Interval Signal Temporal Logic from Natural Inclusion Functions

Luke Baird, *Student Member, IEEE*, Akash Harapanahalli, *Student Member, IEEE*, and Samuel Coogan, *Senior Member, IEEE*

***Abstract*— We propose an interval extension of Signal Temporal Logic (STL) called Interval Signal Temporal Logic (I-STL). Given an STL formula, we consider an interval inclusion function for each of its predicates. Then, we use minimal inclusion functions for the $\min$ and $\max$ functions to recursively build an interval robustness that is a natural inclusion function for the robustness of the original STL formula. The resulting interval semantics accommodate, for example, uncertain signals modeled as a signal of intervals and uncertain predicates modeled with appropriate inclusion functions. In many cases, verification or synthesis algorithms developed for STL apply to I-STL with minimal theoretic and algorithmic changes, and existing code can be readily extended using interval arithmetic packages at negligible computational expense. To demonstrate I-STL, we present an example of offline monitoring from an uncertain signal trace obtained from a hardware experiment and an example of robust online control synthesis enforcing an STL formula with uncertain predicates.**

***Index Terms*— Autonomous systems, constrained control, fault detection.**

I. Introduction

SIGNAL Temporal Logic (STL) is an expressive language for encoding desired dynamic behavior of a system. STL specifications are built from predicate functions over the system output as well as Boolean and temporal connectives. For example, a warehouse robot may be required to visit regions defined by predicate functions in a prescribed order and deadline, or a building HVAC system might be allowed to violate a prescribed temperature range for only a limited period of time. STL is equipped with both qualitative logical semantics [1] and quantitative robustness semantics [2] that quantify the margin by which a specification is violated or satisfied.

Two major applications of STL include monitoring and control synthesis. For monitoring, the goal is to determine whether a given signal satisfies an STL specification [3]. There are several available tools and algorithms in the literature for efficient monitoring of an STL specification [4], [5], [6]. For control synthesis, the goal is to obtain a control strategy such that the resulting system output is guaranteed to satisfy a given STL specification. Control synthesis is often posed as an optimal control problem by including the robustness metric in the cost or constraints. This problem is generally non-convex and non-smooth due to the composition of $\min$ and $\max$ appearing in the definition of the robustness metric and is often converted to a mixed-integer program [7], [8]. For example, a state-of-the-art mixed-integer linear program (MILP) for STL control sythesis over affine predicates with linear costs using a minimal number of binary variables is proposed in [8] and implemented in the `stlpy` Python package. Alternate approaches to control synthesis include under-approximating the non-smooth robustness metric with a smooth approximation [9], [10] and using control barrier functions for certain fragments of STL [11], [12].

One major challenge is accommodating uncertainty in the system dynamics, the system output, and/or the STL specification itself. A variation of STL called pSTL allows satisfaction or violation of a specification over a signal to occur with some probability [13]. Similarly, the paper [14] propagates stochastic robustness intervals of STL robustness with linear predicates for safe motion planning. The paper [15] proposes a monitoring algorithm that accommodates uncertainty and time perturbations using intervals for finite-horizon STL formulas but is limited to monitoring and does not consider uncertainty in the STL predicates. In the context of online monitoring, the paper [6] presents an algorithm where the robustness of a partial signal is predicted as an interval before an entire signal is observed so that satisfaction or violation can be reported early if zero robustness is not in the interval. The paper [16] develops an offline monitoring algorithm for handling common models of sensor uncertainty within an STL framework.

The main contribution of this letter is an interval extension of STL called Interval-STL (I-STL) to accommodate interval-valued uncertainty in the system or specification. Note that we avoid probabilistic considerations such as [17] and use intervals to model uncertainty yielding formal guarantees. The syntax and semantics of I-STL are the same as STL except interval functions replace predicate functions and $\min$ and $\max$ are replaced with their minimal inclusion function counterparts, resulting in interval-valued quantitative robustness semantics and three-valued qualitative logical semantics for I-STL. Unlike previous works, our construction accommodates uncertainty in the predicate functions themselves. Our main theorem is a soundness result establishing that the interval robustness of I-STL over-approximates the usual STL robustness under any realization of the uncertainty, and similarly

This work was supported by the National Science Foundation under grants 1749357 and 2219755 and the Air Force Office of Scientific Research under Grant FA9550-23-1-0303.

L. Baird, A. Harapanahalli, and S. Coogan are with the Electrical and Computer Engineering Department at the Georgia Institute of Technology, Atlanta, GA 30318 USA. {lbaird38,aharapan,sam.coogan}@gatech.edu

for the logical semantics. We identify a class of specifications for which the I-STL robustness interval is minimal. A main feature of I-STL is that, since its definition is built from natural inclusion functions and interval arithmetic, existing algorithms for STL are often easily extended to I-STL using mature interval analysis packages at negligible computational expense. In particular, we extend `stlpy` to I-STL using our interval toolbox `npinterval` [18], and we demonstrate the resulting algorithms on two examples: monitoring an uncertain signal and synthesizing a controller for an uncertain system.

This letter is outlined as follows. Section II presents mathematical preliminaries needed for the interval arithmetic and STL. Section III is the primary theoretic contribution of this letter describing I-STL. Section IV gives a brief discussion of advantages and limitations of I-STL. Section V provides examples of our method applied to monitoring and control synthesis followed by Section VI which concludes this letter.

II. Mathematical Preliminaries

A. Notation

We denote the standard partial order on $\mathbb{R}^n$ by $\leq$, i.e., for $x, y \in \mathbb{R}^n$, $x \leq y$ if and only if $x_i \leq y_i$ for all $i \in \{1, \dots, n\}$. A (bounded) *interval* of $\mathbb{R}^n$ is a set of the form $\{z : \underline{x} \leq z \leq \overline{x}\} =: [\underline{x}, \overline{x}]$ for some endpoints $\underline{x}, \overline{x} \in \mathbb{R}^n$, $\underline{x} \leq \overline{x}$. Let $\mathbb{IR}^n$ denote the set of all intervals on $\mathbb{R}^n$. We also use the notation $[x] \in \mathbb{IR}^n$ to denote an interval when its endpoints are not relevant or implicitly understood to be $\underline{x}$ and $\overline{x}$. For a function $f : \mathbb{R}^n \to \mathbb{R}^m$ and a set $\mathcal{X} \subseteq \mathrm{dom}(f)$, define the set valued extension $f(\mathcal{X}) := \{f(x) : x \in \mathcal{X}\}$.

A discrete-time signal in $\mathbb{R}^n$ is a function $\mathbf{x} : \mathbb{N} \to \mathbb{R}^n$ where $\mathbb{N} = \{0, 1, 2, \dots\}$. A discrete-time interval signal in $\mathbb{IR}^n$ is a function $[\mathbf{x}] : \mathbb{N} \to \mathbb{IR}^n$. If $\mathbf{x}$ and $[\mathbf{x}]$ are such that $\mathbf{x}(t) \in [\mathbf{x}](t)$ for all $t \in \mathbb{N}$, we write $\mathbf{x} \in [\mathbf{x}]$.

B. Interval Analysis

Interval analysis extends operations and functions to intervals [19]. For example, if we know that $a \in [\underline{a}, \overline{a}]$, and $b \in [\underline{b}, \overline{b}]$, it is easy to see that the sum $(a+b) \in [\underline{a}+\underline{b}, \overline{a}+\overline{b}]$. The same idea extends to general functions, using an inclusion function to over-approximate its output.

Definition 1 (Inclusion Function [19])**.** Given a function $f : \mathbb{R}^n \to \mathbb{R}^m$, the interval function $[f] = [\underline{f}, \overline{f}] : \mathbb{IR}^n \to \mathbb{IR}^m$ is an *inclusion function* for f if, for every $[\underline{x}, \overline{x}] \in \mathbb{IR}^n$, $f([\underline{x}, \overline{x}]) \subseteq [f]([\underline{x}, \overline{x}])$, or equivalently

$$\underline{f}([\underline{x}, \overline{x}]) \leq f(x) \leq \overline{f}([\underline{x}, \overline{x}]) \quad \text{for all } x \in [\underline{x}, \overline{x}].$$

An inclusion function is *minimal* if for every $[\underline{x}, \overline{x}]$, $[f]([\underline{x}, \overline{x}])$ is the smallest interval containing $f([\underline{x}, \overline{x}])$, or equivalently

$$[f]_i([\underline{x}, \overline{x}]) = \left[\inf_{x \in [\underline{x}, \overline{x}]} f_i(x), \ \sup_{x \in [\underline{x}, \overline{x}]} f_i(x) \right],$$

for each $i \in \{1, \dots, m\}$.

Of particular relevance to this letter are the minimal inclusion functions for $\min$ and $\max$.

Proposition 1. *The minimal inclusion functions for* $\min(x_1, x_2)$ *and for* $\max(x_1, x_2)$ *with* $x_1 \in [\underline{x}_1, \overline{x}_1] \in \mathbb{IR}$, $x_2 \in [\underline{x}_2, \overline{x}_2] \in \mathbb{IR}$, *denoted as* $[\min]$ *and* $[\max]$, *are given by*

$$[\min]([x_1], [x_2]) = [\min(\underline{x}_1, \underline{x}_2), \min(\overline{x}_1, \overline{x}_2)], \tag{1}$$
$$[\max]([x_1], [x_2]) = [\max(\underline{x}_1, \underline{x}_2), \max(\overline{x}_1, \overline{x}_2)]. \tag{2}$$

Moreover, $[\min]$ *and* $[\max]$ *extend inductively to multiple arguments in the usual way,* e.g., $[\min]([x_1], [x_2], [x_3]) = [\min(\underline{x}_1, \underline{x}_2, \underline{x}_3), \min(\overline{x}_1, \overline{x}_2, \overline{x}_3)]$, *etc.*

For some common functions, the minimal inclusion function is easily defined. For example, if a function is monotone, the minimal inclusion function is simply the interval created by the function evaluated at its endpoints. However, when considering general functions, finding the minimal inclusion function is often not computationally viable. The following proposition provides a more computationally tractable approach.

Proposition 2 (Natural Inclusion Functions)**.** *Given a function* $f : \mathbb{R}^n \to \mathbb{R}^m$ *defined by a composition of functions/operations with known inclusion functions as* $f = e_\ell \circ e_{\ell-1} \circ \cdots \circ e_1$, *an inclusion function for* f *is formed by replacing each composite function with its inclusion function as* $[f] = [e_\ell] \circ [e_{\ell-1}] \circ \cdots \circ [e_1]$, *and is called a natural inclusion function.*

Existing software tools such as `CORA` [20] and `npinterval` [18] automate the construction of natural inclusion functions from general functions. We refer to [19, Section 2.4] for further discussion and other techniques to obtain other inclusion functions.

C. Signal Temporal Logic

Signal Temporal Logic (STL) is defined over a set $\mathcal{P}$ of *predicate functions* where each $\mu \in \mathcal{P}$ is a function $\mu : \mathbb{R}^n \to \mathbb{R}$. STL specifications are formed using the syntax [10], [7]

$$\phi \triangleq (\mu(x) \geq 0) | \neg\phi | \phi \wedge \psi | \phi \mathcal{U}_{[t_1, t_2]} \psi \tag{3}$$

where $\mu \in \mathcal{P}$. The operators conjunction $\wedge$, until $\mathcal{U}$, and negation $\neg$ may be used to define disjunction $\vee$, eventually $\Diamond$, and always $\Box$. We occasionally write $\phi_{\mathcal{P}}$ to emphasize that ϕ is over the set of predicate functions $\mathcal{P}$.

An STL specification ϕ is evaluated over a discrete-time signal $\mathbf{x} : \mathbb{N} \to \mathbb{R}^n$. The quantitative robustness ρ^ϕ of a specification ϕ evaluated over signal $\mathbf{x}$ at time $t \in \mathbb{N}$ is defined and calculated recursively as in [10, Definition 1].

Qualitative semantics of STL formula ϕ evaluated over signal $\mathbf{x}$ are recovered from the robustness as [10]

$$\left[\mathbf{x} \models \phi\right] = \begin{cases} \textsc{True} & \text{if } \rho^\phi(\mathbf{x}, 0) \geq 0 \\ \textsc{False} & \text{if } \rho^\phi(\mathbf{x}, 0) < 0. \end{cases} \tag{4}$$

Note that we adopt the convention that if $\rho^\phi(\mathbf{x}, 0) = 0$, then $\left[\mathbf{x} \models \phi\right] = \textsc{True}$, although this case is sometimes considered an undefined truth evaluation in the literature.

III. Interval Signal Temporal Logic

In standard STL, the robustness ρ^ϕ of a specification ϕ evaluated over a signal $\mathbf{x}$ at a time t is a single number. With the aim of incorporating bounded uncertainty in signal values and in predicate functions, in this section, we define and characterize *Interval Signal Temporal Logic* (I-STL) that is evaluated over interval signals and whose quantitative semantics give an interval of robustness. We connect this to an original STL specification by defining an induced I-STL specification given inclusion functions for the predicates.

I-STL is defined over a set of *interval predicate functions* $\mathcal{I}$ where each $\mathcal{M} \in \mathcal{I}$ is an interval function $\mathcal{M} : \mathbb{IR}^n \to \mathbb{IR}$. I-STL syntax is the same as STL except we exchange predicate functions for interval predication functions.

Definition 2. (I-STL Syntax) Given a set $\mathcal{I}$ of interval predicate functions, I-STL syntax is defined by

$$\phi \triangleq (\mathcal{M}([x]) \subseteq [0, \infty]) | \neg\phi | \phi \wedge \psi | \phi \mathcal{U}_{[t_1,t_2]} \psi \tag{5}$$

for $\mathcal{M} \in \mathcal{I}$.

An I-STL specification ϕ is evaluated over a discrete-time interval signal $[\mathbf{x}] : \mathbb{N} \to \mathbb{IR}^n$ where $[\mathbf{x}](t) \in \mathbb{IR}^n$ for each time $t \in \mathbb{N}$. Using the minimal inclusion functions $[\min]$ and $[\max]$ given in (1) and (2), we now define the quantitative interval robustness semantics of I-STL as follows.

Definition 3. (I-STL Quantitative Semantics) The *interval robustness* $[\rho]^\phi$ of an I-STL specification ϕ evaluated over an interval signal $[\mathbf{x}]$ at time step t is calculated recursively using natural inclusion functions as

$$\begin{aligned}
[\rho]^{\Pi}([\mathbf{x}], t) &= \mathcal{M}([\mathbf{x}](t)), \quad \Pi = (\mathcal{M}([x]) \subseteq [0, \infty]) \\
[\rho]^{\neg\phi}([\mathbf{x}], t) &= -[\rho]^{\phi}([\mathbf{x}], t) \\
[\rho]^{\phi\wedge\psi}([\mathbf{x}], t) &= [\min]\big([\rho]^{\phi}([\mathbf{x}], t), [\rho]^{\psi}([\mathbf{x}], t)\big) \\
[\rho]^{\phi\vee\psi}([\mathbf{x}], t) &= [\max]\big([\rho]^{\phi}([\mathbf{x}], t), [\rho]^{\psi}([\mathbf{x}], t)\big) \\
[\rho]^{\Box_{[t_1,t_2]}\phi}([\mathbf{x}], t) &= \underset{t' \in [t+t_1, t+t_2]}{[\min]} \big([\rho]^{\phi}([\mathbf{x}], t')\big) \\
[\rho]^{\Diamond_{[t_1,t_2]}\phi}([\mathbf{x}], t) &= \underset{t' \in [t+t_1, t+t_2]}{[\max]} \big([\rho]^{\phi}([\mathbf{x}], t')\big) \\
[\rho]^{\phi\mathcal{U}_{[t_1,t_2]}\psi}([\mathbf{x}], t) & \\
= \underset{t' \in [t+t_1, t+t_2]}{[\max]} & [\min]\left([\rho]^{\phi}([\mathbf{x}], t'), \underset{t'' \in [t+t_1, t']}{[\min]} \big([\rho]^{\psi}([\mathbf{x}], t'')\big)\right).
\end{aligned} \tag{6}$$

We also define three-valued logical semantics from the quantitative interval semantics as follows.

Definition 4. (I-STL Three-Valued Logical Semantics) The truth-value of I-STL formula ϕ evaluated over interval signal $[\mathbf{x}]$ is denoted $\big[[\mathbf{x}] \models \phi\big]$ and given by

$$\big[[\mathbf{x}] \models \phi\big] = \begin{cases} \textsc{True} & \text{if } [\rho]^\phi(\mathbf{x}, 0) \subseteq [0, \infty] \\ \textsc{False} & \text{if } [\rho]^\phi(\mathbf{x}, 0) \subseteq [-\infty, 0) \\ \textsc{Undef} & \text{else.} \end{cases} \tag{7}$$

We now establish the key property of I-STL: it provides interval bounds on the robustness of an STL specification given interval uncertainty in the predicate functions and/or signal.

Definition 5 (Predicate interval extensions)**.** Given a set of predicate functions $\mathcal{P}$, a set of interval predicate functions $\mathcal{I}$ is an *interval extension* of $\mathcal{P}$ if for each $\mu \in \mathcal{P}$ there exists a $\mathcal{M} \in \mathcal{I}$ such that $\mathcal{M}$ is an inclusion function for μ.

Example 1. Consider the predicate function $\mu : \mathbb{R}^n \to \mathbb{R}$ such that $\mu(x) := \|x\|_2^2 - r = \sum_{i=1}^n x_i^2 - r$, representing, *e.g.*, a circular obstacle. Then an interval predicate function $\mathcal{M} : \mathbb{IR}^n \to \mathbb{IR}$ can be constructed following the framework from [18]: for each $i = 1, \ldots, n$, define $\underline{y}_i := \begin{cases} 0 & \underline{x}_i \leq 0 \leq \overline{x}_i \\ \min(\underline{x}_i^2, \overline{x}_i^2) & \text{otherwise} \end{cases}$, and $\overline{y}_i := \max(\underline{x}_i^2, \overline{x}_i^2)$; then, $\mathcal{M}([\underline{x}, \overline{x}]) = \left[\sum_{i=1}^n \underline{y}_i - r, \sum_{i=1}^n \overline{y}_i - r\right]$ is an inclusion function for $\mu(x)$.

When $\mathcal{I}$ is an interval extension of $\mathcal{P}$, we can obtain an I-STL specification over $\mathcal{I}$ from an STL specification ϕ over $\mathcal{P}$ by replacing every instance of a predicate function μ with the corresponding $\mathcal{M}$.

Definition 6 (Induced I-STL specification)**.** Given an STL specification $\phi_\mathcal{P}$ over the set of predicate functions $\mathcal{P}$ and a set of interval predicate functions $\mathcal{I}$ that is an extension of $\mathcal{P}$, the I-STL specification that is obtained by replacing each instance of a predicate function $\mu(x)$ in $\phi_\mathcal{P}$ with the corresponding interval predicate function $\mathcal{M}([x])$ is the I-STL specification over $\mathcal{I}$ *induced by* $\phi_\mathcal{P}$ and is denoted $\phi_\mathcal{I}$. When no confusion arises, we sometimes drop the subscript and write ϕ for an STL specification and its induced I-STL specification.

We now present the main theoretical result of this letter, linking the semantics of an STL specification to the semantics of its induced I-STL specification.

Theorem 1 (Soundness of Quantitative Semantics)**.** *Let $\phi_\mathcal{P}$ be an STL specification over the set of predicate functions $\mathcal{P}$. Let $\mathcal{I}$ be an interval extension of $\mathcal{P}$ and let $\phi_\mathcal{I}$ be the I-STL specification over $\mathcal{I}$ induced by $\phi_\mathcal{P}$. Then, for any interval signal $[\mathbf{x}] : \mathbb{N} \to \mathbb{IR}^n$ and any signal $\mathbf{x} \in [\mathbf{x}]$, it holds that*

$$\rho^{\phi_\mathcal{P}}(\mathbf{x}, t) \in [\rho]^{\phi_\mathcal{I}}([\mathbf{x}], t) \quad \textit{for all } t. \tag{8}$$

Moreover,

$$\begin{aligned}
\big[[\mathbf{x}] \models \phi_\mathcal{I}\big] = \textsc{True} \quad &\textit{implies} \quad \big[\mathbf{x} \models \phi_\mathcal{P}\big] = \textsc{True}, \textit{ and} \\
\big[[\mathbf{x}] \models \phi_\mathcal{I}\big] = \textsc{False} \quad &\textit{implies} \quad \big[\mathbf{x} \models \phi_\mathcal{P}\big] = \textsc{False}.
\end{aligned} \tag{9}$$

Proof. Because each $\mathcal{M} \in \mathcal{I}$ is a inclusion function for its corresponding predicate function $\mu \in \mathcal{P}$ and $[\min]$ and $[\max]$ are inclusion functions, each equation in (6) is an inclusion function for the corresponding equation in [10, Definition 1] by Proposition 2. Thus, $[\rho]^{\phi_\mathcal{I}}$ is a natural inclusion function for $\rho^{\phi_\mathcal{P}}$, immediately implying (8). For (9), we observe that

$$\big[[\mathbf{x}] \models \phi_\mathcal{I}\big] = \textsc{True} \implies \underline{\rho}^{\phi_\mathcal{I}}([\mathbf{x}], 0) \geq 0$$

so by (8), $\rho^{\phi_\mathcal{P}}(\mathbf{x}, 0) \geq 0$, that is, $\big[x \models \phi_\mathcal{P}\big] = \textsc{True}$. Symmetrically,

$$\big[[\mathbf{x}] \models \phi_\mathcal{I}\big] = \textsc{False} \implies \overline{\rho}^{\phi_\mathcal{I}}([\mathbf{x}], 0) < 0$$

so by (8), $\rho^{\phi_{\mathcal{P}}}(\mathbf{x}, 0) < 0$, that is, $\left[x \models \phi_{\mathcal{P}}\right] = \textsc{False}$ where $\underline{\rho}^{\phi_{\mathcal{I}}}$ and $\overline{\rho}^{\phi_{\mathcal{I}}}$ are the lower and upper-bounds of $[\rho]^{\phi_{\mathcal{I}}}$, that is, $[\underline{\rho}]^{\phi_{\mathcal{I}}}([\mathbf{x}], 0) = [\underline{\rho}^{\phi_{\mathcal{I}}}([\mathbf{x}], 0), \overline{\rho}^{\phi_{\mathcal{I}}}([\mathbf{x}], 0)]$. □

Note that if $\left[[\mathbf{x}] \models \phi\right] = \textsc{Undef}$ we cannot say anything about the truth value of $\left[\mathbf{x} \models \phi\right]$. Although in general the inclusion function $[\rho]^{\phi_{\mathcal{I}}}([x], t)$ is not minimal, we give an example of a class of specifications for which the I-STL robustness is minimal.

Proposition 3. *Let $\mathcal{P}$ be a set of monotone non-decreasing predicate functions μ_j with associated minimal inclusion functions $\mathcal{M}_j \in \mathcal{I}$. If the specification $\phi_{\mathcal{P}}$ has no negations, then the interval robustness $[\rho]^{\phi_{\mathcal{I}}}$ of the induced I-STL specification $\phi_{\mathcal{I}}$ from Definition 6 is the minimal inclusion function for the STL robustness $\rho^{\phi_{\mathcal{P}}}$ of $\phi_{\mathcal{P}}$.*

Proof. Let $\mu_1, \mu_2 : \mathbb{R}^n \to \mathbb{R}$ be monotone non-decreasing predicate functions with minimal inclusion functions $\mathcal{M}_1, \mathcal{M}_2$. Given an interval $[\underline{x}, \overline{x}]$, due to monotonicity, the interval robustness from I-STL of $\mu_1 \vee \mu_2$ is

$$\begin{aligned}
[\rho]^{\mu_1 \vee \mu_2}([\underline{x}, \overline{x}]) &= \Big[\max\Big(\min_{x \in [\underline{x}, \overline{x}]} \mu_1(x), \min_{x \in [\underline{x}, \overline{x}]} \mu_2(x)\Big), \\
&\qquad \max\Big(\max_{x \in [\underline{x}, \overline{x}]} \mu_1(x), \max_{x \in [\underline{x}, \overline{x}]} \mu_2(x)\Big)\Big], \\
&= [\max(\mu_1(\underline{x}), \mu_2(\underline{x})), \max(\mu_1(\overline{x}), \mu_2(\overline{x}))], \\
&= \Big[\min_{x \in [\underline{x}, \overline{x}]} \rho^{\mu_1 \vee \mu_2}(x), \max_{x \in [\underline{x}, \overline{x}]} \rho^{\mu_1 \vee \mu_2}(x)\Big].
\end{aligned}$$

This follows as $\min$ and $\max$ are monotone non-decreasing, so the composition with μ_1 and μ_2 is monotone non-decreasing. The same holds for $\wedge$ and $\min$. Every operator in I-STL is a composition of $\min$ and $\max$, thus it holds inductively that $[\rho]^{\phi_{\mathcal{I}}}$ is a minimal inclusion function for $\rho^{\phi_{\mathcal{P}}}$. □

IV. Computational Considerations of I-STL

In practice, I-STL specifications most naturally arise by incorporating uncertainty in settings with STL constraints. Aside from the theoretical soundness guarantees of Theorem 1, a key feature of I-STL is that, algorithmically, it is often straightforward to modify existing STL algorithms such as offline monitoring, online monitoring, and control synthesis to incorporate the quantitative semantics in Definition 3. Concretely, as we demonstrate in the case studies, this is often as simple as replacing appropriate numerical computations with their interval counterparts using existing interval arithmetic computation packages, and in many settings, the increase in computational effort is negligible. A contribution of this letter, therefore, is an extension of the `stlpy` package for STL monitoring and control synthesis [8] to allow for I-STL monitoring and control synthesis using our interval arithmetic package `npinterval` [18], which implements intervals as a native datatype in the Python `numpy` package.

For example, consider a setting in which an STL specification is given over known and fixed predicate functions $\mathcal{P}$. Suppose the objective is to monitor offline (*i.e.*, after all measurements are collected) the robustness of ϕ evaluated over a signal, but the true signal is not known exactly—with this uncertainty captured in the interval signal $[\mathbf{x}]$ instead. In this case, we construct a set of interval predicate functions $\mathcal{I}$ as, *e.g.*, the natural inclusion functions of the original predicate functions, $\mathcal{I} = \{[\mu] \mid \mu \in \mathcal{P}\}$, and then $[\rho]^{\phi}$ becomes an inclusion function for ρ^{ϕ}.

We generalize further and consider a setting in which the predicate functions are parameter-dependent, and the parameter is not known exactly but known to be within an interval. For example, consider an affine predicate of the form $\mu(x) = a^\top x - b$ for $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$. If a and b are uncertain and only known to be within the intervals $[a]$ and $[b]$, it is natural to consider an interval predicate

$$\mathcal{M}([x]) = [a]^\top [x] - [b]. \tag{10}$$

As an example, instantiating the predicate $\mu(x) = a^\top x - b$ in `stlpy` is achieved with, *e.g.*,

```
stlpy.STL.LinearPredicate(a,b)
```

for `numpy` arrays `a` and `b`. Creating the interval predicate (10) is achieved with

```
a_int = interval.get_iarray(_a,a_)
b_int = interval.get_iarray(_b,b_)
stlpy.STL.LinearPredicate(a_int,b_int)
```

where `_a`, `a_`, `_b`, `b_` are `numpy` arrays for the lower and upper endpoints of $[a]$ and $[b]$, and `get_iarray` returns the `numpy` array of the `interval` data type.

More generally, given a parameterized predicate function of the form $\mu(x, p)$ where $p \in \mathbb{R}^m$ is an unknown parameter vector known to be within the interval $[p]$, we take as an interval extension the interval predicate function $\mathcal{M}([x]) = [\mu]([x], [p])$ where $[\mu]$ is any inclusion function for μ.

For example, given a parameterized Python function `mu_p` : $\mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ with fixed `p` $\in \mathbb{R}^m$,

```
mu = lambda x : mu_p(x, p=p)
stlpy.STL.NonlinearPredicate(mu,n)
```

instantiates a nonlinear predicate paramerized by `numpy` array `p`. Comparatively, the code

```
p_int = interval.get_iarray(_p,p_)
M = lambda x : mu_p(x, p=p_int)
stlpy.STL.NonlinearPredicate(M,n)
```

instantiates a nonlinear interval predicate obtained from the natural inclusion function of the parameterized predicate function `mu_p` evaluated with an uncertain parameter in the interval `p_int` := [_p, p_] $\in \mathbb{IR}^m$. Note that `npinterval` automatically builds a natural inclusion function for `mu` when arrays of `interval` data-type are passed into the function.

We demonstrate this construction and its application in the examples in Section V. We also illustrate how I-STL can be used for enforcing safety specifications due to the construction from inclusion functions.

V. Examples

In this section, we provide two example use cases of I-STL. First, we demonstrate monitoring on a signal measured from an experiment with a miniature blimp. We consider both linear and nonlinear uncertain predicates and measurement uncertainty. Then, we show how I-STL can be used in conjunction

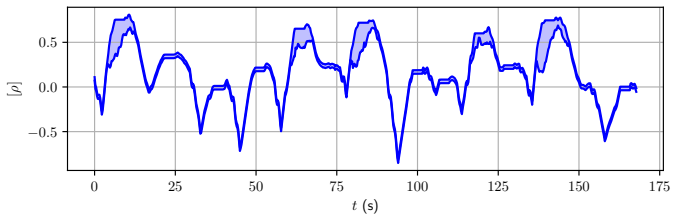

Fig. 1. The offline computed robustness of $\varphi \wedge \gamma$. The trajectory is generated from a way-point following PD controller that regularly violates the specification, suggesting the need for controller redesign, for example.

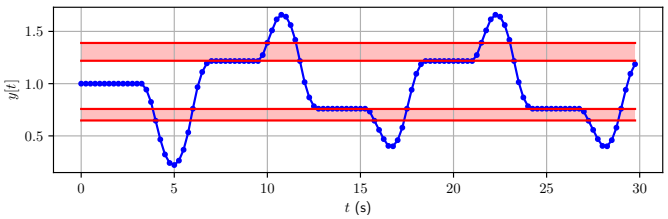

Fig. 2. Synthesized control policy output for a double integrator with uncertain states and predicates. A MILP finds the smallest input in magnitude at each time step such that the lower bound of the interval robustness is non-negative for all time. The uncertain interval predicates are plotted in red.

with theory from [21] for control synthesis of a linear system. Because our implementation for monitoring and control synthesis builds on `stlpy` [8], we convert STL formulas in code into positive normal form (PNF), where negation $\neg$ is only applied to predicates without loss of generality [22]. All simulations were performed on a 2022 Dell Precision 5570 running Ubuntu 22.04.3 LTS[1].

A. Interval Monitoring on a Miniature Blimp

We illustrate monitoring of a signal taken from an experiment with the GT-MAB miniature blimp hardware platform [23]. We wish to monitor the following two specifications, $\varphi = ([x\ y]^\top \notin S) \vee \Diamond_{[0,3/\Delta t]}\Box_{[0,2/\Delta t]}([x\ y]^\top \notin S)$ and $\gamma = \Diamond_{[0,3/\Delta t]}(-\|[v_x\ v_y\ v_z]^\top\|_2 + 2 \geq 0)$, where $\Delta t = 0.2s$ and the expression $[x\ y]^\top \notin S$ is written as $(x \geq d) \vee (x \leq -d) \vee (y \geq d) \vee (y \leq -d)$, where $d = 1.41m$ is half of the width of a square plus the radius of the blimp. The signal is generated from a PD controller with four way-points placed at the coordinates $(0, 1.51)$, $(1.51, 0)$, $(0, -1.51)$ and $(-1.51, 0)$ in the xy-plane. Due to measurement uncertainty, we add an interval of $\pm 0.075m/s$ to each of the velocity states and an interval of $\pm 0.020m$ to each of the position states. We use a natural inclusion function to handle the nonlinear predicate.

The results of monitoring offline for $\varphi \wedge \gamma$ is shown in Figure 1. Note that I-STL adds minimal overhead beyond what is equivalent to monitoring two signals instead of one due to the use of the `npinterval` Python package [18]. Standard STL robustness computations without uncertainty took $0.0035s$ per time step while I-STL computations with uncertainties took $0.0073s$ per time step, which is about 5% more than twice a standard STL robustness computation.

B. Robust Control Synthesis for a Linear System

Consider the following specification adapted from [21], [5]

$$\phi = \Diamond_{[0,\frac{4}{\Delta t}]}\big((y \leq 0.7) \vee (y \geq 1.3)\big) \wedge \big((0.7 \leq y \leq 1.3) \vee \Diamond_{[0,\frac{2}{\Delta t}]}\Box_{[0,\frac{2}{\Delta t}]}(0.7 \leq y \leq 1.3)\big), \tag{11}$$

on the discrete-time double integrator with bounded additive disturbance

$$x(t+1) = \underbrace{\begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}}_{A} x(t) + \underbrace{\begin{bmatrix} 0 \\ \Delta t \end{bmatrix}}_{B} u(t) + w(t), \tag{12}$$

[1]The code for these examples is available at `https://github.com/gtfactslab/Baird_LCSS2024`.

with $x(t) \in \mathbb{R}^2$, for all $t \in \mathbb{N}$, $u(t) \in [-1, 1]$, and $w(t) \in [\underline{w}, \overline{w}] = [-0.001, 0.001]^2$. Set $\Delta t = 0.25$. The horizon of an STL formula ϕ, denoted $\|\phi\|$, is the number of future time steps of a signal necessary to evaluate an STL formula. Its computation is given in [1], yielding $\|\phi\| = 4/0.25 = 16$ time steps for ϕ in (11). The output of the system is the position, $y := x_1$.

The requirement $0.7 \leq y \leq 1.3$ may be written as the conjunction of two affine predicate functions $\alpha y - \beta_1 \geq 0$, and $-\alpha y - \beta_2 \geq 0$ where $\alpha = 1$, $\beta_1 = 0.7$, and $\beta_2 = -1.3$. Similarly, the requirement $(y \leq 0.7) \vee (y \geq 1.3)$ can be written as the disjunction of the negation of the same predicates. Suppose, however, that there is uncertainty in the linear predicates captured with the interval bounds $[\underline{\alpha}, \overline{\alpha}] = [0.95, 1.05]$, $[\underline{\beta}_1, \overline{\beta}_1] = [0.68, 0.72]$, and $[\underline{\beta}_2, \overline{\beta}_2] = [-1.28, -1.32]$ for α, β_1, and β_2. We wish to minimize the control input such that the robustness is non-negative for all possible disturbances and all possible realizations of the interval predicates.

Using Theorem 1 with the I-STL specification induced by (11), our control objective is achieved by requiring that the lower bound on the interval robustness be non-negative. We use the formulation from [21, Algorithm 1], with slight modifications to accommodate I-STL. In particular, we replace the original dynamics constraints with a new embedding system giving lower and upper bounds $\underline{x}$ and $\overline{x}$ on the state trajectory which over-approximates the true behavior of the system, *i.e.*, for all possible disturbances, $x(t) \in [\underline{x}(t), \overline{x}(t)]$. In general, an embedding system may be constructed for a large class of systems using mixed-monotone systems theory [18]. Therefore, from [21, Equation (8)] using instead interval robustness, we obtain the optimization problem

$$\min_{u=\{u(t),\dots,u(t+N-1)\}} |u(t)| \tag{13}$$
$$\text{s.t.} \begin{bmatrix} \underline{x}(\tau+1) \\ \overline{x}(\tau+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix} \begin{bmatrix} \underline{x}(\tau) \\ \overline{x}(\tau) \end{bmatrix} + \begin{bmatrix} B \\ B \end{bmatrix} u(\tau) + \begin{bmatrix} \underline{w} \\ \overline{w} \end{bmatrix}$$
$$\underline{\rho}^{\phi}([y], \tau) \geq 0, \quad \max\{t - \|\phi\|, 0\} \leq \tau \leq t + N - b.$$

where A and B are the matrices from (12). We select $N = 16$, $b = 1$ and solve in a receding horizon fashion as a MILP using Gurobi. The resulting output sequence initialized at the state $x = [1\ 0]^\top$ is plotted in Figure 2. In Figure 3, we provide an empirical analysis of the tightness of the bounds of I-STL and its computational burden compared to computing true robustness intervals from MILPs. For this analysis, we consider the case without uncertainty in the predicates and at each time step, we fix a proposed input sequence from the

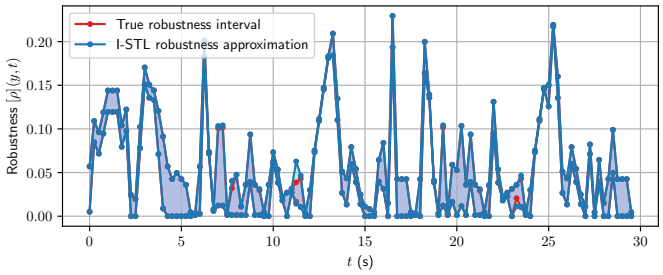

Fig. 3. I-STL robustness vs. exact interval robustness comparison for the double integrator case study without uncertain predicates to avoid bilinear constraints. At each time step, the I-STL interval robustness for a proposed safe trajectory is plotted in blue. The true interval robustness is plotted in red, solved by maximizing and minimizing $\boldsymbol{\rho}$ for the original system (12), with $\boldsymbol{w} \in [-\mathbf{0.001}, \mathbf{0.001}]^{\mathbf{2}}$, $\boldsymbol{u}$ set to the proposed safe input trajectory, and $\boldsymbol{x}$ initialized with historical states. Solving for the true interval robustness with a MILP takes on average $\mathbf{0.015}\boldsymbol{s}$ while computing the I-STL robustness takes on average $\mathbf{0.0014}\boldsymbol{s}$. Out of a total of $\mathbf{119}$ time steps, the I-STL robustness interval is minimal for $\mathbf{107}$ time steps and is no more than $\mathbf{10\%}$ larger than the exact robustness interval for $\mathbf{116}$ time steps.

solution of (13).

Note that the set of predicates for ϕ includes two predicates and their complements. Thus, it is not obvious which realization in the interval is the most conservative assumption. The most conservative realization of an interval predicate function depends on the history and the current time step, *e.g.*, whether to maintain satisfaction the signal must return to within $0.7 \leq y \leq 1.3$ nominally, or leave this range nominally.

The I-STL implementation doubles the state dimension and output dimension due to the embedding system, yielding double the dynamics equality constraints. Enforcing predicates in the I-STL constraint requires additional binary variables. When applying the mixed-integer encoding from [8] with affine interval predicates, the expression

$$\alpha^\top y(t) - b + M(1-z) \geq \rho(t)$$

is modified by using the minimal inclusion function for $[\alpha]^\top[y]$ (where p is the dimension of the output) to

$$\sum_{j=1}^{p} \min\{\underline{\alpha}_j \underline{y}_j, \underline{\alpha}_j \overline{y}_j, \overline{\alpha}_j \underline{y}_j, \overline{\alpha}_j \overline{y}_j\} - \underline{b} + M(1-z) \geq \underline{\rho}(t),$$

which introduces extra binary variables. Otherwise, the number of constraints used to encode I-STL robustness remains the same. Over a 119 time step trajectory in simulation, the I-STL implementation takes $0.46s$ to compute a safe control input per time step, while the STL implementation without disturbances and without uncertain predicates takes $0.15s$ per time step.

VI. Conclusion

We presented an interval extension of STL that uses inclusion functions to give sound interval overestimates of STL robustness. Using the `npinterval` package, I-STL can be efficiently used for robust monitoring or control synthesis with minimal code adaptation and computation time that is approximately twice that of the standard STL counterpart. In contrast, computing exact minimal and maximal robustness bounds is considerably more computationally intensive as demonstrated in the case study.

References

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

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

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

[4] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *International Conference on Computer Aided Verification*, pp. 167–170, Springer, 2010.

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

[6] J. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. Seshia, "Robust online monitoring of signal temporal logic," *Formal Methods in System Design*, vol. 51, no. 1, pp. 5–30, 2017.

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

[8] 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.

[9] K. Leung, N. Aréchiga, and M. Pavone, "Back-propagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods," in *International Workshop on the Algorithmic Foundations of Robotics*, pp. 432–449, Springer, 2020.

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

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

[12] M. Charitidou and D. V. Dimarogonas, "Receding horizon control with online barrier function design under signal temporal logic specifications," *IEEE Transactions on Automatic Control*, 2022.

[13] D. Sadigh and A. Kapoor, "Safe control under uncertainty with probabilistic signal temporal logic," in *Proceedings of Robotics: Science and Systems XII*, 2016.

[14] R. Ilyes, Q. Ho, and M. Lahijanian, "Stochastic robustness interval for motion planning with signal temporal logic," in *2023 IEEE International Conference on Robotics and Automation*, pp. 5716–5722, IEEE, 2023.

[15] B. Zhong, C. Jordan, and J. Provost, "Extending signal temporal logic with quantitative semantics by intervals for robust monitoring of cyber-physical systems," *ACM Transactions on Cyber-Physical Systems*, vol. 5, no. 2, pp. 1–25, 2021.

[16] B. Finkbeiner, M. Fränzle, F. Kohn, and P. Kröger, "A truly robust signal temporal logic: Monitoring safety properties of interacting cyber-physical systems under uncertain observation," *Algorithms*, vol. 15, no. 4, p. 126, 2022.

[17] L. Lindemann, L. Jiang, N. Matni, and G. J. Pappas, "Risk of stochastic systems for temporal logic specifications," *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 3, pp. 1–31, 2023.

[18] A. Harapanahalli, S. Jafarpour, and S. Coogan, "A toolbox for fast interval arithmetic in numpy with an application to formal verification of neural network controlled systems," in *ICML 2023 Workshop on Formal Verification of Machine Learning*, 2023.

[19] L. Jaulin, M. Kieffer, D. Olivier, and E. Walter, *Applied Interval analysis*. Springer, 2001.

[20] M. Althoff and D. Grebenyuk, "Implementation of interval arithmetic in CORA 2016," in *Proc. of the 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems*, pp. 91–105, 2016.

[21] L. Baird and S. Coogan, "Runtime assurance from signal temporal logic safety specifications," in *American Control Conference (ACC)*, pp. 3535–3540, 2023.

[22] J. Ouaknine and J. Worrell, "Some recent results in metric temporal logic," in *Formal Modeling and Analysis of Timed Systems: 6th International Conference, FORMATS 2008, Saint Malo, France, September 15-17, 2008. Proceedings 6*, pp. 1–13, Springer, 2008.

[23] Q. Tao, J. Wang, Z. Xu, T. X. Lin, Y. Yuan, and F. Zhang, "Swing-reducing flight control system for an underactuated indoor miniature autonomous blimp," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 4, pp. 1895–1904, 2021.