2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Modularized Control Synthesis for Complex Signal Temporal Logic Specifications

Zengjie Zhang and Sofie Haesaert

Abstract— **The control synthesis of a dynamic system subject to a signal temporal logic (STL) specification is commonly formulated as a mixed-integer linear/convex programming (MILP/MICP) problem. Solving such a problem is computationally expensive when the specification is long and complex. In this paper, we propose a framework to transform a long and complex specification into separate forms in time, to be more specific, the logical combination of a series of short and simple subformulas with non-overlapping timing intervals. In this way, one can easily modularize the synthesis of a long specification by solving its short subformulas, which improves the efficiency of the control problem. We first propose a syntactic timing separation form for a type of complex specifications based on a group of separation principles. Then, we further propose a complete specification split form with subformulas completely separated in time. Based on this, we develop a modularized synthesis algorithm that ensures the soundness of the solution to the original synthesis problem. The efficacy of the methods is validated with a robot monitoring case study in simulation. Our work is promising to promote the efficiency of control synthesis for systems with complicated specifications.**

I. INTRODUCTION

Signal temporal logic (STL) is widely used to specify requirements for robot systems [1], [2], due to its advantage in specifying real-valued signals with finite timing bounds [3]. System control with STL specifications renders a synthesis problem that can be solved by mixed integer linear/convex programming (MILP/MICP) [3], [4]. Based on this a closed-loop controller can be developed using model predictive control (MPC) [5], [6]. However, solving a MILP/MICP problem is computationally expensive and time-consuming, especially for complex STL formulas with long timing intervals since the computational load grows drastically as the number of the integer variables increases (exponentially in the worst case) [7]. Thus, computational complexity has become a bottleneck of the control synthesis of complex STL specifications, especially those with time-variant specifications [8] and fixed-order constraints [9]. One effective approach is the model-checking-based method which transforms an STL formula into an automaton with strict timing bounds [10]. This method is usually less complex than an optimization problem since it is only concerned with a feasible solution. Control barrier functions (CBF) [9] and funnel functions [11] are also used to simplify the STL synthesis problems.

Another direction of reducing the complexity is to decompose a long and complex STL formula into several shorter and simpler subformulas and solve them sequentially. A subformula refers to a simple STL formula that serves as a primitive unit of a complex formula [12]. This indicates the possibility of splitting a big planning problem into several smaller problems and solving them one by one in the order of time, which forms the essential thought of modularized synthesis. This idea is straightforward from a practical perspective: a complex task is usually composed of a series of smaller subtasks that have independent objectives and are ordered in time. For example, a typical food delivery task includes three subtasks: picking up the order at the restaurant, navigating to the customer, and performing the delivery. Finishing these subtasks means accomplishing the overall task. The advantage of this approach is based on the assumption that solving a subtask may be substantially simpler than directly solving the original overall task.

However, modularized synthesis based on specification decomposition is not trivial and brings up two major challenges. Firstly, the decomposed specification has to ensure *soundness*, i.e., any feasible solution of the modularized synthesis must also be a feasible solution of the original specification. This is important to ensure the efficacy of the specification decomposition and modularized synthesis [13]. Secondly, the subformulas may have overlapping timing intervals which indicate the dependence coupling among these subformulas. In this case, each subformula should not be synthesized independently but should incorporate the coupling with its overlapping subformulas. In existing work, the soundness of specification separation is ensured by *syntactic separation* as partially discussed in [14], [15]. Recently, model checking based on specification decomposition has been studied for a fragment of STL formulas [16]. Nevertheless, the coupling issues among subformulas have not been well resolved by the existing work. To our knowledge, there is no other existing work discussing the modularized synthesis of STL formulas, although we believe it to be a promising technology for the efficient synthesis of complex specifications.

In this paper, we investigate the modularized synthesis of complex STL specifications based on timing separation. We specifically look into a fragment of STL formulas composed of complex temporal operators for which interval overlapping can not be resolved by purely using syntactic separation. Besides proposing several complementary syntactic separation principles to the existing work [14], [15], we also provide a sufficient separation method for this STL fragment with the overlapping between subformulas eliminated. In such a way, we develop a modularized synthesis algorithm for the separated specification by transforming the overall synthesis

This work was supported by the European project SymAware under the grant Nr. 101070802.

Zengjie Zhang and Sofie Haesaert are with the Department of Electrical Engineering, Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, Netherlands. {z.zhang3, s.haesaert}@tue.nl

problem into several small planning problems with reduced complexity, achieving higher efficiency than directly solving the original problem. The main contributions are as follows.

1). Proposing a syntactic timing separation form of a fragment of STL formulas that is proven to be syntactically equivalent to the original specification.

2). Proposing a complete splitting form of this STL fragment which is proven to be sound in semantics.

3). Developing a modularized synthesis algorithm for the complete splitting form, which ensures soundness but less complexity than the original specification synthesis problem.

The rest of the paper is organized as follows. Sec. II introduces the preliminary knowledge of this paper. In Sec. III, we present our main results on specification separation and modularized synthesis. Sec. IV provides a simulation case study to validate the efficacy of the proposed modularized synthesis method. Finally, Sec. V concludes this paper.

Notations: We use $\mathbb{R}$ and $\mathbb{R}^n$ to denote the sets of real scalars and n-dimensional real vectors. We also use $\mathbb{N}$ and $\mathbb{N}^+$ to denote natural numbers and positive natural numbers.

Proofs: Due to lack of space, we provide the proofs of all statements in an online ArXiv version of the paper [17].

II. Preliminaries and Problem Statement

A. Signal Temporal Logic (STL)

Specifications in Signal Temporal Logic (STL) can quantify requirements on real-valued signals. In this paper, we are concerned with discrete-time signals $\mathbf{x}_{[0,L]} := x_0 x_1 \cdots x_L$, where $L \in \mathbb{N}^+$ denotes the length of the signal and $x_k \in \mathbb{R}^n$ is the value of the signal at time $k \in \{0,1,\cdots,L\}$. With $\mathbf{x}_{[k_1,k_2]} := x_{k_1} x_{k_1+1} \ldots x_{k_2}$, we denote a segment of $\mathbf{x}$ or equivalently $\mathbf{x}_{[0,L]}$ with timing points $0 \le k_1 \le k_2 \le L$. The syntax of STL is recursively defined as

$$\varphi ::= \top \,|\, \mu \,|\, \neg\varphi \,|\, \varphi_1 \wedge \varphi_2 \,|\, \varphi_1 \mathsf{U}_{[a,b]} \varphi_2, \tag{1}$$

where φ_1, φ_2 are STL formulas, $\neg$, $\wedge$ are operators *negation* and *conjunction*, μ is a predicate that evaluates a predicate function $\eta : \mathbb{R}^n \to \{\top, \bot\}$ by $\mu = \begin{cases} \top & \text{if } \eta(x_k) \ge 0 \\ \bot & \text{if } \eta(x_k) < 0 \end{cases}$, for discrete time k, and $\mathsf{U}_{[a,b]}$ is the *until* operator bounded with time interval $[a,b]$, where $a, b \in \mathbb{N}$ and $a \le b$.

The semantics of STL are given as follows. We denote the satisfaction of φ at time k by $\mathbf{x}$ as $(\mathbf{x},k) \vDash \varphi$. Furthermore, we have that $(\mathbf{x},k) \vDash \mu \leftrightarrow \eta(x_k) \ge 0$; $(\mathbf{x},k) \vDash \neg\varphi \leftrightarrow \neg((\mathbf{x},k) \vDash \varphi)$; $(\mathbf{x},k) \vDash \varphi_1 \wedge \varphi_2 \leftrightarrow (\mathbf{x},k) \vDash \varphi_1$ and $(\mathbf{x},k) \vDash \phi_2$; $(\mathbf{x},k) \vDash \varphi_1 \mathsf{U}_{[a,b]} \varphi_2 \leftrightarrow \exists k' \in [k+a, k+b]$, such that $(x,k') \vDash \varphi_2$, and $(x,k'') \vDash \varphi_1$ holds for all $k'' \in [k,k']$. Besides, additional operators *disjunction*, *eventually*, and *always* are, respectively, defined as $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\mathsf{F}_{[a,b]}\varphi = \top \mathsf{U}_{[a,b]} \varphi$, and $\mathsf{G}_{[a,b]}\varphi = \neg\mathsf{F}_{[a,b]}\neg\varphi$. When $k=0$, we also write $(\mathbf{x},0) \vDash \varphi$ as $\mathbf{x} \vDash \varphi$. The *length* [18] of an STL, $\mathscr{L}(\varphi)$, is recursively defined as $\mathscr{L}(\mu) = 0$, $\mathscr{L}(\neg\varphi) = \mathscr{L}(\varphi)$, $\mathscr{L}(\varphi_1 \wedge \varphi_2) = \max\{\mathscr{L}(\varphi_1), \mathscr{L}(\varphi_2)\}$, $\mathscr{L}(\varphi_1 \mathsf{U}_{[a,b]} \varphi_2) = b + \max\{\mathscr{L}(\varphi_1), \mathscr{L}(\varphi_2)\}$, which represents the maximum time it takes to determine the truth of the formula φ.

B. Optimization-Based Specification Synthesis

STL formulas are used to specify the requirements of a signal of a dynamic system. In this paper, we consider the following discrete-time dynamic system,

$$x_{k+1} = f(x_k, u_k), \tag{2}$$

where $x_k \in \mathbb{R}^n$ and $u_k \in \mathbb{U}$ are the state and the control input of the system at time k, where $\mathbb{U} \subseteq \mathbb{R}^m$ is the admissible control set, and $f : \mathbb{R}^n \times \mathbb{U} \to \mathbb{R}^n$ is a smooth vector field. Then, the control problem of the system can be formulated as the following optimization problem,

$$\min_{\mathbf{u}} \left(\textstyle\sum_{k=0}^{L-1} u_k^\top u_k - \rho(\mathbf{x}, \varphi) \right) \tag{3a}$$

$$\text{s.t. eq. (2) and } u_k \in \mathbb{U},\ \forall k \in \{0,1,\cdots,L-1\}, \tag{3b}$$

where $L \in \mathbb{N}^+$ is the control horizon, $\mathbf{u} = u_0 u_1 \cdots u_{L-1}$ and $\mathbf{x} = x_0 x_1 \cdots x_L$ are the *open-loop* control and state signals, φ is an STL formula with $\mathscr{L}(\varphi) = L$ to specify the requirements on the state signal $\mathbf{x}$, and $\rho(\mathbf{x},\varphi)$ is the robustness of the satisfaction as defined in [19], [20], with $\rho(\mathbf{x},\varphi) > 0 \leftrightarrow \mathbf{x} \vDash \varphi$. Eq. (3) renders an *open-loop* control problem and can be solved using MICP [7] with an input interface (x_0, L, φ).

C. Problem Statement

Solving problem (3) using MICP usually introduces heavy computational load due to the large number of integer variables brought up by the logical and temporal operators in the specification φ [7]. Usually, longer formulas introduce substantially more integer variables than shorter ones. In the worst case, a specification φ may contain $N \in \mathbb{N}^+$ G or F subformulas with the same length $L = \mathscr{L}(\varphi)$. This requires NL binary variables to determine the logical satisfaction of the complete specification, which leads to an exponential complexity $O(2^{NL})$. Therefore, the computational complexity of the synthesis problem is greatly dependent on the number and the lengths of subformulas.

In this paper, we intend to reduce the complexity of a synthesis problem (3) by separating a long specification φ into shorter subformulas which can be solved by smaller optimization problems. Examples of such separation include the following principle for an until operator $\varphi = \varphi_1 \mathsf{U}_{(a,b)} \varphi_2$ with a separating point $\kappa \in \mathbb{N}^+$, $a < \kappa < b$ [14],

$$\varphi = \varphi_1 \mathsf{U}_{(a,\kappa)} \varphi_2 \vee \left(\mathsf{G}_{(a,\kappa)} \varphi_1 \wedge \mathsf{F}_{\{\kappa\}} \left(\varphi_1 \wedge \varphi_2 \vee \varphi_1 \mathsf{U}_{(0,b-\kappa)} \varphi_2 \right) \right), \tag{4}$$

where the temporal operators F and G in (4) have shorter intervals compared to the original interval (a,b). This form is referred to as *syntactic separation* since it ensures syntactic equivalence [14], [15], i.e., both sides have the same set of satisfying signals.

In this sense, we aim to decompose the overall synthesis problem into several subproblems with shorter horizons and fewer specifications, inspiring the modularized synthesis of the original specification. More precisely, we focus on the following fragment of complex formulas,

$$\varphi := \underbrace{\textstyle\bigwedge_{i=1}^{n_s} \mathsf{G}_{[a_i^s, b_i^s]} \gamma_i^s}_{\text{safety formula}} \wedge \underbrace{\textstyle\bigwedge_{j=1}^{n_p} \mathsf{G}_{[a_j^p, b_j^p]} \mathsf{F}_{[0, c_j^p]} \gamma_j^p}_{\text{progress formula}} \wedge \underbrace{\mathsf{F}_{[a^t, b^t]} \mathsf{G}_{[0, c^t]} \gamma^t}_{\text{target formula}}, \tag{5}$$

where $\mathsf{G}_{[a_i^s, b_i^s]} \gamma_i^s$, $\mathsf{G}_{[a_j^p, b_j^p]} \mathsf{F}_{[0, c_j^p]} \gamma_j^p$, and $\mathsf{F}_{[a^t, b^t]} \mathsf{G}_{[0, c^t]} \gamma^t$ are the *safety*, *progress*, and *target* subformulas, for $i \in$

$\{1,2,\cdots,n_s\}$ and $j\in\{1,2,\cdots,n_p\}$, $n_s,n_p\in\mathbb{N}^+$, γ_i^s, γ_j^p, and γ^t are the boolean formulas that only contain predicates connected with logical operators $\neg$, $\wedge$, and $\vee$, and $[a_i^s,b_i^s]$, $[a_j^p,b_j^p]$, and $[a^t,b^t]$ are the non-empty *syntactic intervals* of the subformulas. We also refer to $[a_i^s,b_i^s]$, $[a_j^p,b_j^p+c_j^p]$, and $[a^t,b^t+c^t]$ which represent the complete coverage of the subformulas as their *complete intervals*, making a clear distinguishment with the *syntactic intervals*.

Similar to the popularly used GR(1) specifications [21], the fragment φ defined above is a complex STL formula composed of three components representing meaningful specifications for practical tasks. The *safety* part consists of a series of *always* subformulas specifying the conditions that should always hold. This could include for example safety rules applicable to the system. The *progress* component contains the always subformulas with eventual operators embedded to represent the tasks that should be performed regularly, such as the monitoring routines. The *target* component describes the task that should be achieved within a strict deadline. An always formula is embedded to ensure the holding of the target condition for a minimum time.

Specifications in the form of eq. (5) already have a natural division in subformulas for individual subtasks. However, modularized synthesis also requires the division of the subformulas in time. Given a set of ordered timing points $\kappa_1<\kappa_2<\cdots<\kappa_l$, $\kappa_z\in\mathbb{N}$, $z\in\{1,2,\cdots,l\}$, $l\in\mathbb{N}$, we intend to split the specification φ into subformulas φ_z with shorter lengths. Then, modularized synthesis expects to efficiently solve the synthesis problem for φ through a sequence of synthesis problems for its subformulas φ_z. To achieve this, the overlapping between the timing intervals of the subformulas should be eliminated to decouple the dependence of different timings. In the next section, we will show that the decoupling of the safety subformulas can be achieved by syntactic separation while ensuring the syntactic equivalence between the separated specification and the original one. However, the progress and the target subformulas are challenging to decouple since the timing overlapping cannot be eliminated by merely using syntactic separation. This has not been well investigated by existing work. We will also show that these subformulas can be decoupled using the complete specification split which ensures soundness but introduces conservativeness. Our work is the first to decompose such STL formulas for modularized synthesis in the literature.

III. Main Results

In this section, we will first show how to split the syntactic intervals of a complex specification while ensuring the syntactic equivalence. Then, we further present a complete split form to eliminate the overlapping of the complete intervals of the subformulas while causing certain conservativeness. Finally, we give the modularized synthesis algorithm based on these separation forms.

A. Syntactic Timing Separation

The syntactic separation form of a complex specification in eq. (5) is defined as follows.

Definition 1 (Syntactic Timing Separation): Given an ordered sequence of timing $0=\kappa_0<\kappa_1<\cdots<\kappa_l=L$, $\kappa_z\in\mathbb{N}$, for $z\in\{1,2,\cdots,l-1\}$, $l\in\mathbb{N}$, the specification φ defined in eq. (5) is said to be in a syntactic timing separation form if

$$\varphi:=\textstyle\bigwedge_{z=1}^{l}\phi_z\wedge\bigvee_{z=1}^{l}\phi_z^t, \tag{6}$$

where, for each $z\in\{1,2,\cdots,l\}$,

$$\phi_z=\underbrace{\textstyle\bigwedge_{i=1}^{n_{s,z}}\mathsf{G}_{[a_{z,i}^s,b_{z,i}^s]}\gamma_i^s}_{\text{safety subformula}}\wedge\underbrace{\textstyle\bigwedge_{j=1}^{n_{p,z}}\mathsf{G}_{[a_{z,j}^p,b_{z,j}^p]}\mathsf{F}_{[0,c_{z,j}^p]}\gamma_j^p}_{\text{progress subformula}},$$

$$\phi_z^t=\underbrace{\mathsf{F}_{[a_z^t,b_z^t]}\mathsf{G}_{[0,c_z^t]}\gamma^t}_{\text{target subformula}}\ \text{or}\ \phi_z^t=\neg\top,$$

and all intervals associated to $z\in\{1,2,\cdots,l\}$ satisfy $[a_{z,i}^s,b_{z,i}^s]\subset[\kappa_{z-1},\kappa_z]$, $\forall i\in\{1,2,\cdots,n_{s,z}\}$, $[a_{z,j}^p,b_{z,j}^p]\subset[\kappa_{z-1},\kappa_z]$, $\forall j\in\{1,2,\cdots,n_{p,z}\}$, and $[a_z^t,b_z^t]\subset[\kappa_{z-1},\kappa_z]$, where $n_{s,z},n_{p,z}\in\mathbb{N}$ are the numbers of the effective safety and progress subformulas for $[\kappa_{z-1},\kappa_z]$. □

Consider the following specifications with the same length 6: $\varphi_1=\mathsf{G}_{[0,4]}\gamma_0\wedge\mathsf{G}_{[2,6]}\gamma_1$, $\varphi_2=\mathsf{G}_{[0,2]}\mathsf{F}_{[0,1]}\gamma_0\wedge\mathsf{G}_{[2,6]}\gamma_1$, $\varphi_3=\mathsf{G}_{[0,4]}\gamma_0\wedge\mathsf{F}_{[0,4]}\mathsf{G}_{[0,2]}\gamma_1$, and $\varphi_4=\mathsf{G}_{[0,2]}\mathsf{F}_{[0,1]}\gamma_0\wedge\mathsf{F}_{[3,5]}\mathsf{G}_{[1,1]}\gamma_1$, where γ_0, γ_1 are boolean formulas. Consider splitting points $\kappa_0=0$, $\kappa_1=2$, $\kappa_2=6$, according to Definition 1, only φ_2 and φ_4 are in a form that is syntactically separated by κ_0, κ_1, κ_2. Formulas φ_1, φ_3 are not since κ_1 splits the intervals $[0,4]$.

We are interested in translating the specification φ (5) into the separated form of (6) using syntactic separation. In this way, the individual numbers of the safety and progress specifications for each timing interval $z\in\{1,2,\cdots,l\}$, denoted as $n_{s,z}$, $n_{p,z}$, can be substantially smaller than those of the corresponding safety and progress specifications, i.e., n_s, n_p. The following syntactic separation principles can be used to transform (5) into (6) with syntactic equivalence guaranteed.

Lemma 1 ([14]): The following properties hold for arbitrary STL formulas φ, φ_1, and φ_2, with $\kappa\in\mathbb{N}$, $\kappa<a$: $\mathsf{F}_{\{\kappa\}}(\neg\varphi)=\neg\mathsf{F}_{\{\kappa\}}\varphi$, $\mathsf{F}_{\{\kappa\}}(\varphi_1\wedge\varphi_2)=\mathsf{F}_{\{\kappa\}}\varphi_1\wedge\mathsf{F}_{\{\kappa\}}\varphi_2$, $\mathsf{F}_{\{\kappa\}}(\varphi_1\mathsf{U}_{(a,b)}\varphi_2)=\mathsf{F}_{\{\kappa\}}\varphi_1\mathsf{U}_{(a,b)}\mathsf{F}_{\{\kappa\}}\varphi_2$, $\varphi\mathsf{U}_{(a,b)}(\varphi_1\vee\varphi_2)=\varphi\mathsf{U}_{(a,b)}\varphi_1\vee\varphi\mathsf{U}_{(a,b)}\varphi_2$. □

Lemma 2: The following conditions hold for arbitrary STL formulas φ, φ_1, and φ_2 defined in Sec. II-A.

1). $\mathsf{F}_{\{\kappa\}}\varphi=\mathsf{G}_{\{\kappa\}}\varphi$, for any $\kappa\in\mathbb{N}$, where both sides are true for signal $\mathbf{x}$, if and only if $(\mathbf{x},\kappa)\vDash\varphi$.

2). $\mathsf{G}_{\{\kappa\}}(\varphi_1\vee\varphi_2)=\mathsf{G}_{\{\kappa\}}\varphi_1\vee\mathsf{G}_{\{\kappa\}}\varphi_2$ holds for any $\kappa\in\mathbb{N}$.

3). For any $a,b\in\mathbb{N}$, $a\leq b$, $\mathsf{G}_{\{\kappa\}}\big(\mathsf{G}_{[a,b]}\varphi\big)=\mathsf{G}_{[\kappa+a,\kappa+b]}\varphi$ and $\mathsf{F}_{\{\kappa\}}\big(\mathsf{F}_{[a,b]}\varphi\big)=\mathsf{F}_{[\kappa+a,\kappa+b]}\varphi$ hold for $\kappa\in\mathbb{N}$, $\kappa<a$. □

Lemma 3: Given $a,b\in\mathbb{N}$, $a\leq b$ and an arbitrary STL formula φ, $\mathsf{G}_{[a,b]}\varphi=\mathsf{G}_{\{a\}}\varphi\wedge\mathsf{G}_{(a,b)}\varphi\wedge\mathsf{G}_{\{b\}}\varphi$ and $\mathsf{F}_{[a,b]}\varphi=\mathsf{F}_{\{a\}}\varphi\vee\mathsf{F}_{(a,b)}\varphi\vee\mathsf{F}_{\{b\}}\varphi$ hold. □

Theorem 1 (Arbitrary Syntactic Separation): Given $\kappa\in\mathbb{N}$, the following conditions hold for an STL formula φ,

$$\mathsf{G}_{[a,b]}\varphi=\mathsf{G}_{[a,\kappa]}\varphi\wedge\mathsf{G}_{[\kappa,b]}\varphi,\ \mathsf{F}_{[a,b]}\varphi=\mathsf{F}_{[a,\kappa]}\varphi\vee\mathsf{F}_{[\kappa,b]}\varphi, \tag{7}$$

with $a\leq\kappa\leq b$. Moreover, the following conditions hold,

$$\mathsf{G}_{[\kappa_0,\kappa_l]}\varphi=\textstyle\bigwedge_{i=1}^{l}\mathsf{G}_{[\kappa_{i-1},\kappa_i]}\varphi,\ \mathsf{F}_{[\kappa_0,\kappa_l]}\varphi=\bigvee_{i=1}^{l}\mathsf{F}_{[\kappa_{i-1},\kappa_i]}\varphi, \tag{8}$$

for κ_0, κ_1, $\cdots$, $\kappa_l\in\mathbb{N}$, $\kappa_0<\kappa_1<\cdots<\kappa_l$. □

Lemmas 2 and 3 provide complementary properties to previous work on syntactic separation [14], [15]. Note that they apply to all STL formulas as introduced in Sec. II-

A, but not only the fragment in eq. (5). Most important is theorem 1 which allows separating a subformula into the logical combination of shorter subformulas with an arbitrary number of timing points. Such separation as eq. (6) does not change the syntax of the specification. i.e., for any signal $\mathbf{x}_{[0,L]}$ with $L=\mathscr{L}(\varphi)$, $\mathbf{x}\vDash\varphi\leftrightarrow\mathbf{x}\vDash\bigwedge_{z=1}^{l}\phi_z\wedge\bigvee_{z=1}^{l}\phi_z^t$. Nevertheless, syntactic timing separation only splits up the syntactic interval of a subformula, which does not eliminate the overlapping between the complete intervals of the subformulas. This is not sufficient for the modularized synthesis of specifications. In the following, we will give an alternative sufficient form of separation that is no longer equivalent to the original specification but ensures the separation of the complete intervals of the subformulas. This form is more conservative than the original specification but ensures the soundness of the solution and allows for modularized solutions to the synthesis problem.

B. Complete Specification Split for Modularized Checking

Before we give the complete splitting form of specification (5), we first explain why eliminating the overlapping of the complete intervals of subformulas is important to modularized model checking which is the foundation of modularized synthesis to be explained in the following. Consider an STL specification φ and a signal prefix $\mathbf{x}$ with the same length. For a given series of timing points $\kappa_0,\kappa_1,\cdots,\kappa_l$, modularized model checking investigates under what conditions and what subformulas $\bar{\varphi}_1$, $\bar{\varphi}_2$, $\cdots$, $\bar{\varphi}_l$, where $\mathscr{L}(\bar{\varphi}_z)=\kappa_z-\kappa_{z-1}$ for all $z\in\{1,2,\cdots,l\}$, it ensures that

$$\mathbf{x}_{[\kappa_{z-1},\kappa_z]}\vDash\bar{\varphi}_z,\ \text{for some } z\in\{1,2,\cdots,l\}\rightarrow\mathbf{x}\vDash\varphi. \tag{9}$$

In such a way, we can split the model checking of the original signal $\mathbf{x}$ and specification φ into l-steps of model checking for shorter signals $\mathbf{x}_{[\kappa_{z-1},\kappa_z]}\vDash\bar{\varphi}_z$ and specifications $\bar{\varphi}_z$. This is only feasible when the coverage or the complete interval of the subformulas $\bar{\varphi}_z$ is confined within the corresponding interval $[\kappa_{z-1},\kappa_z]$ such that it does not overlap with those of other subformulas. Otherwise, the model checking for the left side of (9) can not be performed independently for each $z\in\{1,2,\cdots,l\}$ due to the coupled timing dependence.

Based on this consideration, we give the complete specification split form for formula φ in eq. (5) as

$$\bar{\varphi}:=\bigwedge_{z=1}^{l}\bar{\phi}_z\wedge\bigvee_{z=1}^{l}\bar{\phi}_z^t, \tag{10}$$

where, for any $z\in\{1,2,\cdots,l\}$,

$$\bar{\phi}_z:=\bigwedge_{i=1}^{n_{s,z}}\mathsf{G}_{[a_{z,i}^s,b_{z,i}^s]}\gamma_i^s\wedge\bigwedge_{j=1}^{n_{p,z}}\mathsf{G}_{[a_{z,j}^p,\min\{b_{z,j}^p,\kappa_z-c_{z,j}^p\}]}\mathsf{F}_{[0,c_{z,j}^p]}\gamma_j^p$$
$$\wedge\bigwedge_{r=1}^{\hat{n}_{p,z}}\mathsf{F}_{[\kappa_z-\tau_{z,r},\kappa_z]}\gamma_j^p\wedge\bigwedge_{q=1}^{\hat{n}_{p,z-1}}\mathsf{F}_{[\kappa_{z-1},\kappa_{z-1}+c_{z-1,q}^p-\tau_{z-1,q}]}\gamma_j^p$$
$$\bar{\phi}_z^t:=\mathsf{F}_{[a_z^t,\min\{b_z^t,\kappa_z-c_z^t\}]}\mathsf{G}_{[0,c_z^t]}\gamma^t\ \text{or}\ \bar{\phi}_z^t=\neg\top,$$

where $\hat{n}_{p,z}$ for any $z\in\{1,2,\cdots,l\}$, is the number of $j\in\{1,2,\cdots,n_{p,z}\}$ such that $b_{z,j}^p+c_{z,j}^p>\kappa_z$, i.e., the number of progress formulas that exceed the interval $[\kappa_{z-1},\kappa_z]$, and $\tau_{z,r}\in[0,c_{z,r}^p]$ for any $r\in\{1,2,\cdots,\hat{n}_{p,z}\}$, is a heuristic value to be determined beforehand. It can be verified that $\mathscr{L}(\bar{\varphi})=\kappa_l=\mathscr{L}(\varphi)$. Then, we have the following two theorems to address the relation between the complete split form $\bar{\varphi}$ in eq. (10) and the syntactic separation form φ in eq. (6).

Lemma 4 (Complete Interval Split): Given $0=\kappa_0<\kappa_1<\cdots<\kappa_l=L$, $\kappa_z\in\mathbb{N}$, $z\in\{1,2,\cdots,l-1\}$, $l\in\mathbb{N}$ and a specification $\bar{\varphi}$ in form (10), if $\kappa_z-\kappa_{z-1}\geq c_{z,j}^p$ for all $j\in\{1,2,\cdots,n_{p,z}\}$ and for all $z\in\{1,2,\cdots,l\}$, the complete intervals of $\bar{\phi}_1$, $\bar{\phi}_2$, $\cdots$, $\bar{\phi}_l$ do not overlap, and the complete intervals $\bar{\phi}_1^t$, $\bar{\phi}_2^t$, $\cdots$, $\bar{\phi}_l^t$ do not overlap.

Lemma 5 (Soundness): For a specification φ in eq. (6) and its complete split form $\bar{\varphi}$ in eq. (10) with the splitting timing points κ_0, κ_1, $\cdots$, κ_l as described in *lemma 4*, any signal prefix $\mathbf{x}$ with length $\mathscr{L}(\varphi)$ holds that $\mathbf{x}\vDash\bar{\varphi}\rightarrow\mathbf{x}\vDash\varphi$.

Theorem 2: For a specification φ in eq. (6) and its complete split form $\bar{\varphi}$ in eq. (10) with the splitting timing points κ_0, κ_1, $\cdots$, κ_l as described in *lemma 4*, $\mathbf{x}\vDash\varphi$ holds for signal $\mathbf{x}$ with length $\mathscr{L}(\varphi)$ if the following conditions both hold,
1). $\mathbf{x}_{[\kappa_{z-1},\kappa_z]}\vDash\mathsf{G}_{\{-\kappa_{z-1}\}}\bar{\phi}_z$, $\forall\, z\in\{1,2,\cdots,l\}$;
2). $\mathbf{x}_{[\kappa_{z-1},\kappa_z]}\vDash\mathsf{G}_{\{-\kappa_{z-1}\}}\bar{\phi}_z^t$, $\exists\, z\in\{1,2,\cdots,l\}$. □

Theorem 2 has solved the main problem of modularized model checking for specification φ given in eq. (5) by eliminating the overlapping between the complete intervals of its subformulas, as addressed by lemma 4. From a practical perspective, the overlapping means that the timing coupling between different subtasks specifies that these subtasks need to be executed in parallel. In this sense, theorems (2) provide a solution to decouple such dependence by imposing additional specifications to the subtasks, such that they can be solved independently in sequence. The soundness of the complete interval split is ensured by lemma 5. The timing points that mark the solving sequence can be predetermined according to practical requirements.

C. Modularized Synthesis of Split Specifications

Given the complete specification split form $\bar{\varphi}$ in eq. (10) for modularized model checking, we can further investigate the modularized synthesis by incorporating the constraints brought up by the dynamic systems (2). For this, we develop algorithm 1 for modularized synthesis of a split specification. In algorithm 1, *opt()* is a function of the optimization problem in eq. (3) with interface (x_0,L,φ), and FEASIBLE is a binary variable to indicate whether problem (3) is feasible. Algorithm 1 allows us to perform synthesis for the dynamic system (2) with specification $\bar{\varphi}$ in a modularized way, i.e., by solving a sequence of smaller synthesis problems in a timing order κ_1, κ_2, $\cdots$, κ_l. For each time κ_z, $z\in\{1,2,\cdots,l\}$, the synthesis subproblem requires substantially fewer integers than the original problem since it involves much shorter and fewer specifications.

Complexity Analysis: As addressed in Sec. II-C, the complexity of directly synthesizing the original specification φ in eq. 5 is $O(2^{NL})$, where $L=\mathscr{L}(\varphi)$ is the length of φ and $N:=\max_{x\in\{1,2,\cdot,l\}}(n_s+n_p+1)$ is the total number of subformulas. For its complete split form $\bar{\varphi}$ in eq. (10), assume that the longest subformula has a length $\bar{L}=\max_{z\in\{1,2,\cdots,l\}}(\kappa_z-\kappa_{z-1})$, the complexity of algorithm 1 is $O(l\cdot 2^{\bar{N}\bar{L}})$, where $\bar{N}:=\max_{z\in\{1,2,\cdots,l\}}(n_{s,z}+n_{p,z}+\hat{n}_{p,z}+\hat{n}_{p,z-1})$ denotes the maximum number of subformulas in one synthesis module $z\in\{1,2,\cdots,l\}$. As addressed in Sec. III-A and Sec. III-B, from syntactic separation we can expect

$n_{s,z} \ll n_s$ and $\hat{n}_{p,z} < n_{p,z} \ll n_p$ for any $z \in \{1,2,\cdots,l\}$, which leads to $\bar{N} \ll N$. Moreover, we can also ensure $\bar{L} \ll L$ by properly selecting the timing points κ_0, κ_1, $\cdots$, κ_l. Thus, with $2^{\bar{N}\bar{L}} \ll 2^{NL}$, modularized synthesis can substantially reduce the complexity of the synthesis problem for long and complex specifications and improve its efficiency.

Algorithm 1 Modularized Synthesis of Specification $\bar{\varphi}$

Input: Initial system condition x_0, $\kappa_0 = 0$, splitting timing points κ_z and subformulas $\bar{\phi}_z$, $\bar{\phi}_z^t$, for $z \in \{1,2,\cdots,l\}$.
Output: Control signal $\mathbf{u}$ and state signal $\mathbf{x}$.

$x_{\kappa_0} \leftarrow x_0$
for $z = 1$ **to** l **do**
  $L_z \leftarrow \kappa_z - \kappa_{z-1}$
  if $z > 1$ and $\mathbf{x}_{[\kappa_0,\kappa_{z-1}]} \vDash \vee_w^{z-1} \bar{\phi}_w^t$ **then**
    $\mathbf{x}_{[\kappa_{z-1}+1,\kappa_z]}, \mathbf{u}_{[\kappa_{z-1},\kappa_z-1]} \leftarrow \text{opt}(x_{\kappa_{z-1}}, L_z, \bar{\phi}_z)$
  else
    $\mathbf{x}_{[\kappa_{z-1}+1,\kappa_z]}, \mathbf{u}_{[\kappa_{z-1},\kappa_z-1]} \leftarrow \text{opt}(x_{\kappa_{z-1}}, L_z, \bar{\phi}_z \wedge \bar{\phi}_z^t)$
    if not FEASIBLE **then**
      $\mathbf{x}_{[\kappa_{z-1}+1,\kappa_z]}, \mathbf{u}_{[\kappa_{z-1},\kappa_z-1]} \leftarrow \text{opt}(x_{\kappa_{z-1}}, L_z, \bar{\phi}_z)$
    end if
  end if
end for
$\mathbf{u} \leftarrow \mathbf{u}_{[\kappa_0,\kappa_l-1]}$, $\mathbf{x} \leftarrow \mathbf{x}_{[\kappa_0,\kappa_l]}$

Limitations: Nevertheless, a limitation of algorithm 1 is that it only ensures soundness but not optimality nor completeness to the original problem. This means that if it generates a feasible solution $\mathbf{x}$, it is certainly a feasible solution to the synthesis problem of the original specification, i.e., $\mathbf{x} \vDash \varphi$ (soundness). However, it might not be the optimal solution in terms of the robustness $\rho(\mathbf{x},\varphi)$, i.e., local optimality does not necessarily lead to global optimality. Moreover, if algorithm 1 is not feasible, it does not mean that the original synthesis problem $\mathbf{x} \vDash \varphi$ is also infeasible (completeness). However, this is already sufficient for most practical robotic tasks. It is also worth noting that algorithm 1 might not be feasible for an arbitrary initial system condition x_0. For a system (3) and a specification $\bar{\varphi}$ in eq. (10), the initial condition x_0 that ensures the feasibility of $\mathbf{x} \vDash \varphi$ belongs to a set which is referred to as the *largest satisfaction region* [22]. How to utilize the feasible sets to improve the feasibility of a synthesis problem is also partially discussed in a recent work [16]. We are not providing further discussions on this since it is out of the scope of this paper.

IV. Case Study in Simulation

In this section, we use an essential simulation study to showcase how the proposed modularized synthesis approach can be used to efficiently solve a synthesis problem for a complex specification. As shown in Fig. 1, we consider a scenario where a mobile robot is required to perform a monitoring task in a rectangular space SAFETY sized 8×7 (red) with three square regions TARGET (yellow), HOME (green), and CHANGER (blue) which are centered at $(2,5)$, $(6,5)$, and $(6,2)$ with the same side length 2. The robot is described as the following dynamic model,

$$\zeta_{k+1} = \zeta_k + u_k, \ k \in \mathbb{N}, \tag{11}$$

where $\zeta_k \in \mathbb{R}^2$ denotes the planar coordinate of the robot position at time step k and $u_k \in \mathbb{R}^2$ is the position increments of the robot per step as the control input of the system. The control input of the system is subject to saturation constraints $|u_{k,1}| \le 1$, $|u_{k,2}| \le 1$ for all $k \in \mathbb{N}$, where $u_{k,1}, u_{k,2} \in \mathbb{R}$ are the elements of u_k. The monitoring task is described as follows.

1). Starting from position $(0,5)$, the robot should frequently visit TARGET every 5 steps or fewer until $k = 40$.

2). From $k = 15$ to $k = 45$, once the robot leaves HOME, it should get back to HOME within 5 time steps.

3). After $k = 20$ and before $k = 45$, it should stay in CHANGER continuously for at least 3 time steps to charge.

4). The robot should always stay in the SAFETY region.

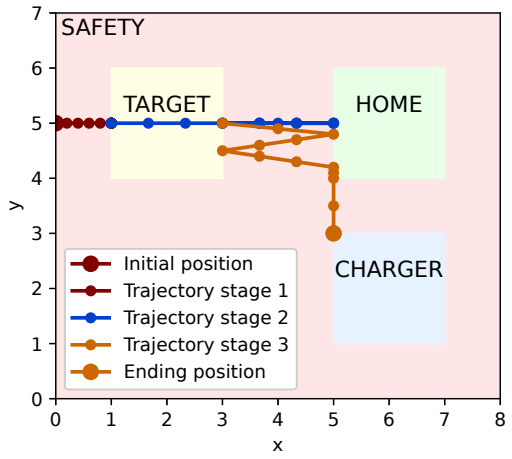

Fig. 1. The illustration of the robot monitoring scenario with the spatial information of the synthesized trajectory subject to specification $\bar{\varphi}$.

These tasks can be specified using the following formulas: $\mathsf{G}_{[0,35]}\mathsf{F}_{[0,5]}\gamma_T$, $\mathsf{G}_{[15,40]}\mathsf{F}_{[0,5]}\gamma_H$, $\mathsf{F}_{[20,42]}\mathsf{G}_{[0,3]}\gamma_C$, and $\mathsf{G}_{[0,45]}\gamma_S$ respectively, where γ_T, γ_H, γ_C, and γ_S are boolean formulas used to specify $\xi_k \in$TARGET, $\xi_k \in$HOME, $\xi_k \in$CHARGER, and $\xi_k \in$SAFETY, for $k \in \mathbb{N}$. Thus, the overall robot task is the conjunction of these formulas. With splitting timing points $\kappa_0 = 0$, $\kappa_1 = 15$, $\kappa_2 = 30$, and $\kappa_3 = 45$, the overall specification can be represented as a syntactic separation form as eq. (6), i.e., $\varphi := \bigwedge_{z=1}^3 \phi_z \wedge \bigwedge_{z=1}^3 \phi_z^t$, where $\phi_1 = \mathsf{G}_{[0,15]}\gamma_S \wedge \mathsf{G}_{[0,15]}\mathsf{F}_{[0,5]}\gamma_T$, $\phi_2 = \mathsf{G}_{[15,30]}\gamma_S \wedge \mathsf{G}_{[15,30]}\mathsf{F}_{[0,5]}\gamma_T \wedge \mathsf{G}_{[15,30]}\mathsf{F}_{[0,5]}\gamma_H$, $\phi_3 = \mathsf{G}_{[30,45]}\gamma_S \wedge \mathsf{G}_{[30,35]}\mathsf{F}_{[0,5]}\gamma_T \wedge \mathsf{G}_{[30,40]}\mathsf{F}_{[0,5]}\gamma_H$, $\phi_1^t = \neg\top$, $\phi_2^t = \mathsf{F}_{[20,30]}\mathsf{G}_{[0,3]}\gamma_C$, $\phi_3^t = \mathsf{F}_{[30,42]}\mathsf{G}_{[0,3]}\gamma_C$.

We transform the overal specification φ into a complete split form as eq. (10), i.e., $\bar{\varphi} := \bigwedge_{z=1}^3 \bar{\phi}_z \wedge \bigwedge_{z=1}^3 \bar{\phi}_z^t$, where $\bar{\phi}_1 = \mathsf{G}_{[0,15]}\gamma_S \wedge \mathsf{G}_{[0,10]}\mathsf{F}_{[0,5]}\gamma_T \wedge \mathsf{F}_{[12,15]}\gamma_T$, $\bar{\phi}_2 = \mathsf{G}_{[15,30]}\gamma_S \wedge \mathsf{F}_{[15,17]}\gamma_T \wedge \mathsf{G}_{[15,25]}\mathsf{F}_{[0,5]}\gamma_T \wedge \mathsf{F}_{[27,30]}\gamma_T \wedge \mathsf{G}_{[15,25]}\mathsf{F}_{[0,5]}\gamma_H \wedge \mathsf{F}_{[27,30]}\gamma_H$, $\bar{\phi}_3 = \mathsf{G}_{[30,45]}\gamma_S \wedge \mathsf{F}_{[30,32]}\gamma_T \wedge \mathsf{G}_{[30,35]}\mathsf{F}_{[0,5]}\gamma_T \wedge \mathsf{F}_{[30,32]}\gamma_H \wedge \mathsf{G}_{[30,40]}\mathsf{F}_{[0,5]}\gamma_H$, $\bar{\phi}_1^t = \neg\top$, $\bar{\phi}_2^t = \mathsf{F}_{[20,27]}\mathsf{G}_{[0,3]}\gamma_S$, $\bar{\phi}_3^t = \mathsf{F}_{[30,42]}\mathsf{G}_{[0,3]}\gamma_S$, where the heuristic τ values are all determined as 3. Then, we use Algorithm 1 to solve an open-loop control signal for system (11) with the split specification $\bar{\varphi}$. The *stlpy* toolbox [7] is used to implement the *opt()*

method in algorithm 1. The program for this simulation study is published at [23]. The resulting robot trajectory ζ_k is shown in Fig. 1 and Fig. 2. The trajectories in different stages are marked with different colors.

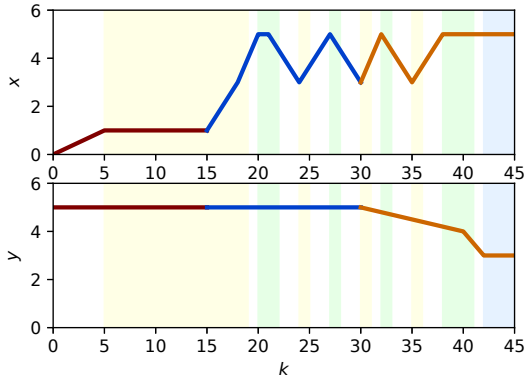

Fig. 2. The x- and y-positions of the robot trajectory in three stages. The color in the background indicates in which region the robot stays, namely yellow for TARGET, green for HOME, and blue for CHARGER, which is consistent with Fig. 1.

From Fig. 1 and Fig. 2, it can be seen that the robot starts from the initial position $(0,5)$, reaches the TARGET at $k=5$ and stays there until $k=15$. From $k=15$, the robot oscillates between TARGET and HOME to satisfy the task requirements 1) and 2). After $k=40$, the robot maintains the visiting frequency to HOME, while taking time to charge itself, which satisfies condition 3). During the entire process, the robot is restricted within the SAFE region, which satisfies specification 4). Therefore, all task specifications are satisfied, which indicates the efficacy of the proposed timing separation approaches and modularized synthesis methods.

V. CONCLUSIONS

In this paper, we discuss how to split a big synthesis problem for a complex and long STL specification into several smaller optimization problems with less complexity. The two proposed separation forms for the specification, namely a syntactically separated form and a complete splitting form, allow us to solve these smaller problems in a modularized manner, which is an important step toward efficient optimization-based specification synthesis. There are still two limitations of our work. One is that we only investigate the modularized synthesis for a certain class of STL formulas, although it is sufficient for many practical tasks. The other one is that the feasibility condition of modularized synthesis has not been deeply studied. Our future work will extend the results to wider fragments of STL formulas. We will also incorporate the feasible sets of specifications to investigate the feasibility of modularized synthesis.

REFERENCES

[1] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI communications*, vol. 29, no. 1, pp. 151–162, 2016.

[2] X. Li, G. Rosman, I. Gilitschenski, C.-I. Vasile, J. A. DeCastro, S. Karaman, and D. Rus, "Vehicle trajectory prediction using generative adversarial network with temporal logic syntax tree features," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3459–3466, 2021.

[3] 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 *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 81–87.

[4] 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.

[5] L. Lindemann, G. J. Pappas, and D. V. Dimarogonas, "Reactive and risk-aware control for signal temporal logic," *IEEE Transactions on Automatic Control*, vol. 67, no. 10, pp. 5262–5277, 2021.

[6] A. Salamati, S. Soudjani, and M. Zamani, "Data-driven verification of stochastic linear systems with signal temporal logic constraints," *Automatica*, vol. 131, p. 109781, 2021.

[7] 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.

[8] M. Srinivasan and S. Coogan, "Control of mobile robots using barrier functions under temporal logic specifications," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 363–374, 2020.

[9] 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.

[10] D. Gundana and H. Kress-Gazit, "Event-based signal temporal logic synthesis for single and multi-robot tasks," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3687–3694, 2021.

[11] S. Liu, A. Saoud, P. Jagtap, D. V. Dimarogonas, and M. Zamani, "Compositional synthesis of signal temporal logic tasks via assume-guarantee contracts," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 2184–2189.

[12] S. Alatartsev, S. Stellmacher, and F. Ortmeier, "Robotic task sequencing problem: A survey," *Journal of intelligent & robotic systems*, vol. 80, pp. 279–298, 2015.

[13] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.

[14] P. Hunter, J. Ouaknine, and J. Worrell, "Expressive completeness for metric temporal logic," in *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*. IEEE, 2013, pp. 349–357.

[15] K. Bae and J. Lee, "Bounded model checking of signal temporal logic properties using syntactic separation," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–30, 2019.

[16] 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.

[17] Z. Zhang and S. Haesaert, "Modularized control synthesis for complex signal temporal logic specifications," *arXiv:2303.17086*, 2023. [Online]. Available: https://arxiv.org/abs/2303.17086

[18] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems: Joint International Conferences on Formal Modeling and Analysis of Timed Systmes, FORMATS 2004, and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24, 2004. Proceedings*. Springer, 2004, pp. 152–166.

[19] L. Nenzi and L. Bortolussi, "Specifying and monitoring properties of stochastic spatio-temporal systems in signal temporal logic," *EAI Endorsed Transactions on Cloud Systems*, vol. 1, no. 4, 2015.

[20] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[21] M. Schlaipfer, G. Hofferek, and R. Bloem, "Generalized reactivity (1) synthesis without a monolithic strategy," in *Haifa Verification Conference*. Springer, 2011, pp. 20–34.

[22] C. Belta, B. Yordanov, E. Aydin Gol, C. Belta, B. Yordanov, and E. Aydin Gol, "Largest satisfying region," *Formal Methods for Discrete-Time Dynamical Systems*, pp. 119–139, 2017.

[23] Z. Zhang and H. Sofie, "Benchmark for modularized synthesis of complex specifications," *GitHub repository*, 2023. [Online]. Available: https://github.com/zhang-zengjie/modustl