(1), 241–246.

Hashimoto, W., Hashimoto, K., & Takai, S. (2022). STL2vec: Signal temporal logic embeddings for control synthesis with recurrent neural networks. *IEEE Robotics and Automation Letters*.

Ho, H., Ouaknine, J., & Worrell, J. (2014). Online monitoring of metric temporal logic. In *International conference on runtime verification* (pp. 178–192).

Jagtap, P., Soudjani, S., & Zamani, M. (2020). Formal synthesis of stochastic systems via control barrier certificates. *IEEE Transactions on Automatic Control*, 66(7), 3097–3110.

Jakšić, S., Bartocci, E., Grosu, R., Nguyen, T., & Ničković, D. (2018). Quantitative monitoring of STL with edit distance. *Formal Methods in System Design*, *53*(1), 83–112.

Kerrigan, E. (2001). *Robust constraint satisfaction: Invariant sets and predictive control* (Ph.D. thesis), University of Cambridge.

Lee, J., Yu, G., & Bae, K. (2021). Efficient SMT-based model checking for signal temporal logic. In *2021 36th IEEE/ACM international conference on automated software engineering (ASE)* (pp. 343–354). IEEE.

Leucker, M. (2012). Sliding between model checking and runtime verification. In *International conference on runtime verification* (pp. 82–87). Springer.

Lindemann, L., & Dimarogonas, D. (2017). Robust motion planning employing signal temporal logic. In *American control conference* (pp. 2950–2955). IEEE.

Lindemann, L., & Dimarogonas, D. (2018). Control barrier functions for signal temporal logic tasks. *IEEE Control Systems Letters*, *3*(1), 96–101.

Lindemann, L., & Dimarogonas, D. (2019). Robust control for signal temporal logic specifications using discrete average space robustness. *Automatica*, *101*, 377–387.

Lindemann, L., Qin, X., Deshmukh, J., & Pappas, G. (2023). Conformal prediction for STL runtime verification. In *ACM/IEEE International Conference on Cyber-Physical Systems* (pp. 142–153).

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

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

Maler, O., & Nickovic, D. (2004). Monitoring temporal properties of continuous signals. In *Formal techniques, modelling and analysis of timed and fault-tolerant systems* (pp. 152–166).

Mascle, C., Neider, D., Schwenger, M., Tabuada, P., Weinert, A., & Zimmermann, M. (2020). From LTL to rLTL monitoring: Improved monitorability through robust semantics. In *International conference on hybrid systems: computation and control* (pp. 1–12).

Mayne, D. (2001). Control of constrained dynamic systems. *European Journal of Control*, *7*(2–3), 87–99.

Mitchell, I., & Tomlin, C. (2003). Overapproximating reachable sets by Hamilton-Jacobi projections. *Journal of Scientific Computing*, *19*, 323–346.

Momtaz, A., Basnet, N., Abbas, H., & Bonakdarpour, B. (2021). Predicate monitoring in distributed cyber-physical systems. In *International conference on runtime verification* (pp. 3–22). Springer.

Pinisetty, S., Jéron, T., Tripakis, S., Falcone, Y., Marchand, H., & Preoteasa, V. (2017). Predictive runtime verification of timed properties. *Journal of Systems and Software*, *132*, 353–365.

Qin, X., & Deshmukh, J. (2020). Clairvoyant monitoring for signal temporal logic. In *International conference on formal modeling and analysis of timed systems* (pp. 178–195). Springer.

Raman, V., Donzé, A., Maasoumy, M., Murray, R., Sangiovanni-Vincentelli, A., & Seshia, S. (2014). Model predictive control with signal temporal logic specifications. In *IEEE conference on decision and control* (pp. 81–87).

Roehm, H., Oehlerking, J., Heinz, T., & Althoff, M. (2016). STL model checking of continuous and hybrid systems. In *Automated technology for verification and analysis: 14th international symposium, ATVA 2016, Chiba, Japan, October 17–20, 2016, proceedings, Vol. 14* (pp. 412–427). Springer.

Sahin, Y., Quirynen, R., & Di Cairano, S. (2020). Autonomous vehicle decision-making and monitoring based on signal temporal logic and mixed-integer programming. In *American control conference* (pp. 454–459). IEEE.

Salamati, A., Soudjani, S., & Zamani, M. (2021). Data-driven verification of stochastic linear systems with signal temporal logic constraints. *Automatica*, *131*, Article 109781.

Stipanović, D., Hwang, I., & Tomlin, C. (2003). Computation of an over-approximation of the backward reachable set using subsystem level set functions. In *2003 European control conference (ECC)* (pp. 300–305). IEEE.

Thati, P., & Roşu, G. (2005). Monitoring algorithms for metric temporal logic specifications. *Electronic Notes in Theoretical Computer Science*, *113*, 145–162.

Waga, M., André, É., & Hasuo, I. (2021). Model-bounded monitoring of hybrid systems. In *ACM/IEEE international conference on cyber-physical systems* (pp. 21–32).

Yoon, H., & Sankaranarayanan, S. (2021). Predictive runtime monitoring for mobile robots using logic-based Bayesian intent inference. In *IEEE international conference on robotics and automation* (pp. 8565–8571). IEEE.

Yu, X., Dong, W., Yin, X., & Li, S. (2022). Online monitoring of dynamic systems for signal temporal logic specifications with model information. In *IEEE conference on decision and control* (pp. 1553–1559). IEEE.

Yu, G., Lee, J., & Bae, K. (2022). STLmc: Robust STL model checking of hybrid systems using SMT. In *Computer aided verification: 34th international conference, CAV 2022, Haifa, Israel, August 7–10, 2022, proceedings, Vol. Part I* (pp. 524–537). Springer.

Yu, X., & Su, R. (2023). Decentralized circular formation control of nonholonomic mobile robots under a directed sensor graph. *IEEE Transactions on Automatic Control*, *68*(6), 3656–3663.

Zhang, X., Leucker, M., & Dong, W. (2012). Runtime verification with predictive semantics. In *NASA formal methods symposium* (pp. 418–432). Springer.

Zhao, D., Zhou, Z., Cai, Z., Yangui, S., & Xue, X. (2022). ASTL: Accumulative STL with a novel robustness metric for IoT service monitoring. *IEEE Transactions on Mobile Computing*.

Xinyi Yu was born in Jiangsu, China, in 1999. She received the B.Eng degree from China University of Petroleum (East China) in Qingdao in 2020, and the M.S. degree from Shanghai Jiao Tong University in Shanghai in 2023, both in automation. She is pursuing the Ph.D. degree in computer science at University of Southern California. Her research interests include formal methods and safety control.

Weijie Dong was born in Xinjiang, China, in 1996. He received the B.S. degree from Huazhong University of Science and Technology in 2019. He is pursuing the Ph.D. degree at Shanghai Jiao Tong University in the Department of Automation. His current research interests include fault diagnosis, state estimation of discrete event systems.

Shaoyuan Li was born in Hebei, China, in 1965. He received the B.S. and M.S. degrees in automation from the Hebei University of Technology, Tianjin, China, in 1987 and 1992, respectively, and the Ph.D. degree from Nankai University, Tianjin, in 1997. Since 1997, he has been with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, where he is currently a chair Professor. His current research interests include model predictive control, dynamic system optimization, and cyber–physical systems. He is the vice-president of the Chinese Association of Automation.
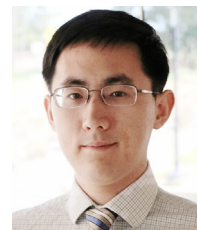
Xiang Yin was born in Anhui, China, in 1991. He received the B.Eng degree from Zhejiang University in 2012, the M.S. degree from the University of Michigan, Ann Arbor, in 2013, and the Ph.D degree from the University of Michigan, Ann Arbor, in 2017, all in electrical engineering. Since 2017, he has been with the Department of Automation, Shanghai Jiao Tong University, where he is an Associate Professor. His research interests include formal methods, discrete event systems and cyber–physical systems. Dr. Yin is serving as the chair of the *IEEE CSS Technical Committee on Discrete Event Systems*, Associate Editors for the *Journal of Discrete Event Dynamic Systems: Theory & Applications*, the *IEEE Control Systems Letters*, the *IEEE Transactions on Intelligent Vehicles*, and a member of the *IEEE CSS Conference Editorial Board*. Dr. Yin received the IEEE Conference on Decision and Control (CDC) Best Student Paper Award Finalist in 2016.