Model Predictive Control for Signal Temporal Logic Specifications with Time Interval Decomposition

Xinyi Yu, Chuwei Wang, Dingran Yuan, Shaoyuan Li, Xiang Yin

Abstract— **In this paper, we investigate the problem of Model Predictive Control (MPC) of dynamic systems for high-level specifications described by Signal Temporal Logic (STL) formulae. Recent works show that MPC has the great potential in handling logical tasks in reactive environments. However, existing approaches suffer from the heavy computational burden, especially for tasks with large horizons. In this work, we propose a computationally more efficient MPC framework for STL tasks based on time interval decomposition. Specifically, we still use the standard shrink horizon MPC framework with Mixed Integer Linear Programming (MILP) techniques for open-loop optimization problems. However, instead of applying MPC directly for the entire task horizon, we decompose the STL formula into several sub-tasks with disjoint time horizons, and shrinking horizon MPC is applied for each short-horizon sub-task iteratively. To guarantee the satisfaction of the entire STL formula and to ensure the recursive feasibility of the iterative process, we introduce new terminal constraints to connect each sub-task. We show how these terminal constraints can be computed by an effective inner-approximation approach. The computational efficiency of our approach is illustrated by a case study.**

I. Introduction

Decision-making under dynamic environment is one of the central problems in control of Cyber-Physical Systems (CPS). In the past years, there has been growing interest in controller synthesis for high-level complex tasks. Specifically, temporal logic, such as Linear Temporal Logic (LTL) and Signal Temporal Logic (STL), provide expressive and user-friendly tools for formal description and automated design of complex tasks involving both continuous variables and discrete logics. Signal temporal logic was firstly developed in [1] for the purpose of behavior monitoring. STL formulae are evaluated over continuous time signals. Compared with LTL, STL semantics are quantitative, and therefore, provide a measure for the degree of the satisfaction or violation in addition to the Boolean satisfaction. Recently, STL has been successfully applied to the analysis and control of many engineering CPSs [2]–[4].

In the context of control synthesis for STL specifications, one of the most widely used approaches is to encode the satisfaction of STL formulae as mixed-integer constraints [5]. Then the STL control problem can be solved by applying Mixed Integer Linear Programming (MILP) techniques together with Model Predictive Control (MPC) framework. The encoding-based approach has also been adopted in [6]–[8]. The main advantage of this approach is that the solution is complete and globally optimal. However, since one needs to introduce decision variables for each time instant, the encoding-based approach suffers from the heavy computational burden, especially for tasks with large horizons.

To mitigate the high complexity in STL control synthesis problem, several computationally efficient methods have been developed recently in the literature. The techniques include, e.g., gradient-based optimizations [9], [10], control barrier functions [11], prescribed performance control [12], [13], referenced way-points [14] and computationally tractable robustness [15]. However, compared with the encoding-based MPC approach, the above mentioned methods are usually not complete and cannot provide formal guarantees for the existence of a solution when facing complex formula. In the context of encoding-based methods, [16] proposed a more efficient encoding approach where disjunction can be encoded using a logarithmic number of binary variables and conjunction can be encoded without binary variables. In [17], the authors showed that binary variables can be further reduced by writing formulae in positive normal forms. However, these improved encoding methods still suffer from the huge computational complexity *when the task horizon increases*. It is worth mentioning [17] used MPC with shorter prediction horizon to solve the *unbounded* STL formulae and then the complexity is reduced, but for long-horizon *bounded* formulae such a efficient mechanism is not applicable and high complexity is still inevitable.

In this paper, we also focus on STL control synthesis using shrinking horizon MPC together with variable encoding techniques. Compared with existing MPC approach that applies directly to the entire task horizon, the contributions of this paper are as follows: (i) We decompose the STL formula into several sub-tasks with disjoint time horizons. Then shrinking horizon MPC is applied for each sub-task iteratively with short-horizons, which significantly reduces computational complexity of the entire synthesis process; (ii) Note that, by focusing on each sub-task only, the subsequent sub-tasks may become infeasible. In order to ensure the recursive feasibility of the entire process, we introduce terminal constraint sets between connected sub-tasks. This guarantees the satisfaction of the global STL task. Also, we provided an effective

This work was supported by the national natural Science Foundation of China (62061136004, 62173226, 61833012).

Xinyi Yu is with Thomas Lord Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA. e-mail: xinyi.yu12@usc.edu

Chuwei Wang, Dingran Yuan, Shaoyuan Li and Xiang Yin are with Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China. e-mail: {wangchuwei, geniustintin, syli, yinxiang}@sjtu.edu.cn

method for offline computation of the inner-approximation the terminal constraint sets. The computational complexity does not increase exponentially as the horizon increases; (iii) Finally, we present a case study of robot motion planning to demonstrate the efficiency of the proposed framework. We show that, compared with the full horizon MPC approach, our approach is more scalable for STL formula with long task horizon.

II. Preliminary

A. System Model

We consider a discrete-time control system of form

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

where $x_k \in \mathcal{X} \subset \mathbb{R}^n$ is the state at instant k, $u_k \in \mathcal{U} \subset \mathbb{R}^m$ is the control input at instant k, $w_k \in \mathcal{W} \subset \mathbb{R}^n$ is the external input or disturbance at instant k and $f : \mathcal{X} \times \mathcal{U} \times \mathcal{W} \to \mathcal{X}$ is a function describing the dynamic of the system. We assume that the disturbances are unknown but belong to a given compact set $\mathcal{W}$. Also, we assume that the dynamic function f is Lipschitz continuous on x, i.e., for all $x, x' \in \mathcal{X}, u \in \mathcal{U}$, there exists positive constant L such that $|f(x', u) - f(x, u)| \leq L|x' - x|$.

Suppose that the system is in state $x_k \in \mathcal{X}$ at instant $k \in \mathbb{Z}_{\geq 0}$. Then given a sequence of control inputs $\mathbf{u}_{k:T-1} = u_k u_{k+1} \cdots u_{T-1} \in \mathcal{U}^{T-k}$ and a sequence of disturbances $\mathbf{w}_{k:T-1} = w_k w_{k+1} \cdots w_{T-1} \in \mathcal{W}^{T-k}$, the solution of the system is a sequence of states $\xi_f(x_k, \mathbf{u}_{k:T-1}, \mathbf{w}_{k:T-1}) = x_{k+1} \cdots x_T \in \mathcal{X}^{T-k}$ such that $x_{i+1} = f(x_i, u_i) + w_i, \forall i = k, \cdots, T-1$, and the solution of nominal system is denoted by $\xi_f(x_k, \mathbf{u}_{k:T-1})$ similarly. Also, we denote by $\xi_f^T(x_k, \mathbf{u}_{k:T-1}, \mathbf{w}_{k:T-1}) = x_T$ the last state in the sequence.

B. Signal Temporal Logic

Given a finite sequence of states $\mathbf{x}$, we use signal temporal logic (STL) formulae [1] with bounded-time temporal operators to describe whether or not the trajectory of the system satisfies some high-level properties. Formally, the syntax of STL formulae is $\Phi ::= \top \mid \pi^\mu \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \mathbf{U}_{[a,b]} \Phi_2$, where $\top$ is the true predicate, π^μ is a 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 if $\mu(x_k) \geq 0$; otherwise it is false. Notations $\neg$ and $\wedge$ are the standard Boolean operators "negation" and "conjunction", respectively, which can further induce "disjunction" $\vee$ and "implication" $\to$. $\mathbf{U}_{[a,b]}$ is the temporal operator "*until*", where $a, b \in \mathbb{R}$.

STL formulae are evaluated on state sequences. We use notation $(\mathbf{x}, k) \models \Phi$ to denote that sequence $\mathbf{x}$ satisfies STL formula Φ at instant k. The reader is referred to [1] for more details on the semantics of STL formulae. We can also further induce temporal operators "*eventually*" $\mathbf{F}_{[a,b]}\Phi := \top \mathbf{U}_{[a,b]}\Phi$ and "*always*" $\mathbf{G}_{[a,b]}\Phi := \neg\mathbf{F}_{[a,b]}\neg\Phi$. We write $\mathbf{x} \models \Phi$ whenever $(\mathbf{x}, 0) \models \Phi$.

Given an STL formula Φ, it is well-known that the satisfaction of Φ can be completely determined only by those states within its *horizon*. Hereafter, we use notation $\Phi^{[S_\Phi, T_\Phi]}$ to emphasize that formula Φ only depends on time horizon $[S_\Phi, T_\Phi]$. With a slight abuse of notation, we also use a broader interval as its superscript, e.g., $\Phi^{[S'_\Phi, T'_\Phi]}$ with $S'_\Phi \leq S_\Phi$ and $T'_\Phi \geq T_\Phi$. In addition to the Boolean satisfaction, we can also evaluate an STL formula *quantitatively* by *space-robustness function* $\rho^\Phi_{\mathbf{x},k}$ [6].

In this paper, we consider the following restrict but expressive enough fragment 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.

C. Feasible Set of Signal Temporal Logic

In our previous works [18], [19], we propose a method to compute the so called *feasible set* of STL formula for a dynamic system without considering disturbances. Here we briefly review this concept and interested readers are referred to [19] for details.

We first rewrite Boolean formula φ in Eq. (2a) in terms of the region of states satisfying the formula. Specifically, we denote π^μ by $\mathcal{H}^\mu = \{x \in \mathcal{X} \mid \mu(x) \geq 0\}$, and we also 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 write it as $x \in \mathcal{H}^\varphi$ or simply $x \in \mathcal{H}$. Also, 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 $\forall k'' \in [k+a, k']$ $(\mathbf{x}, k'') \models \Phi_1$. Compared with $\mathbf{U}$, $\mathbf{U}'$ only requires that Φ_1 holds *from instant* a before Φ_2 holds. Note that, our setting is without loss of generality since we have $(\mathbf{x}, k) \models \Phi_1 \mathbf{U}_{[a,b]} \Phi_2 \Leftrightarrow (\mathbf{x}, k) \models (\Phi_1 \mathbf{U}'_{[a,b]} \Phi_2) \wedge (\mathbf{G}_{[0,a]} \Phi_1)$. We observe that formula Φ of form (2) can be written as

$$\Phi = \bigwedge_{i=1}^{N} \Phi_i^{[a_i, b_i]}, \tag{3}$$

where N denotes the total number of sub-formulae, and for each sub-formula $\Phi_i^{[a_i,b_i]}$, it is effective within time interval $[a_i, b_i]$ and is 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 $x \in \mathcal{H}_i^1 \mathbf{U}'_{[a_i,b_i]} x \in \mathcal{H}_i^2$. We denote by $\mathcal{I} = \{1, \cdots, N\}$ the index set of all sub-formula and by $O_i \in \{\mathbf{G}, \mathbf{F}, \mathbf{U}'\}$ the unique temporal operator in Φ_i. Also, 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.

Let $I \subseteq \mathcal{I}$ be a set of indices representing sub-formulae that have not yet been satisfied. We say I is a remaining set at 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 possible remaining sets at instant k. Then given a remaining set I at instant k, we call formula $\hat{\Phi}_k^I = \bigwedge_{i \in I \cap \mathcal{I}_k} \Phi_i^{[k, b_i]} \wedge \bigwedge_{i \in \mathcal{I}_{>k}} \Phi_i^{[a_i, b_i]}$ the I-remaining formula representing the entire task remained. The set of states from which the remaining formula can be fulfilled is called the I-remaining feasible sets.

Definition 1 (*I*-Remaining Feasible Sets): Given an STL formula Φ of form (3), a subset of indices $I \subseteq \mathcal{I}$ at time instant k, 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 sub-formula at k, i.e.,

$$X_k^I = \left\{ x_k \in \mathcal{X} \;\middle|\; \begin{array}{c} \exists\ \mathbf{u}_{k:T_\Phi - 1} \in \mathcal{U}^{T_\Phi - k} \\ \text{s.t. } x_k \xi_f(x_k, \mathbf{u}_{k:T_\Phi - 1}) \models \hat{\Phi}_k^I \end{array} \right\}. \quad (4)$$

Suppose that I is a remaining set at instant k, we denote by $\mathsf{succ}(I,k) = \{I' \subset I \mid I' \in \mathbb{I}_{k+1}\}$ the set of all possible *successor sets* of I. Intuitively, a transition from I to I' means that sub-formulae in $I \setminus I'$ have been satisfied currently. This means that at instant k the system should be in the region $H_k(I,I') = \bigcap_{i \in I \cap \mathcal{I}_k} H_i$, with

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

where $\mathsf{sat}_{\mathsf{U}}(I,I') = \{i \in I : O_i = \mathbf{U}' \wedge i \notin I'\}$. Then I-remaining feasible set can be computed by Theorem 1 [19].

Theorem 1: I-remaining feasible set X_k^I defined in Definition 1 for the time instant k can be computed as follows

$$X_k^I = \bigcup_{I' \in \mathsf{succ}(I,k)} \Big(H_k(I,I') \cap \Upsilon(X_{k+1}^{I'}) \Big), \quad (5)$$

where $\Upsilon(\cdot)$ is the one-step set defined by: for any $\mathcal{S} \subseteq \mathcal{X}$, $\Upsilon(\mathcal{S}) = \{x \in \mathcal{X} \mid \exists u \in \mathcal{U} \text{ s.t. } f(x,u) \in \mathcal{S}\}$.

III. Model Prediction Control for STL

Our objective is to synthesize a feedback control strategy such that the sequence generated by the closed-loop system satisfies the desired STL formula under all possible disturbances. Furthermore, we want to minimize control effort while maximizing the control performance.

Formally, given state x_k at instant k and input sequence $\mathbf{u}_{k:T-1} = u_k u_{k+1} \cdots u_{T-1}$, we consider a generic cost function $J : \mathcal{X} \times \mathcal{U}^{T-k} \to \mathbb{R}$. For example, J can be defined as the nominal satisfaction degree of the STL formula without disturbance or the total energy of the control inputs.

Our approach for solving the STL control synthesis problem follows the basic framework of model predictive control. Specifically, at each instant, we solve a finite-horizon optimization problem to compute a finite open-loop sequence of inputs such that the cost function is minimized subject to the constraints on both the system's dynamic and the STL formula. Note that we only apply the first input in the computed sequence to the system. Then at the next instant, we recompute the input sequence based on the actually measured current-state and repeat until the terminal instant. Formally, the optimization problem for each instant is formulated as follows.

Problem 1 (Robust STL Optimization Problem): Given system in the form of Equation (1), an STL formula Φ, a cost function J, the current state $x_k \in \mathcal{X}$ at instant k, previous state sequence $\mathbf{x}_{0:k-1}$ and some constant T, find an optimal input sequence $\mathbf{u}^*_{k:T-1}$ that minimizes the cost function subject to constraints on the system's dynamic and the temporal logic requirement. Formally, we have the following optimization problem

$$\underset{\mathbf{u}_{k:T-1}}{\text{minimize}} \quad J(x_k, \mathbf{u}_{k:T-1}) \quad (6a)$$

subject to

$$\forall \mathbf{w}_{k:T-1} \in \mathcal{W}^{T\text{-}k} : \mathbf{x}_{0:k} \xi_f(x_k, \mathbf{u}_{k:T\text{-}1}, \mathbf{w}_{k:T\text{-}1}) \models \Phi, \quad (6b)$$

$$u_k, u_{k+1}, \cdots, u_{T-1} \in \mathcal{U} \quad (6c)$$

Remark 1: Effective approaches have been proposed in the literature for solving the above optimization problem. To handle the STL satisfaction constraint, a basic approach is proposed in [5] by encoding state sequence as well as the satisfaction of the formula using binary variables. Then the logical-constrained optimization problem is converted to a Mixed Integer Linear Program (MILP). For the purpose of robust satisfaction under disturbances, based on the MILP-based approach in [5], [20] further purposes a counterexample-guided inductive synthesis (CEGIS) scheme that finds a robust optimal solution iteratively. There are also some other methods to solve this robust optimization problem, e.g., [7], [21]. In this work, we will adopt the CEGIS-based approach in [20] for solving Problem 1 and its variant. Note that other approaches aforementioned can also be adopted in principle.

In the MPC framework, only the first control input u_k^* in the optimal input sequence $\mathbf{u}^*_{k:T-1}$ computed will be applied to the system. Depending on the actual disturbance w_k occurs, the controller will measure the new state x_{k+1} at the next instant and recompute the optimal sequence based on Problem 1. This procedure is formally provided in Algorithm 1, which is referred to as the *shrinking horizon MPC* because the optimization horizon of Problem 1 is shrinking from T_Φ to 1 as time instant k increases. Here, we assume that Problem 1 is feasible at the initial instant $k = 0$ for the initial state x_0; otherwise, the set of feasible initial states can be calculated by the method that will be introduced in Section V.

Algorithm 1: Shrinking Horizon MPC for STL

Input: STL formula Φ of form (7), dynamic system model of form (1) and cost function J
Output: Control input u_k at each instant k.

```
1 $T \leftarrow T_\Phi$ and $k \leftarrow 0$
2 while $k < T$ do
3     measure current state $x_k$
4     solve Problem 1 based on $\mathbf{x}_{0:k}$, $T$ and $\Phi$ and
        obtain optimal inputs $\mathbf{u}^*_{k:T-1} = u^*_k u^*_{k+1} \cdots u^*_{T-1}$
5     apply control input $u^*_k$
6     $k \leftarrow k + 1$
```

The shrinking horizon MPC provides an effective framework for solving the reactive control synthesis problem for both STL specification and performance optimization. However, the complexity for solving Problem 1 at each instant grow exponentially as the prediction horizon increases. Therefore, the main computational challenging for shrinking horizon MPC is in the first few instants since we need to predict almost the entire horizon the formula, which may make this approach computationally infeasible for STL tasks with large horizons.

IV. Time Interval Decomposition Framework

A. Time Interval Decomposition

As we discussed above, the main computational challenging in shrinking horizon MPC is the predication horizon of the task. In this paper, we still use shrinking horizon MPC as the basis. However, we propose a new time interval decomposition framework, which leverages structural properties of STL formulae in terms of time interval decomposition to further reduce the computational complexity. Specifically, we assume that the STL formula of interest can be further divided into a set of disjoint time intervals. Then we solve the shrinking horizon MPC problem for each sub-task with smaller time horizon. Finally, by putting each time intervals together "correctly", we solve the entire STL control synthesis problem.

To motivate our approach, let us consider the STL formula $\Phi = (\mathbf{G}_{[0,100]}\pi^{\mu_1}) \wedge (\mathbf{F}_{[21,50]}\pi^{\mu_2}) \wedge (\mathbf{F}_{[81,100]}\pi^{\mu_3})$. For the above formula, the time horizon is $T_\Phi = 100$. Therefore, using the standard shrinking horizon MPC, we need to compute an optimization problem with predication horizon 100 initially, which is computationally very challenging. However, we also note that the above formula can be written equivalently as a conjunction of four sub-tasks with disjoint time intervals $\Phi = \Phi_1^{[0,20]} \wedge \Phi_2^{[21,50]} \wedge \Phi_3^{[51,80]} \wedge \Phi_4^{[81,100]}$, where $\Phi_1 = \mathbf{G}_{[0,20]}\pi^{\mu_1}$, $\Phi_2 = (\mathbf{G}_{[21,50]}\pi^{\mu_1}) \wedge (\mathbf{F}_{[21,50]}\pi^{\mu_2})$, $\Phi_3 = \mathbf{G}_{[51,80]}\pi^{\mu_1}$ and $\Phi_4 = (\mathbf{G}_{[81,100]}\pi^{\mu_1}) \wedge (\mathbf{F}_{[81,100]}\pi^{\mu_3})$. As an alternative, we can enforce the satisfaction of the entire STL formula Φ by enforcing the satisfaction of each sub-task Φ_i. For example, initially, we enforce Φ_1 using shrinking horizon MPC by setting the state terminal instant to $T_{\Phi_1} = 20$ (control input terminal instant is 19). Then at instant $k = 20$, we start to enforce Φ_2 using shrinking horizon MPC by setting the state terminal instant to $T_{\Phi_2} = 50$, and so forth. Therefore, we solve four MPC problems with prediction horizons at most 30 instants, which is much more easier than solving a single MPC problem with predication horizon at most 100.

To formalize the above motivation, we assume that the STL formula considered can be written as the conjunction of a set of sub-tasks whose effective horizons are disjoint, i.e., Φ is of the following form

$$\Phi^{[S_\Phi,T_\Phi]} = \Phi_1^{[S_1,T_1]} \wedge \Phi_2^{[S_2,T_2]} \wedge \cdots \wedge \Phi_N^{[S_N,T_N]} \tag{7}$$

where $S_1 = S_\Phi, T_N = T_\Phi$ and for each $i = 1, \cdots, N-1$, we have $T_i \leq S_{i+1}$. To distinguish with the index in equation (3), we say the decomposed part $\Phi_i^{[S_i,T_i]}$ here as *sub-task* and say $\Phi_i^{[a_i,b_i]}$ in equation (3) as *sub-formula*.

Remark 2: The above assumption of time interval decomposition is without loss of generality since we can always take $N = 1$. But no computational reduction will be gained for this case. However, in many practical examples, different requirements in the global task are natural time disjoint. For this scenario, one can decompose the whole task into more than one sub-task.

B. Robust Optimizations with Terminal Constraints

As we mentioned above, at each instant k, suppose that $k \in [T_{i-1}, T_i - 1]$, our approach is to enforce the ith sub-task Φ_i by solving the robust optimization problem with predication horizon from k to T_i as Problem 1. Since each sub-task Φ_i is satisfied within its effective horizon and the entire formula is their conjunction, then the global task Φ is satisfied.

However, the main issue of this approach is that how we can guarantee the *recursive feasibility* during the entire control process. More specifically, for $k \in [T_{i-1}, T_i - 1]$, we consider the optimization problem for Φ_i only up to T_i. This, however, may lead to an optimal solution ending up with a state from which some subsequent sub-tasks $\Phi_j, j > i$ cannot be satisfied. Therefore, for each sub-task period, in addition to the constraints in Problem 1, we also need to take the feasibility of all subsequent sub-tasks into consideration.

To this end, we define $\mathcal{T}_i$ as the set of states from which the subsequent sub-tasks $\bigwedge_{i<j\leq N} \Phi_j$ are feasible in the sense that no matter what the disturbance sequence is, there exists at least one control input sequence such that all the subsequent STL sub-tasks can be satisfied. Formally, for $i = 1, \cdots, N-1$, we define

$$\mathcal{T}_i = \left\{ x_{T_i} \in \mathcal{X} \,\middle|\, \begin{array}{l} \exists \mathbf{u}_{T_i:T_\Phi\text{-}1} \in \mathcal{U}^{T_\Phi\text{-}T_i}, \forall \mathbf{w}_{T_i:T_\Phi\text{-}1} \in \mathcal{W}^{T_\Phi\text{-}T_i}, \\ \text{s.t. } \xi_f(x_{T_i}, \mathbf{u}_{T_i:T_\Phi\text{-}1}, \mathbf{w}_{T_i:T_\Phi\text{-}1}) \models \bigwedge_{i<j\leq N} \Phi_j \end{array} \right\}. \tag{8}$$

Also, we define $\mathcal{T}_N = \mathcal{X}$. Later in Section V, we will discuss in detail how to compute terminal sets $\mathcal{T}_i$. Now, in the context of shrinking horizon MPC, set $\mathcal{T}_i$ can be considered as the *terminal constraint* in the optimization problem for sub-task Φ_i. Therefore, Problem 1 is further modified as the following optimization problem with terminal constraint.

Problem 2: (Robust STL Optimization Problem with Terminal Constraint). Given system in the form of Equation (1), an STL formula Φ, a cost function J, the current state $x_k \in \mathcal{X}$ at instant k, time horizon $[S, T]$, previously state sequence $\mathbf{x}_{S:k-1}$, and a terminal set $\mathcal{T}$, find an optimal input sequence $\mathbf{u}^*_{k:T-1}$ that minimizes the cost function subject to constraints on the system's dynamic, the temporal logic requirement and the terminal constraint. Formally, we have the following optimization problem

$$\underset{\mathbf{u}_{k:T-1}}{\text{minimize}} \quad J(x_k, \mathbf{u}_{k:T-1}) \tag{9a}$$

subject to

$$\forall \mathbf{w}_{k:T-1} \in \mathcal{W}^{T\text{-}k} : \mathbf{x}_{S:k}\xi_f(x_k, \mathbf{u}_{k:T\text{-}1}, \mathbf{w}_{k:T\text{-}1}) \models \Phi, \tag{9b}$$

$$\forall \mathbf{w}_{k:T-1} \in \mathcal{W}^{T\text{-}k} : \xi_f^T(x_k, \mathbf{u}_{k:T\text{-}1}, \mathbf{w}_{k:T\text{-}1}) \in \mathcal{T}, \tag{9c}$$

$$u_k, u_{k+1}, \cdots, u_{T-1} \in \mathcal{U}. \tag{9d}$$

Remark 3: Similar to Problem 1, the CEGIS-based approach can be also applied to solve Problem 2 with the terminal constraint. Basically, CEGIS uses inductive counterexamples of the disturbance sequence to deal with *forall* constraint to compute the optimal solution. Therefore, the terminal constraint (9c) added does affect the solution process. In our algorithm implementation, we still adopt the CEGIS-based method.

C. Overall Framework

Algorithm 2: Shrinking Horizon MPC for STL with Time Interval Decomposition

```
Input: STL formula $\Phi$ of form (7), dynamic system
       model of form (1) and cost function $J$
Output: control input $u_k$ at each instant $k$.
1  for each $i = 1, \cdots, N-1$, compute $\mathcal{T}_i$
2  $i \leftarrow 1$, $k \leftarrow 0$ and $T_0 = 0$
3  while $i \leq N$ do
4      $\Phi \leftarrow \Phi_i, S \leftarrow T_{i-1}, T \leftarrow T_i$
5      while $k < T$ do
6          measure current state $x_k$
7          solve Problem 2 based on $\mathbf{x}_{S:k}$, $T$, $\Phi$ and $\mathcal{T}_i$
             and obtain optimal inputs
             $\mathbf{u}^*_{k:T\text{-}1} = u^*_k u^*_{k+1} \cdots u^*_{T\text{-}1}$
8          apply control input $u^*_k$
9          $k \leftarrow k + 1$
10     $i \leftarrow i + 1$
```

The overall time interval decomposition framework is formalized by Algorithm 2. Specifically, line 1 constructs the terminal constraints in an offline manner for each online optimization stage. The computation will be detailed in the next section. Lines 2-10 aim at synthesizing control strategy for each sub-task iteratively. For each specific sub-task (lines 5-9), we use the shrinking horizon model predictive control and apply the first element of the control input sequence to the system which is similar to Algorithm 1. Note that line 1 is executed offline and the remaining procedures are computed online to resist uncertain disturbances.

The following result shows that the completeness and correctness of Algorithm 2.

Theorem 2: Given dynamic system model (1), STL formula Φ, cost function J and initial state x_0, if we can obtain a control input sequence $\mathbf{u}^*_{0:T_1-1}$ at instant $k=0$ by Algorithm 2, then Algorithm 2 is feasible for all the time instants and the solution with returned $\mathbf{u}^* = [u^*_0, \cdots, u^*_{T_\Phi-1}]$ satisfies the STL formula Φ.

Proof: The existence of an initial control input sequence $\mathbf{u}^*_{0:T_1-1} = u^*_0 u^*_1 \cdots u^*_{T_1-1}$ implies that for all possible disturbance sequences $\mathbf{w}_{0:T_1-1}$, the optimization problem is feasible. At the next time step $k=1$, there exists at least one sub-optimal input sequence $\mathbf{u}_{1:T_1-1} = u^*_{1|0} \cdots u^*_{T_1-1|0}$ obtained last time step that can resist all possible disturbances, i.e., the robust problem at instant $k=1$ is feasible. Then we can prove it recursively in the first stage. For the second stage, we have that, from state x_{T_1}, there exists a control sequence, e.g., $\bar{\mathbf{u}}_{T_1:T_\Phi-1} \in \mathcal{U}^{T_\Phi - T_1}$, such that $\mathbf{x}_{T_1+1:T_\Phi} \models \bigwedge_{1<i\leq N} \Phi_i$ by the definition of terminal constraint (8) since state x_{T_1} is in the terminal set $\mathcal{T}_1$ by constraint (9c) at the first stage. As a result, at time T_1, the first time in the second stage, when solving the optimization problem (9), we can at least find a feasible control input sequence $\bar{\mathbf{u}}^*_{T_1:T_2-1}$ to satisfy (9b), (9c) and (9d). Then, the same as stage 1, the problem is feasible in the second stage. Analogously, we can prove the recursive feasibility in the remaining stages. Furthermore, at the last time step of each stage, constraint (9b) ensures that the state sequence $\mathbf{x}_{T_{i-1}+1:T_i}$ of the current stage i meets the requirements of sub-task $\Phi_i, i \in \{1, \cdots, N\}$. Therefore, STL formula $\Phi = \Phi_1 \wedge \cdots \wedge \Phi_N$ is also satisfied. ■

Remark 4: Let us discuss the computational advantage for the proposed algorithm. First of all, the complexity for computing terminal constraints in (9c) largely depends on the system model and STL formula. Nevertheless, this computation is preformed fully offline and will not be the bottleneck of the online computations. Regarding the online computation, in general, solving Problem 2 is exponential in the length of the prediction horizon. Here, since the time horizon is decomposed, the complexity for online optimization is significantly reduced. The finer the formula can be decomposed, the more complex reduction one can obtain.

Remark 5: Finally, we remark that, if terminal sets $\mathcal{T}_i, i \in \{1, \cdots, N\}$ can be computed precisely, then our framework is both sound and as complete as classic MILP method with the consideration of disturbance. In practice, one may need to compute the inner-approximations of terminal sets. In this case, our framework is sound but not complete since the terminal sets are smaller than what are needed.

V. Computation of Terminal Constraints

In this section, we provide details for the computations of terminal sets as defined in Equation (8), which are essentially feasible set of subsequent sub-tasks taking disturbances into account. A direct approach to compute this set is to use a branch-and-bound algorithm to search for the terminal set as the framework in [22]. However, using this method, at each branch, we need to determine if *there exists* control input sequence $\mathbf{u}_{T_i:T_\Phi-1}$ such that *for all* disturbance sequence $\mathbf{w}_{T_i:T_\Phi-1}$ the subsequent sub-tasks can be satisfied, which is very computational challenging.

Here, instead of computing $\mathcal{T}_i$ explicitly, we seek to compute its inner-approximation $\hat{\mathcal{T}}_i \subseteq \mathcal{T}_i$. In terms of the MPC problem, the satisfaction of $\hat{\mathcal{T}}_i$ implies the satisfaction of $\mathcal{T}_i$. Although considering the inner-approximations of the terminal sets is a bit more conservative, the computation can be done much more efficiently.

Specifically, we define the following set $\hat{\mathcal{T}}_i$ to approximate terminal set $\mathcal{T}_i$

$$\hat{\mathcal{T}}_i = \tag{10}$$
$$\left\{ x_{T_i} \in \mathcal{X} \;\middle|\; \begin{array}{c} \exists u_{T_i} \forall w_{T_i} \in \mathcal{W}\ \exists u_{T_i+1} \forall w_{T_i+1} \in \bigoplus_{i=0}^{1} L^i \mathcal{W} \\ \cdots \exists u_{T_\Phi\text{-}1} \forall w_{T_\Phi\text{-}1} \in \bigoplus_{i=0}^{T_\Phi - T_i - 1} L^i \mathcal{W} \\ \text{s.t. } \xi_f(x_{T_i}, \mathbf{u}_{T_i:T_\Phi\text{-}1}, \mathbf{w}_{T_i:T_\Phi\text{-}1}) \models \bigwedge_{i<j\leq N} \Phi_j \end{array} \right\}.$$

The above defined set $\hat{\mathcal{T}}_i$ differs from the original terminal set $\mathcal{T}_i$ in two folds: (i) in the definition of $\mathcal{T}_i$, we require the existence of an open-loop control input sequence robust to all possible disturbance sequences. However, in the definition of $\hat{\mathcal{T}}_i$, the control inputs can be determined after observing each specific states (or disturbances). Therefore, there are alternations between the existential quantifiers and the universal quantifiers. This allows us to compute the set inductively in a backward manner; (ii) Second, the disturbance set grows at each instant according to the Lipschitz constant. Specifically, set $\bigoplus_{i=0}^{k} L^i\mathcal{W}$ upper bounds possible disturbances of the system at instant $T_i + k$. Clearly, this set is easy to compute but larger than the actual disturbance to the system. Therefore, the corresponding terminal set is smaller than the actual one.

The following result formally shows that $\hat{\mathcal{T}}_i$ is indeed an inner-approximation of $\mathcal{T}_i$.

Proposition 1: For each terminal set $\mathcal{T}_i$, we have $\hat{\mathcal{T}}_i \subseteq \mathcal{T}_i$.

Proof: The proof can be found in Appendix A in [23].

Then, we present the computation methods of $\hat{\mathcal{T}}_i$ for each $i\in\{1,\cdots,N-1\}$ as follows with the help of I-remaining *robust* feasible sets whose notion is similar to I-remaining feasible sets in Definition 1.

Definition 2 (I-Remaining Robust Feasible Sets): Given an STL formula Φ of form (3), a subset of indices $I \subseteq \mathcal{I}$, a starting instant s and current instant k with $s \le k$, then starting from instant s, I-remaining robust feasible set at k, denoted by $\hat{X}^I_{s,k} \subseteq \mathcal{X}$, is the set of states as follows,

$$\hat{X}^I_{s,k}=\left\{ x_k\in\mathcal{X} \;\middle|\; \begin{array}{c} \exists u_k \forall w_k \in \bigoplus_{i=0}^{k\text{-}s} L^i\mathcal{W}\ \cdots \\ \exists u_{T_\Phi\text{-}1} \forall w_{T_\Phi\text{-}1} \in \bigoplus_{i=0}^{T_\Phi\text{-}1\text{-}s} L^i\mathcal{W} \\ \text{s.t. } x_k\xi_f(x_k,\mathbf{u}_{k:T_\Phi-1},\mathbf{w}_{k:T_\Phi\text{-}1}) \models \hat{\Phi}^I_k \end{array} \right\}. \quad (11)$$

In this above definition, parameter s represents the time instant when a sub-task starts. This information is used in order to determine how many times the disturbance set should be magnified. Then parameter k is still the current instant from which the control sequence is applied. Then by definition, when $s = k = T_i$, the I-remaining robust feasible set is indeed the inner-approximation of terminal set that we want, i.e.,

$$\hat{\mathcal{T}}_i = \hat{X}^I_{T_i,T_i},$$

where I is the set of sub-formulae index corresponding to the set of sub-task index $\{i+1,\cdots,N\}$. For the computation of $\hat{X}^I_{s,k}$, we can apply a similar approach for computing X^I_k in Equation (5). Specifically, for each fixed s, we use $\hat{X}^{I'}_{s,k+1}$ to compute $\hat{X}^I_{s,k}$ in a backwards manner until $k = s$. The computation is summarized by the following theorem.

Theorem 3: For each time instant s, the I-remaining robust feasible set $\hat{X}^I_{s,k}$ defined in Definition 2 for time instant k can be computed as follows

$$\hat{X}^I_{s,k} = \bigcup_{I'\in\mathsf{succ}(I,k)} \Big(H_k(I,I') \cap \Upsilon_r(\hat{X}^{I'}_{s,k+1}, h(\mathcal{W},s)) \Big), \quad (12)$$

where $h(\mathcal{W},s) = \bigoplus_{i=0}^{k-s} L^i\mathcal{W}$ and $\Upsilon_r(\cdot,\cdot)$ is the robust one-step set defined by: for any $\mathcal{S} \subseteq \mathcal{X}$, we have $\Upsilon_r(\mathcal{S},\mathbf{W}) = \{x \in \mathcal{X} \mid \exists u \in \mathcal{U}\ \forall w \in \mathbf{W} \text{ s.t. } f(x,u)+w \in \mathcal{S}\}$.

Proof: The proof can be found in Appendix B in [23].

More implementation details of Theorem 3 can be found in Section 5.4 in [19] which is omitted here.

VI. Case Study

In this section, we apply the proposed time interval decomposition framework to a case study of planar motion of a single robot with double integrator dynamics. Online simulations are conducted in Python 3 and we use Gurobi [24] to solve the optimization problem. The terminal constraints are computed offline in Julia with the help of existing packages JuliaReach [25], [26]. All simulations are carried out by a laptop computer with i7-10510U CPU and 16 GB of RAM. Our codes are available at `https://github.com/Xinyi-Yu/MPC4STL-TID`, where more details can be found.

System Model: The model with a sampling period of 0.5 seconds is $x_{k+1} = Ax_k + Bu_k + w_k$, where $A = \begin{bmatrix} 1 & 0.5 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 0.125 & 0 \\ 0.5 & 0 \\ 0 & 0.125 \\ 0 & 0.5 \end{bmatrix}$ and 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 $x \in \mathcal{X} = [0,10] \times [-2.5,2.5] \times [0,10] \times [-2.5,2.5]$, and $u \in \mathcal{U} = [-3,3]^2$. The disturbance $w_k = [w_x\ w_{vx}\ w_y\ w_{vy}]^T$ is assumed to be bounded by a compact set $[-0.01,0.01]^4$.

Planning Objectives: We assume that the initial state of the robot is $x_0 = [3,0,8,0]$ shown as the red point in Fig. 1. The control objective of the robot is to visit region A_1 at least once between instants 0 to 6 (from 0s to 3s), always stay at region A_2 between instants 14 to 15 (from 7s to 7.5s) and finally reach region A_3 at least once between instants 22 to 25 (from 11s to 12.5s). Such an objective can be specified by STL formula, $\Phi = \mathbf{F}_{[0,6]}A_1 \wedge \mathbf{G}_{[14,15]}A_2 \wedge \mathbf{F}_{[22,25]}A_3$, where $A_1 = (x \in [7.5,10]) \wedge (y \in [7.5,10])$, $A_2 = (x \in [0,3]) \wedge (y \in [0,3])$ and $A_3 = (x \in [7.5,10]) \wedge (y \in [0,2.5])$.

Simulation Results: In our framework, we can decompose the formula Φ into $\Phi_1^{[0,12]}$ and $\Phi_2^{[13,25]}$ as $\Phi_1^{[0,12]} = \mathbf{F}_{[0,6]}A_1$, $\Phi_2^{[13,25]} = \mathbf{G}_{[14,15]}A_2 \wedge \mathbf{F}_{[22,25]}A_3$. For this case, there is only one terminal set at instant 12 needs to be computed, whose inner approximation is shown as the red shaded part in Fig. 1. The online executed trajectory is also shown in Fig. 1. Specifically, the simulation result of the first stage is represented by blue dots and solid line; the simulation result of the second stage is printed by black dots and dashed line. Clearly, the black rectangle dot x_{12} falls into the approximated terminal set $\hat{X}^{\{2,3\}}_{12,12}$ in order to ensure the continuation of the task. Note that the offline results shown in the figure is the projection to the first and third dimensions but the terminal set is still constrained in the complete 4-dimensional state space.

TABLE I
Comparison of simulation results

Algorithms	**Computation time**				**Cost**
	k=0 (s)	k=1 (s)	...	total (s)	
Our method	0.2741	0.0625	...	1.487	-1.2476
General method	3.4345	5.1492	...	21.911	-1.2499

In terms of computation time and performance, we compared our method with the standard MILP method in [5].

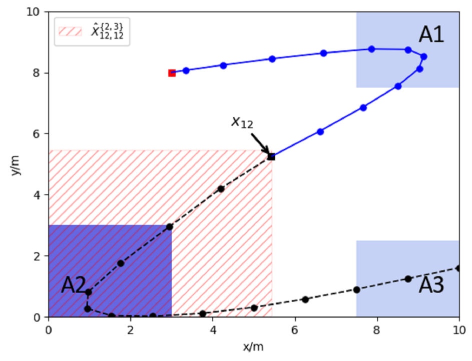

Fig. 1. The result of simulation trace.

The comparison result is shown in Table 1, where the cost function we used is $J(x_k, \mathbf{u}_{k:T-1}) = -\rho^{\Phi}_{\mathbf{x}_{0:k}\xi_f(x_k,\mathbf{u}_{k:T-1}),k} + 0.67\times 10^{-7} \times \Sigma_{i=k}^{T-1}(u_x^2+u_y^2)$. Note that all results in Table 1 are averaged results among 50 simulations.[1] Compared with the standard MILP approach, one can see that the cost of the trajectory using our approach is slightly higher. This is because we do not solve the optimization problem globally. However, using our approach, the computation time for the online optimization problem is significantly smaller than that of the MILP approach, especially for the initial stages. In particular, the MILP approach cannot keep pace with the sampling time 0.5s of the system. However, our approach can ensure online control for this system.

VII. Conclusion

In this work, we proposed a new framework for model predictive control of STL specifications. We showed that, by effectively computing the terminal set of each subsequent sub-task, the long horizon MPC problem can be decomposed into a set of MPC problems with shorter horizon. In this work, we assume that the STL formula can be naturally decomposed into several sub-tasks with disjoint time intervals. In the future, we aim to relax this assumption by tackling the case with any receding prediction horizon.

References

[1] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pp. 152–166, 2004.

[2] L. Lindemann and D. Dimarogonas, "Robust motion planning employing signal temporal logic," in *American Control Conference*, pp. 2950–2955, 2017.

[3] N. Mehr, D. Sadigh, R. Horowitz, S. Sastry, and S. Seshia, "Stochastic predictive freeway ramp metering from signal temporal logic specifications," in *American Control Conference*, pp. 4884–4889, 2017.

[4] M. Ma, E. Bartocci, E. Lifland, J. Stankovic, and L. Feng, "SaSTL: Spatial aggregation signal temporal logic for runtime monitoring in smart cities," in *ACM/IEEE 11th International Conference on Cyber-Physical Systems*, pp. 51–62, 2020.

[1]Here, we use CEGIS to find a control input sequence which can resist all possible disturbances. The efficiency of this method relies on the given initial disturbance sequences. Therefore, we use average time of multiple simulations to better justify the computation time and performance.

[5] 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*, pp. 81–87, 2014.

[6] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Formal Modeling and Analysis of Timed Systems*, pp. 92–106, 2010.

[7] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *Annual Allerton Conference on Communication, Control, and Computing*, pp. 772–779, 2015.

[8] S. Farahani, R. Majumdar, V. Prabhu, and S. Soudjani, "Shrinking horizon model predictive control with chance-constrained signal temporal logic specifications," in *American Control Conference*, pp. 1740–1746, 2017.

[9] N. Mehdipour, C. Vasile, and C. Belta, "Arithmetic-geometric mean robustness for control from signal temporal logic specifications," in *American Control Conference*, pp. 1690–1695, 2019.

[10] Y. Gilpin, V. Kurtz, and H. Lin, "A smooth robustness measure of signal temporal logic for symbolic control," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 241–246, 2020.

[11] L. Lindemann and D. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE control systems letters*, vol. 3, no. 1, pp. 96–101, 2018.

[12] L. Lindemann, C. Verginis, and D. Dimarogonas, "Prescribed performance control for signal temporal logic specifications," in *IEEE Conference on Decision and Control*, pp. 2997–3002, 2017.

[13] P. Varnai and D. Dimarogonas, "Prescribed performance control guided policy improvement for satisfying signal temporal logic tasks," in *American Control Conference*, pp. 286–291, 2019.

[14] D. Sun, J. Chen, S. Mitra, and C. Fan, "Multi-agent motion planning from signal temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3451–3458, 2022.

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

[16] V. Kurtz and H. Lin, "Mixed-integer programming for signal temporal logic with fewer binary variables," *IEEE Control Systems Letters*, vol. 6, pp. 2635–2640, 2022.

[17] S. Sadraddini and C. Belta, "Formal synthesis of control strategies for positive monotone systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 2, pp. 480–495, 2018.

[18] X. Yu, W. Dong, X. Yin, and S. Li, "Online monitoring of dynamic systems for signal temporal logic specifications with model information," in *IEEE Conference on Decision and Control*, pp. 1553–1559, 2022.

[19] X. Yu, W. Dong, X. Yin, and S. Li, "Model predictive monitoring of dynamic systems for signal temporal logic specifications," *arXiv preprint arXiv:2209.12493*, 2022.

[20] V. Raman, A. Donzé, D. Sadigh, R. Murray, and S. Seshia, "Reactive synthesis from signal temporal logic specifications," in *International conference on hybrid systems: Computation and control*, pp. 239–248, 2015.

[21] S. Farahani, V. Raman, and R. Murray, "Robust model predictive control for signal temporal logic synthesis," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 323–328, 2015.

[22] J. Bravo, D. Limón, T. Alamo, and E. Camacho, "On the computation of invariant sets for constrained nonlinear systems: An interval arithmetic approach," *Automatica*, vol. 41, no. 9, pp. 1583–1589, 2005.

[23] X. Yu, C. Wang, D. Yuan, S. Li, and X. Yin, "Model predictive control for signal temporal logic specifications with time interval decomposition," *arXiv preprint arXiv:2211.08031*, 2022.

[24] I. G. Optimization *et al.*, "Gurobi optimizer reference manual, 2018," *URL http://www. gurobi. com*, 2018.

[25] M. Forets and C. Schilling, "LazySets.jl: Scalable Symbolic-Numeric Set Computations," *Proceedings of the JuliaCon Conferences*, vol. 1, no. 1, p. 11, 2021.

[26] 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*, pp. 39–44, 2019.