Model Predictive Monitoring of Dynamic Systems for Signal Temporal Logic Specifications

Xinyi Yu a,b, Weijie Dong a,b, Xiang Yin a,b, Shaoyuan Li a,b

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

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 dynamic system whose model is known. 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 dynamic 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.

Key words: Signal temporal logic; online monitoring; feasible set.

1 Introduction

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 [2]. 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) [16], Metric Temporal Logic (MTL) [13, 42] and Signal Temporal Logic (STL) [11, 14]. Recent applications of specification-based monitoring techniques include, e.g., robot systems [8], autonomous vehicles [40], fuel control system [25], smart cities [31], Internet of Things [48] and intelligent medicines [38].

Depending on what information can be utilized by the monitor, the monitoring problem can be categorized as offline (e.g., [14, 15, 17]) and online (e.g., [11, 13, 23]). 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 \([1, 3, 34]\) or MTL formulae \([23]\) In \([11, 13]\), 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 dynamic system. In this scenario, by leveraging the model information of the dynamic 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.

There are already many works along the line of leveraging model information for the purpose of online monitoring in the discrete domain during the last decade. For example, \([27, 47]\) introduce predictive semantics for monitoring of untimed LTL specifications for systems that are not black boxes. In \([37]\), the authors proposed a predictive runtime verification framework for systems with timing requirements. Recently, \([18]\) extends the single-model predictive monitoring approach to the multi-model case for both centralized and compositional settings. In \([44]\), the authors introduce a Bayesian intent inference framework leveraging the robot’s intent information to predict future positions.

For signals in the continuous domain, however, there are only a few existing works along this line these years. For example, \([43]\) and \([20]\) use prior knowledges 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 \([36]\), the authors adopt a lightweight mechanism for incorporating bounds on system dynamics to reduce monitoring overhead. For data-driven STL predictive monitoring, \([38]\) proposes to use statistical time-series analysis techniques to predicate future states. Also, \([32]\) uses Bayesian recurrent neural networks learned from data to predict future states with uncertainties. However, these approaches either use some rough information of the model or consider a purely unknown system dynamic.

In this paper, we propose a new model predictive monitoring approach of dynamic 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 \([21, 22, 29, 30, 33, 41]\). 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 dynamic 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.

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. Preliminary version of some results in this paper are presented in \([45]\). However, \([45]\) only considers a restrictive case that the horizons of different temporal operators have no overlap, which is relaxed in this work with more uniform techniques. Furthermore, in the present work, we provide detailed case studies to illustrate the proposed algorithm.
2 Preliminary

2.1 System Model

We consider a discrete-time control system of form

\[ x_{k+1} = f(x_k, u_k), \]

where \( x_k \in X \subseteq \mathbb{R}^n \) is the state at time \( k \), \( u_k \in U \subseteq \mathbb{R}^m \) is the control input at time \( k \) and \( f : X \times U \to X \) is the dynamic function of the system.

Suppose that the system is in state \( x_k \in X \) at time instant \( k \in \mathbb{Z}_{\geq 0} \). Then given a sequence of control inputs \( u_{k:N-1} = u_k u_{k+1} \ldots u_{N-1} \in U^{N-k} \), the solution of the system is a sequence of states

\[ \xi^f(x_k, u_{k:N-1}) = x_k x_{k+1} \ldots x_N \in X^{N-k} \]

such that \( x_{i+1} = f(x_i, u_i), i = k, \ldots, N - 1 \).

2.2 Signal Temporal Logic

We use Signal Temporal Logic (STL) formulae with bounded-time temporal operators \cite{33} to describe some desired high-level properties. Formally, the syntax of STL formulae is as follows

\[ \Phi ::= \top | \pi^\mu | \neg \Phi | \Phi_1 \land \Phi_2 | \Phi_1 U_{[a,b]} \Phi_2, \]

where \( \top \) is the true predicate, \( \pi^\mu \) is an atomic predicate whose truth value is determined by the sign of its underlying predicate function \( \mu : \mathbb{R}^n \to \mathbb{R} \) and it is true at state \( x_k \) when \( \mu(x_k) \geq 0 \); otherwise it is false. Notations \( \neg \) and \( \land \) are the standard Boolean operators “negation” and “conjunction”, respectively, which can further induce “disjunction” by \( \Phi_1 \lor \Phi_2 ::= \neg(\neg \Phi_1 \land \neg \Phi_2) \) and “implication” by \( \Phi_1 \to \Phi_2 ::= \neg \Phi_1 \lor \Phi_2 \). \( U_{[a,b]} \) is the temporal operator “until”, where \( a, b \in \mathbb{N} \) are two time instants with \( a \leq b \) and \( [a, b] \subseteq \mathbb{N} \) denotes the set of all integers between \( a \) and \( b \) including \( a \) and \( b \).

STL formulae are evaluated on state sequence \( x = x_0 x_1 \ldots \). We use notation \((x, k) \models \Phi\) to denote that sequence \( x \) satisfies STL formula \( \Phi \) at time instant \( k \). The reader is referred to \cite{33} for more details on the semantics of STL formulae. Particularly, we have \((x, k) \models \pi^\mu \) iff \( \mu(x_k) \geq 0 \), i.e., \( \mu(x_k) \) is non-negative for the current state \( x_k \), and \((x, k) \models \Phi_1 U_{[a,b]} \Phi_2 \) iff \( \exists k' \in [a+k, b+k] \) such that \((x, k') \models \Phi_2 \) and \( \forall k'' \in [k, k'] \), we have \((x, k'') \models \Phi_1 \), i.e., \( \Phi_2 \) will hold at some instant between \([a+k, b+k] \) in the future and before that \( \Phi_1 \) always holds. Furthermore, we can also induce temporal operators

- “always” \( G_{[a,b]} \Phi ::= \top U_{[a,b]} \Phi \) such that it holds when \((x, k) \models \Phi \) for some \( k' \in [k+a, k+b] \); and

- “eventually” \( F_{[a,b]} \Phi ::= \top U_{[a,b]} \Phi \) such that it holds when \((x, k) \models \Phi \) for any \( k' \in [k+a, k+b] \).

We write \( x \models \Phi \) whenever \((x, 0) \models \Phi \).

Given an STL formula \( \Phi \), in fact, it is well-known that the satisfaction of \( \Phi \) 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 \( \Phi \) only depends on time horizon \([S,T] \), where \( S \) is the starting instant of \( \Phi \) which is the minimum terminal instant that appears in the formula and \( T \) is the terminal instant of \( \Phi \) which is the maximum sum of all nested upper bounds. For example, for \( \Phi = F_{[2,7]} x^{\mu_1} \land G_{[3,12]} x^{\mu_2} \), 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:

\[ \varphi ::= \top \mid \pi^\mu \mid \neg \varphi \mid \varphi_1 \land \varphi_2, \]

\[ \Phi ::= F_{[a,b]} \varphi \mid G_{[a,b]} \varphi \mid \varphi_1 U_{[a,b]} \varphi_2 \mid \Phi_1 \land \Phi_2, \]

where \( \varphi_1, \varphi_2 \) are formulae of class \( \varphi \), and \( \Phi_1, \Phi_2 \) are formulae of class \( \Phi \). Specifically, we only allow the temporal operators to be applied once for Boolean formulae.

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

\[ (x, k) \models \Phi_1 U_{[a,b]} \Phi_2 \iff (x, k) \models (\Phi_1 U'_{[a,b]} \Phi_2) \land (G_{[0,a]} \Phi_1). \]

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

Based on the above discussion, STL formulae \( \Phi \) in Eq. (2)
can be expressed equivalently by:
\[
\Phi := \bigwedge_{i=1}^{N} \Phi_{i}^{[a_{i}, b_{i}]},
\]
where \( \Phi_{i}^{[a_{i}, b_{i}]} \) is a sub-formula that applies within time interval \([a_{i}, b_{i}]\) in the form of (3) as
\[
\Phi = (G_{[0,2]} x \in H_{G}) \land (G_{[3,7]} x \in H_{G} \cap H_{F}) \land (F_{[5,15]} x \in H_{F}) \land (G_{[8,11]} x \in H_{G}) \land (\neg \Phi_{1}) \land \Phi_{2},
\]
which is shown in Figure 1(b). In this case, we have \( I = \{1, 2, 3, 4, 5\} \) and \( O_{1}, O_{2}, O_{3} = G, O_{3} = F \) and \( O_{5} = U' \). Also, for time instant \( k = 9 \), we have \( I_{k} = \{3, 4, 5\} \).

### 3.2 Online Monitoring of STL

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

- **violated** if for any control input \( u_{k:T-1} \), we have \( x_{0:k} \xi_{f}(x_{k}, u_{k:T-1}) \not\models \Phi \);
- **feasible** if for some control input \( u_{k:T-1} \), we have \( x_{0:k} \xi_{f}(x_{k}, u_{k:T-1}) \models \Phi \).

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 \( G_{[0,T]} x \in H \), once the system reaches a state \( x_{k} \notin H \) for \( k < T \), we know immediately that the formula is violated. Also, if the system is in state \( x_{k} \) from which no solution \( \xi_{f}(x_{k}, u_{k:T-1}) \) can be found such that each state is in region \( H \), then we can also claim the formula cannot be satisfied anymore, i.e., it is violated.

Therefore, an online *monitor* is a function
\[
\mathcal{M} : \mathcal{X}^{*} \to \{0, 1\}
\]
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 dynamic system of form (1) and an STL formula \( \Phi \) as in (4), design an online monitor \( \mathcal{M} : \mathcal{X}^{*} \to \{0, 1\} \) such that for any prefix signal \( x_{0:k} \) where \( k \leq T \), we have \( \mathcal{M}(x_{0:k}) = 1 \) iff \( x_{0:k} \) is a violated prefix.

**Remark 1** We note that, for any prefix signal \( x_{0:k} \), it is a violated prefix if we cannot find a sequence of control inputs \( u_{k:T-1} \) such that \( x_{0:k} \xi_{f}(x_{k}, u_{k:T-1}) \models \Phi \). The existence of such a control sequence can be determined by
the binary encoding technique proposed in [39]. Therefore, a naive approach for designing an online monitor is to solve the above constrained satisfaction problem based on $x_{0:k}$. 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.

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 $\Phi$ 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

$$
\Phi^I_k = \bigwedge_{i \in I \cap \mathcal{I}_k} \Phi_{i[k,b_i]}^{[k,b_i]} \land \bigwedge_{i \in \mathcal{I}_{>k}} \Phi_{i[a_i,b_i]}^{[a_i,b_i]},
$$

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

Intuitively, $\Phi^I_k$ 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, we are only interested in the satisfiability in the future, the formulae is truncated from the given instant $k$.

**Example 2 (Cont.)** Let us consider the STL formula $\Phi$ in Equation (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 some sequence $x_{0:6}$. Then $I$-remaining formula at $k = 7$ is $\check{\Phi}^I_7 = (F_{[7,15]} x \in \mathcal{H}_F) \land (G_{[8,11]} x \in \mathcal{H}_G) \land (x \in \mathcal{H}^I_7 U^I_{[8,14]} x \in \mathcal{H}^I_7).

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 $\Phi$ 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 $X^I_k$, is the set of states from which there exists a solution that satisfies the $I$-remaining formula at $k$, i.e.,

$$
X^I_k = \left\{ x_k \in \mathcal{X} \mid \exists u_{k:T-1} \in U^{I-k} \text{ s.t. } x_k \xi_f(x_k, u_{k:T-1}) = \check{\Phi}^I_k \right\}. 
$$

In what follows, we will present the main online monitoring algorithm by using the $I$-remaining feasible sets. The computation of set $X^I_k$ 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 $F_{[a_i,b_i]} x \in \mathcal{H}_i$ can be expressed as $x \in \mathcal{X} U^I_{[a_i,b_i]} x \in \mathcal{H}_i$. Therefore, technically, we only need to handle temporal operators $G$ and $U$. Specifically, in terms of the satisfaction:

- for sub-formula of form $\Phi_{i[a_i,b_i]} = G_{[a_i,b_i]} x \in \mathcal{H}_i$ with operator $G$, 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 $\Phi_{i[a_i,b_i]} = x \in \mathcal{H}_i U^I_{[a_i,b_i]} x \in \mathcal{H}_i'$, 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 $I$ (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 $\Phi_i$ is 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 $I \cap I_k$ since sub-formulae with indices in $I_{>k}$ 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 $M_k$
1 $k \leftarrow 0$
2 $I \leftarrow I$
3 while $I \neq \emptyset$ do
4     read new current state $x_k$
5     if $x_k \notin X_k^I$ then
6         $M_k = 1$
7         return "prefix is violated"
8     else
9         $M_k = 0$
10        forall $i \in I \cap I_k$ do
11           if $[O_i = G \land k = b_i \land x_k \in H_i]$ or $[O_i = U' \land x_k \in H_i^1 \land H_i^2]$ then
12              $I \leftarrow I \setminus \{i\}$
13        $k \leftarrow k + 1$
14 return "$\Phi$ has been satisfied"
```

Remark 2 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 Offline Computation 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 $x_{0:k-1}$. Since the state trajectory $x_{0:k-1}$ 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 $2^I$ 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 $I_{<k}$ and that in $I_{>k}$ are all contained in $I$ since they have not been evaluated. Second, for each index in $I_k$, if the temporal operator is "Always" or $k$ is the first instant of "Until", then their index must be in $I$ 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 $I \subseteq I$ is a potential index set for instant $k$ if

(i) $I_{<k} \cap I = \emptyset$; and
(ii) $I_{>k} \subseteq I$; and
(iii) $\{i \in I_k \mid [O_i = G] \lor [O_i = U' \land k = a_i] \} \subseteq I$.

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

Example 3 (Cont.) Let us consider the STL formula $\Phi$ in Equation (6). For instant $k = 7$, the set of all potential index sets and all potential feasible sets are $I_7 = \{\{2, 3, 4, 5\}\}$ and $X_7 = \{X_7^\{2,3,4,5\}\}$, respectively. For instant $k = 8$, we have $I_8 = \{\{4, 5\}, \{3, 4, 5\}\}$ and $X_8 = \{X_8^\{4,5\}, X_8^\{3,4,5\}\}$.

5.2 Backwards Computation of Feasible Regions

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

Note that for each remaining set $I \in I_k$ for instant $k$, there are multiple choices $I' \in I_{k+1}$ for the next instant depending on which currently effective sub-formulae are satisfied. Clearly, given the current remaining set $I \in I_k$
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

$$\forall i \in \mathcal{I}_{k+1} : [O_i = U' \land i \notin I] \Rightarrow i \notin I'. \quad (9)$$

We denote by $\text{succ}(I, k) \subseteq \mathbb{I}_{k+1}$ 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

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

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
  $$\text{sat}_U(I, I') = \{ i \in I : O_i = U' \land i \notin I' \}, \quad (10)$$
  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
  $$\mathcal{H}_k(I, I') = \bigcap_{i \in \text{sat}_U(I, I')} \mathcal{H}_i, \quad (11)$$
  where
  $$\mathcal{H}_i = \begin{cases} 
  \mathcal{H}^1_i \cap \mathcal{H}^2_i & \text{if } i \in \text{sat}_U(I, I') \\
  \mathcal{H}^1_i \setminus \mathcal{H}^2_i & \text{if } O_i = U' \land i \notin \text{sat}_U(I, I') \\
  \mathcal{H}_i & \text{if } O_i = G.
  \end{cases} \quad (12)$$

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 $\text{sat}_U(I, I')$ captures the indice set of “Until” formulae that are satisfied at instant $k$. Since $I$ is considered as an element in $\mathbb{I}_k$, we naturally have $\text{sat}_U(I, I') \subseteq \mathcal{I}_k$. 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 $\mathcal{H}_k(I, I')$. 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 Equation (12):

- If $i \in \text{sat}_U(I, I')$, then it means that the system must be in region $\mathcal{H}^1_i \cap \mathcal{H}^2_i$ in order to satisfy sub-formula $i$;
- If $O_i = U'$ but $i \notin \text{sat}_U(I, I')$, then it means that the sub-formula $i$ is not yet satisfied, and, therefore, the system should be in region $\mathcal{H}^1_i \setminus \mathcal{H}^2_i$;
- If $O_i = G$, 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, $\mathcal{H}_k(I, I')$ 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 $X^I_k$ can be divided into two parts:

- First, it should stay in the consistent region $\mathcal{H}_k(I, I')$ at instant $k$;
- Also, it needs to be able to reach region $X^I_{k+1}$ 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

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

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 $\mathcal{H}(I, I') \cap \mathcal{Y}(X^I_{k+1})$ at instant $k$. However, the remaining set $I'$ for the next instant depends on the current-state of the system. Therefore, to compute $X^I_{k+1}$, we need to consider all possible successor sets $I' \in \text{succ}(I, k)$, and take the union of these regions. This is formalized by the following theorem.

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

$$X^I_k = \bigcup_{I' \in \text{succ}(I, k)} \left( \mathcal{H}(I, I') \cap \mathcal{Y}(X^I_{k+1}) \right). \quad (14)$$

**Proof.** When $k = T$, since $\text{succ}(I, k) = \{ \emptyset \}$ and
satisfies all possible \( k' \in \text{succ}(I, k) \). Therefore, we have 
\( X_k' = \bigcup_{k' \in \text{succ}(I, k)} (H_k(I, k') \cap \mathcal{Y}(X_{k+1}')) \)
which is the same as Equation (14), i.e., the theorem is proved.

**Example 4 (Cont.)** Let us consider the STL formula \( \Phi \) in Equation (6). For instance \( k = 11 \), assume that remaining set is \( I = \{3,4,5\} \in I_1 \). In this case, any set in \( I_2 = \{\emptyset, \{3\}, \{5\}, \{3,5\}\} \) could be a possible successor set as defined in Definition 4, i.e., \( \text{succ}(I, 11) = I_1 \), and we denote by \( I_1 = \emptyset, I_2' = \{3\}, I_2 = \{5\} \) and \( I_2 = \{3,5\} \). Note that for sub-formula \( \Phi_3 = \mathcal{F}_{[5,15]}x \in \mathcal{H}_F \), we write it as \( x \in \mathcal{X}U_{[5,15]}x \in \mathcal{H}_F \) here for four possible successor sets, their satisfaction sets are
\begin{align*}
\text{sat}_u(I, I_1') = \{3,5\}, \text{sat}_u(I, I_2') = \{5\}, \\
\text{sat}_u(I, I_1) = \{3\} \text{ and } \text{sat}_u(I, I_2') = \emptyset
\end{align*}
respectively, and their consistent regions are
\begin{align*}
H_k(I, I_1') &= (\mathcal{H}_U \cap \mathcal{H}_U') \cap (\mathcal{X} \cap \mathcal{H}_F) \cap \mathcal{H}_G, \\
H_k(I, I_2') &= (\mathcal{H}_U \cap \mathcal{H}_U') \cap (\mathcal{X} \cap \mathcal{H}_F) \cap \mathcal{H}_G, \\
H_k(I, I_1') &= (\mathcal{X} \cap \mathcal{H}_F) \cap (\mathcal{H}_U' \cap \mathcal{H}_U) \cap \mathcal{H}_G, \\
H_k(I, I_2') &= (\mathcal{X} \cap \mathcal{H}_F) \cap (\mathcal{H}_U' \cap \mathcal{H}_U) \cap \mathcal{H}_G,
\end{align*}
respectively. Then, we have \( I \)-remaining feasible set at instant 11 is
\begin{align*}
X_{k+1}' &= (H_k(I, I_1') \cup \mathcal{Y}(X_{k+1}')) \cup (H_k(I, I_2') \cup \mathcal{Y}(X_{k+1}')) \\
&\quad \cup (H_k(I, I_1) \cup \mathcal{Y}(X_{k+1}')) \cup (H_k(I, I_2') \cup \mathcal{Y}(X_{k+1}')).
\end{align*}

### 5.3 Offline Computation Algorithm

Now, we present the complete procedure for offline computation of feasible sets in Algorithm 2. Initially, for each instant \( k \in [0, T] \), we set \( X_k \) as the empty set since no \( X_k \) is computed for some \( I \in \mathbb{I}_k \) (line 1). Then for the unique element \( X_k' \) in \( X_{T+1} \), 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 \( X_k' \) according to Equation (14) by considering all possible successor sets at instant \( k+1 \) (lines 4-8).
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. Nevertheless, it is still worth mentioning 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 $x_{0:k}$ is satisfied if for any control input $u_{k:T−1}$, we have $x_{0:k} \xi_f(x_{k}, u_{k:T−1}) \models \Phi$. 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:

$$Y_k^I = \left\{ x_k \in X \mid \forall u_{k:T−1} \in \mathcal{U}^{T−k} \text{ s.t. } x_k \xi_f(x_{k}, u_{k:T−1}) \models \hat{\Phi}_k^I \right\}.$$  

The only difference between the $I$-remaining feasible set and the satisfiable set is their quantifiers “$\forall$” and “$\exists$”. 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”.

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:

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

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

$$Y_k^I = \bigcup_{I' \in \text{suc}(I,k)} (H_k(I, I') \cap \hat{\Upsilon}(X_k^{I'+1})), $$

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 [5] with the help of existing package JuliaReach [4,7,19]. Regarding the computation of one-step sets, we use the generic method in [9] with some slight modifications for the purpose of approximations. The main idea of [9] is to start with a large interval box that contains the feasible (satisfiable) sets for sure and then to iteratively divide the box in smaller boxes by tracking which boxes are still within the feasible (satisfiable) sets; the readers are referred to [9] for more details. 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 [9] 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 prefix since most of their I-remaining satisfiable sets are empty. We show that, by leveraging the model information of the dynamic 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

\[ x_{k+1} = x_k + \tau_x (T_c - x_k) + \alpha_e (T_e - x_k) u_k, \]

where state \( x_k \in \mathcal{X} = [0, 45] \) denotes the zone temperature of building at instant \( k \), control input \( u_k \in \mathcal{U} = [0, 1] \) is the ratio of the heater valve, \( \tau_x = 1 \) minute is the sampling time, \( T_h = 55^\circ C \) is the heater temperature, \( T_c = 0^\circ C \) is outside temperature, and \( \alpha_e = 0.06 \) and \( \alpha_h = 0.08 \) are the heat exchange coefficients. The model is adopted from [24].

The objective of the temperature control system is to warm the single room to specific comfortable environment between \( 20^\circ C - 25^\circ C \) in 8 minutes, and then keep the temperature in this interval from 10 to 15 minutes, which can be described by the following STL formula

\[ \Phi = F_{[0,8]x_k \in [20, 25]} \land G_{[10,15]}x_k \in [20, 25]. \]

Before starting the online monitoring process, we first compute the I-remaining feasible and satisfiable sets of STL formula \( \Phi \) 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 blue dotted lines and pure 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 was caused by one-step set computation since the computation needs to divide boxes into smaller ones. Note that satisfied sets \( Y_{\{1,2\}}^k \) for all instants and \( Y_{\{2\}}^k \) 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 \( x_{4} \notin X_{\{1,2\}} \) for the below blue line and \( x_{10} \notin X_{\{2\}} \) 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 [11, 23], 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 \( x_{14} \in Y_{\{2\}}^{14} \) 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 [28]. The system model with a sampling period of 0.5 seconds is as follows

\[
\begin{bmatrix}
1 & 0.5 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0.5 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_k \\
u_k
\end{bmatrix}
\]

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

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

\[
\Phi = \Phi_1 \land \Phi_2 \land \Phi_3,
\]

where \( \Phi_1 = F_{[5,20]}(x \in [0, 2] \land y \in [8, 10]), \Phi_2 = F_{[5,20]}(x \in [8, 10] \land y \in [8, 10]) \) and \( \Phi_3 = F_{[5,20]}(x \in [8, 10] \land y \in [0, 2]) \).

We consider two trajectories of the robot starting from two rectangle points up to instant \( k = 13.5 \) and \( k = 9.5 \) shown in Fig. 3, respectively, where the feasible sets \( X_{1.5}^{[2.3]} \) and \( X_{2.5}^{[1.2,3]} \) computed offline are also depicted. For the blue trajectory, at instant \( k = 6 \) 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 13.5, we observe that \( x_{13.5} \notin X_{1.5}^{[2.3]} \) 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 \( k = 9.5 \) and also \( x_{9.5} \notin X_{2.5}^{[1.2,3]} \). Then we can claim in advance that it will not complete

Fig. 3. Two trajectories for the double integrator system.

the task. Note that the offline results shown in the figure is 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 [46] in the form of

\[
\dot{x} = v \cos \theta, \quad \dot{y} = v \sin \theta, \quad \dot{\theta} = \omega,
\]

and we discretize it with sampling time 0.5 instants 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 \( x \in X = [0, 100] \times [0, 100] \times [-\pi, \pi] \) and \( u \in U = [-10, 10] \times [-0.3, 0.3] \) if \( k \leq 10 \) and \( U = [-3, 3] \times [-0.3, 0.3] \) 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 and then to stay in A3 (green) between instants 10 and 15. Meanwhile, the robot should always avoid the obstacles (grey) in the map. The task can be described by the following STL formula

\[
\Phi = \Phi_1 \land \Phi_2 \land \Phi_3,
\]

where \( \Phi_1 = G_{[0,15]}(y < x + 85 \land y > -x + 15 \land y < -x + 185 \land y > x - 85 \land \neg(\{x \in [40, 50] \land y \in [42.5, 47.5]\}), \Phi_2 = F_{[0,8]}((\{x \in [15, 30] \land y \in [20, 35] \land \theta \in [-1.4, -0.2]\} \lor (\{x \in [15, 30] \land y \in [65, 80] \land \theta \in [-1.4, -0.2]\}) \land \Phi_3 = G_{[10,15]}(x \in [60, 100] \land 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 \( x_5^{\text{blue}} \notin X_{5}^{[1.2,3]} \). Also, we can claim that red trajectory will satisfy the specification for sure at \( k = 13 \) since \( x_5^{\text{red}} \in Y_{13}^{(1,3)} \).
Also, the chaser must always remain with a line-of-sight
and the following nonlinear dynamic equations describe
the two-dimensional, planar motion of chaser spacecraft
on an orbital plane towards a target spacecraft
\[
\begin{align*}
\dot{x} &= v_x, \\
\dot{y} &= v_y, \\
\dot{v}_x &= n^2 x + 2nv_y + \frac{\mu}{r^2} - \frac{\mu}{r_c}(r+x) + \frac{u_x}{m_c}, \\
\dot{v}_y &= n^2 y - 2nv_x - \frac{\mu}{r_c^2} y + \frac{u_y}{m_c}, 
\end{align*}
\]
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 minutes. The parameters are \( \mu = 3.986 \times 10^{14} \times 60^2 [m^3/min^2], r = 42164 \times 10^3 [m], m_c = 500[kg], n = \sqrt{\frac{\mu}{r}} \) and \( r_c = \sqrt{(r+x)^2 + y^2} \). When the
target and the chaser spacecrafts’ separation distance
is less than 100m, the chaser needs to continue to be
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 \( 10N \) of force in any single direction in this period,
i.e., the physical constraints are \( x \in X' = [-100, 0] \times
[-100, 0] \times [0, 3.5] \times [0, 3.5] \) and \( u \in U = [0, 10] \times [0, 10] \).

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 40 minutes, i.e., \( P = \{x, y, v_x, v_y| x \in [-5, 0] \land
y \in [-5, 0] \land v_x \in [0, 2.5] \land v_y \in [0, 2.5]\} \), 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 \( L = \{(x, y) \mid (y \geq x \tan(30^\circ)) \land (-y \geq x \tan(30^\circ))\} \)
to keep a good position for docking preparation. Such a
task can be described by the following formula
\[
\Phi = \Phi_1 \land \Phi_2,
\]
where \( \Phi_1 = F_{(0,30)}(x_k \in P) \) and \( \Phi_2 = G_{[0,30]}((x, y) \in L) \).

Considering a trajectory of a spacecraft starting from a
rectangle points up to \( k = 9.5 \) minutes 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 \( x_{9.5} \notin X_{3.5}^{(1,2)} \). 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 tem-
poral 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 offline computation characterized by
the notion of feasible and satisfiable sets. The offline
computed information is used during the online moni-
toring 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 re-
sults. Note that, in this work, we only consider the STL
formula in the form of conjunction of sub-formulae in
which the temporal operator will only be applied once
for Boolean formulae. In the future, we would like to con-
sider more general formula, including nested temporal
operators, to further generalize our results.

Acknowledgement

The authors would like to thank Christian Schilling at
Aalborg University and Marcelo Forets at Universidad
Tecnológica del Uruguay, two of main contributors of
JuliaReach, for their great helps when completing our
case studies.

References

[1] M. Abate, E. Feron, and S. Coogan. Monitor-based
runtime assurance for temporal logic specifications. In IEEE
Conference on Decision and Control, pages 1997–2002, 2019.

[2] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler,
D. Ničković, and S. Sankaranarayanan. Specification-based
monitoring of cyber-physical systems: a survey on theory,
tools and applications. In Lectures on Runtime Verification,
pages 135–175, 2018.

[3] A. Bauer, M. Leucker, and C. Schallhart. Runtime
verification for LTL and TLTL. ACM Transactions on
Software Engineering and Methodology, 20(4):1–64, 2011.
[4] L. Benet, M. Forets, D. Sanders, and C. Schilling. TaylorModels.jl: Taylor models in Julia and their application to validated solutions of ODEs. In *SWIM*, 2019.

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

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

[7] S. Bogomolov, M. Forets, G. Frehse, K. Potomkin, and C. Schilling. JuliaReach: a toolbox for set-based reachability. In *ACM International Conference on Hybrid Systems: Computation and Control*, pages 39–44, 2019.

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

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

[10] N. Chan and S. Mitra. Verifying safety of an autonomous spacecraft rendezvous mission. *EPI-C Series in Computing*, 48:20–32, 2017.

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

[12] A. Devonport and M. Arcak. Data-driven reachable set computation using adaptive gaussian process classification and Monte Carlo methods. In *American Control Conference*, pages 2629–2634. IEEE, 2020.

[13] A. Dokhanchi, B. Hoxha, and G. Fainekos. On-line monitoring for temporal logic robustness. In *International Conference on Runtime Verification*, pages 231–246. Springer, 2014.

[14] A. Donzé, T. Ferrere, and O. Maler. Efficient robust monitoring for STL. In *International Conference on Computer Aided Verification*, pages 264–279, 2013.

[15] A. Donzé and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 92–106. Springer, 2010.

[16] C. Eisner, D. Fisman, J. Havlicek, Y. Lustig, A. McIsaac, and D. Campenhout. Reasoning with temporal logic on truncated paths. In *International Conference on Computer Aided Verification*, pages 27–39, 2003.

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

[18] A. Ferrando, R. Cardoso, M. Farrell, M. Luckcuck, F. Papacchini, M. Fisher, and V. Mascardi. Bridging the gap between single-and multi-model predictive runtime verification. *Formal Methods in System Design*, pages 1–33, 2022.

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

[20] B. Ghosh and E. André. Offline and online monitoring of scattered uncertain logs using uncertain linear dynamical systems. *arXiv preprint arXiv:2204.11505*, 2022.

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

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

[23] H. Ho, J. Ouaknine, and J. Worrell. Online monitoring of metric temporal logic. In *International Conference on Runtime Verification*, pages 178–192, 2014.

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

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

[26] E. Kerrigan. Robust constraint satisfaction: Invariant sets and predictive control. PhD thesis, University of Cambridge, 2001.

[27] M. Leucker. Sliding between model checking and runtime verification. In *International Conference on Runtime Verification*, pages 82–87. Springer, 2012.

[28] L. Lindemann and D. Dimarogonas. Robust motion planning employing signal temporal logic. In *American Control Conference*, pages 2950–2955. IEEE, 2017.

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

[30] L. Lindemann and D. Dimarogonas. Robust control for signal temporal logic specifications using discrete average space robustness. *Automatica*, 101:377–387, 2019.

[31] M. Ma, E. Bartocci, E. Liffland, J. 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.

[32] 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*, 20(5):1–25, 2021.

[33] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152-166, 2004.

[34] C. Mascle, D. Neider, M. Schwenger, P. Tabuada, A. Weinert, and V. Preoteasa. Predictive runtime verification of timed systems. In *International Conference on Hybrid Systems: Computation and Control*, pages 1–12, 2020.

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

[36] A. Montaz, N. Basnet, H. Abbas, and B. Bonakdarpour. Predicate monitoring in distributed cyber-physical systems. In *International Conference on Runtime Verification*, pages 3–22. Springer, 2021.

[37] S. Pinisetty, T. Jéron, S. Tripakis, Y. Falcone, H. Marchand, and V. Preoteasa. Predictive runtime verification of timed properties. *Journal of Systems and Software*, 132:355–365, 2017.

[38] X. Qin and J. Deshmukh. Clairvoyant monitoring for signal temporal logic. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 178–195. Springer, 2020.

[39] V. Raman, A. Donzé, M. Maasoumy, R. Murray, A. Sangiovanni-Vincentelli, and S. Seshia. Model predictive control with signal temporal logic specifications. In *IEEE Conference on Decision and Control*, pages 81–87, 2014.
[40] Y. Sahin, R. Quirynen, and S. Di Cairano. Autonomous vehicle decision-making and monitoring based on signal temporal logic and mixed-integer programming. In *American Control Conference*, pages 454–459. IEEE, 2020.

[41] A. Salamati, S. Soudiani, and M. Zamani. Data-driven verification of stochastic linear systems with signal temporal logic constraints. *Automatica*, 131:109781, 2021.

[42] P. Thati and G. Roșu. Monitoring algorithms for metric temporal logic specifications. *Electronic Notes in Theoretical Computer Science*, 113:145–162, 2005.

[43] M. Waga, É. André, and I. Hasuo. Model-bounded monitoring of hybrid systems. In *ACM/IEEE International Conference on Cyber-Physical Systems*, pages 21–32, 2021.

[44] H. Yoon and S. Sankaranarayanan. Predictive runtime monitoring for mobile robots using logic-based Bayesian intent inference. In *IEEE International Conference on Robotics and Automation*, pages 8565–8571. IEEE, 2021.

[45] X. Yu, W. Dong, X. Yin, and S. Li. Online monitoring of dynamic systems for signal temporal logic specifications with model information. *IEEE Conference on Decision and Control*, accepted. arXiv preprint arXiv:2203.16267, 2022.

[46] X. Yu and R. Su. Decentralized circular formation control of nonholonomic mobile robots under a directed sensor graph. *IEEE Transactions on Automatic Control*, pages 1–8, 2022.

[47] X. Zhang, M. Leucker, and W. Dong. Runtime verification with predictive semantics. In *NASA Formal Methods Symposium*, pages 418–432. Springer, 2012.

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