Mixed Integer Linear Programming Approach for Control Synthesis with Weighted Signal Temporal Logic

Gustavo A. Cardona
Lehigh University
Bethlehem, PA, USA
gcardona@lehigh.edu

Disha Kamale
Lehigh University
Bethlehem, PA, USA
ddk320@lehigh.edu

Cristian-Ioan Vasile
Lehigh University
Bethlehem, PA, USA
cvasile@lehigh.edu

ABSTRACT

This work presents an optimization-based control synthesis approach for an extension of Signal Temporal Logic (STL) called weighted Signal Temporal Logic (wSTL). wSTL was proposed to accommodate user preferences for importance and priorities over concurrent and sequential tasks as well as satisfaction times denoted by weights over the logical and temporal operators, respectively. We propose a Mixed Integer Linear Programming (MILP) based approach for synthesis with wSTL specifications. These specifications have the same qualitative semantics as STL but differ in their quantitative semantics, which is recursively modulated with weights. Additionally, we extend the formal definition of wSTL to include the semantics for *until* and *release* temporal operators and present an efficient encoding for these operators in the MILP formulation. As opposed to the original implementation of wSTL, where the arithmetic-geometric mean robustness was used with gradient-based methods prone to local optima, our encoding allows the use of a weighted version of traditional robustness and efficient global MILP solvers. We demonstrate the operational performance of the proposed formulation using multiple case studies, showcasing the distinct functionalities over Boolean and temporal operators. Moreover, we elaborate on multiple case studies for synthesizing controllers for an agent navigating a non-convex environment under different constraints highlighting the difference in synthesized control plans for STL and wSTL. Finally, we compare the time and complexity performance of encodings for STL and wSTL.

CCS CONCEPTS

• **Computer systems organization** → **Robotic control**; • **Hardware** → *Safety critical systems*; • **Computing methodologies** → Computational control theory.

KEYWORDS

Temporal Logic, Control Synthesis, Mixed Integer Linear Programming, User Satisfaction Preferences

ACM Reference Format:
Gustavo A. Cardona, Disha Kamale, and Cristian-Ioan Vasile. 2023. Mixed Integer Linear Programming Approach for Control Synthesis with Weighted Signal Temporal Logic. In *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '23), May 09–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3575870.3587120

HSCC '23, May 09–12, 2023, San Antonio, TX, USA

ACM ISBN 979-8-4007-0033-0/23/05.
https://doi.org/10.1145/3575870.3587120

1 INTRODUCTION

Owing to their rich expressivity, temporal logic formalisms have proved to be useful in capturing complex, precise high-level mission specifications with logical and temporal modalities [5, 20]. There is a vast body of literature on verifying and synthesizing temporal logic formulae to provide guarantees on the system behavior. We consider the problem of formal control synthesis, which deals with generating a control signal that leads the system to reach the desired behavior per the given temporal logic specification. The synthesized control strategies can be optimized by considering a relevant cost function, thereby optimizing the behavior of the system subject to correctness constraints.

Linear Temporal Logic (LTL) [3] and its fragments have been widely used for control synthesis of dynamical systems [6, 33, 37, 38]. Although highly expressive and close to natural language, LTL is unable to capture explicit time constraints. Several temporal logic formalisms such as Signal Temporal Logic (STL) [23], Time Window Temporal Logic (TWTL) [36], Bounded Linear Temporal Logic (BLTL) [35], Metric Temporal Logic (MTL) [19] have been proposed that accommodate explicit time constraints. Among these, STL is defined over continuous signals and real predicates, and it has been widely used for verification [2], continuous-monitoring [10], multi-agent control synthesis [7, 9, 34], etc. Moreover, STL provides quantitative semantics, also known as the robustness of satisfaction, that indicates the margin of satisfaction or violation of the given specification [30]. Thus, the control synthesis of STL can be posed as an optimization problem where the objective is to maximize robustness. Various robustness functionals have been proposed in the STL literature, including traditional robustness, [12], cumulative robustness [14], AGM robustness [25], discrete-average space robustness [22], etc. In this paper, we limit the discussion to the definition of traditional robustness. However, STL implicitly assumes that all sub-parts of the specification have the same importance. For instance, consider a service robot tasked with the following mission specification:

Example 1. *"Always between 9 am to 10 am bring tea or preferably coffee to the office desk. Always between 12 to 1 pm, be as far from the office room as possible and visit the living room between times 2 pm to 3 pm, preferably towards the end of the period."*

Here the preferences for a particular alternative (e.g., coffee) and the preference for the execution of an activity within a given time interval (e.g., towards the end of the period) cannot be readily

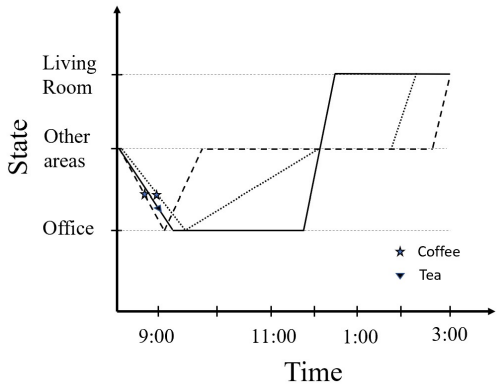


Figure 1: Possible trajectories for the specification in Example 1

expressed using standard temporal logic formalisms thus, limiting the specification of complex and demanding missions. As shown in Fig. 1, all three trajectories are valid, but only one closely follows the user specification. This has motivated the research toward satisfying temporal logic specifications robustly while also capturing user preferences. Existing works can be broadly classified into expressing user preferences using temporal logic formulae [4, 18], regular expressions [16], or integrating the preferences into the planning problem as (soft) constraints [1, 27, 28]. Weighted Signal Temporal Logic (wSTL) was proposed as an extension of STL to accommodate user preferences for satisfaction, priorities, and importance of subformulae directly within the specification captured using weights over logical and temporal operators.

The authors in [24] employed a gradient-based approach to synthesize control policies for maximizing the robustness of satisfaction. Several other approaches for optimization-based synthesis include heuristics or control barrier functions [39]. These methods are sound (a solution returned by the algorithm satisfies the specification) but not complete (it is not guaranteed to obtain a solution even if one exists) [21]. Given LTL specifications, [17] formulated as a Mixed Integer Linear Program (MILP) applied to LTL and has since been widely adopted for STL [8, 29, 31], MTL[32], etc. Although MILPs have exponential time complexity in the number of binary variables, it has been shown that efficiently encoding the constraints results in an efficient controller allowing for real-time synthesis [32].

In this work, we consider the problem of control synthesis for wSTL specifications that express mission specifications as well as user preferences for control synthesis of a given linear discrete-time dynamical system. Restricting ourselves to linear predicates, the problem is cast as a mixed integer linear program (MILP) where the goal is to maximize the robustness of satisfaction while minimizing the cost function associated with control execution. Note that our framework can capture user preferences for priorities over Boolean and temporal operators with the help of weights.

The main contributions of this work are

(1) We propose a MILP formulation that captures the qualitative and quantitative semantics of wSTL developed in [26]. Our MILP encoding captures weighted traditional robustness to quantify satisfaction instead of the AGM and gradient-based approach in [26].
(2) We extend the definition of wSTL [26] to include *Until* and *Release* operators.
(3) We demonstrate the versatility of wSTL specifications and the functionality of weights to modify the solution of an equivalent STL specification. Additionally, we present case studies for control synthesis using the proposed wSTL MILP formulation and compare it against equivalent STL specifications.
(4) Finally, we present the time performance comparison between the proposed MILP formulation and the STL MILP encoding presented in [31].

2 PRELIMINARIES AND NOTATION

Let $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{B}$ denote the sets of integer, real, and binary numbers sets, respectively. The set of integers and real numbers greater than a are $\mathbb{Z}_{\geq a}$ and $\mathbb{R}_{\geq a}$, respectively. For a set $\mathcal{S}$, $2^{\mathcal{S}}$ and $|\mathcal{S}|$ represent its power set and cardinality. For $\mathcal{S} \subseteq \mathbb{R}$ and $\alpha \in \mathbb{R}$, we denote by $\alpha + \mathcal{S}$ the set $\{\alpha + x \mid x \in \mathcal{S}\}$. The sign function is denoted by $\text{sign} : \mathbb{R} \to \{-1, 0, 1\}$. The empty set is denoted by $\emptyset$. Let $[a..b] = \mathbb{Z} \cap [a, b]$ denote the range of integers between a and b (inclusive). For a range $I = [a..b]$, we use $\underline{I} = a$ and $\bar{I} = b$.

2.1 Signal Temporal Logic (STL)

STL was introduced in [23] to monitor the temporal properties of real-valued signals. A *signal* s is a function $s : \mathbb{Z}_{\geq 0} \to \mathbb{R}^n$ that maps each time point $k \in \mathbb{Z}_{\geq 0}$ to an n-dimensional vector of real values $s(k)$, with s_i being its i-th component.

Definition 1 (STL syntax). *The syntax of STL in Backus-Naur form over linear predicates is*

$$\phi ::= \top \mid \bot \mid \mu \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \Diamond_I \phi \mid \Box_I \phi \mid \phi_1 \mathcal{U}_I \phi_2 \mid \phi_1 \mathcal{R}_I \phi_2$$

where $\top$ and $\bot$ are the logical *True* and *False*; μ is a linear *predicate* of the form $s_i \geq \pi$, with threshold π over the i-th component of signal s; $\neg$, $\wedge$, and $\vee$ are the Boolean *negation*, *conjunction*, and *disjunction* operators, respectively. $\Diamond$ (*eventually*), $\Box$ (*always*), $\mathcal{U}$ (*until*), and $\mathcal{R}$ (*release*) are temporal operators with time bound in the range I. $\Diamond_I \phi$ is satisfied if "at least at some time point in I the specification ϕ is true" while $\Box_I \phi$ requires "ϕ must be true at all times in I". The satisfaction of $\phi_1 \mathcal{U}_I \phi_2$ requires that "ϕ_1 must hold at least until ϕ_2 turns true within time range I". Whereas $\phi_1 \mathcal{R}_I \phi_2$ means that "ϕ_1 releases ϕ_2 if ϕ_2 holds within I up until and including a time instance in I at which ϕ_1 is true". Note that if ϕ_1 does not become true, ϕ_2 must hold for the entire duration I.

The (qualitative) semantics of STL formulae describes whether a signal s satisfies ϕ, and it is defined recursively for every Boolean and temporal operator as follows

Definition 2 (Qualitative semantics). *Given an STL specification ϕ and a signal s satisfaction of the specification at time k by*

signal s is defined by

$$\begin{aligned}
(s,k) &\models (s_i \geq \mu) \equiv s_i(k) \geq \mu,\\
(s,k) &\models \neg\phi \equiv (s,k) \not\models \phi,\\
(s,k) &\models \phi_1 \wedge \phi_2 \equiv \big((s,k) \models \phi_1\big) \wedge \big((s,k) \models \phi_2\big),\\
(s,k) &\models \phi_1 \vee \phi_2 \equiv \big((s,k) \models \phi_1\big) \vee \big((s,k) \models \phi_2\big),\\
(s,k) &\models \Diamond_I \phi \equiv \exists k' \in k+I \text{ s.t. } (s,k') \models \phi,\\
(s,k) &\models \Box_I \phi \equiv \forall k' \in k+I \text{ s.t. } (s,k') \models \phi,\\
(s,k) &\models \phi_1 \mathcal{U}_I \phi_2 \equiv \exists k' \in k+I \text{ s.t. } \big((s,k') \models \phi_2\\
&\qquad \wedge \forall k'' \in [k..k'] \ (s,k'') \models \phi_1\big),\\
(s,k) &\models \phi_1 \mathcal{R}_I \phi_2 \equiv \forall k' \in k+I \text{ s.t. } \big((s,k') \models \phi_2\\
&\qquad \vee \exists k'' \in [k..k'] \text{ s.t. } (s,k'') \models \phi_1\big),
\end{aligned} \tag{1}$$

where $\models$ and $\not\models$ denote satisfaction and violation, respectively. A signal s satisfying ϕ, denoted as $s \models \phi$, is true if $(s,0) \models \phi$.

In addition to the qualitative semantics of STL, the quantitative semantics, also called *robustness*, measures *how much* the signal s is satisfying or violating the specification ϕ at time k.

Definition 3 (Traditional Robustness). *Given a specification ϕ and a signal s, the robustness score $\rho(\phi, s, k)$ at time k is recursively defined as [12]:*

$$\begin{aligned}
\rho(\mu, s, k) &:= s_i(k) - \pi,\\
\rho(\neg\phi, s, k) &:= -\rho(\phi, s, k),\\
\rho(\phi_1 \wedge \phi_2, s, k) &:= \min\{\rho(\phi_1, s, k), \rho(\phi_2, s, k)\},\\
\rho(\phi_1 \vee \phi_2, s, k) &:= \max\{\rho(\phi_1, s, k), \rho(\phi_2, s, k)\},\\
\rho(\Diamond_I \phi, s, k) &:= \max_{k' \in k+I} \rho(\phi, s, k'),\\
\rho(\Box_I \phi, s, k)) &:= \min_{k' \in k+I} \rho(\phi, s, k'),\\
\rho(\phi_1 \mathcal{U}_I \phi_2, s, k) &:= \max_{k' \in k+I} \Big\{\min\{\rho(\phi_2, s, k'), \min_{k'' \in I'} \rho(\phi_1, s, k'')\}\Big\},\\
\rho(\phi_1 \mathcal{R}_I \phi_2, s, k) &:= \min_{k' \in k+I} \Big\{\max\{\rho(\phi_2, s, k'), \max_{k'' \in I'} \rho(\phi_1, s, k'')\}\Big\},
\end{aligned} \tag{2}$$

where $I' = [k..k']$.

Theorem 1. *The robustness score for an STL specification is sound, meaning that $\rho(\phi, s, k) > 0$ implies $s \models \phi$ that signal s satisfies ϕ at time k, and $\rho(\phi, s, k) < 0$ implies $s \not\models \phi$ that s violates ϕ at time k.*

We denote the robustness score of specification ϕ at time $k = 0$ with respect to the signal S by $\rho(\phi, s)$. We refer to this definition as the traditional STL robustness score.

The time horizon of an STL formula [11] is the minimum time horizon required to decide on the satisfaction or violation of any given signal. It is computed recursively as follows

$$\|\phi\| = \begin{cases} 0, & \text{if } \phi = s_i \geq \pi,\\ \|\phi_1\|, & \text{if } \phi = \neg\phi_1,\\ \max\{\|\phi_1\|, \|\phi_2\|\}, & \text{if } \phi \in \{\phi_1 \wedge \phi_2,\ \phi_1 \vee \phi_2\},\\ \bar{I} + \max\{\|\phi_1\|, \|\phi_2\|\}, & \text{if } \phi \in \{\phi_1 \mathcal{U}_I \phi_2,\ \phi_1 \mathcal{R}_I \phi_2\},\\ \bar{I} + \|\phi_1\|, & \text{if } \phi \in \{\Diamond_I \phi_1,\ \Box_I \phi_1\}. \end{cases} \tag{3}$$

3 WEIGHTED SIGNAL TEMPORAL LOGIC

This section describes an extension of STL referred to as Weighted STL (wSTL) introduced in [26] that allows specifications to capture user preferences, priorities, and importance associated with the Boolean and temporal operators.

Definition 4 (Weighted Signal Temporal Logic (wSTL) [26]). *The syntax of wSTL[1] in [26] with the inclusion of until and release temporal operators is defined as follows:*

$$\varphi ::= \top \mid \bot \mid \mu \mid \neg\varphi \mid \bigwedge_{i \in [1..N]}{}^{p}\varphi_i \mid \bigvee_{i \in [1..N]}{}^{p}\varphi_i \mid \Diamond_I^w \varphi \mid \Box_I^w \varphi \mid \varphi_1 \mathcal{U}_I^w \varphi_2 \mid \varphi_1 \mathcal{R}_I^w \varphi_2 \tag{4}$$

where all Boolean, temporal operators, true $\top$, false $\bot$, and predicate μ have the same definitions as for STL. The positive weights over conjunction and disjunction formulae are assigned by weight functions $p : [1..N] \to \mathbb{R}_{>0}$, where N is the number of sub-formulae under the operator. For sub-formulae in conjunction, denoted as $(\wedge^p(\varphi_1, \ldots, \varphi_N))$, the weights capture the importance of parallel tasks, whereas, for the sub-formulae in disjunction, denoted as $(\vee^p(\varphi_1, \ldots, \varphi_N))$, the weights indicate priorities for alternatives. The positive weight functions $w : I \to \mathbb{R}_{>0}$ capture user preferences for satisfaction times for the eventually operator and the importance of satisfaction times in the case of the always operator. Similarly, the weights over until and release operators indicate the preference for the time instance when switching from satisfying one subformula to the other should happen.

Note that the qualitative (Boolean) semantics of a wSTL formula is the same as the associated STL formula without the attached weight functions.

Remark 1. *In case a wSTL specification φ has all unit weights, i.e., $p = 1$ and $w = 1$ for every sub-formula, then it does not impose any preference, priorities, or importance over the Boolean or temporal operators. Thus, STL formulae are wSTL formulae with unit weights.*

Example 2 (Continuation of Example (1)). *Thus, the example (1) can now be expressed as $\varphi = \wedge^{p_1}(\varphi_1, \varphi_2, \varphi_3)$ where $\varphi_1 = \Box^{w_1}_{[9,10]}(\vee^{p_2}(\textit{tea},\ \textit{coffee}))$, $\varphi_2 = \Box^{w_2}_{[12,1]}\neg\textit{office}$, $\varphi_3 = \Diamond^{w_3}_{[2,3]}\textit{living\_room}$. p_1, p_2, w_1, w_2, w_3 are weight vectors with $p_2 = [p_{21}, p_{22}]$, $p_{21} < p_{22}$ and $w_3 = [w_{31}, \ldots, w_{3n}]$, n is the number of timesteps considered between $[2, 3]$ and $w_{3n} > w_{3i}$, $\forall i \in [1..(n-1)]$. Since all three sub-tasks are equally important, p_1 is a vector of all ones, $p_1 = [1, 1, 1]$. Since there are no temporal preferences for φ_1 and φ_2, w_1 and w_2 are vectors of all ones of appropriate lengths.* □

The time horizon of a wSTL specification $\|\varphi\|$ is the same as for STL and computed using (3).

3.1 Weighted traditional robustness for wSTL

Similar to STL, wSTL admits quantitative semantics. However, in this case, it is modulated by the weights on the Boolean and temporal operators that capture user satisfaction preferences. Informally, it is a weighted version of the transitional robustness in (2). The definition below extends the wSTL traditional robustness from [26] to include the *until* and *release* temporal operators.

Definition 5 (Weighted Traditional Robustness). *Given a wSTL specification φ and a signal s, the weighted robustness score*

[1] From now on, we refer to an STL specification as ϕ and a wSTL specification as φ

$\tilde{\rho}(\varphi, s, k)$ *at time* k *is recursively defined as follows*

$$\begin{aligned}
\tilde{\rho}(\mu, s, k) &:= s_i(k) - \pi, \\
\tilde{\rho}(\neg\varphi, s, k) &:= -\tilde{\rho}(\varphi, s, k), \\
\tilde{\rho}\left(\bigwedge_i^p \varphi_i, s, k\right) &:= \min_i \left\{p_i^{\wedge} \cdot \tilde{\rho}(\varphi_i, s, k)\right\}, \\
\tilde{\rho}\left(\bigvee_i^p \varphi_i, s, k\right) &:= \max_i \left\{p_i^{\vee} \cdot \tilde{\rho}(\varphi_i, s, k)\right\}, \\
\tilde{\rho}\left(\Box_I^w \varphi, s, k\right) &:= \min_{k' \in k+I} \left\{w^{\Box}(k'-k) \cdot \tilde{\rho}(\varphi, s, k')\right\}, \\
\tilde{\rho}\left(\Diamond_I^w \varphi, s, k\right) &:= \max_{k' \in k+I} \left\{w^{\Diamond}(k'-k) \cdot \tilde{\rho}(\varphi, s, k')\right\}, \\
\tilde{\rho}\left(\varphi_1 \mathcal{U}_I^w \varphi_2, s, k\right) &:= \max_{k' \in k+I} \left\{\min\left\{w^{\mathcal{U}}(k'-k) \cdot \tilde{\rho}(\varphi_2, s, k'), \right.\right. \\
&\qquad \left.\left. \min_{k'' \in I'} \left\{\tilde{\rho}(\varphi_1, s, k'')\right\}\right\}\right\}, \\
\tilde{\rho}\left(\varphi_1 \mathcal{R}_I^w \varphi_2, s, k\right) &:= \min_{k' \in k+I} \left\{\max\left\{w^{\mathcal{R}}(k'-k) \cdot \tilde{\rho}(\varphi_2, s, k'), \right.\right. \\
&\qquad \left.\left. \max_{k'' \in I'} \left\{\tilde{\rho}(\varphi_1, s, k'')\right\}\right\}\right\},
\end{aligned} \tag{5}$$

where $I' = [k..k']$, *For Boolean operators, we have*

$$p_i^{\wedge}(r_i) = \left(\frac{1}{2} - \bar{p}_i\right)\operatorname{sign}(r_i) + \frac{1}{2}, \qquad p_i^{\vee}(r_i) = 1 - p_i^{\wedge}(r_i),$$

where $r_i = \tilde{\rho}(\varphi_i, s, k)$ *is the weighted robustness of the subformula* φ_i, *and* $\bar{p}_i = \frac{p_i}{\mathbf{1}^T p}$ *is its normalized weight. Similarly, for the unary temporal operators, we have*

$$w^{\Box}(r_{k'}) = \left(\frac{1}{2} - \bar{w}(k-k')\right)\operatorname{sign}(r_{k'}) + \frac{1}{2}, \; w^{\Diamond}(r_{k'}) = 1 - w^{\Box}(r_{k'}),$$

where $r_{k'} = \tilde{\rho}(\varphi, s, k')$, $\bar{w}(\mathrm{t}) = \frac{w(\mathrm{t})}{W}$ *and* $W = \sum_{\mathrm{t}=0}^{k'-k} w(\mathrm{t})$. *For the binary temporal operators until and release, we define*

$$w^{\mathcal{U}}(r_{k'}) = w^{\Diamond}(r_{k'}), \qquad w^{\mathcal{R}}(r_{k'}) = w^{\Box}(r_{k'}),$$

where $r_{k'} = \tilde{\rho}(\varphi_2, s, k')$. *The weights* $p^{\wedge}$, $p^{\vee}$, $w^{\Box}$, *and* $w^{\Diamond}$ *are defined as DeMorgan aggregation functions [26].*

Theorem 2 (wSTL soundness [26]). *The wSTL weighted robustness is sound iff weights are positive. Let STL specification* ϕ *be the wSTL specification* φ *with all weights equal* 1 *then*

$$\begin{aligned}
\tilde{\rho}(\varphi, s, k) > 0 &\iff \rho(\phi, s, k) > 0 \rightarrow s \models \varphi, \\
\tilde{\rho}(\varphi, s, k) < 0 &\iff \rho(\phi, s, k) < 0 \rightarrow s \not\models \varphi.
\end{aligned} \tag{6}$$

soundness proof outline is provided in [26].

Note that if all weights in the wSTL specification are positive, then the robustness score is a scaled value of the equivalent STL robustness.

Example 3. *Consider a fully actuated robot with a scalar state in the interval* $[1, 12]$, *initial position* 7.5, *and no control bounds except for state saturation, e.g., 1D cart, conveyor belt. Let the specification require that "The robot should be - 1) either at a distance of 7 to 8 units from the starting point; 2) or between 0 to 5 minutes, it should always be at least two units away from the starting point. Moreover, it is five times more important to satisfy the first choice (priority between alternatives), and it is especially important to satisfy the second choice between 3 to 5 minutes (importance for satisfaction times). " This specification can be described using wSTL as* $\varphi = \vee^{p_2}\left(\wedge^{p_1}((s \leq 8), (s \geq 7)), (\Box_{[0,5]}^{w_1}(s \geq 2))\right)$ *where* $p_1 = [1, 1]$, $p_2 = [5, 1]$, $w_1 = [0.5, 0.5, 0.8, 0.8, 0.9]$. *Note that the weights are defined relative to each other and thus can be any positive value as long as they reflect the relative preference (importance).*

Let $\varphi_1 = \wedge^{p_1}((s \leq 8), (s \geq 7))$ *and* $\varphi_2 = \Box_{[0,5]}^{w_1}(s \geq 2)$. *The time horizon is given by* $\max\{\|\varphi_1\|, \|\varphi_2\|\} = \max\{1, 5\} = 5$. *The plan thus obtained is* $[7.5, 12, 12, 12, 12]$ *where it is observed that at the first timestep, the robot satisfies* φ_1 *with equal importance to the conditions* $(s \leq 8)$ *and* $(s \geq 7)$. *For the subsequent timesteps, it can then choose to satisfy* φ_2 *where staying at 12 yields the maximum robustness. The overall robustness thus obtained is* $\tilde{\rho} = 2.5$ *which can be calculated as follows:* $\tilde{\rho}(\varphi, s) = \max\{\min\{p \cdot \tilde{\rho}(\varphi_1, s)\}, \min\{w_1 \cdot \tilde{\rho}(\varphi_2, s)\}\} = \max\{5 \times \min\{(8-7.5), (7.5-7)\}, 1 \times \min\{[0.5 \times 5.5, 0.5 \times 10, 0.8 \times 10, 0.8 \times 10, 0.9 \times 10]\}\} = \max\{2.5, 2.25\} = 2.5$.

4 PROBLEM STATEMENT

This section introduces the control synthesis problem subject to temporal and logical constraints captured as wSTL specifications. Let us consider a discrete-time dynamical system modeled as

$$s(k+1) = A\, s(k) + B\, u(k), \qquad s(0) = s_\circ, \tag{7}$$

where $s(k) \in \mathbf{S} \subseteq \mathbb{R}^n$ is the state of the system and $u(k) \in \mathbf{U} \subseteq \mathbb{R}^m$ is the control input at the k-th time step, $k \in \mathbb{Z}_{\geq 0}$, $s_\circ \in S$ is the initial state, and A and B of appropriate sizes are the state transition and input gain matrices, respectively, capturing the system's dynamics. Let us consider a system trajectory $s = s(0)s(1)\ldots$ generated by a control sequence $u = u(0)u(1)\ldots$ starting with the initial state $s_\circ$. Additionally, let $J(s, u)$ be a cost function associated with generating control signal u while being at state s. We consider cost functions as sums of weighted 1-norm, ∞-norm, and linear terms.

The control synthesis problem requires finding an input control sequence $u^* : [0..K-1] \rightarrow \mathbf{U}$ that minimizes cost J and maximizes wSTL robustness of φ such that the specification φ is satisfied. Formally, the control synthesis problem subject to logical and temporal constraints is defined as follows.

Problem 1. *Given a discrete linear dynamical system of the form* (7), *the initial state* $s_\circ$, *and a specification with user preferences given as a wSTL formula* φ *as in* (4), *synthesize a sequence of control inputs* u^* *such that*

$$\begin{aligned}
u^* = \; & \operatorname*{argmin}_u J(s, u) - \lambda\, \tilde{\rho}(\varphi, s) \\
& s.t. \quad (7) \quad \textit{(linear discrete dynamics)}, \\
& \qquad s \models \varphi \quad \textit{(mission specification satisfaction)}, \\
& \qquad \underline{u} \leq u \leq \bar{u} \quad \textit{(control signal saturation)},
\end{aligned} \tag{8}$$

where $\underline{u}$ *and* $\bar{u}$ *are lower and upper control bounds and* $\lambda \geq 0$ *is a tuning parameter that defines the trade-off between the cost function* J *and specification satisfaction captured via maximization of the robustness.*

The planning time horizon is denoted by K such that $K \geq \|\varphi\|$.

Remark 2. *Note that when cost function* $J(s, u)$ *is not defined in* (8) *Problem 1 reduces to maximizing the robustness* $\tilde{\rho}(\varphi, s)$. *It leads to satisfaction of* φ *while taking into account the user satisfaction preferences and importance over subformulae.*

5 CONTROL SYNTHESIS

In this section, we formulate Problem 1 as an optimization problem and introduce a Mixed Integer Linear Program (MILP) encoding for wSTL. We consider the following assumptions throughout the rest of the paper.

Assumption 1. *wSTL specifications are over linear predicates.*

While wSTL formulae can be defined over general, non-linear predicates $l_\mu(s(k)) \geq \pi$ for some function $l : \mathbb{R}^n \to \mathbb{R}$, we limit them to simple linear functions $s_i(k) \geq \pi$ (or $s_i(k) \leq \pi$, see below). Our MILP encoding can still be employed by introducing output variables $y_\mu = l_\mu(s(k))$ for all non-linear predicates μ, and using piecewise-linear approximations of the output functions l_μ.

STL was originally intended to express the desired behavior of systems in the continuous-time domain. However, solving synthesis problems with STL constraints as MILPs requires discrete-time abstraction. For clarity and brevity, we consider uniform time discretization. However, the MILP encoding works for non-uniform encoding as well, provided that the system dynamics are linear or linearized. In both cases, the synthesized discrete-time control policy may not guarantee the satisfaction of the wSTL specification for continuous-time signals [31]. To ensure continuous-time satisfaction, e.g., collision avoidance for robots, either state constraints (such as funnels or tubes) or a robustness lower bound can be used to ensure safety.

Assumption 2. *wSTL specifications are in positive normal form.*

The assumption is not limiting because any STL, and by extension, the wSTL formula, can be put in positive normal form, where the negation operators are only in front of predicates. In the following, we eliminate negations completely by considering predicates defined with either $\geq$ and $\leq$ comparison operators.

Remark 3. *Any wSTL formula can be represented using an Abstract Syntax Tree (AST)*[2] *in which intermediate nodes correspond to logical and temporal operators, and leaves to predicates [15]. Our MILP encoding relies on a recursion definition that uses the AST to compute the overall robustness score of the specification formula φ.*

5.1 MILP encoding for wSTL

The synthesis Problem 1 requires that the specification be satisfied (8), and by Assumption 2 wSTL specifications do not contain negation operators. Thus, we simplify the definition of the weighted traditional robustness (5). The gain functions in Definition 5 for Boolean operators are constants $p_i^\wedge = 1 - \bar{p}_i$ and $p_i^\vee = \bar{p}_i$, and for temporal operators are $w^\square(k') = 1 - \bar{w}(k - k')$, and $w^\diamond(k') = \bar{w}(k - k')$. For simplicity, below, we consider that weights have been normalized and changed to match the above gains, e.g., $p_i = p_i^\wedge$.

The MILP encoding is defined recursively over the nodes of the AST of wSTL formula φ starting from the leaves, the *predicates*.

Let μ be a predicate. We define the variables $z_k^\mu \in \mathbb{B}$ that take value one if the predicate $\mu \in \mathbb{R}$ is taken into account in the satisfaction of formula φ at time $k \in [0..K]$, zero otherwise, and $\tilde{\varrho}_k^\mu \in \mathbb{R}$ the robustness score of the predicate μ at time $k \in [0..K]$, where $K \geq \|\varphi\|$. Let M be a large enough number (e.g., larger than the largest upper bound of signals used in wSTL specification φ). The following constraints capture the satisfaction of predicates of the type $s(k) \geq \pi$

$$\varphi = \mu \Rightarrow \begin{cases} s(k) + M(1 - z_k^\mu) \geq \pi + \tilde{\varrho}_k^\mu \\ s(k) - M z_k^\mu \leq \pi + \tilde{\varrho}_k^\mu \end{cases}, \tag{9}$$

Notice that the set of constraints for predicates checks whether or not the predicate is considered in the overall satisfaction and does not constrain the variables $s(k)$ and $\tilde{\varrho}_k^\mu$ otherwise. They can take any value, but as the objective is to maximize robustness, $s(k)$ is forced to take the value that maximizes $\tilde{\varrho}_k^\mu$, i.e., as far as possible from threshold value π. In the case of predicates of the form $s(k) \leq \pi$, the set of constraints is modified as follows.

$$\varphi = \mu \Rightarrow \begin{cases} s(k) - M(1 - z_k^\mu) \leq \pi - \tilde{\varrho}_k^\mu \\ s(k) + M z_k^\mu \geq \pi - \tilde{\varrho}_k^\mu \end{cases}. \tag{10}$$

For *conjunction* operator, let us define $z_k^\varphi \in [0, 1]$ as the variable taking value one if all the subformulae φ_i hold at time $k \in [0..K]$, and zero otherwise, where $i \in [1..N]$ and N is the total number of subformulae in the conjunction. Let $z_k^{\varphi_i} \in [0, 1]$, $\tilde{\varrho}_k^\varphi \in \mathbb{R}$, and $p_i \in \mathbb{R}_{\geq 0}$ be the variables defining satisfaction or violation, robustness score, and weights of subformulae φ_i at time $k \in [0..K]$, respectively. The set of constraints capturing the conjunction semantics and robustness is

$$\varphi = \bigwedge_i^p \varphi_i \Rightarrow \begin{cases} \tilde{\varrho}_k^\varphi \leq p_i \cdot \tilde{\varrho}_k^{\varphi_i}, \forall i \\ z_k^\varphi \leq z_k^{\varphi_i}, \forall i \\ z_k^\varphi \geq 1 - N + \sum_i z_k^{\varphi_i} \end{cases}. \tag{11}$$

The objective here is to impose the satisfaction of z_k^φ if all subformulae φ_i are satisfied. The first equation will impose an upper bound in the overall robustness score as the minimum weighted robustness of the component subformulae $\tilde{\varrho}_k^{\varphi_i}$. The second equation requires that the variable capturing the conjunction operator z_k^φ takes the value of zero, indicating violation, if at least one or more of the subformulae satisfaction variables $z_k^{\varphi_i}$ are zero. Finally, we indicate that z_k^φ takes value one only if the total number of subformulae N equals the summation of $z_k^{\varphi_i}$ variables, i.e., all of the subformulae are satisfied.

In the case of *disjunction*, we use the same type of variables as for conjunction. We also introduce auxiliary variables $\hat{z}_{k,i}^\varphi \in \mathbb{B}$ used for the linear formulation of the "max" operator. $M \in \mathbb{R}_{\geq} 0$ is a sufficiently large value (see description for predicates).

$$\varphi = \bigvee_i^p \varphi_i \Rightarrow \begin{cases} \tilde{\varrho}_k^\varphi \leq p_i \cdot \tilde{\varrho}_k^{\varphi_i} + M \cdot (1 - \hat{z}_{k,i}^\varphi), \forall i \\ \hat{z}_{k,i}^\varphi \leq z_k^{\varphi_i}, \forall i \\ \sum_i \hat{z}_{k,i}^\varphi \geq z_k^\varphi \\ z_k^\varphi \geq z_k^{\varphi_i}, \forall i \\ z_k^\varphi \leq \sum_i z_k^{\varphi_i} \end{cases}. \tag{12}$$

The first equation captures the maximum robustness among all subformulae φ_i that are taken into account $z_k^{\varphi_i} = 1$. The $\hat{z}_{k,i}^\varphi$ variables select the maximum robustness, while the second equation ensures that the subformula is considered. At least one subformula is selected by the third equation if φ holds. The last two equations

[2] A formula can have many equivalent ASTs [15] determined by the parsing methods used.

ensure that z^{φ} is set to one if and only if at least one subformula is satisfied.

The *always* operator follows the same as the *conjunction* operator, but with weights and variables over time rather than over subformulae. For temporal weights $w(\mathrm{t}) \in \mathbb{R}_{\geq 0}$ with $\mathrm{t} \in I$, we have

$$\varphi = \Box_I^w \psi \Rightarrow \begin{cases} \tilde{\varrho}_k^{\varphi} \leq w(\mathrm{t}) \cdot \tilde{\varrho}_{k+\mathrm{t}}^{\psi}, \forall \mathrm{t} \in I \\ z_k^{\varphi} \leq z_{k+\mathrm{t}}^{\psi}, \forall \mathrm{t} \in I \\ z_k^{\varphi} \geq 1 - |I| + \sum_{\mathrm{t} \in I} z_{k+\mathrm{t}}^{\psi}, \end{cases} \tag{13}$$

where $|I|$ denotes the length of the interval.

The set of constraints for the *eventually* operator are similar to the ones for disjunction.

$$\varphi = \Diamond_I^w \psi \Rightarrow \begin{cases} \tilde{\varrho}_k^{\varphi} \leq w(\mathrm{t}) \cdot \tilde{\varrho}_{k+\mathrm{t}}^{\psi} + M \cdot (1 - \hat{z}_{k,\mathrm{t}}^{\varphi}), \forall \mathrm{t} \in I \\ \hat{z}_{k,\mathrm{t}}^{\varphi} \leq z_{k+\mathrm{t}}^{\psi}, \forall \mathrm{t} \in I \\ \sum_{\mathrm{t} \in I} \hat{z}_{k,\mathrm{t}}^{\varphi} \geq z_k^{\varphi} \\ z_k^{\varphi} \geq z_{k+\mathrm{t}}^{\psi}, \forall \mathrm{t} \in I \\ z_k^{\varphi} \leq \sum_{\mathrm{t} \in I} z_{k+\mathrm{t}}^{\psi} \end{cases} \tag{14}$$

As before, the first three equations select the maximum robustness of ψ over time interval I if φ holds, while the latter two ensure that z_k^{φ} is one if and only if ψ is satisfied at least once in I.

The weights over the temporal operator $\mathcal{U}$ denote the preference for the timestep at which φ_2 becomes true. It is required that φ_1 should remain true from time 0 until the beginning of I, and thus, for $\mathrm{t} \in [0, \underline{I} - 1]$, the robustness is upper bounded by the robustness of φ_1. For $\mathrm{t} \in I$, the constraints on $\tilde{\varrho}_k^{\varphi}$ ensure that the robustness is upper bounded by the minimum of the robustness of φ_1 and φ_2. Furthermore, the constraint on the auxiliary variable $\hat{z}_{k,\mathrm{t}}^{\varphi}$ ensures that φ_2 holds at some timestep t whereas the constraint on z_k^{φ} ensures violation is encoded correctly.

$$\varphi = \varphi_1 \mathcal{U}_I \varphi_2 \Rightarrow \begin{cases} \tilde{\varrho}_k^{\varphi} \leq \tilde{\varrho}_{k+\mathrm{t}}^{\varphi_1}, \mathrm{t} \in [0, \underline{I} - 1] \\ \tilde{\varrho}_k^{\varphi} \leq w(\mathrm{t}) \tilde{\varrho}_{k+\mathrm{t}}^{\varphi_2} + M \cdot (1 - \hat{z}_{k,\mathrm{t}}^{\varphi}), \forall \mathrm{t} \in I \\ \tilde{\varrho}_k^{\varphi} \leq \tilde{\varrho}_{k+\mathrm{t}}^{\varphi_1} + M \cdot \sum_{\ell=0}^{\mathrm{t}} \hat{z}_{k,\ell}^{\varphi}, \forall \mathrm{t} \in I \\ z_k^{\varphi} \leq z_{k+\mathrm{t}}^{\varphi_1}, \mathrm{t} \in [0, \underline{I} - 1] \\ \hat{z}_{k,\mathrm{t}}^{\varphi} \leq z_{k+\mathrm{t}}^{\varphi_2}, \forall \mathrm{t} \in I \\ \sum_{\mathrm{t} \in I} \hat{z}_{k,\mathrm{t}}^{\varphi} \geq z_k^{\varphi} \\ z_k^{\varphi} \leq \sum_{\mathrm{t} \in I} z_{k+\mathrm{t}}^{\varphi_2} \\ z_k^{\varphi} \leq z_{k+\mathrm{t}}^{\varphi_1} + \sum_{\ell=0}^{\mathrm{t}} z_{k+\ell}^{\varphi_2}, \forall \mathrm{t} \in I \end{cases} \tag{15}$$

We leverage the cumulative nature of the until operator in terms of satisfaction of φ_1 to reduce the number of constraints. Specifically, the third and last equations constrain the variables for φ only until φ_2 holds.

For the *release* operator, the weights denote the preference for the timestep at which φ_2 may be released, i.e., when φ_1 holds. The robustness $\tilde{\varrho}_k^{\varphi}$ is upper-bounded by the robustness of the φ_2 times its weight until φ_1 becomes true in equation one. The maximum value of the robustness of φ_1 is also a bound if φ_1 holds before $\bar{I}$ (equation two). The $\hat{z}_{k,\mathrm{t}}^{\varphi}$ variables ensure that the correct maximum is selected among all times φ_1 holds via equations three and four.

$$\varphi = \varphi_1 \mathcal{R}_I \varphi_2 \Rightarrow \begin{cases} \tilde{\varrho}_k^{\varphi} \leq w(\mathrm{t}) \cdot \tilde{\varrho}_{k+\mathrm{t}}^{\varphi_2} + M \cdot \sum_{\ell=0}^{k+\mathrm{t}} \hat{z}_{k,\ell}^{\varphi}, \forall \mathrm{t} \in I \\ \tilde{\varrho}_k^{\varphi} \leq \tilde{\varrho}_{k+\mathrm{t}}^{\varphi_1} + M \cdot (1 - \hat{z}_{k,\mathrm{t}}^{\varphi}), \mathrm{t} \in [0, \bar{I}] \\ \hat{z}_{k,\mathrm{t}}^{\varphi} \leq z_{k+\mathrm{t}}^{\varphi_1}, \forall \mathrm{t} \in [0, \bar{I}] \\ \sum_{\mathrm{t}=0}^{\bar{I}} \hat{z}_{k,\mathrm{t}}^{\varphi} \leq 1 \end{cases} \tag{16}$$

Lastly, to ensure the satisfaction of the overall wSTL specification, we impose that the top-level formula is taken into account and its robustness is positive

$$z^{\varphi} = 1 \ \wedge \ \tilde{\varrho}_0^{\varphi} \geq 0. \tag{17}$$

Proposition 1 (Correctness). *The constraints (9) - (17) satisfy the following properties:*

1. $s \models \varphi$ *if constraints are feasible;*
2. $s \not\models \varphi$ *if constraints are infeasible;*
3. *the maximum* $\tilde{\varrho}_0^{\varphi}$ *such that the set of constraints are feasible is given by* $\tilde{\varrho}_0^{\varphi} = \tilde{\rho}(\varphi, s, 0)$.

Proof Sketch. The first two properties follow similarly to the proof in [31]. The last property follows structural induction over the AST. The base case for predicates is trivially true. The induction for the Boolean and temporal operators follows from the definition of max and min operators. For the until and release operator, we leverage their cumulative structure to reduce the number of variables and constraints needed. □

6 CASE STUDIES

In this section, we showcase and test the functionality of the wSTL MILP encoding. First, we show how the variation of weights can lead to multiple solutions compared to its equivalent STL specifications. Second, we test the control synthesis for an agent satisfying a specification of navigating in a non-convex two-dimensional environment with temporal and logical constraints. Lastly, we show the time performance and complexity comparison between STL and wSTL MILP solutions. All computations of the case studies were performed on a PC with 20 cores at 3.7 GHz with 64 GB of RAM. We used Gurobi [13] as an MILP solver.

6.1 Functionality of weights

For the three case studies considered here, we consider that signals are bounded but unconstrained, meaning there is no dynamic governing the variables.

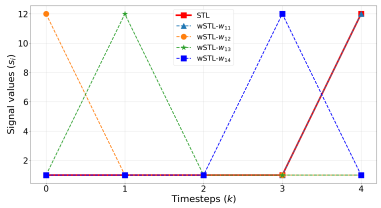


Figure 2: The weights w_{1j}, w_{2j} indicate the preference for satisfaction over sub-formulae resulting in the generation of different signals.

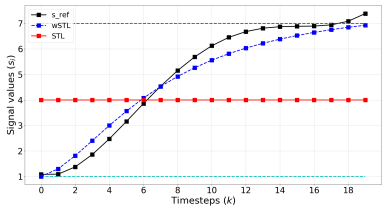


Figure 3: Generation of a signal close to the reference by suitably designing time-varying weights w.

6.1.1 Satisfaction preferences over temporal operators. Consider the wSTL specification

$$\varphi = \wedge^{p}\left(\diamond_{[0,4]}^{w_{1j}}(s_i \geq 4), \diamond_{[0,4]}^{w_{2j}}(s_i \leq 3)\right). \tag{18}$$

With unit weights over the conjunction operator, we vary temporal weights w_{1j}, w_{2j} indicating different preferences for satisfaction instances. The set of weights are as follows: 1) $w_{11} = [0.1, 0.2, 0.3, 0.4, 0.5]$, 2) $w_{12} = [0.9, 0.9, 0.8, 0.1, 0.1]$, 3) $w_{13} = [0.5, 0.5, 0.5, 0.5, 0.5]$, 4) $w_{14} = [0.1, 0.1, 0.5, 0.9, 0.9]$. In all four cases, $w_{2j} = [1, \ldots, 1] - w_{1j}$ where $j \in [1, 4]$. Higher weights indicate a higher preference for the satisfaction of the corresponding subformula at that timestep.

Fig. 2 shows the generated wSTL signals compared to the STL signal generated for the equivalent STL formula

$$\phi = \diamond_{[0,4]}(s_i \geq 4) \wedge \diamond_{[0,4]}(s_i \leq 3). \tag{19}$$

Thus, it can be seen that wSTL φ with w_{12} satisfies the predicate $(s_i \geq 4)$ earlier during the interval, whereas with w_{14} it prefers the satisfaction of $(s_i \leq 3)$ during the initial timesteps as the weights w_{14} indicate that the satisfaction of $(s_i \geq 4)$ is preferred towards the end of the interval.

6.1.2 Effect of time-varying weights on control synthesis. The weights p, w can be time-varying over the interval leading to generating the desired behavior. Consider two predicates $s_i \geq 1$ and $s_i \leq 7$. The goal is to generate a signal close to a reference signal s_{ref} (solid black) as shown in Fig. 3 where

$$\varphi = \square_{[0,20]}^{w}\left(\wedge^{p(k)}\left((s_i \leq 7), (s_i \geq 1)\right)\right). \tag{20}$$

The time-varying weights $p(k) = [p_1(k)\ p_2(k)]$ are as follows:

$$p_1(k) = \frac{\sqrt{k^3}}{|I|}; \quad p_2(k) = 1 - \frac{k}{|I|}, \tag{21}$$

where $k \in [0..20]$ is the current time-step and $|I|$ denotes the total duration of the specification. Since the desired behavior is over the entire interval without preferences to the specific sub-interval, w can be chosen to be a vector of all ones, $w = [1, \ldots, 1]_{1\times|I|}$. Note that the weights in (21) are design choices and, thus, not unique.

The equivalent STL formula is

$$\phi = \square_{[0,20]}((s_i \leq 7) \wedge (s_i \geq 1)). \tag{22}$$

The resulting signals are shown in Fig. 3. In the case of STL, both subformulae in conjunction receive equal importance, and thus, the synthesized signal (solid red) stays constant at 4. On the contrary, the time-varying weights on these subformulae, in the case of the wSTL formula, allow the regulation of the generated behavior (dashed blue) more closely with respect to the reference signal.

6.1.3 Generating time-varying weights for desired behavior. Consider a scenario wherein some desired behavior is to be generated as a combination of given component signals. Fig. 4 shows a solid black reference signal to be generated with the help of the component signals s_A and s_B. We consider two cases: 1) The component signals need to be utilized at all times. 2) The component signals need to be utilized at some instance in the given interval. Let $\varphi_{11} = \wedge^{p_1}((s \leq 12), (s \geq 3)), \varphi_{12} = \wedge^{p_1}((s \leq 9), (s \geq 2)), \varphi_{13} = \wedge^{p_1}((s \leq 12), (s \geq 2))$. The STL and wSTL formulae are:

$$\begin{aligned}\phi_1 = \square_{[0,4]}((s \leq 12) \wedge (s \geq 3)) \wedge \square_{[5,10]}((s \leq 9) \\ \wedge (s \geq 2)) \wedge \square_{[11,15]}((s \leq 12) \wedge (s \geq 2)).\end{aligned} \tag{23}$$

$$\begin{aligned}\phi_2 = \diamond_{[0,4]}((s \leq 12) \wedge (s \geq 3)) \wedge \diamond_{[5,10]}((s \leq 9) \\ \wedge (s \geq 2)) \wedge \diamond_{[11,15]}((s \leq 12) \wedge (s \geq 2)).\end{aligned} \tag{24}$$

$$\varphi_1 = \wedge^{p_2}\left(\square_{[0,4]}^{w_1}\varphi_{11}, \square_{[5,10]}^{w_1}\varphi_{12}\square_{[11,15]}^{w_1}\varphi_{13}\right). \tag{25}$$

$$\varphi_2 = \wedge^{p_2}\left(\diamond_{[0,4]}^{w_1}\varphi_{11}, \diamond_{[5,10]}^{w_1}\varphi_{11}, \diamond_{[11,15]}^{w_1}\varphi_{13}\right). \tag{26}$$

The time-varying weights are generated as follows:

$$p_1(k) = [\gamma(k)\ ,\ 1 - \gamma(k)] \forall k \in I, \quad \gamma(k) = \frac{|s\_ref(k) - s_A(k)|}{|s_B(k) - s_A(k)|}$$

We consider the following set of temporal weights w_{1i} to further regulate the generated behavior. The weights indicate the importance of satisfaction for the sub-formulae and are as follows:

- $wSTL - w_{11} = w = [1, \ldots, 1]_{1\times 15}$
- $wSTL - w_{12} = [1, 1, 2, 2, 2, 3, 9, 5, 1, 1, 9, 9, 5, 1, 1, 9] \times 10^{-1}$
- $wSTL - w_{13} = [5, 5, 5, 5, 5, 1, 1, 5, 9, 9, 1, 1, 5, 9, 9, 9] \times 10^{-1}$
- $wSTL - w_{14} = [1, 1, 5, 9, 9, 1, 1, 5, 9, 9, 1, 1, 5, 9, 9, 9] \times 10^{-1}$

Higher weights indicate higher preference. Simply put, $wSTL - w_{11}$ gives equal importance for satisfaction over the entire time interval. According to $wSTL - w_{12}$, it is more important to satisfy the subformulae towards the ends of intervals $[5, 10]$ and $[11, 15]$. The weights $wSTL - w_{13}$ and $wSTL - w_{14}$ can be interpreted accordingly. Fig.(4a) shows the generated wSTL signals for φ_1 where higher importance on the temporal operator results in generating a signal closer to s_ref. We employ curve-fitting in Fig. (4b, 4c) to smooth out the generated signals.

6.2 Control Synthesis

We consider three cases of the Problem 1. First, examples where we set the cost function $J(s, u) = 0$. Thus the problem is to find a control signal $u(k)$ that maximizes the robustness $\tilde{\rho}(\varphi, s)$. Second, an infeasible case where an agent is tasked with traveling to multiple places simultaneously. Finally, we add a cost function to show how it could affect the synthesized controller. Consider a single robot navigating in a planar environment $\mathcal{M} \subset \mathbb{R}^2$. Thus, $s(k) = [s_x(k), s_y(k)]^\top \in \mathbb{R}^2$. We define regions of interest $\mathcal{A} = [-9.5, -5.5]^2$, $\mathcal{B} = [5.5, 9.5] \times [-9.5, -5.5]$, $\mathcal{C} = [5.5, 9.5]^2$, $\mathcal{D} = [-9.5, -5.5] \times [5.5, 9.5]$ in $\mathcal{M}$, and region $\mathcal{E} = [-2.5, 2.5]^2 \subseteq \mathcal{M}$ that robot $s(k)$ needs to avoid. We arbitrarily choose $A = B = I_{2\times 2}$ for the robot dynamics as in (7). We consider initial position as $s(0) = (s_x(0), s_y(0)) = (-9, -9)$.

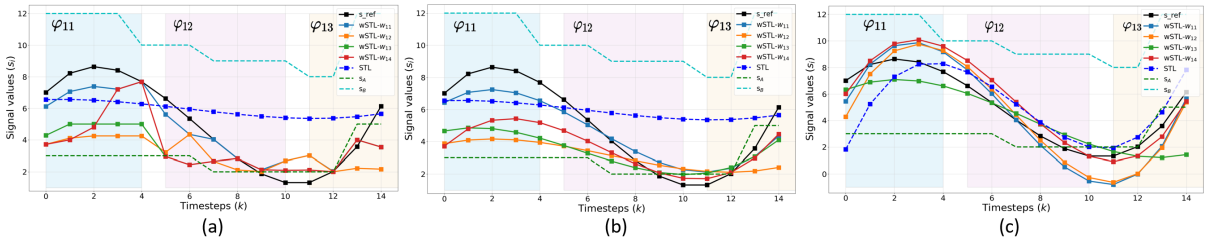


Figure 4: The time-varying weights p, w result in generating signals that closely follow the reference signal s_ref (a) generated raw signals for specification (25); (b) signals generated for (25) by employing curve fitting; (c) signals generated for (26) by employing curve fitting.

6.2.1 Control synthesis without cost function. For the following case studies, we consider the cost function $J(s,u) = 0$, making the control synthesis problem to find control signals that maximize robustness. We consider the STL ϕ and wSTL φ specifications

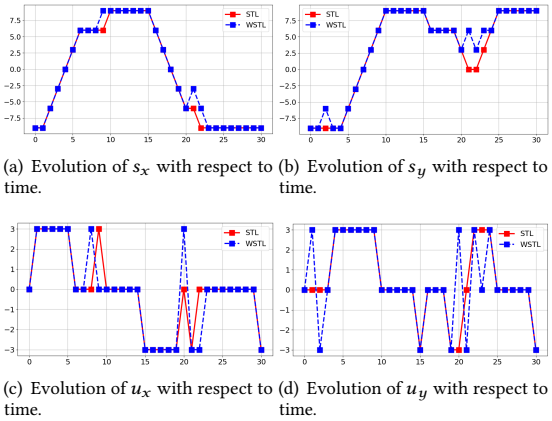
$$\phi = (\square_{[0,1]}\mathcal{A}) \wedge (\square_{[10,15]}C) \wedge (\square_{[25,30]}\mathcal{D}) \wedge (\square_{[0,30]}\mathcal{E}^c), \quad (27)$$

$$\varphi = \wedge^{p_i}\left((\square^{w}_{[0,1]}\mathcal{A}), (\square^{w}_{[10,15]}C), (\square^{w}_{[25,30]}\mathcal{D}), (\square^{w}_{[0,30]}\mathcal{E}^c)\right), \quad (28)$$

Going to regions of interest can be specified in STL and wSTL form by constraining s_x and s_y inside the region boundaries. For simplicity, with a slight abuse of notation, we define the formulae directly over the regions instead of defining predicates over all four boundaries of each region. Also, notice that instead of $\mathcal{E}$ the predicate we use to avoid this region is its complement $\mathcal{E}^c = \mathcal{M} \setminus \mathcal{E}$, i.e., robot position is allowed to be in every region outside of $\mathcal{E}$. The variable bounds considered in these case studies are $-10 \leq s_x \leq 10$, $-10 \leq s_y \leq 10$, $-3 \leq u_x \leq 3$, $-3 \leq u_y \leq 3$.

The first scenario we consider is where all the weights $p_1 = [1,1,1,1]$ and $w = [1,\ldots,1]_{1\times b}$ for b in time intervals $[a,b]$ in φ, then wSTL is expected to behave as its equivalent STL specification. In Fig. 5, we can see how the evolution of states s_x and s_y with its corresponding control signals u_x and u_y for wSTL (blue line) is very similar to the outcome of STL (red line). The trajectory of the signals in the environment is shown in Fig. 8(a), where it can be seen that the solution found for wSTL differs from the one in STL in small intervals. Even though both of them satisfy the specification by avoiding region $\mathcal{E}$ all the time and starting in the region $\mathcal{A}$ and then visiting regions C and $\mathcal{D}$.

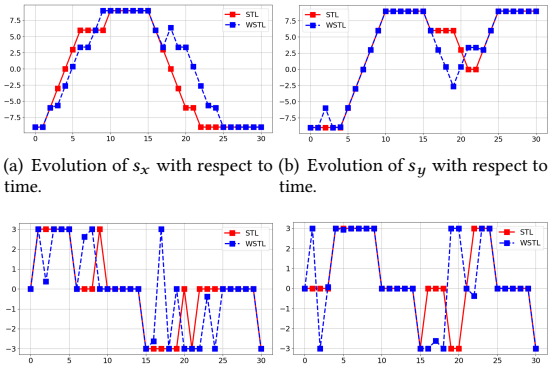
However, the more interesting behaviors can be expressed when the weight in the wSTL specification describes some preferences. For instance, by imposing the following two different weights vectors $p_2 = [0.1, 0.1, 0.1, 0.8]$ and $p_3 = [0.9, 0.9, 0.9, 0.1]$ over the conjunction operator in φ, the control signals and positions described differ significantly from the solution of the equivalent STL, as can be seen in Fig. 6 and Fig. 7, respectively. In Fig. (8b) and Fig. (8c), the generated trajectories can be seen where the preferences of avoiding the obstacle as much as possible are relaxed. Then, the robot tries to avoid regions of interest until it is necessary to satisfy the specification for the first set of weights. Whereas for the second set of weights, the priority is given to staying as far as possible from region $\mathcal{E}$ and maximizing the robustness in regions of interest, which is achieved by being at the farthest boundary of region $\mathcal{E}$.



(a) Evolution of s_x with respect to time. (b) Evolution of s_y with respect to time.

(c) Evolution of u_x with respect to time. (d) Evolution of u_y with respect to time.

Figure 5: Position and control signals when all weights of wSTL are set to be one such that the solution behaves very close to its equivalent STL.

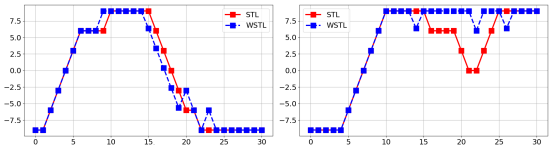


(a) Evolution of s_x with respect to time. (b) Evolution of s_y with respect to time.

(c) Evolution of u_x with respect to time. (d) Evolution of u_x with respect to time.

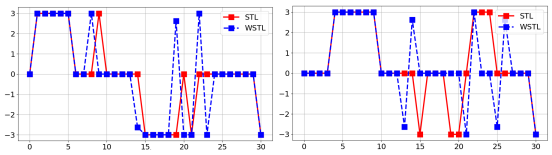
Figure 6: Position and control signals when the weights in wSTL are adjusted to marginally satisfy going to regions and softly constraint avoiding region $\mathcal{E}$.

Another case study we consider is when the specification is infeasible; this can happen when subformulae are conflicting or impossible to satisfy within their time windows given a bounded control signal. In the case of an infeasible STL specification, its



(a) Evolution of s_x with respect to time.

(b) Evolution of s_y with respect to time.



(c) Evolution of u_x with respect to time.

(d) Evolution of u_y with respect to time.

Figure 7: Position and control signals when the weights in wSTL are adjusted to avoid region $\mathcal{E}$ as much as possible.

robustness score is negative, with the smallest overall violation margin. Examples of infeasible STL and wSTL specifications are

$$\phi = (\square_{[0,1]}\mathcal{A}) \wedge (\square_{[8,10]}\mathcal{B}) \wedge (\square_{[8,10]}\mathcal{D}), \tag{29}$$

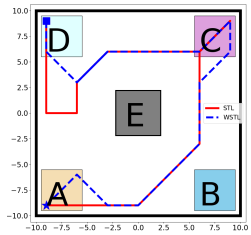
$$\varphi = \wedge^{p_i}\left((\square^{w}_{[0,1]}\mathcal{A}), (\square^{w}_{[8,10]}\mathcal{B}), (\square^{w}_{[8,10]}\mathcal{D})\right). \tag{30}$$

Note that the infeasibility is given by requesting the agent to be in two different places during the same time intervals. Here we show how varying the weights can lead to different solutions to the minimal violation of conflicting subformulae. We set $-2 \leq u_x \leq 2$, $-2 \leq u_y \leq 2$, $w = [1, \dots 1]_{1\times b}$ for all temporal operators and $p_1 = [1, 1, 1]$: (wSTL behaves the same as STL), $p_2 = [1, 20, 1]$: (higher preference to going to $\mathcal{B}$), and $p_3 = [1, 1, 20]$: (higher preference to going to $\mathcal{D}$). In Fig. 8(d), we can see that solution for STL and wSTL with unit weights remains in the middle of the two regions of interest since it will minimally violate both subformulae. However, choosing a significant difference of weights over the conjunction operator in φ leads to choosing the option with higher weight and ignoring the other conflicting subformula. Since the robustness of this subformula is the one that maximizes the robustness and therefore minimizes the biased violation. It is worth mentioning that we can not think of wSTL as a global solution for the partial satisfaction problem since it will depend on the predefined significant difference of weights between conflicting specifications. A slight difference in weights will not induce partial satisfaction (i.e., choosing at least one of the subformulae to satisfy); instead, the solution will only deviate close to the one with higher weight but will still not satisfy any.

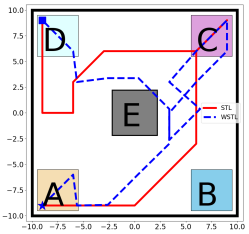
Lastly, let us consider the initial position of the agent as $s_x(0) = -5$, $s_y(0) = -1$, and the following wSTL specification

$$\varphi = \wedge^{p_i}\left((\square^{w}_{[0,7]}\mathcal{E}^c), (\square^{w}_{[8,18]}C)\right), \tag{31}$$

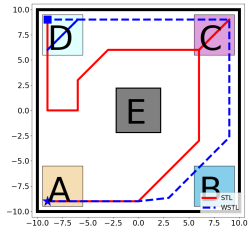
we generate four trajectories that avoid getting close to region $\mathcal{E}$ and go to region C in the time intervals specified by varying weights p_i. Let us define the following different weights for the conjunction operator in φ as $p_1 = [1, 0.1]$, $p_2 = [2, 0.9]$, $p_3 = [1, 0.8]$, $p_4 = [0.1, 2]$, and $p_5 = [0.1, 2]$. Note that the latter two are the same.



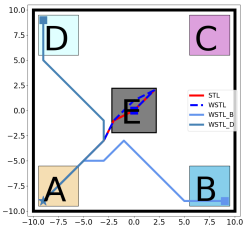
(a) Unit weights for wSTL φ and STL ϕ trajectories.



(b) Adjusting weights for softly constraint avoiding region $\mathcal{E}$ in wSTL trajectory.



(c) Adjusting weights for being as far as possible of region $\mathcal{E}$ in wSTL trajectory.



(d) Defining preferences for visiting a place in an infeasible specification.

Figure 8: Trajectories for wSTL φ and STL ϕ specifications in a two-dimensional environment.

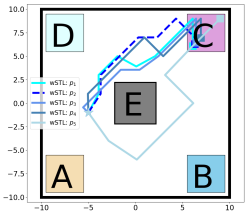


Figure 9: Trajectories around an obstacle changing the weights in the wSTL specification.

The control bounds for the first four cases are $-2 \leq u_x \leq 2$ and $-2 \leq u_y \leq 2$, and for p_5 as $-3 \leq u_x \leq 3$ and $-3 \leq u_y \leq 3$. The outcome of this case is shown in Fig. 9. The trajectories constrained by control inputs bounds of two units are shorter. Some of them are getting closer or farther to $\mathcal{E}$ according to the weights. However, for the case where the control bounds are three units, the solution is to go around the bottom boundary of $\mathcal{E}$, keeping distance to the obstacle and reaching the maximum robustness. Thus, varying weights may produce trajectories that are topologically the same or different from the STL one.

6.2.2 Control synthesis with cost functions. In previous case studies, we have not considered any cost function affecting the objective function in (8) since our main goal was to highlight the behavior when varying the weights in the robustness. However, any linear cost function $J(s, u)$ can be added and implemented in Gurobi. For instance, cost $J(s) = \|\Gamma^\top s\|_1 = \sum_i |\gamma_i s_i|$ is captured using variables v_i and $J = \sum_i v_i$ and constraint set $\{v_i \geq \gamma_i s_i, v_i \geq -\gamma_i s_i \mid \forall i\}$, where γ_i are regularization values and v_i are variable bounds. Similarly for $J(s) = \|\Gamma^\top s\|_\infty = \max_i |\gamma_i s_i|$, we add variable J and

constraint set $\{J \geq \gamma_i s_i,\ J \geq -\gamma_i s_i \mid \forall i\}$. In both cases, the objective function of the MILP becomes $J - \lambda\tilde{\rho}_0^{\varphi}$. The same constructions hold for control cost terms.

6.3 Time performance and complexity comparison

In this section, we want to analyze the complexity regarding the number of binary and continuous variables needed to capture wSTL specifications, its equivalent STL, and its time performance. Firstly, in Table 1, we show the robustness score [3], the number of binary and continuous variables required, and the time performance for every case study performed previously. In most cases, the encoding used for STL is faster in time than the one used for wSTL because the latter requires more binary and continuous variables to capture preferences and importance. Usually, binary variables are the ones that lead to complexity and affect the time performance of the encoding. So we could say that every specification that includes multiple *eventually*, *disjunction*, *until*, and *release* operators will diminish the performance due to the necessity of inclusion of additional binary variables. The wSTL synthesis problem is NP-complete, similar to the STL case. The structure of the problem determines the performance (e.g., close to totally unimodular) and the solver (e.g., Gurobi). Thus, a good measure/predictor of performance is the number of binary variables, but not always. The number of binary variables is $O(K|\phi|)$, where K is the time horizon and $|\phi|$ is the size of the formula.

Table 1: Complexity, time performance, and robustness value for every case study.

Specification	Robustness	Binary	Continuous	Time (s)
STL (19)	$\rho = 3$	40	58	0.010
wSTL (18)	$\tilde{\rho}$ =0	40	102	0.061
STL (22)	ρ =2	40	58	0.008
wSTL (20)	$\tilde{\rho}$ =1.8	40	100	0.021
STL (23)	$\rho = 3.5$	30	51	0.012
wSTL (25)	$\tilde{\rho} = 0.002$	32	88	0.027
STL (24)	ρ =3.5	30	51	0.017
wSTL (26)	$\tilde{\rho}$ =1.748	48	88	0.788
STL (27)	$\rho = 3$	152	237	0.118
wSTL (28)-p_1	$\tilde{\rho}$ =3	276	438	0.143
wSTL (28)-p_2	$\tilde{\rho}$ =0.3	276	438	0.197
wSTL (28)-p_3	$\tilde{\rho} = 0.3$	276	438	0.264
STL (29)	ρ =-6	16	79	0.034
wSTL (30)-p_1	$\tilde{\rho}$ =-20	16	106	0.057
wSTL (30)-p_2	$\tilde{\rho}$ =-20	16	106	0.034
wSTL (30)-p_3	$\tilde{\rho}$ =-6	16	106	0.030
wSTL (31)-p_1	$\rho = 0.090$	78	206	0.084
wSTL (31)-p_2	$\tilde{\rho}$ =0.620	78	206	0.140
wSTL (31)-p_3	$\tilde{\rho}$ =0.444	78	206	0.064
wSTL (31)-p_4	$\tilde{\rho}$ =0.095	78	206	0.084
wSTL (31)-p_5	$\tilde{\rho} = 0.3$	78	206	0.057

Finally, we show the run time performance comparison between STL and wSTL encodings with unit and random weights by gradually increasing the size of the mission specification. Let us consider

[3]Note that ρ and $\tilde{\rho}$ cannot be numerically compared since the weights introduce a scaling effect on $\tilde{\rho}$ and thus the magnitude of the $\tilde{\rho}$ can be considerably smaller or greater even for very similar STL and wSTL trajectories.

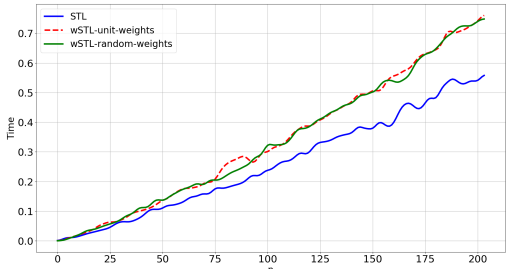


Figure 10: Time performance comparison between STL and wSTL with unit and random weights.

six variables x, y, z, u, v, and w, all with a lower-bound of -9 and upper-bound of 9. The STL ϕ and wSTL φ specification for the time performance comparison are the following

$$\phi = \bigwedge_{1}^{n} \mathcal{T}_I \left((\mathfrak{s}_1 \otimes_1 \Xi_1)\ \mathcal{L}\ (\mathfrak{s}_2 \otimes_2 \Xi_2) \right),$$

$$\varphi = \bigwedge_{1}^{n\ \ p} \mathcal{T}_I^{w} \left(\mathcal{L}^{p} \left((\mathfrak{s}_1 \otimes_1 \Xi_1),\ (\mathfrak{s}_2 \otimes_2 \Xi_2) \right) \right),$$

where $\mathcal{T} \in \{\Box, \Diamond\}$, $\mathcal{L} \in \{\wedge, \vee\}$, $\mathfrak{s}_1$ and $\mathfrak{s}_2 \in \{x, y, z, u, v, w\}$, $\otimes \in \{<, \leq, >, \geq\}$, Ξ_1 and $\Xi_2 = \text{rand}(-8, 8)$ are variables randomly chosen, n is an iterator that grows from 1 to 200, and the time interval of the temporal operator is defined randomly as $I = [n+4, \ldots, n+4+\text{rand}(1,5)]$. For the case where the weights are not unit, the weights for Boolean p and temporal w operators are randomly chosen in an interval $(0, 1]$. In Fig. 10, we compare time performance for growing STL and unit weights in wSTL and wSTL with random weights. Note that the STL performance grows linearly and is faster than the other two. However, there is a slight difference between STL and wSTL, which is expected since more constraints are required in the encoding to capture the importance or preferences in the specification. Note that there is no significant change in using unit weights on the wSTL and random weights in the wSTL.

7 CONCLUSIONS

We presented a Mixed Integer Linear Programming formulation for Weighted Signal Temporal Logic. The weighted robustness associated with the wSTL improved the system's optimal behavior in a control synthesis framework where prioritized tasks and time precedence are critical. We compare features that wSTL specifications can capture that STL cannot, for instance, preferences and importance over Boolean and temporal operators. Another difference is shown when the specifications are infeasible standard STL definition will result in a trajectory that averages the violation within the conflicting sub-formulae, whereas wSTL can make the solution to reduce violation towards a preferred option. In extreme cases, if the weight difference is significant, it can lead to partial satisfaction. Finally, the time performance of the MILP encoding is shown compared with STL showing a small cost to pay at the expense of being able to specify preferences and importance.

REFERENCES

[1] Sofie Ahlberg and Dimos V Dimarogonas. 2019. Human-in-the-loop control synthesis for multi-agent systems under hard and soft metric interval temporal logic specifications. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE, 788–793.

[2] Hammad Ahmad and Jean-Baptiste Jeannin. 2021. A program logic to verify signal temporal logic specifications of hybrid systems. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*. 1–11.

[3] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of model checking*. MIT press.

[4] Fernando S Barbosa, Daniel Duberg, Patric Jensfelt, and Jana Tumova. 2019. Guiding autonomous exploration with signal temporal logic. *IEEE Robotics and Automation Letters* 4, 4 (2019), 3332–3339.

[5] Pierfrancesco Bellini, Riccardo Mattolini, and Paolo Nesi. 2000. Temporal logics for real-time system specification. *ACM Computing Surveys (CSUR)* 32, 1 (2000), 12–42.

[6] Andrea Bisoffi and Dimos V. Dimarogonas. 2021. Satisfaction of Linear Temporal Logic Specifications Through Recurrence Tools for Hybrid Systems. *IEEE Trans. Automat. Control* 66, 2 (2021), 818–825. https://doi.org/10.1109/TAC.2020.2984724

[7] Gustavo A Cardona, David Saldaña, and Cristian-Ioan Vasile. 2022. Planning for Modular Aerial Robotic Tools with Temporal Logic Constraints. In *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2878–2883.

[8] Gustavo A Cardona and Cristian-Ioan Vasile. 2022. Partial Satisfaction of Signal Temporal Logic Specifications for Coordination of Multi-robot Systems. In *Algorithmic Foundations of Robotics XV: Proceedings of the Fifteenth Workshop on the Algorithmic Foundations of Robotics*. Springer, 223–238.

[9] Hongkai Chen, Scott A Smolka, Nicola Paoletti, and Shan Lin. 2022. An STL-based Approach to Resilient Control for Cyber-Physical Systems. *arXiv preprint arXiv:2211.02794* (2022).

[10] Jyotirmoy V Deshmukh, Alexandre Donzé, Shromona Ghosh, Xiaoqing Jin, Garvit Juniwal, and Sanjit A Seshia. 2017. Robust online monitoring of signal temporal logic. *Formal Methods in System Design* 51, 1 (2017), 5–30.

[11] Adel Dokhanchi, Bardh Hoxha, and Georgios Fainekos. 2014. *5th International Conference on Runtime Verification, Toronto, ON, Canada.* Springer, Chapter On-Line Monitoring for Temporal Logic Robustness, 231–246.

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

[13] LLC Gurobi Optimization. 2020. Gurobi Optimizer Reference Manual. http://www.gurobi.com

[14] Iman Haghighi, Noushin Mehdipour, Ezio Bartocci, and Calin Belta. 2019. Control from signal temporal logic specifications with smooth cumulative quantitative semantics. In *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 4361–4366.

[15] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. 2001. Introduction to automata theory, languages, and computation. *Acm Sigact News* 32, 1 (2001), 60–65.

[16] Disha Kamale, Eleni Karyofylli, and Cristian-Ioan Vasile. 2021. Automata-based optimal planning with relaxed specifications. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 6525–6530.

[17] Sertac Karaman, Ricardo G Sanfelice, and Emilio Frazzoli. 2008. Optimal control of mixed logical dynamical systems with linear temporal logic specifications. In *2008 47th IEEE Conference on Decision and Control*. IEEE, 2117–2122.

[18] Jesper Karlsson, Fernando S Barbosa, and Jana Tumova. 2020. Sampling-based motion planning with temporal logic missions and spatial preferences. *IFAC-PapersOnLine* 53, 2 (2020), 15537–15543.

[19] Ron Koymans. 1990. Specifying real-time properties with metric temporal logic. *Real-time systems* 2, 4 (1990), 255–299.

[20] Hadas Kress-Gazit, Morteza Lahijanian, and Vasumathi Raman. 2018. Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), 211–236.

[21] Vincent Kurtz and Hai Lin. 2022. Mixed-Integer Programming for Signal Temporal Logic with Fewer Binary Variables. *IEEE Control Systems Letters* 6 (2022), 2635–2640.

[22] Lars Lindemann and Dimos V Dimarogonas. 2019. Robust control for signal temporal logic specifications using discrete average space robustness. *Automatica* 101 (2019), 377–387.

[23] Oded Maler and Dejan Nickovic. 2004. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 152–166.

[24] Noushin Mehdipour, Farzaneh Abdollahi, and Morteza Mirzaei. 2015. Consensus of multi-agent systems with double-integrator dynamics in the presence of moving obstacles. In *2015 IEEE Conference on Control Applications (CCA)*. IEEE, 1817–1822.

[25] Noushin Mehdipour, Cristian-Ioan Vasile, and Calin Belta. 2019. Average-based robustness for continuous-time signal temporal logic. In *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 5312–5317.

[26] Noushin Mehdipour, Cristian-Ioan Vasile, and Calin Belta. 2020. Specifying user preferences using weighted signal temporal logic. *IEEE Control Systems Letters* 5, 6 (2020), 2006–2011.

[27] Aniruddh Puranic, Jyotirmoy Deshmukh, and Stefanos Nikolaidis. 2021. Learning from demonstrations using signal temporal logic. In *Conference on Robot Learning*. PMLR, 2228–2242.

[28] Hazhar Rahmani and Jason M O'Kane. 2019. Optimal temporal logic planning with cascading soft constraints. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2524–2531.

[29] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia. 2014. Model predictive control with signal temporal logic specifications. In *53rd IEEE Conference on Decision and Control*. 81–87. https://doi.org/10.1109/CDC.2014.7039363

[30] Alëna Rodionova, Lars Lindemann, Manfred Morari, and George J Pappas. 2022. Combined left and right temporal robustness for control under stl specifications. *IEEE Control Systems Letters* 7 (2022), 619–624.

[31] Sadra Sadraddini and Calin Belta. 2015. Robust temporal logic model predictive control. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 772–779.

[32] Sayan Saha and A Agung Julius. 2016. An MILP approach for real-time optimal controller synthesis with metric temporal logic specifications. In *2016 American Control Conference (ACC)*. IEEE, 1105–1110.

[33] Philipp Schillinger, Mathias Bürger, and Dimos V Dimarogonas. 2017. Multi-objective search for optimal multi-robot planning with finite LTL specifications and resource constraints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 768–774.

[34] Dawei Sun, Jingkai Chen, Sayan Mitra, and Chuchu Fan. 2022. Multi-agent motion planning from signal temporal logic specifications. *IEEE Robotics and Automation Letters* 7, 2 (2022), 3451–3458.

[35] Ilya Tkachev and Alessandro Abate. 2013. Formula-Free Finite Abstractions for Linear Temporal Verification of Stochastic Hybrid Systems. In *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control (HSCC '13)*. Association for Computing Machinery, New York, NY, USA, 283–292. https://doi.org/10.1145/2461328.2461372

[36] Cristian-Ioan Vasile, Derya Aksaray, and Calin Belta. 2017. Time window temporal logic. *Theoretical Computer Science* 691 (2017), 27–54.

[37] Cristian-Ioan Vasile, Jana Tumova, Sertac Karaman, Calin Belta, and Daniela Rus. 2017. Minimum-violation scLTL motion planning for mobility-on-demand. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1481–1488.

[38] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M Murray. 2010. Receding horizon control for temporal logic specifications. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*. 101–110.

[39] Guang Yang, Calin Belta, and Roberto Tron. 2020. Continuous-time signal temporal logic planning with control barrier functions. In *2020 American Control Conference (ACC)*. IEEE, 4612–4618.

APPENDIX A

MILP encoding for STL

Below is the MILP formulation for STL specifications as proposed in [31].

For each predicate $\mu = (y - \pi)$, we define a binary variable $z_k^\pi \in 0, 1$ where 1 indicates satisfaction. M is chosen to be sufficiently large, $M \geq \max_i y_i$ for $i \in \{1, \ldots, n_y\}$. The constraints for predicate satisfaction are encoded as follows:

$$y + M(1 - z^\mu) \geq \rho, \tag{32}$$

$$y - Mz^\mu \leq \rho, \tag{33}$$

$$z_k^\pi \in \{0, 1\}, \forall \mu = y - \pi, \forall k \in \{0, \ldots, K\} \tag{34}$$

The logical operators - conjunction and disjunction are defined as follows:

$$z = \bigwedge_i z_i \Rightarrow z \leq z_i, \forall i \tag{35}$$

$$z = \bigvee_i z_i \Rightarrow z \leq \sum_i z_i \tag{36}$$

$$z \in [0, 1] \tag{37}$$

where $z \in [0,1]$ is a continuous variable, but as it follows from the encoding above, it can only take binary values. Let us introduce $z_k^{\varphi} \in \{0,1\}$ as a variable denoting the satisfaction of the given STL formula. Thus, an STL formula can be recursively defined using the following constraints:

$$\phi = \bigwedge_i \phi_i \Rightarrow z_k^{\phi} = \bigwedge_i z_k^{\phi_i} \tag{38}$$

$$\phi = \bigvee_i \phi_i \Rightarrow z_k^{\phi} = \bigvee_i z_k^{\phi_i} \tag{39}$$

$$\phi = \Box_I \psi \Rightarrow z_k^{\phi} = \bigwedge_{k' \in k+I} z_{k'}^{\psi} \tag{40}$$

$$\phi = \Diamond_I \psi \Rightarrow z_k^{\phi} = \bigvee_{k' \in k+I} z_{k'}^{\psi} \tag{41}$$

$$\phi = \psi_1 \mathcal{U} \psi_2 \Rightarrow z_k^{\phi} = \bigvee_{k' \in k+I} \left(z_{k'}^{\psi_2} \wedge \bigwedge_{k'' \in [k,k']} z_{k''}^{\psi_1} \right) \tag{42}$$

$$z_k^{\phi} \in [0,1], \forall \phi \neq \mu, \forall k \in \{0, \dots, K\} \tag{43}$$

$$z_0^{\phi} = 1 \tag{44}$$

$$\rho \geq 0 \tag{45}$$

Equations (45) and (46) impose the constraint of satisfaction.