2024 IEEE 20th International Conference on Automation
Science and Engineering (CASE)
August 28 - September 1, 2024. Bari, Italy

2024 IEEE 20th International Conference on Automation Science and Engineering (CASE) | 979-8-3503-5851-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/CASE59546.2024.10711435

Self-Triggered Model Predictive Control for Signal Temporal Logic Tasks

Junyue Huang, Chuwei Wang, Xinyi Yu, Xiang Yin

***Abstract*— In this paper, we investigate the reactive control synthesis problem for discrete-time dynamic systems under signal temporal logic (STL) specifications. Our focus is on addressing this problem within the model predictive control (MPC) framework, which involves iteratively solving optimization problems at each time instant. Traditional MPC controllers for STL tasks necessitate sampling the system state at every time instant. To mitigate sensing costs and conserve communication bandwidth during sensor measurement transmission, we introduce a novel concept termed self-triggered MPC for STL tasks. Our proposed approach aims to reduce the overall sampling rate of the system, resulting in considerable energy savings. We provide case studies to demonstrate the effectiveness of the proposed algorithm.**

I. Introduction

Autonomous systems have become one of the key ingredients in our society, with applications ranging from warehouses, healthcare facilities, to manufacturing plants. Particularly, by equipping with smart sensing modules and flexible manipulation capabilities, these systems can navigate intricate environments and perform complex tasks. Decision-making and task planning are central challenges in the development of autonomous systems, as they play a crucial role in ensuring the efficiency, safety, and adaptability of these systems in various domains.

Recently, there has been a growing interest in applying *formal methods* to decision-making in autonomous systems, especially for high-level tasks; see, e.g., recent surveys and references therein [1]–[3]. Particularly, Signal Temporal Logic (STL) is one such formalism that offers a comprehensive yet user-friendly approach for specifying spatio-temporal requirements over real-valued signals. It was initially developed in [4] as a robust formalism that extends temporal logic over Boolean domain into the continuous domain. Recently, STL has been successfully applied to the analysis and control of many engineering systems including, e.g., mobile robots [5]–[8], intelligent transportation systems [9], [10] and smart buildings [11].

In the context of control synthesis for STL tasks, several effective methodologies have been developed, which can generally be categorized into control barrier function (CBF)-based approach [12]–[14], learning-based approach [15]–[19], and optimization-based approach [20]–[24]. The optimization-based approach, in particular, encodes the satisfaction of STL as Mixed Integer Linear Programming (MILP) constraints, providing a sound and complete solution to the control synthesis problem. This approach allows for the integration of the optimization-based framework within the receding horizon control framework. In this framework, the optimization problem for the open-loop trajectory is solved iteratively at each time instant, and only the first control input is applied. This integration essentially leads to model predictive control (MPC) for STL tasks, which provides a reactive control policy capable of dealing with dynamic environments [20], [25]–[29].

It is worth noting that in the aforementioned works on MPC for STL tasks, there is a requirement to recompute the open-loop optimization problem at each time instant using a new state measurement as feedback information. However, this approach can be computationally costly and energy inefficient. Specifically, maintaining constant state measurement requires deploying sensors continuously, leading to high energy or bandwidth consumption. In many cases, frequent re-computation of the plan may be unnecessary. This strategy can significantly save both energy consumption in sensing and computational resources.

In this paper, we address the STL control synthesis problem using model predictive control. However, unlike existing approaches that necessitate state information acquisition for each time instant, we propose a more efficient approach known as self-triggered model predictive control. This approach automatically determines the next time instant for state sampling and re-computation of the optimization problem. The contributions of this paper can be outlined as follows. First, we present the framework of self-triggered model predictive control for signal temporal logic specifications. Then we show that our algorithm can ensure the robustness and can find the largest time interval to the next observation-prediction operation within the predefined bound. Finally, a case study of drone control is presented to demonstrate the efficiency of the proposed framework.

We would like to emphasize that the concept of self-triggered control has been extensively explored in the literature for various purposes. For instance, self-triggered control protocols have been proposed for the stabilization of linear or nonlinear systems; see, for example, [30], [31]. In the context of model predictive control, self-triggered mechanisms have also been investigated to reduce computational burden [32]–

This work was supported by the National Natural Science Foundation of China (623B1011, 62173226, 62061136004).

J. Huang, C. Wang and X. Yin are with Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China. J. Huang is also with the Zhiyuan College, Shanghai Jiao Tong University. e-mail: {hjy-564904993, wangchuwei, yinxiang}@sjtu.edu.cn

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

[34]. However, to our knowledge, the application of self-triggered mechanisms for control synthesis of STL specifications has not yet been implemented, and this constitutes the main contribution of our work.

II. Preliminary

A. System Model

We consider a discrete-time dynamic system of form

$$x_{t+1} = f(x_t, u_t) + w_t, \tag{1}$$

where $x_t \in \mathcal{X} \subseteq \mathbb{R}^n$, $u_t \in \mathcal{U} \subseteq \mathbb{R}^m$, $w_t \in \mathcal{W} \subset \mathbb{R}^n$ are the system state, control input and external input or disturbance at time instant t, respectively. The system dynamic is described by a function $f : \mathcal{X} \times \mathcal{U} \times \mathcal{W} \to \mathcal{X}$. We assume that the disturbances are unknown but are confined within a specified compact set $\mathcal{W}$. Additionally, we assume Lipschitz continuity of the dynamic function f with respect to the state variable x. That is, for any $x, x' \in \mathcal{X}$ and $u \in \mathcal{U}$, there exists a positive constant L such that $|f(x', u) - f(x, u)| \leq L|x' - x|$.

Suppose that the system is in state $x_t \in \mathcal{X}$ at instant $t \in \mathbb{Z}_{\geq 0}$. Then given a sequence of control inputs $\mathbf{u}_{t:T-1} = u_t u_{t+1} \cdots u_{T-1} \in \mathcal{U}^{T-t}$ and a sequence of disturbances $\mathbf{w}_{t:T-1} = w_t w_{t+1} \cdots w_{T-1} \in \mathcal{W}^{T-t}$, the solution trajectory of the system is a sequence of states $\xi_f(x_t, \mathbf{u}_{t:T-1}, \mathbf{w}_{t:T-1}) = x_{t+1} \cdots x_T \in \mathcal{X}^{T-t}$ such that $x_{i+1} = f(x_i, u_i) + w_i, \forall i = t, \cdots, T-1$. The solution of nominal system is denoted by $\xi_f(x_t, \mathbf{u}_{k:T-1})$ by assuming that there is no disturbance.

B. Signal Temporal Logic

The formal specifications of the system are described by STL formulae with bounded-time with the following syntax

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

where $\top$ represents the `true` predicate, and π^μ is a predicate whose truth value is determined by the sign of its underlying predicate function $\mu : \mathbb{R}^n \to \mathbb{R}$, i.e., it holds true if $\mu(x_t) \geq 0$; otherwise, it is false. Notations $\neg$ and $\wedge$ denote the standard Boolean operators "negation" and "conjunction", respectively, which induce "disjunction" and "implication". Additionally, $\mathbf{U}_{[a,b]}$ represents the temporal operator "until", where $a, b \in \mathbb{R}$.

STL formulae are evaluated on state sequences. Given a sequence $\mathbf{x}$, we use notation $(\mathbf{x}, t) \models \Phi$ to denote the satisfaction for STL formulae Φ at time instant t. In particular, we have $(\mathbf{x}, t) \models \pi^\mu$ if and only if $\mu(x_t) \geq 0$, and $(\mathbf{x}, t) \models \Phi_1 \mathbf{U}_{[a,b]} \Phi_2$ if and only if there exists $t' \in [t + a, t + b]$ such that $(\mathbf{x}, t') \models \Phi_2$ and for all $t'' \in [t, t']$, we have $(\mathbf{x}, t'') \models \Phi_1$. Furthermore, we can also introduce temporal operators "eventually" and "always" by $\mathbf{F}_{[a,b]}\psi := \top \mathbf{U}_{[a,b]} \psi$ and $\mathbf{G}_{[a,b]} := \neg \mathbf{F}_{[a,b]} \neg\psi$, respectively.

Apart from the Boolean satisfaction, an STL formula can also be *quantitatively* evaluated based on the *robust semantics*. Formally, for any STL formula Φ, state sequence $\mathbf{x}$ and time instant k, we denote by $\rho^\Phi_{\mathbf{x},k}$ the *space-robustness function* the same as [35].

In this paper, we consider the following restricted 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. Specifically, we only allow the temporal operators to be applied once for Boolean formula and nested temporal operators are not allowed.

We express the satisfaction region of predicate π^μ as $\mathcal{H}^\mu = \{x \in \mathcal{X} \mid \mu(x) \geq 0\}$. Then we have $\mathcal{H}^{\neg\varphi} = \mathcal{X} \setminus \mathcal{H}^\varphi$ and $\mathcal{H}^{\varphi_1 \wedge \varphi_2} = \mathcal{H}^{\varphi_1} \cap \mathcal{H}^{\varphi_2}$. Also, we introduce a new temporal operator $\mathbf{U}'$ defined by $(\mathbf{x}, t) \models \Phi_1 \mathbf{U}'_{[a,b]} \Phi_2$ iff $\exists t' \in [t + a, t + b]$ such that $(\mathbf{x}, t') \models \Phi_2$ and $\forall t'' \in [t + a, t']$ $(\mathbf{x}, t'') \models \Phi_1$. Compared with $\mathbf{U}$, $\mathbf{U}'$ only required that Φ_1 holds *from instant* a before Φ_2 holds.

Note that we can express the standard "until" by $\Phi_1 \mathbf{U}_{[a,b]} \Phi_2 = (\Phi_1 \mathbf{U}'_{[a,b]} \Phi_2) \wedge (\mathbf{G}_{[0,a]} \Phi_1)$. Furthermore, we can also express "eventually" by $\mathbf{F}_{[a,b]}\varphi$ by $\top \mathbf{U}_{[a,b]}\varphi$. Hereafter, we will only consider two operators $\mathbf{G}$ and $\mathbf{U}'$.

C. Fragment of Signal Temporal Logic and Remaining Sets

Formula Φ in the form of Equation (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 each Φ_i is either (i) $\mathbf{G}_{[a_i,b_i]} x \in \mathcal{H}_i$ or (ii) $x \in \mathcal{H}_i^1 \mathbf{U}'_{[a_i,b_i]} x \in \mathcal{H}_i^2$. Each sub-formula $\Phi_i^{[a_i,b_i]}$ is effective within time interval $[a_i, b_i]$. We denote by $\mathcal{I} = \{1, \cdots, N\}$ the index set of all sub-formulae and by $O_i \in \{\mathbf{G}, \mathbf{U}'\}$ the unique temporal operator in Φ_i. Also, we denote by $\mathcal{I}_t = \{i \in \mathcal{I} \mid a_i \leq t \leq b_i\}$ the index set of sub-formulae that are effective at instant t. Similarly, we denote by $\mathcal{I}_{<t} = \{i \in \mathcal{I} \mid b_i < t\}$ and $\mathcal{I}_{>t} = \{i \in \mathcal{I} \mid t < a_i\}$ the index sets of sub-formulae that are effective strictly before and after instant t respectively. We introduce the definition of the remaining set.

Definition 1 (Remaining Sets): Let $I \subseteq \mathcal{I}$ be a set of indices representing those sub-formulae that have not yet been satisfied. We say I is a *remaining set* at instant t if it satisfies the following conditions: (i) $\mathcal{I}_{<t} \cap I = \emptyset$; and (ii) $\mathcal{I}_{>t} \subseteq I$; and (iii) $\{i \in \mathcal{I}_t \mid [O_i = \mathbf{G}] \vee [O_i = \mathbf{U}' \wedge t = a_i]\} \subseteq I$. Furthermore, we denote by $\mathbb{I}_t$ the set of all possible remaining sets at instant t.

Definition 2 (I-Remaining Robust Feasible Sets): Given a remaining set I at instant t, we call formula $\hat{\Phi}_t^I = \bigwedge_{i \in I \cap \mathcal{I}_t} \Phi_i^{[t,b_i]} \wedge \bigwedge_{i \in I \cap \mathcal{I}_{>t}} \Phi_i^{[a_i,b_i]}$ the *I-remaining formula* representing the entire task remained, where $\Phi_i^{[t,b_i]}$ is obtained from $\Phi_i^{[a_i,b_i]}$ by replacing the starting instant of temporal operator from a_i to t.

We define $\mathfrak{X}_t^I$ as the set of states from which the subsequent sub-tasks $\hat{\Phi}_t^I$ are feasible at instant t 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, we we have

$$\mathfrak{X}_t^I = \left\{ x_t \in \mathcal{X} \;\middle|\; \begin{array}{l} \exists \mathbf{u}_{t:T_\Phi\text{-}1} \in \mathcal{U}^{T_\Phi - t}, \forall \mathbf{w}_{t:T_\Phi\text{-}1} \in \mathcal{W}^{T_\Phi - t}, \\ \text{s.t. } x_t \xi_f(x_t, \mathbf{u}_{t:T_\Phi\text{-}1}, \mathbf{w}_{t:T_\Phi\text{-}1}) \models \hat{\Phi}_t^I \end{array} \right\}.$$

D. Augmented and Belief States

The system information can be more precisely described by a tuple (x_t, I_t), which is referred to as the *augmented state* at instant t, where x_t is the current state and I_t is the remaining index set at instant t. Note that, within each augmented state (x_t, I_t), the advancement of the task I_t does not incorporate the influence of the current state x_t; instead, it will counted by the subsequent time instant.

A *belief state* is a set of augmented states. We denote by $\mathbb{B} := 2^{\mathcal{X}\times\mathcal{I}}$ the set of all possible belief states. Given a belief state $\boldsymbol{b}$, each augmented state (x_t, I_t) in it is an *explanation* of the possible current status of the system. Therefore, a belief state essentially captures all possible explanations based on the partial information.

III. Self-Triggered MPC Algorithm

Our objective is to synthesize a feedback control strategy such that the sequence generated by the closed-loop system satisfies the desired STL formula Φ under all possible disturbances. Furthermore, we want to minimize control effort while maximizing the control performance. Formally, given state x_t at instant t and input sequence $\mathbf{u}_{t:T_\Phi-1}$, we consider a generic cost function $J : \mathcal{X} \times \mathcal{U}^{T_\Phi - t} \to \mathbb{R}$.

Our approach for solving the STL control synthesis problem follows a self-triggered framework of model predictive control. Specifically, a self-triggered MPC controller works recursively as follows:

- At a trigger instant t_s determined in the previous round, we observe the current system state and solve a finite-horizon optimization problem to compute a finite open-loop sequence of inputs $\mathbf{u}_{t_s:T_\Phi-1}$ such that the cost function is minimized subject to the constraints on both the system dynamic and the robustness.
- We employed a Trigger-time procedure (introduced in section IV) to compute the trigger interval τ.
- During the interval τ, the controller will apply $\mathbf{u}_{t_s:t_s+\tau-1}$ out of $\mathbf{u}_{t_s:T_\Phi-1}$ to the system until $t_s + \tau$, at which the above steps will be repeated.

In this paper, we obtain the open-loop control input $\mathbf{u}_{t_s:T_\Phi-1}$ by enforcing the spatial robustness larger than a positive number K. Basically, K is a *response* to the future unknown disturbance. Larger K implies more robust control input to some extent, which brings longer trigger interval τ. However, this will decrease the performance of the solution as the system tends to operate in a more open-loop fashion. Furthermore, if K is selected to be too large, then the optimization problem may have no solution. This is a trade-off relationship. In other words, user-defined K captures how robust the open-loop controller is and determines how often the agent needs to turn on the sensors. We will show the relation of K and τ by some examples in Section V. The optimization problem at instant t is formulated as follows.

Algorithm 1: Self-Triggered MPC for STL

Input: STL formula Φ, dynamic system model of form (1), cost function J and threshold K
Output: Control input u_t at each instant t

1 $t_s \leftarrow 0; I \leftarrow \mathcal{I}, \hat{\boldsymbol{b}} \leftarrow \{(x_0, I)\}$
2 **while** $t_s < T_\Phi$ **do**
3 measure current state x_{t_s}
4 $\boldsymbol{b} \leftarrow \mathsf{refine}(\hat{\boldsymbol{b}}, x_{t_s})$
5 $\mathbf{u}^*_{t_s:T_\Phi-1} \leftarrow$ solve Problem 1 based on t_s, x_{t_s}, K and $\hat{\Phi}^I_{t_s}$, where $(x_{t_s}, I) = \boldsymbol{b}$
6 $\tau, \hat{\boldsymbol{b}} \leftarrow \mathsf{Trigger\text{-}Time}(\boldsymbol{b}, t_s, \mathbf{u}^*_{t_s:T_\Phi-1})$
7 $\mathbf{u}_{t_s:t_s+\tau-1} \leftarrow \mathbf{u}^*_{t_s:t_s+\tau-1}$; # apply to the system
8 $t_s \leftarrow t_s + \tau$

Problem 1 (STL Optimization Problem): Given system of form (1), a cost function J, current state $x_t \in \mathcal{X}$ at instant t, the remaining set I at instant t and I-remaining formula $\hat{\Phi}^I_t$ at instant t, find an optimal input sequence $\mathbf{u}^*_{t:T_\Phi-1}$ that minimizes the cost function while guarantees that the robustness of the predicted sequence $x_t\xi_f(x_t, \mathbf{u}_{t:T_\Phi-1})$ is no less than the threshold K. Formally, we have the following optimization problem

$$\begin{aligned} \underset{\mathbf{u}_{t:T_\Phi-1}}{\text{minimize}} \quad & J(x_t, \mathbf{u}_{t:T_\Phi-1}) \\ \text{subject to} \quad & \rho^{\hat{\Phi}^I_t}_{\mathbf{x}_t\xi_f(x_t,\mathbf{u}_{t:T_\Phi-1}),0} \geq K \\ & u_t, u_{t+1}, \cdots, u_{T_\Phi-1} \in \mathcal{U}. \end{aligned}$$

Now we present the main algorithm for self-trigger MPC in Algorithm 1. It works as follows. First, line 1 initializes the system, where $\hat{\boldsymbol{b}}$ denotes the current belief state of the system. Then at each trigger instant, which is determined by the previous decision round, line 3 takes a new state observation and line 4 uses this information to refine the belief state (how the refinement is done will be specified in the next section). Then based on the refined information of the system (current state and remaining set), line 5 solves Problem 1 to obtain control input $\mathbf{u}^*_{t_s:T_\Phi-1}$. Furthermore, based on the computed control input, line 6 computes how long it can be applied in an open-loop fashion, i.e., the trigger interval τ and predicts the belief state $\hat{\boldsymbol{b}}$ by Trigger-Time Procedure, which will also be specified in the next section. Finally, in lines 7-8, we apply the control input $\mathbf{u}_{t_s:t_s+\tau-1}$ and wait until the next trigger instant $t_s + \tau$, so that the algorithm can run recursively.

Remark 1: The open-loop control input $\mathbf{u}_{t:T_\Phi-1}$ in Problem 1 can be computed by any of the existing methods for optimizations-based STL control synthesis; see, e.g., [12], [21], [23], [26], [29], [36] and they are all compatible with our self-triggered control framework.

IV. Trigger Time Procedure

The basic MPC approach of solving a finite-horizon optimization at each sampling instant is computational and communicational expensive. However, the self-triggered mechanism can adjust the triggered interval according to the change of inputs. Leveraging our previous work in [37] for online

monitoring, the triggered interval will be the maximum time instant satisfying certain properties. In this section, we will first introduce the update of belief state space where the procedure is built upon. Then we will define the properties the triggered interval should satisfy and then present the whole procedure.

In accordance to the semantics of STL formulae, the index set should be renewed upon a new state. For each time instant t, the *index update function*, denoted as $\mathsf{update}_t : \mathcal{I} \times \mathcal{X} \to \mathcal{I}$, is define by: for any $I \subseteq \mathcal{I}, x \in \mathcal{X}$, we have

$$\mathsf{update}_t(I,x) = \left\{ i \in I \;\middle|\; \begin{matrix} [i \in \mathcal{I}_t^{\mathbf{U}} \wedge x \notin \mathcal{H}_i^1 \cap \mathcal{H}_i^2] \\ \vee [i \in \mathcal{I}_t^{\mathbf{G}} \wedge t \neq b_i] \vee [i \notin \mathcal{I}_t] \end{matrix} \right\},$$

where, for each $i \in \mathcal{I}_t^{\mathbf{U}}$, we have $\Phi_i = x \in \mathcal{H}_i^1 \mathbf{U}_{[a_i,b_i]} x \in \mathcal{H}_i^2$.

Intuitively, $\mathsf{update}_t(I,x)$ can only change the the index of sub-formulae that is effective at instant t by removing the index for sub-formulae with temporal operator $\mathbf{U}$ that have already been fulfilled and sub-formulae with expired temporal operator $\mathbf{G}$.

Given an augmented state (x, I) at instant t and input u, we say (x', I') is a *successor augmented state* of (x, I) from instants t to $t+1$ if the following conditions are met:

- $\exists w \in \mathcal{W}$ such that $x' = f(x,u) + w$; and
- $I' = \mathsf{update}_t(I, x)$.

We denote by $\mathsf{succ}_t(x, I, u) \in \mathbb{B}$ the set of all *successor augmented states* of (x, I) from instants t to $t+1$ with input u. In the interval between two trigger instants where no observation is made, the belief states can only be *predicted*.

Definition 3 (Belief Predictions with Inputs): We define the (one-step) *belief prediction function* from instants t to $t+1$, denoted as $\mathsf{pred}_t^{t+1} : \mathbb{B} \to \mathbb{B}$, by: for any $\boldsymbol{b} \in \mathbb{B}$, $u \in \mathcal{U}$, we have $\mathsf{pred}_t^{t+1}(\boldsymbol{b}, u) = \bigcup_{(x,I) \in \boldsymbol{b}} \mathsf{succ}_t(x, I, u)$.

The multi-step belief prediction function is defined recursively by: for any $\boldsymbol{b} \in \mathbb{B}$ and $k \geq 1$, we have

$$\begin{aligned} \mathsf{pred}_t^{t+k}(\boldsymbol{b}, \mathbf{u}_{t:t+k-1}) &= \mathsf{pred}_{t+k-1}^{t+k}(\boldsymbol{b}_{t+k-1}, u_{t+k-1}). \\ \boldsymbol{b}_{t+k-1} &= \mathsf{pred}_{t+k-2}^{t+k-1}(\boldsymbol{b}_{t+k-2}, u_{t+k-2}) \\ &\cdots \\ \boldsymbol{b}_{t+1} &= \mathsf{pred}_t^{t+1}(\boldsymbol{b}, u_t). \end{aligned}$$

Once observations are made at those triggered instants, the belief states can be *refined* by the *belief refinement function*, denoted as $\mathsf{refine} : \mathbb{B} \times \mathcal{X} \to \mathbb{B}$, which is defined by: for any $\boldsymbol{b} \in \mathbb{B}, x \in \mathcal{X}$, we have $\mathsf{refine}(\boldsymbol{b}, x) = \{(x', I') \in \boldsymbol{b} \mid x' = x\} = \boldsymbol{b} \cap (\{x\} \times \mathcal{I})$.

Intuitively, the belief refinement function confines a belief state to a smaller set that aligns with the current observed state and is time-independent.

Now let us discuss how the self-triggered MPC controller determines the trigger interval τ. Given the current belief state $\boldsymbol{b}_t$ and input sequence $\mathbf{u}_{t:T_\Phi-1}$, it aims to select an integer τ as the time interval before the next optimization and observation. During the interval from t to $t+\tau$, no observation is made. Consequently, the controller can only predict the belief state $\hat{\boldsymbol{b}}_{t+\tau} = \mathsf{pred}_t^{t+\tau}(\boldsymbol{b}_t, \mathbf{u}_{t:t+\tau-1})$. Therefore, the predicted belief state $\hat{\boldsymbol{b}}_{t+\tau}$ should satisfy the following two properties:

- *Determinacy:* For any two augmented states $(x, I), (x, I') \in \hat{\boldsymbol{b}}_{t+\tau}$ with the same state component, we have $\mathsf{update}_t(I, x) = \mathsf{update}_t(I', x)$.
- *Safety:* For any augmented state $(x, I) \in \hat{\boldsymbol{b}}_{t+\tau}$, we have $x \in \mathfrak{X}_t^I$.

We require the above two properties for the following reasons. First, we need to know how far a task has progressed at time $t+\tau$. This means we have to figure out if each subformula is satisfied exactly. Essentially, we should precisely update the remaining set I upon observing $x_{t+\tau}$, which is the meaning of *determinacy*. Second, we want to be sure that the task can still be completed for the next optimization. We noticed in the earlier discussion that having $x \in \mathfrak{X}_t^I$ means the task could still be finished later. Therefore, we also need to ensure *safety*. With these considerations, we can define the Self-Triggered MPC Problem.

Problem 2 (Self-Triggered MPC Problem): Given system in the form of Equation (1), an STL formula Φ, augmented state (x_t, I_t), upper bound of trigger interval T_{max} and input sequence $\mathbf{u}_{t:T_\Phi-1}$, maximize trigger interval $\tau \in \{1, \ldots, T_{max}\}$ under the premise of guaranteeing *Determinacy* and *Safety*, and compute the belief state $\hat{\boldsymbol{b}}$ at instant $t+\tau$.

Now let us discuss how to solve Problem 2. As previously indicated, the belief state $\hat{\boldsymbol{b}}$ at time $t+\tau$ must satisfy both *Safety* and *Determinacy*. It is important to note that while it is strictly required for $\hat{\boldsymbol{b}}$ to satisfy *Safety* for all instants between t and $t+\tau$, it is only necessary for it to fulfill the requirement of *Determinacy* at time $t+\tau$. This is because our primary concern lies in determining the system's status at instant $t+\tau$. Therefore, we denote $\mathsf{pred}_t^{t+k}(\boldsymbol{b}_t, \mathbf{u}_{t:t+k-1})$ as $\hat{\boldsymbol{b}}_{t+k}$. Then the largest trigger interval τ can be select is

$$\tau = \max\{k \leq T_{max} : \hat{\boldsymbol{b}}_{t+k} \text{ satisfies safety and determinacy}\}. \tag{5}$$

With the above discussion, we formally introduce the trigger time computation method in Trigger-Time Procedure (Procedure 1). In the Trigger-Time Procedure, line 1 initializes the procedure. Lines 2-9 calculate trigger interval τ based on Equation (5). Specifically, lines 3-4 predict the belief state $\hat{\boldsymbol{b}}$. Line 5 and line 6 check *safety* and *determinacy*, respectively, and we update τ only when the above two properties are both satisfied. Note that, if *safety* is violated, then we return τ and $\hat{\boldsymbol{b}}$ immediately.

Theorem 1: Given system in the form of Equation (1) and STL formula Φ, when applying Algorithm 1 and Trigger-Time procedure, if the optimization problem (4) is feasible at each t_s, then after applying the control input, that is the concatenation of all input sequences $\mathbf{u}_{t_s:t_s+\tau-1}$ in line 7 of Algorithm 1, we have $\mathbf{x}_{0:T_\Phi} \models \Phi$.

V. Case Studies

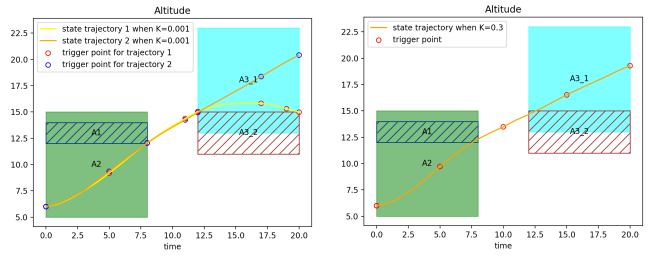
In this section, we demonstrate our approach by a case study. Specifically, the general self-triggered MPC algorithm (1) is conducted in Python 3, and we solve the STL Optimization Problem (1) by Pyscipopt. The Trigger-Time Procedure is conducted in Julia with the help of

Procedure 1: Trigger Time

Input: STL formula Φ, dynamic system model in the form of Equation (1), initial instant t_s, upper bound T_{max}, original input sequence $\mathbf{u}^*_{t_s:T_\Phi-1}$, and augmented state $\boldsymbol{b}$

Output: Trigger interval τ and belief state $\hat{\boldsymbol{b}}$

$k \leftarrow 1$; $\tau \leftarrow 0$; $\hat{\boldsymbol{b}} \leftarrow \boldsymbol{b}$
while $k < T_{max}$ **do**
  $\hat{\boldsymbol{b}} \leftarrow \text{pred}^{t_s+k}_{t_s+k-1}(\hat{\boldsymbol{b}}, \mathbf{u}^*_{t_s+k-1})$
  if $\hat{\boldsymbol{b}}$ *satisfies* safety **then**
    if $\hat{\boldsymbol{b}}$ *satisfies* determinacy **then**
      $\tau \leftarrow k$
  else
    return $\tau, \hat{\boldsymbol{b}}$
  $k \leftarrow k + 1$



(a) Trajectory when threshold K=0.001. (b) Trajectory when threshold K=0.3.

Fig. 1: Simulation results.

existing package JuliaReach. The communication between the Python and Julia languages is facilitated by the Pyjulia library. All simulations were performed on a computer with an Intel Core i5-12500H CPU and 16 GB of RAM. All codes are available at `https://github.com/JunyueHuang/ST_MPC-for-STL`

We consider a drone for the purpose of gathering air data at various altitudes. Therefore, we consider its altitude state z and velocity v_z, i.e., $x = [z, v_z]^\top \in \mathcal{X} = [0, 100] \times [-5, 5]$. The discrete-time dynamic of the drone is

$$x_{t+1} = Ax_t + Bu_t + w_t,$$

where $A = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$ and $\mathcal{U} \subseteq [-2.5, 2.5], \mathcal{W} \subseteq [-0.05, 0.05]$. The drone will start at initial state $[6, 0]$, and its objective is as follows. First, within time interval $[0, 8]$, the drone needs to maintain its altitude between $[5, 15]$ and the altitude within the range $[12, 14]$. Then, within time interval $[12, 20]$, the drone should maintain its altitude between $[13, 23]$ until it arrives at $z \in [11, 15]$ at some instant. We set the time horizon $T = 20$, and the overall mission of the drone is captured by the following STL formula: $\psi = \mathbf{F}_{[0,8]} z \in [12, 14] \wedge \mathbf{G}_{[0,8]} z \in [5, 15] \wedge z \in [15, 25] \mathbf{U}_{[13,23]} z \in [11, 15]$.

Also, we consider the following the cost function to minimize the control efforts: $J(\mathbf{u}_{t_s:T_\Phi-1}) = \sum_{i=t_s}^{T_\Phi-1} ||u_i||^2$.

In Fig. 1(a), we show two possible trajectories generated by our algorithm, where dots represent the trigger instants and their corresponding states. Here each colored zones matches predicates in the STL formula. Using regions in the figure, the STL formula can be rewritten as $\psi = \mathbf{F}_{[0,8]} A1 \wedge \mathbf{G}_{[0,8]} A2 \wedge A3\_1 \mathbf{U}_{[13,23]} A3\_2$

Here, we start by selecting a small value $K = 0.001$ and set $T_{max} = 5$ as the the upper bound of trigger interval τ. Note that due to the presence of disturbances, the trajectories for different instances executing the same control policy may vary. For example, in trajectory 1, during the 4th trigger instant, the controller realizes that region $A3\_2$ has not been reached yet. Consequently, it needs to trigger more frequently in the subsequent instants to ensure that $A3\_2$ can be visited. However, in trajectory 2, during the 4th trigger instant, the controller observes that it has already reached region $A3\_2$. As a result, it only needs to trigger every T_{max} time units in the remaining stage.

TABLE I: Avg. Trigger intervals with different thresholds.

	$T_{max} = 5$				$T_{max} = 10$			
Threshold K	0.3	0.15	0.05	0.001	0.3	0.05	0.03	0.001
Avg. Trigger Interval $\bar{\tau}$	5.0	4.08	3.67	3.31	6.67	5.58	5.16	4.83

In order to examine the impact of the threshold value K on the performance of the solution, we increase K from 0.001 to 0.3. We illustrate a trajectory generated under the synthesized MPC controller in Fig. 1(b). It is worth mentioning that this MPC controller consistently triggers every T_{max} time units. This behavior arises because we chose a considerably large value for K, resulting in a highly robust plant at each trigger instant. Consequently, even when operating in an open-loop manner, the system can maintain the fulfillment of the task requirements.

However, as we mentioned, a large value of K can potentially lead to a decrease in overall performance. To analyze this effect more thoroughly, we increment K from 0 to 0.5 gradually. Additionally, we consider two different values for T_{max}. For each fixed combination of K and T_{max}, we execute the synthesized MPC controller 20 times to account for the influence of disturbances. Subsequently, we calculate the average trigger intervals and the average cost, which are presented in Table I and illustrated in Fig. 2, respectively. As depicted in the figure, when K is relatively large, an incremental increase in K imposes stricter constraints in the optimization problem. This can lead to a deterioration in optimality and subsequently an increase in cost. Conversely, when K is relatively small, a high cost is also observed. This is attributed to excessively small values of K compromising robustness, resembling situations similar to trajectory 2 in Fig. 1(a), which can result in significant costs.

VI. Conclusion

In this paper, we introduced a novel self-triggered model predictive control framework for control synthesis of signal temporal logic specifications. Instead of acquiring information and performing optimization at each time instant, our approach can automatically determine the next trigger instant on demand. We provided a systematic approach for determining the trigger time such that the STL specification can be

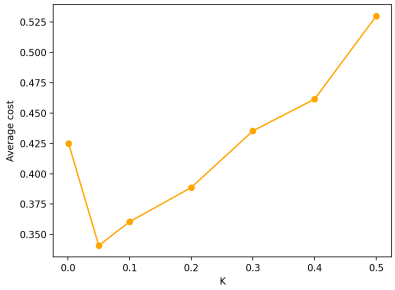


Fig. 2: Relationship between average cost and threshold K when $T_{max} = 5$.

enforced while minimizing the observation or computation-related burden. It is important to note that, in this paper, we do not explicitly address the issue of recursive feasibility in the MPC iteration. Generally, the STL optimization problem could become unsolvable after a few iterations. We plan to investigate this feasibility issue in our future work.

REFERENCES

[1] M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher, "Formal specification and verification of autonomous robotic systems: A survey," *ACM Computing Surveys*, vol. 52, no. 5, pp. 1–41, 2019.

[2] N. Mehdipour, M. Althoff, R. D. Tebbens, and C. Belta, "Formal methods to comply with rules of the road in autonomous driving: State of the art and grand challenges," *Automatica*, vol. 152, p. 110692, 2023.

[3] X. Yin, B. Gao, and X. Yu, "Formal Synthesis of Controllers for Safety-Critical Autonomous Systems: Developments and Challenges," *Annual Reviews in Control*, p. 100940, 2024.

[4] 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, Springer, 2004.

[5] L. Lindemann, J. Nowak, L. Schönbächler, M. Guo, J. Tumova, and D. V. Dimarogonas, "Coupled multi-robot systems under linear temporal logic and signal temporal logic tasks," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 2, pp. 858–865, 2019.

[6] Z. Liu, B. Wu, J. Dai, and H. Lin, "Distributed communication-aware motion planning for networked mobile robots under formal specifications," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 4, pp. 1801–1811, 2020.

[7] X. Yu, X. Yin, and L. Lindemann, "Efficient stl control synthesis under asynchronous temporal robustness constraints," in *IEEE Conference on Decision and Control*, pp. 6847–6854, 2023.

[8] X. Yu, W. Dong, S. Li, and X. Yin, "Model predictive monitoring of dynamical systems for signal temporal logic specifications," *Automatica*, vol. 160, p. 111445, 2024.

[9] 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.

[10] S. V. Patil, K. Hashimoto, and M. Kishida, "Traffic flow control at signalized intersections using signal spatio-temporal logic," in *IEEE Conference on Decision and Control*, pp. 1051–1058, IEEE, 2022.

[11] 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.

[12] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 96–101, 2018.

[13] W. Xiao, C. A. Belta, and C. G. Cassandras, "High order control lyapunov-barrier functions for temporal logic specifications," in *American Control Conference*, pp. 4886–4891, 2021.

[14] S. Liu, A. Saoud, P. Jagtap, D. V. Dimarogonas, and M. Zamani, "Compositional synthesis of signal temporal logic tasks via assume-guarantee contracts," in *IEEE Conference on Decision and Control*, pp. 2184–2189, 2022.

[15] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *IEEE Conference on Decision and Control*, pp. 6565–6570, IEEE, 2016.

[16] W. Hashimoto, K. Hashimoto, and S. Takai, "Stl2vec: Signal temporal logic embeddings for control synthesis with recurrent neural networks," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5246–5253, 2022.

[17] S. Wang, X. Yin, S. Li, and X. Yin, "Tractable reinforcement learning for signal temporal logic tasks with counterfactual experience replay," *IEEE Control Systems Letters*, 2024.

[18] S. Wang, S. Li, and X. Yin, "Synthesis of temporally-robust policies for signal temporal logic tasks using reinforcement learning," in *IEEE International Conference on Robotics and Automation*, 2024.

[19] R. Liu, S. Li, and X. Yin, "NNgTL: Neural network guided optimal temporal logic task planning for mobile robots," in *IEEE International Conference on Robotics and Automation*, 2024.

[20] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *IEEE Conference on Decision and Control*, pp. 81–87, 2014.

[21] 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.

[22] V. Kurtz and H. Lin, "A more scalable mixed-integer encoding for metric temporal logic," *IEEE Control Systems Letters*, vol. 6, pp. 1718–1723, 2021.

[23] 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.

[24] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 115–140, 2019.

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

[26] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *International Conference on Hybrid Systems: Computation and Control*, pp. 239–248, 2015.

[27] S. S. Farahani, R. Majumdar, V. S. Prabhu, and S. Soudjani, "Shrinking horizon model predictive control with signal temporal logic constraints under stochastic disturbances," *IEEE Transactions on Automatic Control*, vol. 64, no. 8, pp. 3324–3331, 2018.

[28] Y. Meng and C. Fan, "Signal temporal logic neural predictive control," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7719–7726, 2023.

[29] X. Yu, C. Wang, D. Yuan, S. Li, and X. Yin, "Model predictive control for signal temporal logic specifications with time interval decomposition," in *IEEE Conference on Decision and Control*, pp. 7849–7855, 2023.

[30] M. Mazo Jr, A. Anta, and P. Tabuada, "An iss self-triggered implementation of linear controllers," *Automatica*, vol. 46, no. 8, pp. 1310–1314, 2010.

[31] A. Anta and P. Tabuada, "To sample or not to sample: Self-triggered control for nonlinear systems," *IEEE Transactions on automatic control*, vol. 55, no. 9, pp. 2030–2042, 2010.

[32] F. D. Brunner, M. Heemels, and F. Allgöwer, "Robust self-triggered MPC for constrained linear systems: A tube-based approach," *Automatica*, vol. 72, pp. 73–83, 2016.

[33] X. Mi, Y. Zou, S. Li, and H. R. Karimi, "Self-triggered DMPC design for cooperative multiagent systems," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 1, pp. 512–520, 2019.

[34] L. Lu and J. M. Maciejowski, "Self-triggered MPC with performance guarantee using relaxed dynamic programming," *Automatica*, vol. 114, p. 108803, 2020.

[35] 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.

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

[37] C. Wang, X. Yu, J. Zhao, L. Lindemann, and X. Yin, "Sleep when everything looks fine: Self-triggered monitoring for signal temporal logic tasks," 2023. arXiv: 2311.15531.