Acta Cybernetica **27** (2026) 383–403.

Guaranteed Satisfaction of a Signal Temporal Logic Formula on Tubes*

Joris Tillet[ab], Antoine Besset[ac],
and Julien Alexandre dit Sandretto[ad]

Abstract

This paper considers the issue of how to deal with Signal Temporal Logic (STL) when taking into account uncertainties. The STL is a formalism with a large expressiveness to describe real-time properties on real-value signals. It is particularly used for system verification. This work focuses on extensions of STL that handle bounded uncertainties on predicates or on the signal itself, by using tubes to represent the sets of signals. In this way, it becomes possible to robustly check the satisfaction of specifications for a noisy system. However, some cases are undecidable due to uncertainty, and other ones are too complex to determine. Mainly, this paper provides a literature review and compares the few state-of-the-art STL monitors able to deal with tubes. In addition, it proposes to go further by introducing Boolean intervals to formalize undecidable cases, and by implementing a new STL formalism applied to sets in DynIbex, a guaranteed integration tool. Thus, STL specifications can be validated in a guaranteed way for a simulated system. As a result, we obtain the same reliable result as the state-of-the-art, but faster. A robotic application with a drone is proposed to illustrate the concept.

Keywords: STL, interval methods, system verification, tubes

1 Introduction

Cyber-physical systems (CPS) are engineered, physical or biological systems (continuous and real-value systems) with a numerical attached system (discrete states) as a monitor or a controller. Designers and users of such systems often require guarantees on the system behavior, hence the need of a runtime verification process. This formal verification can be done using temporal logic, a formalism able to specify the system requirements with temporal constraints.

*This work was supported by the CIEDS (French Interdisciplinary Center for Defense and Security) within the STARTS project.

[a]ENSTA, Institut Polytechnique de Paris, Palaiseau Cedex, France
[b]E-mail: joris.tillet@ensta.fr, ORCID: 0000-0002-1955-6725
[c]E-mail: antoine.besset@ensta.fr, ORCID: 0009-0004-2662-306X
[d]E-mail: alexandre@ensta.fr, ORCID: 0000-0002-6185-2480

DOI: 10.14232/actacyb.315054

In 1977, Linear Temporal Logic (LTL) [24] is introduced, allowing to describe discrete time properties on a discrete signal. This is mainly used for formal verification of software or digital hardware, but reaches its limits when considering real-time constraints. Metric Temporal Logic [15] goes further by providing a semantics to speak of real-time properties on discrete signal (Boolean traces over continuous time). When considering CPS, man needs an extension of LTL able to handle real-time properties on real-value signal. This is the purpose of Signal Temporal Logic (STL) [22].

Let $\mathbf{x} = (x_1, \ldots, x_n) : \mathbb{R}^+ \to \mathbb{R}^n$ be a signal of dimension n and depending on time t. Let φ be an STL formula. We note the satisfaction of the formula φ by the trace of the signal $\mathbf{x}$ starting at time t by: $(\mathbf{x}, t) \vDash \varphi$.

The Signal Temporal Logic syntax is defined iteratively by:

$$\varphi := \top \mid \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \, \mathcal{U}_{[t_1,t_2]} \, \varphi_2 \tag{1}$$

with $\top$ denoting the *true* Boolean value, and μ is an atomic predicate: it is a constraint on the signal at a time t:

$$(\mathbf{x}, t) \vDash \mu \iff f(x_1(t), \ldots, x_n(t)) > 0. \tag{2}$$

$\neg$ is the logical *not* operator, $\wedge$ the logical *and* and $\mathcal{U}$ is the temporal *until* operator, defined below.

We also have:

$$\bot = \neg\top \qquad \textit{(False)} \tag{3}$$

$$\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2) \qquad \textit{(Or, using De Morgan's law)} \tag{4}$$

Example 1. Let $\mathbf{x} : \mathbb{R}^+ \to \mathbb{R}^6$ be the state of a drone in 3 dimension (position and speed):

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}. \tag{5}$$

We can express a limited speed when the drone is too close to the ground as follows:

$$\varphi = \mu_{\text{low\_height}} \implies \mu_{\text{speed}} \tag{6}$$

$$= \neg\mu_{\text{low\_height}} \vee \mu_{\text{speed}} \tag{7}$$

with

$$\mu_{\text{speed}} = \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} \leq 1 \tag{8}$$

$$\mu_{\text{low\_height}} = z \leq 10. \tag{9}$$

Then, if we consider the signal of a drone flying from time $t = 0$ s, and we check the satisfaction of the formula φ, we obtain (from (2)):

$$(\mathbf{x}, 0) \vDash \varphi \iff \begin{cases} z(0) > 10 \\ \text{or } \sqrt{\dot{x}(0)^2 + \dot{y}(0)^2 + \dot{z}(0)^2} \leq 1. \end{cases} \tag{10}$$

Our specification is only checked at a specific time of the signal (here at 0 s, *i.e.* the beginning of the signal). If we want to specify modalities with respect to time, we have to use the *until* operator of the STL.

Until The temporal operator $\mathcal{U}$ is the *until* operator defined on the time interval $[t_1, t_2]$ with $t_1, t_2 \in \mathbb{R}^+$ and $t_1 \leq t_2$, as follows:

$$(\mathbf{x}, t) \vDash \varphi_1 \, \mathcal{U}_{[t_1,t_2]} \, \varphi_2 \iff \exists t' \in [t_1, t_2] \; (x, t + t') \vDash \varphi_2 \text{ and } \forall t'' \in [t, t'] \; (x, t'') \vDash \varphi_1. \tag{11}$$

In other words, the until operator states that φ_1 must stay true until a time t' within $[t_1, t_2]$ when φ_2 is true. Note that nothing is specified on what happen after t'.

We can derive other useful operators from this operator:

$$\mathcal{F}_{[t_1,t_2]} \, \varphi = \top \, \mathcal{U}_{[t_1,t_2]} \, \varphi \qquad \textit{(Finally)} \tag{12}$$

$$\mathcal{G}_{[t_1,t_2]} \, \varphi = \neg \left(\mathcal{F}_{[t_1,t_2]} \, \neg \varphi \right) \qquad \textit{(Globally)} \tag{13}$$

Finally The *finally* operator $\mathcal{F}$ corresponds to the satisfaction of φ at a time t' within $[t_1, t_2]$:

$$(\mathbf{x}, t) \vDash \mathcal{F}_{[t_1,t_2]} \, \varphi \iff \exists t' \in [t_1, t_2] \; (\mathbf{x}, t + t') \vDash \varphi. \tag{14}$$

Globally The *globally* operator $\mathcal{G}$ corresponds to the satisfaction of φ during the whole period $[t_1, t_2]$:

$$(\mathbf{x}, t) \vDash \mathcal{G}_{[t_1,t_2]} \, \varphi \iff \forall t' \in [t_1, t_2] \; (\mathbf{x}, t + t') \vDash \varphi. \tag{15}$$

Example 2. If we consider again the example of the drone with limited speed, we can now express the same constraint but on the whole duration of the flight (final time is noted T), using the globally operator:

$$\varphi = \mathcal{G}_{[0,T]} \left(\mu_{\text{low\_height}} \implies \mu_{\text{speed}} \right). \tag{16}$$

STL enables a system designer to express properties on real-time and real-valued signals. This formalism is very flexible and can adapt easily to many systems, and yet it is concise.

Example 3. Now we can express more subtle specifications for our drone. For instance:

$$\mathcal{F}_{[0,60]}\ \mu_{\text{goal}} \quad \textit{(The drone must be at less than 5 m from the goal within 1 min)} \tag{17}$$

$$\mathcal{F}_{[0,60]}\ (\mathcal{G}_{[0,10]}\ \mu_{\text{goal}}) \quad \textit{(Same as (17) but the drone must stay around the goal during 10 s at least)} \tag{18}$$

$$\varphi_{\text{takeoff\_area}}\ \mathcal{U}_{[5,10]}\ \neg\mu_{\text{low\_height}} \quad \textit{(The drone must stay in its takeoff area until it reaches 10 m height wihtin the period 5 s – 10 s)} \tag{19}$$

with

$$\mu_{\text{goal}} = \sqrt{(x-20)^2 + (y+50)^2} \leq 5 \tag{20}$$

$$\varphi_{\text{takeoff\_area}} = |x| \leq 2 \wedge |y| \leq 2. \tag{21}$$

There are two ways to monitor a system based on an STL specification. Firstly, the *qualitative* monitoring returns a boolean corresponding to the satisfaction of a signal with an STL formula φ, like in equation (2). Secondly, the *quantitative* monitoring results in a real number ρ which is called the *robustness degree* [8]. The sign of ρ gives the qualitative information, and the absolute value is the robustness: the higher this number, the more robust. It can be viewed as the distance to the boundary between the set of signals satisfying φ and the one violating it.

This quantitative monitoring with its robustness is the mainly used approach to deal with aleatoric uncertainty of systems. However, if the signals are not known and only bounds are available, this approach becomes insufficient. Indeed, in the context of system verification based on STL, we should be able to deal with tubes, which is a robust representation for uncertain signals.

In this paper, a comprehensive overview of the state-of-the-art is proposed to show the need of a new formalism able to handle bounded uncertainties. The ability to consider sets of trajectories instead of a single one allows providing guaranteed results even when the system is too uncertain. The formalism proposed in this paper paves the way for using set-membership approaches when dealing with STL specifications. It can address every CPS whose trajectories can be bounded. There is no assumption of continuity. Theoretically, there is no limitation for this formalism. The limitations of the proposed approach are linked to the implementation. Indeed, nested temporal operators in the STL formula increase the complexity exponentially, and pessimism is added by the abstraction of sets.

The next section (2) of this paper proposes a literature review on STL in the context of handling uncertainties. Then, Section 3 presents how to deal with uncertainties inherent to tubes and introduces Boolean intervals, and an STL formalism adapted to set-membership approaches. The Section 4 is about the implementation of this newly introduced STL formalism and proposes a robotic application with the monitoring of a drone.

2 Literature review

Specifications of CPS have been widely studied. In particular, a survey can be found in [4] where qualitative and quantitative monitoring are reviewed, along with existing tools and applications in different domains. However, it does not deeply review the uncertainty issue. In this section, we propose to review more specifically how to deal with uncertainties, based on quantitative satisfaction, stochastic methods or set-membership approaches.

2.1 Quantitative satisfaction

Uncertainties have already been studied when considering the satisfaction of an STL formula (monitoring). The quantitative satisfaction can be used to handle some uncertainties, as in [6] where spatial and temporal uncertainties are considered, sometimes with the help of intervals. The space robustness is roughly the distance to border between satisfaction and violation, and the time robustness is the period around a time during which a formula is satisfied, *i.e.* the times θ^-, θ^+ representing the duration of the satisfaction of the formula at time t. Then the space-time robustness is the largest rectangle of height c around t. To compute the robustness degree on continuous time and space, the signal is discretized into a piecewise linear function. The complexity is linear with respect to the input signal size and the formula length. If we define the space robustness $\rho(\varphi, \mathbf{x}, t)$ for an STL formula φ and a trace $(\mathbf{x}, t)$, then we have the following Theorem, from [8]:

$$\rho(\varphi, \mathbf{x}, t) > 0 \implies (\mathbf{x}, t) \vDash \varphi \tag{22}$$

$$(\mathbf{x}, t) \vDash \varphi \text{ and } \|\mathbf{x} - \mathbf{x}'\|_\infty < \rho(\varphi, \mathbf{x}, t) \implies (\mathbf{x}', t) \vDash \varphi. \tag{23}$$

Equation (23) ensures us that if the considered signal satisfies the specifications with a robustness greater than the distance with the actual signal, then the real system satisfies the specifications.

In a few papers, intervals have been used to represent these uncertainties. For instance, in [30], an offline monitoring algorithm with intervals for finite time STL formulas is proposed. At each time step, an interval containing the estimated robustness value is computed. In [5], an online monitoring is proposed using an interval of all the possible quantitative satisfaction of a partial signal with unbounded future.

Finally, the robustness as described in the temporal logic literature corresponds to how far, in space or time, a signal is from violating or satisfying a property. In [6], a robustness sensitivity is also defined, so that we can compute the sensitivity of a formula to a given parameter. This robustness is a great tool to get a metric for evaluating signals, or do some specification mining (falsification problem to tune specifications and elicitation). However, this formalism corresponds to the robustness of one specification to one signal. It is not about the robustness to an uncertain hybrid system which presents uncountable many traces. In this paper, we are interested in describing temporal properties for CPS while considering noises and uncertain signals.

There are two main representations for uncertain systems: stochastic and set-membership approaches, whose literature is reviewed in the two next sections.

2.2 Stochastic STL

In this section, we focus on approaches to specify temporal properties over uncertain systems by using probabilities to represent errors due to noises and estimations. The main idea is to add probabilities on predicates, such that an STL formula can be considered as satisfied if the probability of violation is low enough.

For instance, in [27], a probabilistic STL is defined to allow violation of a specification with a given probability. A threshold ϵ on probability is defined to set the tolerance level in satisfaction of the properties. The paper deals with control design only, and monitoring is not addressed.

Another formalism to express probabilities on predicates can be found in [13]. The introduced temporal logic is called "chance-constrained temporal logic". However, the approach is restricted to deterministic and linear dynamical system. In [20], random STL is also proposed, with random predicates. Robot dynamics is modelled by a discrete-time, continuous space Markov decision process. Monitoring a stochastic system with STL is dealt within [10]. It gives a robustness measure (intervals) of stochastic trajectories. A monitor and motion planning are proposed using a sampling-based approach. The stochastic intervals of linear predicates are propagated on the STL.

To go further with the stochastic approach, some researchers have considered a risk-based STL which is able to consider the probability a property is violated weighted by the cost of such a violation. For instance, in [28], stochastic dynamical systems are studied. Constraints on atomic predicates are reformulated into deterministic affine constraints, using axiomatic risk theory. Other papers [18, 21] handle stochastic signals: the STL robustness risk is estimated for the value-at-risk. Some applications to data-driven approach are proposed.

2.3 Set-membership STL

Some researchers have worked on using set-membership approaches to deal with STL. Most papers propose to build robustness intervals, which hold every possible quantitative value for a given signal and an STL formula. It is the case in [30], where a monitoring algorithm with intervals for finite time STL formulas is proposed. It considers spatial deviation and time delay in the signal, but not the uncertainty in the STL predicates. On the other hand, the paper [3] uses intervals on predicates and on traces. It is based on a three-valued logical semantics, and relies on inclusion functions from interval arithmetic. The satisfaction of the STL formula is given by the resulting robustness interval I as follows: it is `true` if $I \subset [0, +\infty]$, `false` if $I \subset [-\infty, 0]$, and `undef` otherwise. This approach is also used in [5] in the context of online monitoring.

Finkbeiner paper's [9] introduces an offline monitoring algorithm, tailored for handling prevalent sensor uncertainty within the framework of STL. The approach

integrates error models, including bounded measurement inaccuracies, directly into the evaluation process of an STL formula. By systematically accounting for uncertainties, the algorithm provides robust verdicts regarding the satisfaction or violation of temporal properties. In [29], the authors introduce an online monitoring approach known as model predictive monitoring or model-based monitoring. This approach uses a dynamical model to estimate future states and provides qualitative evaluations for partially observed signals. This enables early certification or falsification of STL specifications.

Tubes Up to this point, presented set-membership approaches consider single signals as input, and then compute sets to consider uncertainties. However, in the context of real systems, signals cannot be exact, and we might want to consider a set of signals to be sure the actual trace is taken into account. Thus, if we are sure the actual trace is included in our set of signals, we can guarantee the final result. Such a set of signals can be represented by a *tube*, often used to represent intervals of trajectories [26]. In the remaining of this paper, a tube is noted $[x](t)$, and tube vectors are noted $[\mathbf{x}](t)$. A tube is represented by intervals of value at each time interval, as explained in Figure 1. The main drawback of this representation is that some additional traces might be taken into account, making the approach pessimistic. For instance, the orange signal in the figure is taken into account as it is included in the tube, even if it is physically impossible. So, a tube is not a signal with some added uncertainties, it is a set containing an infinite number of signals, but including every possible signal.

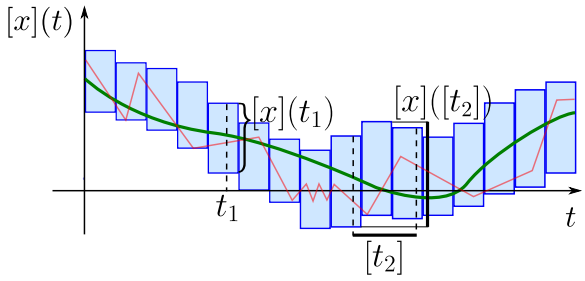

Figure 1: Representation of a set of traces by a tube. Two traces belonging to the tube are represented in green and in red. The red trace is considered whereas it might be physically impossible.

In the literature, a few papers consider sets of traces instead of single traces. Firstly, the Interval STL introduced in [3] considers uncertainties in predicates. Signals of intervals are handled, and quantitative satisfaction is returned as an interval.

In [25], Roehm *et al.* introduce a "Reachset Temporal Logic" (RTL) to address infinite set of traces. It requires to compute the reachable set. Then, the STL is sampled in time and converted into RTL. The tubes are cut in the time dimension

and STL formulas are interpreted as successive discrete states. When there exists a trace in the set of trace that does not satisfy the constraint, then the proposed monitor returns false (it is conservative).

At last, the approach proposed by Ishii *et al.* in [11] is closer to the first proposed STL monitor for single traces [22]: it searches times when properties become true or false, and then propagates the information in a bottom up manner to deduce the result for the whole STL formula. To deal with uncertain results, the algorithm can return `unknown` when there is an ambiguity on the satisfaction or the violation of every trace.

3 Monitoring tubes

When considering tubes instead of single traces, the monitor algorithm to check the satisfiability of an STL formula must take into account every possible trace within the tubes, which is infinite. The two papers introduced in last in the previous section are the only ones able to handle tubes, to our knowledge. These papers are reviewed more deeply in the following, along with an example.

3.1 Uncertainty on satisfaction of tubes

The main difficulty when considering the satisfaction of an STL formula for a tube (instead of a single trace) is that the tube may contain traces satisfying and traces not satisfying the formula at the same time. Then the result is neither completely false, nor completely true. To stay as conservative as possible, these undetermined cases can be considered as false. This is the choice of Roehm *et al.* [25], even if it introduces some pessimism, and prevents from knowing when every trace is false. The other option is to keep the undetermined case as a possible result, leading to a three-valued logic, as done by Ishii *et al.* [11].

However, in some cases it can be intricate to determine the true result. Indeed, there are an infinite number of potential traces in a tube, and there are also an infinite times to check for most STL formulas. This is why the state-of-the-art algorithms can fail to find the true answer and just consider it as an undetermined case even if it is not the case. This is illustrated in the following example.

Consider the tube $[x](t)$ represented in the Figure 2. We have a set point $\rho \in \mathbb{R}$ that we want our system to reach in the time interval $[t_1, t_2] \subset \mathbb{R}^+$. It means that we want the system to stay below ρ before t_1, and be equal or greater than ρ at least at one time during $[t_1, t_2]$. We first consider the STL atomic predicate:

$$\psi = ([x] \geq \rho). \tag{24}$$

In this example, during the period $[0, t_1]$, the predicate ψ is never satisfied for every possible trace included in the tube (every point in the blue area and before t_1 is below the ρ value). Similarly, after time t_2, the predicate ψ is always satisfied. On the other hand, during the period $[t_1, t_2]$, the satisfaction of the predicate ψ is ambiguous: some traces always satisfy ψ, others always violate it, and the remaining

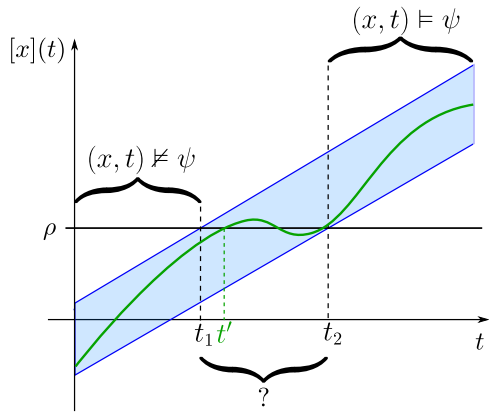

Figure 2: Simple tube representing a set of traces. A trace belonging to the tube is painted in green.

ones violate the predicate before satisfying it. This is typically an undetermined case.

Now, if we consider the full STL formula φ, corresponding to our problem:

$$\varphi = \neg\psi \, \mathcal{U}_{[t_1,t_2]}\psi, \tag{25}$$

it is well satisfied by any trace taken in the tube $[x](t)$ from time zero. Indeed, if we look at the definition of the *until* operator (see (11)), we have:

$$\begin{aligned}(x,0) \vDash \varphi &\overset{(25)}{\iff} (x,0) \vDash \neg\psi \, \mathcal{U}_{[t_1,t_2]} \, \psi \\ &\overset{(11)}{\iff} \exists t' \in [t_1,t_2] \begin{cases} (x,t') \vDash \psi \\ \wedge \, \forall t'' \in [0,t'] \, (x,t'') \nvDash \psi \end{cases}.\end{aligned}$$

It means that for every trace $(x,0)$ in the tube, we can find a time $t' \in [t_1,t_2]$ such that $x(t') \geq \rho$ and for every time t'' in $[0,t']$, we have $x(t'') < \rho$. Such a t' is given for the example of the green trajectory in the Figure 2. We can remark that the until formula does not specify anything on what happen after the time t'. Thus, we have the following proposition:

Proposition 1. *The STL formula $\varphi = \neg\psi \, \mathcal{U}_{[t_1,t_2]}\psi$, with $\psi = [x] \geq \rho$, is* `true` *for every signal in the blue tube of Figure 2.*

Proof. Let $(x,0)$ be a trace in the tube. Let t' be the smallest time within $[t_1,t_2]$ such that $x(t') \geq \rho$. We know such a t' exists because we have $x(t_2) \geq \rho$. Indeed, the interval of possible values for x at time t_2 is above ρ, according to Figure 2. Then we have $\forall t'' \in [0,t'], x(t'') < \rho$. □

However, the difficulty in finding an algorithm checking such a formula lies in the fact that the time t' may be different for every possible trace included in the tube. So, there is an infinite number of trace and time to check.

The two approaches proposed in the state-of-the-art both failed on this example to provide the expected `true` result.

Firstly, when applying the approach proposed by Ishii *et al.* [11] (Figure 3, left), we obtain the three-Boolean signal for the predicate ψ as already described. Then we apply the logical `not` operator ($\neg$), and finally we compute the three-Boolean signal for the complete formula from the two previously computed signals using a logical `and` ($\wedge$) operator and a time shifting. As we have an undefined result for the predicate ψ (depicted by the dark area in the figure), the ambiguity is propagated until the complete formula and is never removed. Thus, the final result is the undefined value for the tube at time zero. This result is not wrong, but it is not the expected one.

Secondly, the proposed approach of Roehm *et al.* [25] (Figure 3, right) uses only binary Boolean values. The time is discretized, and the predicate satisfaction is computed for every slice, resulting in a `false` value when there is a doubt (traces both validating and violating the formula in a same slice). Then the logical `not` operator ($\neg$) is easy to compute, and finally the `until` operator ($\mathcal{U}$) is transformed into a disjunction of cases depending only on the slices. At the end, the obtained result is `false` for this example, which is the conservative result of this approach.

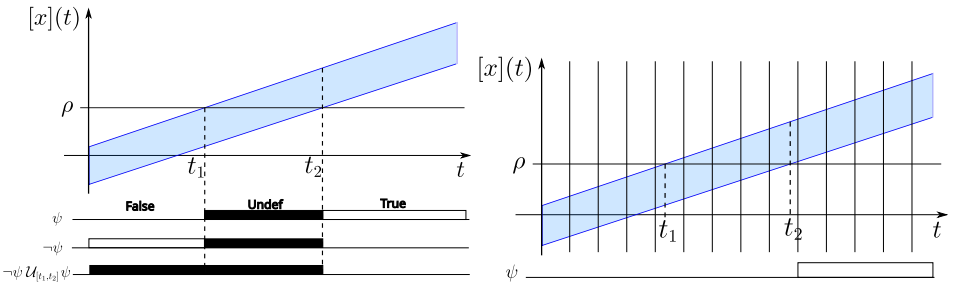

Figure 3: Illustration of the approaches from Ishii *et al.* [11] (left) and Roehm *et al.* [25] (right).

This simple example shows that the monitoring of a tube can easily become intricate, even with a small formula and a linear tube. In this example, and even more generally, in order to conclude the satisfaction rather than an unknown or false result, a solution is to use reachability analysis [16]. However, it is not in the scope of this paper, as reachability requires dealing directly with dynamical equations, and the computation cost can increase quickly when considering high-dimensional systems.

3.2 Boolean intervals

As stated above, the three-valued logic is more suitable when dealing with tubes to separate the cases where we are sure that every trace satisfies or every trace violates the formula, and when it is undetermined. In this paper, we propose to

use the formalism of Boolean intervals to represent such a three-valued logic. It is a well established formalism, based on interval analysis, with an arithmetic already known [12]. In addition, intervals are used as an abstraction for representing tubes, so the arithmetic stays the same.

Definition 1. *A Boolean interval is a subset of the set of Boolean* $\{true, false\}$*, i.e. an element of* $\mathbb{IB} = \{\emptyset, [0], [1], [0,1]\}$ *with:*

- $\emptyset$ *means impossible;*
- $[0]$ *means false* ($\bot$)*;*
- $[1]$ *means true* ($\top$)*;*
- $[0,1]$ *means undetermined (uncertain).*

The following Table 1 presents the different results with respect to the logical and temporal operators for the $[0,1]$ interval.

Table 1: Logical and temporal operators on the Boolean interval $[0,1]$.

$\neg[0,1] = [0,1]$	$[0,1] \vee 1 = 1$	$[0,1]\,\mathcal{U}_{[t_1,t_2]}\,0 = 0$
$[0,1] \wedge 1 = [0,1]$	$[0,1] \vee 0 = [0,1]$	$1\,\mathcal{U}_{[t_1,t_2]}\,[0,1] = [0,1]$
$[0,1] \wedge 0 = 0$	$[0,1]\,\mathcal{U}_{[t_1,t_2]}\,1 = [0,1]$	$0\,\mathcal{U}_{[t_1,t_2]}\,[0,1] = 0.$

What is important in this table is that some operations remove the uncertainty (*e.g.* $[0,1] \wedge 0 = 0$). So, depending on the form of the STL formula, the final result can be more or less pessimistic. Indeed, in practice, we hope to not obtain this uncertain case, as it does not contain actual information.

3.3 Set-Membership STL Formalism

In order to properly deal with tubes using STL specifications, we need to establish a specific formalism based on sets. Thus, we propose to slightly change the STL syntax to directly handle tubes and use Boolean intervals. As tubes are based on sets, the natural atomic predicates are the set operators.

In this way, we define the set predicate $\mu = \mathcal{S} \mid \mathcal{I}$ for boxes $[\mathbf{x}], [\mathbf{A}] \in \mathbb{IR}^n$ with:

- $\mathcal{S}$ the subset test ($[\mathbf{x}] \overset{?}{\subset} [\mathbf{A}]$);
- $\mathcal{I}$ the empty intersection test ($[\mathbf{x}] \cap [\mathbf{A}] \overset{?}{=} \emptyset$).

Finally, the new set-membership STL syntax can be written as:

$$\varphi := \beta \mid \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1\,\mathcal{U}_{[t_1,t_2]}\varphi_2. \tag{26}$$

If we abstract sets by using level sets, this new syntax is not less general than previously. For example, we can represent a predicate $\mu = x > 0$ by $\mathcal{X}^\mu = \{x \in \mathbb{R} \mid x > 0\}$. Then, we have, for $t \in \mathbb{R}$:

$$([\mathbf{x}], t) \vDash \mu = \begin{cases} 1 & \text{if } [\mathbf{x}](t) \subset \mathcal{X}^\mu \quad (\mathcal{S}); \\ 0 & \text{if } [\mathbf{x}](t) \cap \mathcal{X}^\mu = \emptyset \quad (\mathcal{I}); \\ [0,1] & \text{otherwise.} \end{cases} \tag{27}$$

The differences introduced in this STL are only on Boolean values and atomic predicates. Thus, there are no changes on the temporal operators.

4 Application

Using the set-membership STL, it becomes possible to implement a monitor for tubes and use it on real systems. This is presented in this section.

4.1 Implementation

Tubes are used to represent sets of traces. The main advantage is to consider bounded uncertainties in a guaranteed way. Tubes are well-suited for addressing dynamical systems, and several set-based libraries already exist, as Vnode [23], DynIbex [2] or CAPD [14].

DynIbex offers a set of validated numerical integration methods based on Runge-Kutta schemes to solve initial value problem. In the library, there are already available objects and functions required for the set-membership STL. We build a subset of the set-membership STL syntax. Indeed, in this syntax, nested temporal operators are not allowed, contrary to (1). For instance, an STL formula like the following one[1] is not allowed here as there is a *finally* composed with an *until* operator:

$$(x \leq 10)\, \mathcal{U}_{[0,20]}\, (\mathcal{F}_{[0,5]}\, (x \geq 15)).$$

The full syntax is a work in progress. The definition is spread on several lines, with different variables to prevent nested temporal operators:

$$\beta := \emptyset \mid [0] \mid [1] \mid [0,1] \tag{28}$$
$$\mu := \mathcal{S} \mid \mathcal{I} \mid \beta \tag{29}$$
$$\pi := \neg\mu \mid \mu \tag{30}$$
$$\sigma := \mathcal{G}_{[t_1,t_2]}\, \pi \mid \mathcal{F}_{[t_1,t_2]}\, \pi \mid \pi_1\, \mathcal{U}_{[t_1,t_2]}\, \pi_2 \mid \pi \tag{31}$$
$$\theta := \sigma \wedge \theta \mid \sigma \vee \theta \mid \sigma \tag{32}$$

with $\mathcal{S}, \mathcal{I}$ the subset and empty intersection tests, respectively.

[1] This formula states that x must reach 10 within 20 s, and as soon as it happens x must become greater than 15 in less than 5 s.

All these operators have been implemented in DynIbex. For the sake of efficiency, some compositions of operators are directly implemented as one function. The following Table 2 gives the corresponding functions for each operator, based on the $\mathcal{S}$ predicate (it is very similar for the $\mathcal{I}$ predicate). It corresponds to the terms σ in the syntax (Equation 31).

Table 2: Corresponding DynIbex functions for each available set-membership STL. I is an interval of time.

Set-membership STL operator	**DynIbex function**
$\mathcal{S}$	`subset`
$\mathcal{G}_I\,\mathcal{S}$	`globally_subset`
$\mathcal{G}_I\,\neg\mathcal{S}$	`globally_not_subset`
$\mathcal{F}_I\,\mathcal{S}$	`finally_subset`
$\mathcal{F}_I\,\neg\mathcal{S}$	`finally_not_subset`
$\mathcal{S}_1\,\mathcal{U}_I\,\mathcal{S}_2$	`until`

4.2 Example with a simple tube

If we take again the example used in Section 3 and implement it in DynIbex, we obtain the same result as with the algorithm proposed by Ishii *et al.* in [11], *i.e.* `undefined` result ($[0, 1]$ Boolean interval). The simulation time (to generate the tube) takes less than 7 ms in average, and the verification of the STL formula takes less than 0.02 ms on a classical laptop (Intel i5-1335U, up to 4.6 GHz).

Figure 4 illustrates the obtained result. Blue boxes have been estimated in a guaranteed way using DynIbex: the real tube (represented in transparency as a reference) is well included inside the boxes.

Discussion This example enables comparing the two state-of-the-art methods able to deal with tubes (Ishii *et al.* [11] and Roehm *et al.* [25], see Section 3.1) and our approach. We obtain the same uncertain result as Ishii *et al.*, which is more accurate than the conservative false result of Roehm *et al.* The main advantage of our approach is that we evaluate only the satisfaction of the STL formula at the required specific time, which allows to be faster than Ishii *et al.* when the formula does not require to study all the signal duration. Indeed, in their work, Ishii *et al.* always compute the whole satisfaction signal for every sub-formula. The following Table 3 recaps the differences between presented approaches.

In order to obtain the expected true result, we may use tools from reachability analysis. However, it is not in the scope of this paper, as it requires the knowledge of the system dynamical equations, and often asks heavy computations. In our context, we do not need the system model since the tube may have been obtained only with measurements.

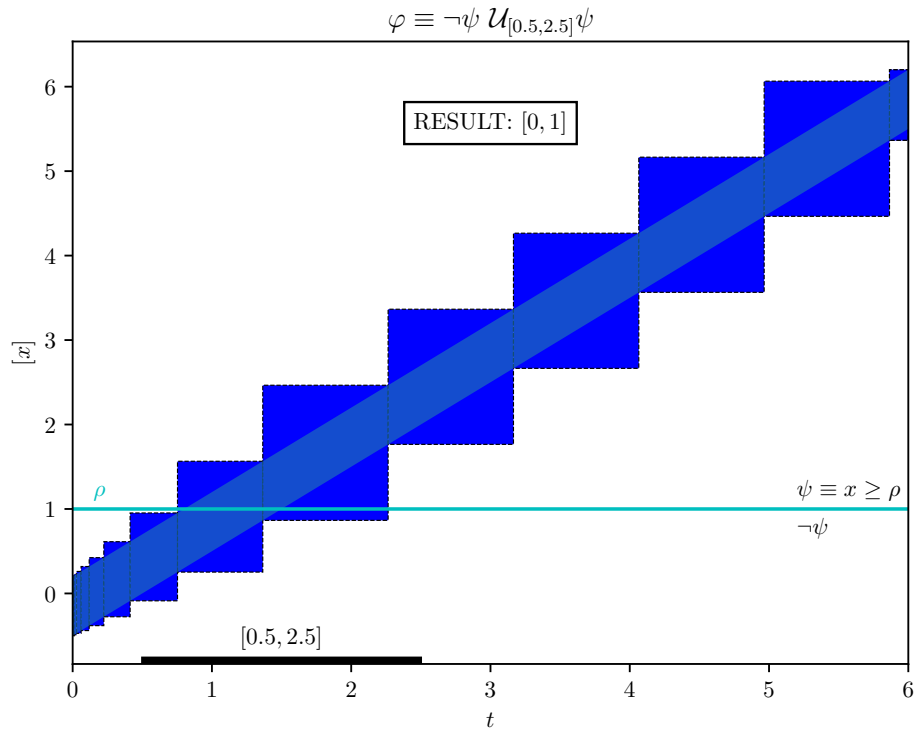

Figure 4: Result of the simulation of the simple tube example using DynIbex. The cyan line represents the ρ parameter. The result is "uncertain" ($[0, 1]$).

Table 3: Comparison with the state-of-the-art approaches on this example

Method	**Result**	**Running time for computing satisfaction**
Roehm *et al.* [25]	False	not implemented
Ishii *et al.* [11]	Undef	12 ms
Ours	[0,1]	0.02 ms

4.3 A test-case: monitoring of a drone

In this section is presented a test-case example where a drone (DJI Tello) is monitored with respect to an STL formula. This is a real-time monitoring: while the drone is flying, its state is estimated for the next 4 seconds and the STL specifications are verified on this estimation. Thus, if the satisfaction is not guaranteed, the drone can be slowed down or even stopped soon enough. In this example, tubes are used to represent uncertainties, mostly due to model approximation. Thus, STL

specifications, which are necessary to guarantee the safety of a critical system like a drone, have to be verified on the tubes. Finally, the generation of the tubes and the verification of the STL must be done in real-time, which requires short runtimes (with respect to the drone velocity).

Setup and modelling The drone state is denoted by y and its control is u. The drone is commanded in velocity along x, y and z axis. The control is based on a switched model, which consists on motion primitives that guide one robot movement, ensuring transitions between trajectories while adhering to its dynamical constraints [19]. Trajectories in the experiment are composed of 8 possible modes, and modes switching are only possible at every τ seconds.

We denote $y_{\text{ref}}(t)$ a reference trajectory and $u_{\text{ref}}(t)$ a reference control. They represent the precomputed movement and control of each motion primitive.

To track the given trajectory, the drone velocity control is:

$$u(t) = K_{\text{p}}(y_{\text{ref}}(t) - y(t)) + u_{\text{ref}}(t), \tag{33}$$

with K_{p} a P corrector.

Even though the behavior of a quadcopter can be represented by a highly nonlinear dynamical model, we consider the drone to be holonomic and use a simplified dynamical model for model-based monitoring. The model is a first-order system controlled by speed along each axis. This model is simple compared to dynamical models that account for the rotation of the quadrotor [1].

$$\dot{y} = u. \tag{34}$$

The main advantage of using a simple model is reducing computation time and complexity. To account for the simplifications in the dynamical models, we will introduce interval parameters as in [17].

$$[\dot{y}] = [\alpha_1] \cdot [u] + [\beta_1] \tag{35}$$

The presence of $[\alpha_1]$:

- Adjusts the assumption of immediate command responses by incorporating the drone speed profile,
- Accounts for uncertainties related to forces that depend on velocity,
- Considers uncertainties associated with the tuning of the corrector to track the trajectory.

The presence of $[\beta_1]$:

- Corrects uncertainties arising from sudden perturbations,
- Accounts for ambient noise and bias in the control inputs, including noise induced by the drone itself [17].

Furthermore, when reading sensor values used in the initial state of the dynamic equation, the initial state $\mathbf{y}_0$ becomes $[\mathbf{y}_0]$ as we introduce bounded uncertainties in the measurement or state estimation from filtering.

Monitoring The monitoring process aims to recognize if faults occur. In our case, it is done by verifying some constraints. Consider the following specification: "the drone must avoid obstacles within a given time horizon $h = N \times \tau$ and remain inside a designated area". The corresponding STL formula is given by:

$$\varphi_{\text{flag}} = \mathcal{G}_{[0,h]} \neg \mathcal{S}_{\text{collision}} \wedge \mathcal{G}_{[0,h]} \mathcal{S}_{\text{environment}}. \tag{36}$$

To monitor the system, the dynamical model is used to predict the future states within the time horizon [7]. The predicted state $[\tilde{\mathbf{y}}]$ is evaluated using a set-based simulation approach to reliably determine if a fault occurs. For example, it is important to ensure that the system stays within safe operating limits. In our setup, we have for a time $t \in \mathbb{R}$:

$$[\tilde{\mathbf{y}}](t) \subset [\mathbf{A}] \implies ([\tilde{\mathbf{y}}], t) \vDash \mathcal{S}_{\text{environment}}, \tag{37}$$

with $[\mathbf{A}]$ the bounding box of the authorized environment. If the predicted state violates such a constraint, the monitor flags a potential fault.

Finally, the satisfaction function C, which now represents the monitor flag for the STL formula φ_{flag}, is defined for an n-dimensional system as:

$$\begin{aligned} C : (\mathbb{R}^+ \to \mathbb{IR}^n) &\to \mathbb{IB} \\ [\tilde{\mathbf{y}}](t) &\mapsto \begin{cases} 1 & \text{if } [\tilde{\mathbf{y}}](t) \text{ satisfies the formula,} \\ 0 & \text{if } [\tilde{\mathbf{y}}](t) \text{ violates the formula,} \\ [0,1] & \text{otherwise (uncertain result).} \end{cases} \end{aligned} \tag{38}$$

Results The Figure 5 presents the results of the experimentation. The monitoring process is operated in real time at a given rate of 1 Hz. Thus, each second, the next 4 seconds of time horizon are estimated and the obtained tube is used to check the satisfaction of the STL formula. The "Monitor Go_flag" is the result of the satisfaction function. It has been implemented in the ROS[2] framework. The drone is localized using a motion capture system (OptiTrack); the actual trajectory is represented by green dots. The authorized environment is the green frame. Obstacles are in gray. The reference trajectory is drawn in thick red strokes (following the possible modes). Finally, the estimated trajectory is the tube starting at the last known position (end of the green dots) with a small blue box filled in transparent red. This box has been integrated using the dynamic model of the drone, along with the uncertainties, leading to bigger boxes.

The performance benchmark was conducted over the first four seconds of the trajectory ($h = 4$ s). We measured the total execution time on a laptop (Intel i5-8265u, 8gb RAM). It took 0.211 s, making it suitable for online monitoring.

[2]ROS: Robot Operating System (see https://www.ros.org/).

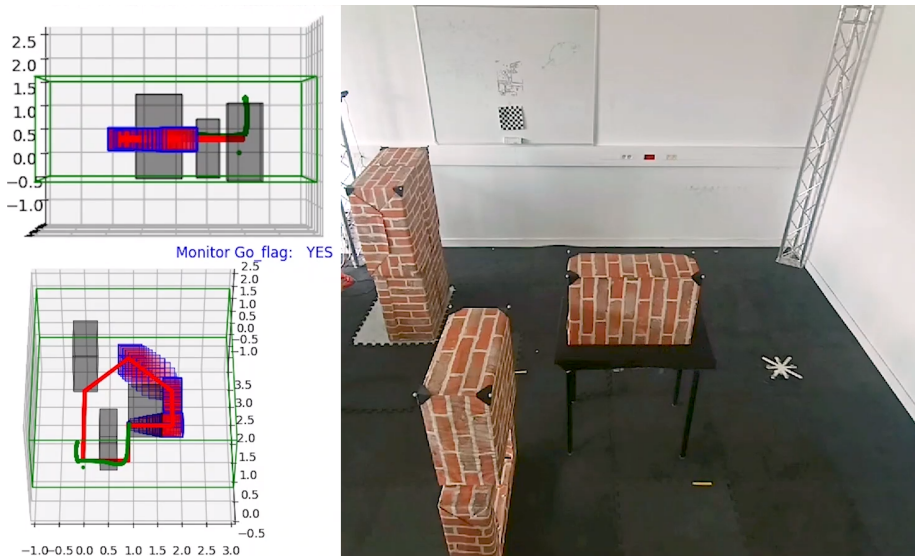

Figure 5: Experimentation results. Left: 2 different views of the computed tube on a given time horizon (red boxes); Right: actual drone flying.

Discussion A simple but realistic STL-based monitor has been tested on a real system for illustration purpose. This experimentation proves the interest of the approach: i) the use of a tube to take into account uncertainties and model simplification provides a guaranteed result; ii) the proposed set-based approach for STL verification is sound and sufficiently fast for real-time application. This example can be generalized, and more complex STL formulas can be tested because no limitations have been imposed.

5 Conclusion

In this paper, literature about dealing with uncertainties has been reviewed in the context of STL. We focused on how to handle tubes instead of single traces, as it allows considering sets of traces from real systems with a strong robustness to uncertainties. Only a few papers have already addressed this issue, and there is still room for improvements, as shown with a simple example whose true result is difficult to obtain with existing approaches. Then, a formalism directly applied on sets is proposed, along with Boolean intervals to rigorously deal with ineluctable uncertain results. This formalism is used as a base for an implementation within the DynIbex tool, in order to have a direct link between STL and a guaranteed integration tool. It is also the base for future works, including dealing with nested formulas and links with reachability analysis.

Acknowledgments

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions.

References

[1] Abadi, A., El Amraoui, A., Mekki, H., and Ramdani, N. Guaranteed trajectory tracking control based on interval observer for quadrotors. *International Journal of Control*, 93(11):2743–2759, 2020. DOI: 10.1080/00207179.2019.1610903.

[2] Alexandre Dit Sandretto, J. and Chapoutot, A. Validated explicit and implicit Runge-Kutta methods. *Reliable Computing electronic edition*, 22, 2016. URL: https://hal.science/hal-01243053.

[3] Baird, L., Harapanahalli, A., and Coogan, S. Interval Signal Temporal Logic From Natural Inclusion Functions. *IEEE Control Systems Letters*, 7:3555–3560, 2023. DOI: 10.1109/LCSYS.2023.3337744.

[4] Bartocci, E., Deshmukh, J., Donzé, A., Fainekos, G., Maler, O., Ničković, D., and Sankaranarayanan, S. Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications. In Bartocci, E. and Falcone, Y., editors, *Lectures on Runtime Verification*, Volume 10457, pages 135–175. Springer International Publishing, Cham, 2018. DOI: 10.1007/978-3-319-75632-5_5.

[5] Deshmukh, J. V., Donzé, A., Ghosh, S., Jin, X., Juniwal, G., and Seshia, S. A. Robust online monitoring of signal temporal logic. *Formal Methods in System Design*, 51(1):5–30, 2017. DOI: 10.1007/s10703-017-0286-7.

[6] Donzé, A. and Maler, O. Robust Satisfaction of Temporal Logic over Real-Valued Signals. In Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J. M., Mattern, F., Mitchell, J. C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M. Y., Weikum, G., Chatterjee, K., and Henzinger, T. A., editors, *Formal Modeling and Analysis of Timed Systems*, Volume 6246, pages 92–106. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. DOI: 10.1007/978-3-642-15297-9_9.

[7] Dvorak, D. and Kuipers, B. Model-based monitoring of dynamic systems. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Volume 2 of *IJCAI'89*, pages 1238–1243, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc. URL: https://www.ijcai.org/Proceedings/89-2/Papers/062.pdf.

[8] Fainekos, G. E. and Pappas, G. J. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009. DOI: 10.1016/j.tcs.2009.06.021.

[9] Finkbeiner, B., Fränzle, M., Kohn, F., and Kröger, P. A Truly Robust Signal Temporal Logic: Monitoring Safety Properties of Interacting Cyber-Physical Systems under Uncertain Observation. *Algorithms*, 15(4):126, 2022. DOI: 10.3390/a15040126.

[10] Ilyes, R. B., Ho, Q. H., and Lahijanian, M. Stochastic robustness interval for motion planning with Signal Temporal Logic. In *Proceedings of the 2023 IEEE International Conference on Robotics and Automation*, pages 5716–5722, 2023. DOI: 10.1109/ICRA48891.2023.10161409.

[11] Ishii, D., Yonezaki, N., and Goldsztejn, A. Monitoring Temporal Properties using Interval Analysis. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E99.A(2):442–453, 2016. DOI: 10.1587/transfun.E99.A.442.

[12] Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. Applied Interval Analysis. In Jaulin, L., Kieffer, M., Didrit, O., and Walter, E., editors, *Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics*, pages 11–43. Springer, London, 2001. DOI: 10.1007/978-1-4471-0249-6_2.

[13] Jha, S., Raman, V., Sadigh, D., and Seshia, S. A. Safe autonomy under perception uncertainty using Chance-Constrained Temporal Logic. *Journal of Automated Reasoning*, 60(1):43–62, 2018. DOI: 10.1007/s10817-017-9413-9.

[14] Kapela, T., Mrozek, M., Wilczak, D., and Zgliczyński, P. CAPD::DynSys: A flexible C++ toolbox for rigorous numerical analysis of dynamical systems. *Communications in Nonlinear Science and Numerical Simulation*, 101:105578, 2021. DOI: 10.1016/j.cnsns.2020.105578.

[15] Koymans, R. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990. DOI: 10.1007/BF01995674.

[16] Kurzhanski, A. B. and Varaiya, P. *Dynamics and Control of Trajectory Tubes: Theory and Computation*, Volume 85 of *Systems & Control: Foundations & Applications*. Springer International Publishing, Cham, 2014. DOI: 10.1007/978-3-319-10277-1.

[17] Largent, S. and Alexandre Dit Sandretto, J. Trajectory monitoring for a drone using interval analysis. In *Proceedings of the Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, Kyoto, Japan, 2023. Philippe Martinet. URL: https://hal.science/hal-04194830.

[18] Lars, L., Lejun, J., Nikolai, M., and J., P. G. Risk of stochastic systems for Temporal Logic Specifications. *ACM Transactions on Embedded Computing Systems*, 2023. DOI: 10.1145/3580490.

[19] Le Coënt, A., Alexandre dit Sandretto, J., Chapoutot, A., and Fribourg, L. An improved algorithm for the control synthesis of nonlinear sampled switched systems. *Formal Methods in System Design*, 53(3):363–383, 2018. DOI: 10.1007/s10703-017-0305-8.

[20] Lee, K. M. B., Yoo, C., and Fitch, R. Signal Temporal Logic synthesis as probabilistic inference. In *Proceedings of the 2021 IEEE International Conference on Robotics and Automation*, pages 5483–5489, 2021. DOI: 10.1109/ICRA48506.2021.9560929.

[21] Lindemann, L., Matni, N., and Pappas, G. J. STL robustness risk over Discrete-Time Stochastic processes. In *Proceedings of the 2021 60th IEEE Conference on Decision and Control*, pages 1329–1335, 2021. DOI: 10.1109/CDC45484.2021.9683305, ISSN: 2576-2370.

[22] Maler, O. and Nickovic, D. Monitoring temporal properties of continuous signals. In Lakhnech, Y. and Yovine, S., editors, *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Volume 3253 of *Lecture Notes in Computer Science*, pages 152–166, Berlin, Heidelberg, 2004. Springer. DOI: 10.1007/978-3-540-30206-3_12.

[23] Nedialkov, N. S. and Jackson, K. R. ODE software that computes guaranteed bounds on the solution. In Langtangen, H. P., Bruaset, A. M., and Quak, E., editors, *Advances in Software Tools for Scientific Computing*, pages 197–224, Berlin, Heidelberg, 2000. Springer. DOI: 10.1007/978-3-642-57172-5_6.

[24] Pnueli, A. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE, 1977. DOI: 10.1109/SFCS.1977.32.

[25] Roehm, H., Oehlerking, J., Heinz, T., and Althoff, M. STL model checking of continuous and hybrid systems. In Artho, C., Legay, A., and Peled, D., editors, *Automated Technology for Verification and Analysis*, Volume 9938 of *Lecture Notes in Computer Science*, pages 412–427, Cham, 2016. Springer International Publishing. DOI: 10.1007/978-3-319-46520-3_26.

[26] Rohou, S., Jaulin, L., Mihaylova, L., Le Bars, F., and Veres, S. M. Guaranteed computation of robot trajectories. *Robotics and Autonomous Systems*, 93:76–84, 2017. DOI: 10.1016/j.robot.2017.03.020.

[27] Sadigh, D. and Kapoor, A. Safe control under uncertainty with Probabilistic Signal Temporal Logic. In *Robotics: Science and Systems XII*. Robotics: Science and Systems Foundation, 2016. DOI: 10.15607/RSS.2016.XII.017.

[28] Safaoui, S., Lindemann, L., Dimarogonas, D. V., Shames, I., and Summers, T. H. Control design for risk-based Signal Temporal Logic specifications. *IEEE Control Systems Letters*, 4(4):1000–1005, 2020. DOI: 10.1109/LCSYS.2020.2998543.

[29] Yu, X., Dong, W., Li, S., and Yin, X. Model predictive monitoring of dynamical systems for signal temporal logic specifications. *Automatica*, 160:111445, 2024. DOI: 10.1016/j.automatica.2023.111445.

[30] Zhong, B., Jordan, C., and Provost, J. Extending Signal Temporal Logic with quantitative semantics by intervals for robust monitoring of cyber-physical systems. *ACM Transactions on Cyber-Physical Systems*, 5(2):1–25, 2021. DOI: 10.1145/3377868.