Uncertainty Removal in Verification of Nonlinear Systems against Signal Temporal Logic via Incremental Reachability Analysis

Antoine Besset*, Joris Tillet* and Julien Alexandre dit Sandretto*

Abstract— **A framework is presented for the verification of Signal Temporal Logic (STL) specifications over continuous-time nonlinear systems under uncertainty. Based on reachability analysis, the proposed method addresses indeterminate satisfaction caused by over-approximated reachable sets or incomplete simulations. STL semantics is extended via Boolean interval arithmetic, enabling the decomposition of satisfaction signals into unitary components with traceable uncertainty markers. These are propagated through the satisfaction tree, supporting precise identification even in nested formulas. To improve efficiency, only the reachable sets contributing to uncertainty are refined, identified through the associated markers. The framework allows online or offline monitoring to adapt to incremental system evolution while avoiding unnecessary recomputation. A case study on a nonlinear oscillator demonstrates a significant reduction in satisfaction ambiguity, highlighting the effectiveness of the approach.**

I. Introduction

Ensuring the reliability and safety of dynamical and controlled systems is critical across domains such as autonomous systems, cyber-physical systems, and industrial automation. Despite careful design, uncertainties like model inaccuracies or external disturbances can cause deviations from expected behavior. Model-based verification methods support system analysis by simulating state evolution and identifying discrepancies between intended and actual behavior [1], [2].

Signal Temporal Logic (STL) [3] provides a formal framework for specifying both timing constraints and state-dependent properties, making it well suited for verifying dynamical behaviors. For systems with modeling and disturbance uncertainties, stochastic methods are commonly used, extending single-trajectory temporal logic verification to a probabilistic setting as in [4]. However, these methods may miss rare or worst-case behaviors, thus offering limited safety guarantees. To overcome this limitation, we adopt reachability analysis, which systematically over-approximates all possible system trajectories under bounded disturbances [5]–[7]. This approach ensures that no behavior is overlooked, enabling sound verification of STL specifications across both temporal and state domains.

In reachability analysis frameworks such as [5], [8], adaptive step size selection is employed to satisfy precision requirements. Although smaller step sizes may increase computational cost and introduce conservatism, they are sometimes essential to reduce the uncertainty in STL satisfaction. Nonetheless, given that the objective of this work is to verify temporal properties through STL specifications, any uncertainty in the satisfaction must stem from the intrinsic system dynamics, rather than being an artifact of the numerical step size selection.

Related Work. Various approaches have been proposed for verifying STL formulas against reachable sets [9]–[13] or interval of signals [14]. Some proposed a new formalism known as Reachset Temporal Logic (RTL) providing a sound and complete transformation from an STL formula to an RTL formula [10] and an automatic refinement strategy in [15]. However, this approach constrains the time step and time range of the formula. In Ishii et al. [12], the concept of consistent time intervals is introduced, where the reachable set allows for a definitive determination of whether an STL formula is satisfied. These approaches fall under offline monitoring [3], [10], [12], as it requires the prior computation of the system evolution before verifying the formula. Conversely, Lercher et al. [9] propose an online monitoring approach, which incrementally computes only the necessary reachable sets for verification, thereby improving efficiency in long horizon system. To address uncertainties in satisfaction, three-valued semantics for temporal logic were introduced in [11], [12], [16]–[18], while four-valued satisfaction was proposed in [9] to differentiate between incomplete observations and intrinsic system ambiguity. To mitigate the uncertainty caused by large integration step sizes, the approach in [9] employs smaller step sizes, potentially requiring multiple reachable set computations before evaluating formula satisfaction.

In this paper, we build upon these concept by introducing a method based on Boolean interval arithmetic [11], [19] for evaluating STL formulas over reachable sets. Our approach supports the verification of nested STL formulas by using satisfaction signal, not always supported in prior works [11], [13]. We decompose the satisfaction signal into certain and uncertain unitary Boolean signals, enabling uncertainty tracking within the satisfaction tree. Our approach identifies whether an uncertainty arises from specific reachable sets or from incomplete simulation, and selectively contracts or refines only the relevant sets. This method is orthogonal to approaches where verification is static on the generated sequence of reachable sets [10]–[12], [16]. We assume access to a model of the system dynamics, allowing us to compute and contract reachable sets over specified time intervals. Our automatic refinement approach also addresses the over-approximation of reachable sets, similar to [15], but remains fully within the STL formalism and extends to nonlinear systems. In contrast to [15], our method is designed for online monitoring under incomplete time horizons. By refining reachable sets through localized backtracking and parsimo-

*U2IS, ENSTA, Institut Polytechnique de Paris, Palaiseau, France

nious forward computation, it enables efficient verification, in line with the incremental strategy of [9].

Our main contribution is a dynamic STL verification framework that reduces uncertainty by selectively refining only the reachable sets responsible for indeterminate satisfaction. The approach combines offline and online monitoring to improve precision while avoiding unnecessary computations.

Initially, Section II introduces the key concepts used throughout the paper. In Section III, we extend STL satisfaction on reachable sets using Boolean interval arithmetic and present a decomposition of the satisfaction signal for uncertainty tracking. Section IV details the tracking of uncertainty to the predicate level within the satisfaction tree, enabling identification of its origin from specific reachable sets. In Section V, we redefine time intervals and contract the relevant reachable sets. We finally present the proposed algorithm and an experiment illustrating the efficiency of our approach.

II. Preliminaries

A. Reachability analysis

Let us consider a continuous dynamical system modeled by a differential equation of the form:

$$\dot{y}(t) = f(y(t), w(t)), \quad y(t) \in \mathbb{R}^n, \ w(t) \in \mathcal{W}, \tag{1}$$

where $y(t)$ denotes the system state, $w(t)$ is a bounded external input, $\mathcal{W} \subseteq \mathbb{R}^p$ is a compact set. A function f is a nonlinear function, such that for any initial state $y_0 \in \mathbb{R}$ and any measurable input signal $w : \mathbb{R}^+ \to W$, the system admits a unique trajectory in $\mathbb{R}^n$ denoted by $\xi(\cdot, y_0, w)$.

Definition 1: The *reachable set* at time $t \in \mathbb{R}^+$ from a set of initial states $\mathcal{Y}_0 \subseteq \mathbb{R}^n$ is defined as [5], [6]:

$$\text{Reach}_t(\mathcal{Y}_0, \mathcal{W}) = \{\xi(t, y_0, w) \mid y_0 \in \mathcal{Y}_0, \ w(s) \in \mathcal{W}, \ \forall s \in [0, t]\} \tag{2}$$

Definition 2: The *reachable tube* over a time interval $[0, T] \subseteq \mathbb{R}^+$ is given by [5], [6]:

$$\text{Reach}_{[0,T]}(\mathcal{Y}_0, \mathcal{W}) = \bigcup_{s \in [0,T]} \text{Reach}_s(\mathcal{Y}_0, \mathcal{W}). \tag{3}$$

In the process of solving an Initial Value Problem for a differential equation using validated numerical integration methods [5], an enclosure $[\tilde{y}_j]$ of the solution is computed for each time interval $[t_j, t_{j+1}]$, where $t_{j+1} = t_j + h_j$ and h_j is the integration step size. This enclosure satisfies:

$$\text{Reach}_{[t_j, t_{j+1}]} \subseteq [\tilde{y}_j], \ \text{with} \ \bigcup_{j=0}^{N-1} [t_j, t_{j+1}] = [t_0, T], \tag{4}$$

with t_0 the start time of the simulation and T the end time.

The set of trajectories over the interval $[t_0, T]$ captures all possible executions of the system. This continuous-time representation, referred to as a *tube* and denoted $[\tilde{y}](t)$ for $t \in [t_0, T]$, is essential to preserve system behavior that may be lost through sparse time sampling. This sequence of reachable sets is the object of our analysis for STL formula satisfaction.

B. Signal temporal logic

STL is an extension of Linear Temporal Logic designed to specify properties of continuous-time systems [2], [3], for robotics and control applications. STL formulas enable the expression of temporal properties in terms of time bounds, combining logical operators with bounded temporal operators. The syntax of STL is defined recursively as follows:

$$\phi := \mu \mid T \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \, U_{[a,b]} \, \phi_2, \tag{5}$$

where ϕ is an STL formula, T denotes the tautology, $\neg$ the negation, and μ is an atomic predicate specifying signal constraints (e.g., $x > 3$) [3].

The semantics is the following:

$$\begin{aligned} (s,t) \models \mu &\iff \pi_\mu(s)[t] = T \\ (s,t) \models \neg\phi &\iff (s,t) \not\models \phi \\ (s,t) \models \phi_1 \vee \phi_2 &\iff (s,t) \models \phi_1 \text{ or } (s,t) \models \phi_2 \\ (s,t) \models \phi_1 U_{[a,b]} \phi_2 &\iff \exists t' \in [t+a, t+b], \ (s,t') \models \phi_2 \\ &\qquad \text{and } \forall t'' \in [t, t'], \ (s, t'') \models \phi_1 \\ F_{[a,b]}\phi = TU_{[a,b]}\phi, &\quad G_{[a,b]}\phi = \neg(F_{[a,b]}\neg\phi), \end{aligned} \tag{6}$$

where (s,t) represents the trace of a signal s at time t, and $\pi_\mu(s)$ is a function that evaluates a predicate on the signal. $U_{[a,b]}$ is the "until" operator bounded by the time interval $[a,b]$. The semantic is interpreted as: "if a formula ϕ_2 is satisfied at a time $t' \in [t+a, t+b]$, then another formula ϕ_1 must be true from the evaluating time t to the time t' ". The Finally operator $F_{[a,b]}$ expresses the existence of a satisfaction within a time interval. The Globally operator $G_{[a,b]}$ specifies that the subformula must hold during the entire time interval. Other logical operators such as $\wedge$ and $\implies$ are defined using standard logical equivalences:

$$\phi_1 \wedge \phi_2 \equiv \neg(\neg\phi_1 \vee \neg\phi_2), \quad \phi_1 \implies \phi_2 \equiv \neg\phi_1 \vee \phi_2. \tag{7}$$

C. Satisfaction signals

To unify predicate evaluation and temporal reasoning, the concept of satisfaction signal $\mathcal{S}_\phi(t)$ is defined in Eq. (8) from [3], [9]. This representation tracks the time-dependent satisfaction of each nested subformula, allowing Boolean operations such as $\vee$, $\wedge$ and $\neg$ to be applied directly to these signals.

Definition 3 (satisfaction signal [3], [9]):

$$\mathcal{S}_\phi : \mathbb{R}^+ \to \mathbb{B}, \ t \mapsto \begin{cases} 1 & \text{if } (s,t) \models \phi_i \\ 0 & \text{if } (s,t) \not\models \phi_i \end{cases}. \tag{8}$$

As defined in [3], the satisfaction signal of the Until operator is constructed using *unitary* Boolean signals. A satisfaction signal $\mathcal{S}^*_\phi$ is said to be **unitary** if it is true only over a single, continuous and positive time interval I_ϕ, denoted unitary time interval in this paper. This is a key concept as it allows for the verification of any STL formula with temporal operator.

The satisfaction signal of any formula can be given by a disjunction of N unitary signals [3]. The decomposition in unitary signals is illustrated in Fig. 1. $\mathcal{S}_w(\cdot)$ is the satisfaction signal of the predicate w. It is decomposed in two unitary Boolean signals: $\mathcal{S}^*_{w,1}(\cdot)$ and $\mathcal{S}^*_{w,2}(\cdot)$

Definition 4 (unitary decomposition [3]):

$$\mathcal{S}_\phi(t) := \bigvee_{j=0}^{N} \mathcal{S}^*_{\phi,j}(t),\ t \in \mathbb{R}^+. \tag{9}$$

For simplicity in notation, $\mathcal{S}_\phi(t), \forall t \in I$ corresponds to $\mathcal{S}_\phi(I)$.

Definition 5: Following [3], let $I = [m, n)$ and $[a, b]$ be intervals in $\mathbb{T} = \mathbb{R}^+$. The $[a, b]$-back shifting of I is defined as $I \ominus [a, b] = [m - b, n - a) \cap \mathbb{T}$.

Consider two non-unitary satisfaction signals, $\mathcal{S}_{\phi_1}(t) = \bigvee_{i=1}^{n} \mathcal{S}^*_{\phi_1,i}(t)$ and $\mathcal{S}_{\phi_2}(t) = \bigvee_{j=1}^{m} \mathcal{S}^*_{\phi_2,j}(t)$, each composed of n, m unitary signals with corresponding unitary intervals $I_{\phi_1,i}$ and $I_{\phi_2,j}$, The satisfaction signal of $\psi = \phi_1 \mathcal{U}_{[a,b]} \phi_2$ is defined on multiple unitary time interval $I_{\psi,i,j}$ as follows [3].

Lemma 1 (unitary decomposition of $U_{[a,b]}$ [3]):

$$\begin{aligned}
&\mathcal{S}_\psi(t) := \bigvee_{i=1}^{m} \bigvee_{j=1}^{n} \mathcal{S}^*_{\psi,i,j}(t),\ t \in \mathbb{T} \text{ with,}\\
&I_{\psi,i,j} = ((I_{\phi_1,i} \cap I_{\phi_2,j}) \ominus [a, b]) \cap I_{\phi_1,i}, \text{ and}\\
&\mathcal{S}^*_{\psi,i,j}(t) := \begin{cases} \mathcal{S}^*_{\psi,i,j}(I_{\psi,i,j}) = 1 \\ \mathcal{S}^*_{\psi,i,j}(\mathbb{T} \setminus I_{\psi,i,j}) = 0 \end{cases}.
\end{aligned} \tag{10}$$

In the context of offline monitoring, we look for the satisfaction of an STL formula. To achieve this, we recursively construct satisfaction signals in a bottom-up manner, starting from a satisfaction signal on predicates and proceeding until the "root" formula. This approach enables tracking the temporal properties of the signal. Here is an example of an STL formula: $\phi = G_{[0,2]}(\neg p) \wedge (q \implies w)$, where p, q and w are predicates. Figure 1 represents the satisfaction tree.

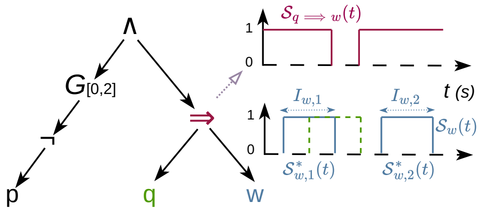


Fig. 1. Satisfaction tree [3] and associated satisfaction signals.

III. STL SATISFACTION ON REACHABLE SETS

The reachable tube, denoted $([\tilde{y}], t)$, encloses all system behaviors. The STL formalism, originally defined over a single trace, is extended to evaluate satisfaction over this set of traces.

A. Boolean intervals

While considering sets, constraints may be satisfied for some executions and violated for others, so the result cannot be simply true or false. To handle this, Boolean intervals are used, which extend traditional Boolean logic to manage uncertainties in set-based contexts [19]. A Boolean interval is a subset of $\mathbb{B} = \{1, 0\}$, belonging to the interval Boolean set $\mathbb{IB} = \{\emptyset, [0, 0], [1, 1], [0, 1]\}$. Here, the values $\emptyset$ and $[0, 1]$ represent, respectively, impossible and undetermined states. In this paper, $[1, 1]$ will be denoted as 1 (true value), and $[0, 0]$ as 0 (false). The logical operations on Boolean values are extended to Boolean intervals $[a]$ and $[b]$ as follows.

Definition 6 (Boolean interval arithmetic [19], [20]):

$$\begin{aligned}
[a] \wedge [b] &= \{a \wedge b \mid a \in [a], b \in [b]\}\\
[a] \vee [b] &= \{a \vee b \mid a \in [a], b \in [b]\}\\
\neg[a] &= \{\neg a \mid a \in [a]\}.
\end{aligned} \tag{11}$$

For example, the behavior of an interval $[0, 1]$ is:

$$\begin{aligned}
0 \wedge [0, 1] = 0, &\quad 0 \vee [0, 1] = [0, 1],\\
1 \wedge [0, 1] = [0, 1], &\quad 1 \vee [0, 1] = 1.
\end{aligned}$$

B. Satisfaction on set predicates

To extend predicate evaluation to a set-based context, two set predicates are defined from [9], [11], the inclusion predicate μ_i, and the exclusion predicate μ_d, corresponding to, $y(t) \in \mathcal{X}^\mu$, and $y(t) \notin \mathcal{X}^\mu$, respectively.

Here, $\mathcal{X}^\mu \in \mathbb{IR}^n$ is a set on an n-dimensional space, and $y(t) \in \mathbb{R}^n$ is the state of the system at time t. Abstracting sets by using level sets, this new syntax is not less general than previously. For example, a predicate $\mu = x > 0$ is represented by an inclusion in a set $\mathcal{X}^\mu = \{x \in \mathbb{R} \mid x > 0\}$. Moreover, these predicates can be time-varying, provided that an exact model of their dynamics is available. In practice, the introduced predicates are redundant but offer flexibility while defining constraints on the system.

Definition 7 (set-based extension of predicate [9], [11]): For the inclusion predicate:

$$([\tilde{y}], t) \vDash \mu_i := \begin{cases} 1, & \text{if } [\tilde{y}](t) \subset \mathcal{X}^\mu, t \in [t_0, T],\\ 0, & \text{if } [\tilde{y}](t) \cap \mathcal{X}^\mu = \emptyset, t \in [t_0, T],\\ [0, 1], & \text{otherwise.} \end{cases} \tag{12}$$

At the formula level, the following equivalence is established:

$$(y, t) \vDash \neg\mu_i \equiv (y, t) \vDash \mu_d, \tag{13}$$

with (y, t) being an execution of our system.

If the predicate cannot be conclusively evaluated over the reachable set $[\tilde{y}_j]$ on $[t_j, t_{j+1}]$, the satisfaction is marked as undetermined and the value $[0, 1]$ is assigned. Lastly, the set-membership STL syntax is written as:

$$\phi := \mu_i \mid T \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1\ \mathcal{U}_{[a,b]} \phi_2. \tag{14}$$

C. Unitary decomposition in uncertain context

As stated in Section II-C, handling temporal operators requires decomposing the satisfaction signal into unitary signals. Following the formalism introduced in Eq. (9), we define two unitary decompositions that are essential for evaluating the $U_{[a,b]}$ operator in uncertain context:

Definition 8 (uncertain and certain unitary signals):

$$\overline{\mathcal{S}^*_\phi} : \mathbb{R}^+ \to \mathbb{IB},\ t \mapsto \begin{cases} [0, 1] & \text{if } t \in \overline{I_\phi}\\ 0 & \text{if } t \in \mathbb{T} \setminus \overline{I_\phi} \end{cases} \tag{15}$$

$$\underline{\mathcal{S}^*_\phi} : \mathbb{R}^+ \to \mathbb{IB},\ t \mapsto \begin{cases} 1 & \text{if } t \in \underline{I_\phi} = 1 \\ 0 & \text{if } t \in \mathbb{T} \setminus \underline{I_\phi}, \end{cases} \tag{16}$$

where $\overline{I_\phi}$ denotes unitary time intervals with only uncertain satisfaction $[0,1]$, and $\underline{I_\phi}$ those with guaranteed satisfaction 1.

Finally, we define the satisfaction signal at t by:

Definition 9 (uncertain unitary decomposition):

$$[\mathcal{S}_\phi] : \mathbb{R}^+ \to \mathbb{IB},\ t \mapsto \bigvee_{j=0}^{N} \overline{\mathcal{S}^*_{\phi,j}}(t) \vee \bigvee_{i=0}^{K} \underline{\mathcal{S}^*_{\phi,i}}(t). \tag{17}$$

An illustrative example is provided in Fig. 3. This decomposition captures how satisfaction is composed. The satisfied portions of the signal are computed as $\bigvee_{i=0}^{K} \underline{\mathcal{S}^*_{\phi,i}}(\cdot)$, consistent with the construction in [3], while $\bigvee_{j=0}^{N} \overline{\mathcal{S}^*_{\phi,j}}(\cdot)$ captures intervals where satisfaction remains uncertain. Lastly, portions of $[\mathcal{S}_\phi](\cdot)$ with a satisfaction value of 0 correspond to guaranteed violations.

If at time t, $\underline{\mathcal{S}^*_{\phi,i}}(t) = 0$ and $\overline{\mathcal{S}^*_{\phi,i}}(t) = [0,1]$, then the Boolean interval arithmetic, defined in Eq. (11), gives $0 \vee [0,1] = [0,1]$, indicating that the signal may still be satisfied. Conversely, if $\underline{\mathcal{S}^*_{\phi,i}}(t) = 1$ and $\overline{\mathcal{S}^*_{\phi,i}}(t) = [0,1]$, then $1 \vee [0,1] = 1$, meaning that the satisfaction is definitively true, and the uncertainty has no further impact—thus maintaining consistency with the semantics from [3] in Eq. (9).

Finally, Boolean operations on satisfaction signals are extended following the arithmetic introduced in Eq. (11):

Definition 10 ($\vee$ operator on interval satisfaction signal):

$$[\mathcal{S}_{\phi_1 \vee \phi_2}](t) := \begin{cases} 1, \text{ if } [\mathcal{S}_{\phi_1}](t) \vee [\mathcal{S}_{\phi_2}](t) = 1 \\ 0, \text{ if } [\mathcal{S}_{\phi_1}](t) \vee [\mathcal{S}_{\phi_2}](t) = 0 \\ [0,1], \text{ otherwise.} \end{cases} \tag{18}$$

Definition 11 ($\neg$ operator on interval satisfaction signal):

$$[\mathcal{S}_{\neg\phi_1}](t) := \begin{cases} 1, \text{ if } [\mathcal{S}_{\phi_1}](t) = 0 \\ 0, \text{ if } [\mathcal{S}_{\phi_1}](t) = 1 \\ [0,1], \text{ otherwise.} \end{cases} \tag{19}$$

D. Computing unitary signals from predicates

To compute the overall formula satisfaction, it starts at the predicate level by identifying two unitary signals. The unitary time interval $\underline{I_\mu}$ and $\overline{I_\mu}$ are constructed by merging consecutive time intervals $[t_j, t_{j+1}]$ where the predicate is consistently satisfied, i.e., $\left(\bigcup_{j_0}^{j_f} [\tilde{y}_j]\right) \subset \mathcal{X}^\mu$, or consistently uncertain, i.e., $\left(\bigcup_{j_0}^{j_f} [\tilde{y}_j] \cap \mathcal{X}^\mu\right) \neq \emptyset$ and $\left(\bigcup_{j_0}^{j_f} [\tilde{y}_j]\right) \not\subset \mathcal{X}^\mu$. This forms a merged unitary time interval $I_\mu = [t_{j_0}, t_{j_f}]$.

Finally, we compute the satisfaction signal of a predicate:

$$[\mathcal{S}_\mu](t) := \bigvee_{j=0}^{N} \overline{\mathcal{S}^*_{\mu,j}}(t) \vee \bigvee_{i=0}^{K} \underline{\mathcal{S}^*_{\mu,i}}(t), \text{ with}$$

$$\overline{\mathcal{S}^*_\mu}(t) := \begin{cases} [0,1], & \text{if } t \in \overline{I_\mu} \\ 0, & \text{if } t \in \mathbb{T} \setminus \overline{I_\mu} \end{cases} \tag{20}$$

$$\underline{\mathcal{S}^*_\mu}(t) := \begin{cases} 1, & \text{if } t \in \underline{I_\mu} \\ 0, & \text{if } t \in \mathbb{T} \setminus \underline{I_\mu}. \end{cases}$$

E. Until operator under uncertainty

The definition in Eq. (10) provides a unitary Boolean decomposition of the Until operator, as proposed by Maler. While this decomposition allows us to trace uncertainties within the satisfaction tree, it was originally formulated for a deterministic setting and thus requires adaptation to account for uncertainty.

Building on the satisfaction semantics of Until defined by [9], the Until satisfaction is as follows:

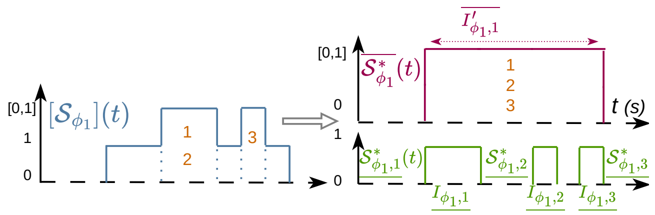
Definition 12 ($U_{[a,b]}$ operator in uncertain context [9]):

$$([\tilde{y}], t) \vDash \phi_1 U_{[a,b]} \phi_2 := \begin{cases} 1, & \text{if } \exists t' \in [t+a, t+b],\ [\mathcal{S}_{\phi_2}](t') = 1 \\ & \quad \text{and } \forall t'' \in [t, t'],\ [\mathcal{S}_{\phi_1}](t'') = 1, \\ 0, & \text{if } \forall t' \in [t+a, t+b],\ [\mathcal{S}_{\phi_2}](t') = 0 \\ & \quad \text{or } \exists t'' \in [t, t'],\ [\mathcal{S}_{\phi_1}](t'') = 0, \\ [0,1], & \text{otherwise.} \end{cases} \tag{21}$$

A unitary decomposition for the Until operator, consistent with the structure defined in Eq. (17), is proposed. We define four unitary signals with corresponding unitary time intervals $\overline{I'_{\phi_1}}$, $\underline{I_{\phi_1}}$, $\overline{I_{\phi_2}}$ and $\underline{I_{\phi_2}}$ defined on positive times with $\overline{\mathcal{S}^*_{\phi_1}}(\cdot)$, $\underline{\mathcal{S}^*_{\phi_1}}(\cdot)$, $\overline{\mathcal{S}^*_{\phi_2}}(\cdot)$ and $\underline{\mathcal{S}^*_{\phi_2}}(\cdot)$. As stated in the definition of $U_{[a,b]}$, the subformula ϕ_1 must hold over the entire interval $[t, t']$. If ϕ_1 holds everywhere except at some $t'' \in [t, t']$ where $\mathcal{S}_{\phi_1}(t'') = [0,1]$, then the overall satisfaction of ϕ_1 is conservatively set to $[0,1]$. However, this uncertainty may be resolved to 1 or 0 through further refinement or additional computations.

Then the definition of the unitary time interval $\overline{I_{\phi_1}}$ is different from Eq. (15) in this approach. Considering a new unitary time interval $\overline{I'_{\phi_1}}$, it is defined as: $\forall t \in \overline{I'_{\phi_1}}, [\mathcal{S}_{\phi_1}](t) = 1$ or $[\mathcal{S}_{\phi_1}](t) = [0,1]$. An illustration is proposed in Fig. 2.

$$\overline{\mathcal{S}^*_{\phi_1}}(t) := \begin{cases} [0,1] & \text{if } t \in \overline{I'_{\phi_1}}, \\ 0, & \text{if } t \in \mathbb{T} \setminus \overline{I'_{\phi_1}}. \end{cases} \tag{22}$$



$\overline{I'_{\phi_1,1}}$
[0,1]
$\overline{\mathcal{S}^*_{\phi_1}}(t)$
1
2
3
0
t (s)
[0,1]
$[\mathcal{S}_{\phi_1}](t)$
1
2
3
1
0
$\underline{\mathcal{S}^*_{\phi_1,1}}(t)$
$\underline{\mathcal{S}^*_{\phi_1,2}}$
$\underline{\mathcal{S}^*_{\phi_1,3}}$
$\underline{I_{\phi_1,1}}$
$\underline{I_{\phi_1,2}}$
$\underline{I_{\phi_1,3}}$

Fig. 2. Unitary signal decomposition of $[\mathcal{S}_{\phi_1}](t)$ in $\phi_1 U_{[a,b]} \phi_2$; uncertainty indices in orange.

Finally, we propose the interval satisfaction signal of Until as follows.

Proposition 1 (satisfaction signal of $\psi = \phi_1 \mathcal{U}_{[a,b]} \phi_2$):

$$[\mathcal{S}_\psi](t) := \underline{\mathcal{S}_\psi}(t) \vee \overline{\mathcal{S}_\psi}(t), \tag{23}$$

with the signal $\overline{\mathcal{S}_\psi}(\cdot)$:

$$\overline{\mathcal{S}_\psi}(t) = \overline{\mathcal{S}_{\psi}}_{\underline{\phi_2}}(t) \vee \overline{\mathcal{S}_{\psi}}_{\overline{\phi_2}}(t), \text{ with,}$$

$$\begin{cases} \overline{\mathcal{S}_{\psi}}_{\underline{\phi_2}}(t) = \bigvee_{i=1}^{m} \bigvee_{j=1}^{n} \overline{\mathcal{S}^*_{\psi}}_{\underline{\phi_2},i,j}(t), \\ \text{with, } \overline{I_{\psi}}_{\underline{\phi_2},i,j} = \left(\left(\overline{I'_{\phi_1,i}} \cap \underline{I_{\phi_2,j}}\right) \ominus [a,b]\right) \cap \overline{I'_{\phi_1,i}}, \end{cases}$$

and,

$$\begin{cases} \overline{\mathcal{S}_{\psi}}_{\overline{\phi_2}}(t) = \bigvee_{i=1}^{m} \bigvee_{j=1}^{n} \overline{\mathcal{S}^*_{\psi}}_{\overline{\phi_2},i,j}(t), \\ \text{with, } \overline{I_{\psi}}_{\overline{\phi_2},i,j} = \left(\left(\overline{I'_{\phi_1,i}} \cap \overline{I_{\phi_2,j}}\right) \ominus [a,b]\right) \cap \overline{I'_{\phi_1,i}}, \end{cases} \tag{24}$$

with the signal $\underline{\mathcal{S}_\psi}(\cdot)$:

$$\underline{\mathcal{S}_\psi}(t) = \bigvee_{i=1}^{m} \bigvee_{j=1}^{n} \underline{\mathcal{S}^*_{\psi,i,j}}\ , \text{ with,} \tag{25}$$

$$\underline{I_{\psi,i,j}} = \left(\left(\underline{I_{\phi_1,i}} \cap \underline{I_{\phi_2,j}}\right) \ominus [a,b]\right) \cap \underline{I_{\phi_1,i}}.$$

Sketch of Proof: We need to prove the soundness of the decomposition $[\mathcal{S}_\psi](t) := \underline{\mathcal{S}_\psi}(t) \vee \overline{\mathcal{S}_\psi}(t)$. Starting from the decomposition of unitary signals given in Eq. (10) [3], our approach differs in the treatment of temporal quantifiers on ϕ_1 and ϕ_2. As previously defined, $\overline{I_{\phi_2}}$ or $\underline{I_{\phi_2}}$ indicates continuous satisfaction of ϕ_2, but the existential quantifier $\exists t'$ removes dependency on successive unitary signals. Conversely, for ϕ_1, the universal quantifier $\forall t \in [t, t']$ introduces dependencies between successive unitary signals, potentially transitioning as $1 \to [0,1] \to 1$.

$\overline{\mathcal{S}_\psi}(\cdot)$ is composed by case distinction, aiming to capture time intervals with guaranteed 0 satisfaction. Due to the universal quantifier on ϕ_1, any combination of 1 and $[0,1]$ over $\overline{I'_{\phi_1}}$ yields $[0,1]$, but never 0, justifying the semantics of $\overline{\mathcal{S}^*_{\phi_1}}(\cdot)$. Specifically, $\overline{I_{\psi}}_{\underline{\phi_2}}$ defines a unitary time interval $\overline{I_\psi}$ of an uncertain Until unitary signal where ϕ_1 is uncertain, but ϕ_2 is certain. Similarly, $\overline{\mathcal{S}^*_{\psi}}_{\overline{\phi_2}}(\cdot)$ represents an uncertain Until unitary signal where both condition on ϕ_1 and ϕ_2 are uncertain.

Additionally, in our semantics, the signal $\overline{\mathcal{S}^*_{\phi_1}}(\cdot)$ already accounts for time intervals where the satisfaction is 1. When intersecting the shorter unitary interval $\underline{I_{\phi_1}} \subset \overline{I'_{\phi_1}}$ with $\overline{I_{\phi_2}}$, the resulting interval $\overline{I_{\psi}}_{\underline{\phi_1},\overline{\phi_2}} = \left(\left(\underline{I_{\phi_1}} \cap \overline{I_{\phi_2}}\right) \ominus [a,b]\right) \cap \underline{I_{\phi_1}}$ is necessarily included in $\overline{I_{\psi}}_{\overline{\phi_2}}$, due to the semantics of back-shifting and intersection applied to these unitary intervals, moreover, the satisfaction of ψ over $\overline{I_{\psi}}_{\underline{\phi_1},\overline{\phi_2}} \subset \overline{I_{\psi}}_{\overline{\phi_2}}$ remains $[0,1]$.

Finally, $\underline{\mathcal{S}_\psi}(\cdot)$ corresponds to a unitary signal with guaranteed 1 satisfaction, consistent with the initial decomposition (Eq. (10)) [3] and the semantics for Until in uncertain contexts (Eq. (21)) [9]. Thus, $[\mathcal{S}_\psi](t) := \underline{\mathcal{S}_\psi}(t) \vee \overline{\mathcal{S}_\psi}(t)$ maintains the same logical composition as Eq. (17), completing the proof.

IV. Uncertainty tracking in logical Operators

Uncertainties in STL satisfaction may stem from incomplete simulation or over-approximated reachable sets. To distinguish these sources and enable targeted refinement, a method is introduced to track the origin of uncertainties through unitary signal decomposition. Uncertain time intervals on predicates, represented by the corresponding reachable set indices, are memorized and propagated through the satisfaction tree (Section II-C) using simple rules as detailed in the following. These memorized uncertain time intervals are called "markers". They point out the reachable sets that contribute to uncertainty in the final satisfaction.

A. Tracking from predicate level to the root

Starting from Eq. (17) and Eq. (12), if a reachable set $[\tilde{y}_j]$ defined on $[t_j, t_{j+1}]$ satisfies $[\tilde{y}_j] \cap \mathcal{X}^\mu \neq \emptyset$ and $[\tilde{y}_j] \not\subset \mathcal{X}^\mu$ as in Fig. 4, an uncertainty arises. As discussed in Section III-D, $\overline{I_\mu}$ is computed as the union of multiple time intervals $[t_j, t_{j+1}]$. If multiple reachable sets intersect the predicate, all corresponding indices j are associated with this unitary signal $\overline{S^*_\mu}$ using a marker denoted $\mathcal{M}_{\overline{I_\mu}}$:

$$\mathcal{M}_{\overline{I_\mu}} = \bigcup_{j_0}^{j_f} \{j\}, \quad \text{with } \overline{I_\mu} = [t_{j_0}, t_{j_f}].$$

B. Boolean operator

a) OR: To track uncertainties, we retrieve the markers originating from the unitary signals of the subformulas. A logic is then established to extract and propagate these markers. Practically, when computing new unitary signals (i.e. $\overline{S^*_{p,2}}$ in Fig. 3), if a new unitary time interval $\overline{I} = [t_i, t_{i+1}]$ is associated with $[0,1]$, we assign a marker to this interval containing all uncertainty indices j from reachable sets $[\tilde{y}_j]$ (i.e. coming from $\overline{S^*_{q,2}}$). Specifically, when computing $[0,1] \vee 1 = 1$, all markers from the $[0,1]$ satisfaction are discarded for the resulting time interval. Finally, identical satisfaction time intervals are merged to form unitary signals, incorporating all corresponding markers (i.e. $\overline{S^*_{p,1}}$ in Fig. 3).

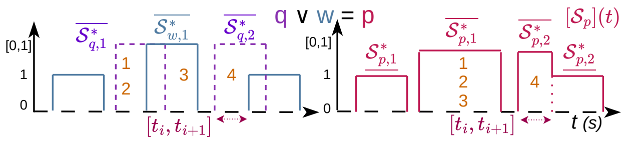


Fig. 3. Uncertainty tracking in the $\vee$ operation between two satisfaction signal (in color) using unitary signals; uncertainty indices in orange.

Let $\phi = \phi_1 \vee \phi_2$ be a formula. Given a newly computed unitary interval $\overline{I_\phi}$ and contributing unitary intervals $\overline{I_{\phi_1}}$ and $\overline{I_{\phi_2}}$, the marker is assigned as:

$$\mathcal{M}_{\overline{I_\phi}} = \begin{cases} \mathcal{M}_{\overline{I_{\phi_1}}}, & \text{if } [S_{\phi_2}](\overline{I_\phi}) = 0 \\ \mathcal{M}_{\overline{I_{\phi_2}}}, & \text{if } [S_{\phi_1}](\overline{I_\phi}) = 0 \\ \mathcal{M}_{\overline{I_{\phi_2}}} \cup \mathcal{M}_{\overline{I_{\phi_1}}}, & \text{if } \overline{I_{\phi_1}} \cap \overline{I_{\phi_2}} \neq \emptyset. \end{cases}$$

b) NEG: Negation is straightforward: $(s,t) \models \neg\phi \iff (s,t) \not\models \phi$. Therefore, $\neg[\mathcal{S}_\phi](\cdot)$ corresponds to the negation of the satisfaction for each time interval in $[\mathcal{S}_\phi](\cdot)$. The markers are preserved, as the negation of an uncertain satisfaction remains uncertain.

C. Temporal operators

a) Until: Starting from Eq. (22) with $\psi = \phi_1 \mathcal{U}_{[a,b]} \phi_2$, the computation of the new unitary signal $\overline{\mathcal{S}^*_{\phi_1,i}}(\cdot)$ is associated with the union of contributing markers, forming a new marker on $\overline{I'_{\phi_1,i}}$ (see Fig. 2).

From Eq. (24), during the computation of the Until operator, successive back-shifting and intersections are applied over unitary time intervals $\overline{I'_{\phi_1,i}}$ and $\overline{I_{\phi_2,j}}$. Throughout this process, the corresponding markers $\mathcal{M}_{\overline{I'_{\phi_1,i}}}$ or $\mathcal{M}_{\overline{I_{\phi_2,j}}}$ are merged to form new markers $\mathcal{M}_{\psi_{\underline{\phi_2},i,j}}$ and $\mathcal{M}_{\psi_{\overline{\phi_2},i,j}}$ corresponding to the unitary signals $\overline{\mathcal{S}^*_{\psi_{\underline{\phi_2},i,j}}}$, $\overline{\mathcal{S}^*_{\psi_{\overline{\phi_2},i,j}}}$ respectively.

To compute $\overline{\mathcal{S}_\psi}(\cdot)$, the disjunction ($\vee$) of all resulting unitary signals $\overline{\mathcal{S}^*_\psi}(\cdot)$ is taken, with the uncertainty tracking mechanism from Section IV-B applied to mark each resulting unitary interval with its sources of uncertainty. This same tracking logic is used when combining with $\underline{\mathcal{S}_\psi}(\cdot)$ in $[\mathcal{S}_\psi](t) := \underline{\mathcal{S}_\psi}(t) \vee \overline{\mathcal{S}_\psi}(t),\ t \in \mathbb{T}$, ensuring consistent propagation of uncertainty throughout the evaluation.

b) Finally and Globally: Since the Until operator is handled, and the temporal operators $F_{[a,b]}$ and $G_{[a,b]}$ are derived from it as:

$$F_{[a,b]}\phi = T\, U_{[a,b]}\phi, \quad G_{[a,b]}\phi = \neg(F_{[a,b]}\neg\phi),$$

the tracking mechanism naturally extends to both operators.

D. Handling incomplete simulation

Following [3], signal comparison relies on the minimal signal length defined by the formula semantics. To ensure conservativeness and enable further computation, shorter signals are extended with $[0,1]$ to the required length. This avoid truncation errors particularly in temporal operators like $U_{[a,b]}$, preserving information and preventing premature conclusions.

This conservative logic maintains consistency, e.g., $[0,1] \wedge 1 = [0,1]$, $[0,1] \wedge 0 = 0$, $[0,1] \vee 1 = 1$.

To distinguish such completions, a specific marker with a common negative indices is assigned to $[0,1]$ values introduced by signal extension, separating them from uncertainty originating in reachable sets.

V. Adaptative sampling approach

A. Recursive bisection using tracked error

The step size in our approach is governed by a precision criterion. Specifically, validated numerical integration presented in Section II-A, dynamically adjusts the step size h_j to satisfy a given precision constraint, which is closely related to the Local Truncation Error (LTE). This aligns with the approach in [5], where LTE is computed using the order conditions of Runge-Kutta methods to ensure numerical stability and accuracy.

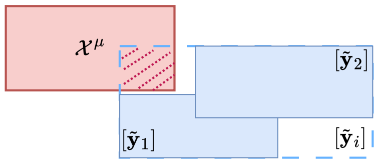


Fig. 4. Time bisection and contraction, $\mathcal{X}_\mu$ a set predicate.

However, excessively small step sizes not only increase computational cost but may also lead to overly pessimistic results, as the simulation becomes longer and the accumulated LTE may grow beyond what is necessary. To mitigate this, our approach balances step size selection to maintain accuracy while avoiding excessive computational overhead.

This causes two problems. First, some boxes may exhibit very large step times (in the context of dynamic equations), leading to uncertainty in transition times when strict timing conditions are considered. Second, these boxes may appear excessively large due to the need to cover the system state set over an extended time interval. This property introduces timing uncertainty and leads to overly conservative enclosures, potentially causing false crossing statements.

The overall refinement procedure is summarized in Algorithm 1. First, an initial tube is computed with `compute_tube_init()` (line 3). Then, an iterative loop progressively refines the tube until the STL satisfaction can be determined or the stopping criterion is met.

At each iteration, the current tube is evaluated against the STL specification using `stl_verifier()` (line 6). The resulting satisfaction signal, which may contain uncertain values, is analyzed by `uncertainty_tracker()` (line 7), propagating uncertainty through the satisfaction tree and identifying the reachable sets that are responsible. If no uncertainty markers remain, the procedure terminates. Otherwise, the corresponding boxes are refined: their time intervals are bisected and the results are contracted using `bisect_and_contract()` (line 11). This refinement is performed by recomputing two new *Picard boxes* with an additional integration scheme. The contraction step reduces as much as possible the diameter of the enclosures, thereby tightening the approximation of the trajectories [5]. For example, a single box $[\tilde{\mathbf{y}}_i]$ is replaced with two smaller boxes $[\tilde{\mathbf{y}}_1]$ and $[\tilde{\mathbf{y}}_2]$, as illustrated in Fig. 4.

If the uncertainty originates from an incomplete simulation, detected via `exists_negative_index()` (line 12), the horizon is extended by h and the missing portion of the tube is appended with `complete_tube()` (line 14). This process repeats until all uncertainties are resolved or the stopping threshold is reached, and the algorithm finally returns the refined satisfaction.

B. Case Study

In our approach, to perform reachability analysis, we are using a tool based on affine arithmetic and Runge-Kutta methods called DynIbex[1]. Other tools, such as CORA[2] [9], [10] and CAPD[3] [7], enable the reachability analysis of continuous dynamic systems and hybrid systems. The presented method generalizes to systems described by ODEs and any STL formula expressed with set predicates.

[1]https://perso.ensta-paris.fr/~chapoutot/dynibex/
[2]https://tumcps.github.io/CORA/
[3]http://capd.ii.uj.edu.pl/

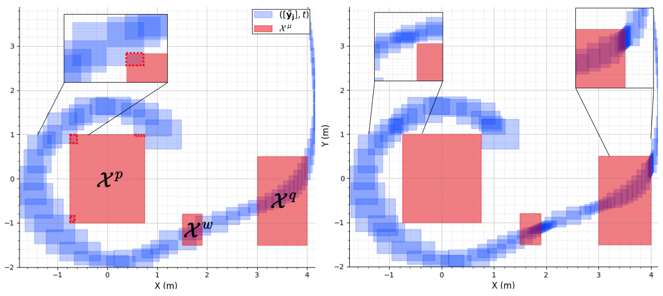


Fig. 5. Initial tube on the left and refined tube on the right.

To demonstrate the efficiency of the proposed approach, an STL formula is verified on a tube, similar to the one studied in Section II-C: $\phi = G_{[0,1]}(G_{[0,2]}(\neg p) \wedge (q \implies F_{[1,2]}w))$. The tube dynamics are derived from a system governed by the Van der Pol oscillator and simulated over a 10 second horizon. Described as:

$$\begin{cases} \dot{x} &= y, \\ \dot{y} &= \mu(1-x^2)y - x. \end{cases} \tag{26}$$

A comparison between the initial (left) and refined (right) tubes is shown in Fig. 5. The predicate q is associated with the region on the right (red box), $\mathcal{X}^w$ with the central region, and $\mathcal{X}^p$ with the leftmost region. These correspond to spatial constraints in the STL specification. In natural language, this formula states that on a 1 second horizon, whenever the system encounters the $\mathcal{X}^q$ zone, it must reach the $\mathcal{X}^w$ zone within $[1,2]$ seconds. Additionally, within a $[0,2]$ time horizon, the system must never enter the p box, which represents a forbidden zone.

As observed in the upper-left zoom, rediscretizing and contracting the boxes reduced pessimism and eliminated the uncertainty regarding the crossing of the forbidden zone. The upper-right zoom highlights a finer temporal resolution that mitigates the uncertainty on the moment of crossing $\mathcal{X}^q$, which enforces a temporal constraint in $\mathcal{X}^w$.

Finally, analyzing the satisfaction signals (Fig. 6) over a 10 s time horizon (simulation duration) demonstrates that,

Algorithm 1 Adaptive Refinement of a Tube under STL Satisfaction

```
1: Input: System parameters params, initial final time final_time, STL
   formula, set of predicates, stopping threshold stopping_criterion
2: Output: Refined satisfaction
3: tube ← compute_tube_init(params, final_time)
4: simulation_complete ← False
5: while not simulation_complete do
6:     satisfaction ← stl_verifier(predicates, formula, tube)
7:     uncertainty_marks ← uncertainty_tracker(satisfaction,
       stopping_criterion)
8:     if uncertainty_marks is empty then
9:         simulation_complete ← True
10:    else
11:        tube ← bisect_and_contract(tube, uncertainty_marks)
12:        if exists_negative_index(uncertainty_marks) then
13:            final_time ← final_time + h
14:            tube ← complete_tube(param, final_time, tube)
15:        end if
16:    end if
17: end while
18: return satisfaction
```

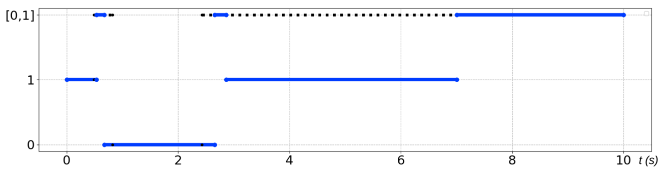


Fig. 6. Original satisfaction signal $\mathcal{S}_\phi(t)$ in black dots vs. refined one in blue.

despite an initial 4.5 s uncertainty between 0 s and 7 s, our approach successfully mitigates uncertainty, reducing it to 0.5 s of uncertainty and 6.5 s of guaranteed satisfaction within this interval. In this approach, the tube enables formula computation from 0 s to 7 s due to a 3 s shift (Fig. 6). In our program, the remaining uncertainty $[0,1]$ from 7 s to 10 s is flagged as originating from an incomplete simulation.

Computation Time: On the same model and specification, we compared three approaches using the same hardware setup (an HP EliteBook with an Intel Core i7 processor): our uncertainty tracking mechanism implemented with DynIbex, an incremental approach inspired by [9] that we also implemented within DynIbex for a fair comparison, and the four-valued semantics approach using CORA [9]. The corresponding computation times are reported in Table I. In each sum, the first value corresponds to the simulation time, and the second to the STL verification time.

TABLE I
COMPUTATION TIME (IN SECONDS) FOR TWO VERIFICATION PROBLEMS. THE FIRST WAS SOLVED ONLY WITH DYNIBEX, WHILE THE SECOND COMPARES DYNIBEX AND CORA.

$([\tilde{y}], t = 2.63s) \models \phi$ (Dynibex only)		
	Uncertainty Tracking	Incremental Only
Dynibex	0.485 + 0.222 = 0.707s	3.37 + 0.15 = 3.52s

$([\tilde{y}], t = 3s) \models \phi$ (Dynibex vs CORA)		
	Uncertainty Tracking	Incremental Only
Dynibex	0.600 + 0.156 = 0.756s	4.0 + 0.15 = 4.15s
CORA	—	11.9 + 4.2 = 16.1s

In this case, verifying the formula at $t = 2.63\,$s is among the most precision-demanding points. To obtain a conclusive verification result, we increased the precision criterion for the LTE by a factor of 10^{-5} and employed a higher-order integration scheme. Consequently, the simulation took approximately $7\times$ longer than with a less precise configuration. The increase in verification time (from 0.15s to 0.222s) under uncertainty tracking is due to repeated bisection and contraction of the reachable sets during the STL evaluation.

From a broader perspective, at a less demanding time point ($t = 3\,$s), we compared execution times between our tracking implementation and CORA, using identical initial time steps (Table I). For each approach, we reported the fastest time required to reach a conclusive result, our tracking method is about $21\times$ faster than the incremental one using CORA. This performance gain may enable precise, on-the-fly verification of nonlinear behaviors against STL specifications in online monitoring contexts.

VI. Discussion

One limitation of our approach lies in the propagation of uncertainty through unitary signals under the disjunction ($\vee$) operator. Contributing reachable set indices are merged across the entire resulting unitary signal, potentially leading to over-approximation. As shown in Fig. 3, indices (1,2) affect only part of the signal but are attributed to the whole, which may trigger unnecessary refinements in early iterations, especially for complex formulas. However, this limitation is inherent to the propagation process within unitary decomposition. As future work, incorporating time-dependence shifting into the markers could help reduce this over-approximation.

VII. Conclusion

This work introduces a novel method for managing uncertainty in STL verification by embedding Boolean interval arithmetic within a reachability analysis framework. Our method systematically tracks uncertainty within the satisfaction tree, distinguishing between uncertainties arising from incomplete observations and those caused by reachable set over-approximation. This enables an offline/online refinement strategy, reducing unnecessary computations while ensuring precise verification.

A key feature of our method is its ability to contract and resample only the necessary reachable sets, thereby refining STL satisfaction evaluation. Through an experimental case study, we demonstrate the effectiveness of our approach in reducing uncertainty.

ACKNOWLEDGMENT

The authors acknowledge support from the CIEDS[4] with the STARTS project.

References

[1] D. Dvorak and B. Kuipers, "Model-based monitoring of dynamic systems," in *Proceedings of the 11th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 1989, pp. 1238–1243. [Online]. Available: https://ijcai.org/Proceedings/89-2/Papers/068.pdf

[2] 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," *Lecture Notes in Computer Science*, vol. 10457, pp. 135–175, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-75632-5_5

[3] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, ser. Lecture Notes in Computer Science, vol. 3253. Springer, 2004, pp. 152–166. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-30206-3_12

[4] T. Nghiem, S. Sankaranarayanan, G. Fainekos, F. Ivancić, A. Gupta, and G. J. Pappas, "Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems," in *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 211–220. [Online]. Available: https://doi.org/10.1145/1755952.1755983

[5] J. Alexandre Dit Sandretto and A. Chapoutot, "Validated explicit and implicit runge-kutta methods," *Reliable Computing*, vol. 22, 2016, special issue devoted to material presented at SWIM 2015. [Online]. Available: https://hal.science/hal-01243053

[6] M. Althoff, "An introduction to CORA 2015," in *Proc. of the 1st and 2nd Workshop on Applied Verification for Continuous and Hybrid Systems*. EasyChair, December 2015, pp. 120–151. [Online]. Available: https://easychair.org/publications/paper/xMm

[7] T. Kapela, M. Mrozek, D. Wilczak, and P. Zgliczyński, "Capd::dynsys: A flexible c++ toolbox for rigorous numerical analysis of dynamical systems," *Communications in Nonlinear Science and Numerical Simulation*, vol. 101, p. 105578, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1007570420304081

[8] A. Le Coënt, J. Alexandre dit Sandretto, A. Chapoutot, and L. Fribourg, "An improved algorithm for the control synthesis of nonlinear sampled switched systems," *Formal Methods in System Design*, vol. 53, no. 3, pp. 363–383, 2018. [Online]. Available: https://doi.org/10.1007/s10703-017-0305-8

[9] F. Lercher and M. Althoff, "Using four-valued signal temporal logic for incremental verification of hybrid systems," in *Proc. of Computer Aided Verification (CAV)*, ser. Lecture Notes in Computer Science, A. Gurfinkel and V. Ganesh, Eds., vol. 14683. Springer Nature Switzerland, 2024, pp. 259–281. [Online]. Available: https://doi.org/10.1007/978-3-031-65633-0_12

[10] H. Roehm, J. Oehlerking, T. Heinz, and M. Althoff, "STL model checking of continuous and hybrid systems," in *Automated Technology for Verification and Analysis (ATVA 2016)*, ser. Lecture Notes in Computer Science, vol. 9938. Springer, 2016, pp. 412–427. [Online]. Available: https://doi.org/10.1007/978-3-319-46520-3_26

[11] J. Tillet, A. Besset, and J. A. D. Sandretto, "Guaranteed satisfaction of a signal temporal logic formula on tubes," *Acta Cybernetica*, 2025, accepted, to appear.

[12] D. Ishii, N. Yonezaki, and A. Goldsztejn, "Monitoring temporal properties using interval analysis," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E99.A, no. 2, pp. 442–453, 2016. [Online]. Available: https://doi.org/10.1587/transfun.E99.A.442

[13] X. Yu, W. Dong, S. Li, and X. Yin, "Model predictive monitoring of dynamical systems for signal temporal logic specifications," *Automatica*, vol. 160, p. 111445, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S000510982300612X

[14] L. Baird, A. Harapanahalli, and S. Coogan, "Interval signal temporal logic from natural inclusion functions," *IEEE Control Systems Letters*, vol. 7, pp. 3555–3560, 2023.

[15] N. Kochdumper and S. Bak, "Fully automated verification of linear systems against temporal logic specifications," in *Nonlinear Analysis: Hybrid Systems*. IFAC, 2024.

[16] T. Wright and I. Stark, "Property-directed verified monitoring of signal temporal logic," in *Runtime Verification (RV 2020)*, ser. Lecture Notes in Computer Science, J. Deshmukh and D. Ničković, Eds., vol. 12399. Springer, Cham, 2020, pp. 337–353. [Online]. Available: https://doi.org/10.1007/978-3-030-60508-7_19

[17] M. Abate, E. Feron, and S. Coogan, "Monitor-based runtime assurance for temporal logic specifications," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 5, pp. 1932–1947, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/9093858

[18] A. Bauer, M. Leucker, and C. Schallhart, "Runtime verification for ltl and tltl," *ACM Transactions on Software Engineering and Methodology*, vol. 20, no. 4, pp. 14:1–14:64, 2011. [Online]. Available: https://doi.org/10.1145/2000799.2000800

[19] J. Alexandre Dit Sandretto and A. Chapoutot, "Logical differential constraints based on interval boolean tests," in *Fuzzy Techniques: Theory and Applications*. Springer International Publishing, 2019, vol. 1000, pp. 788–792.

[20] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, "Applied interval analysis," in *Applied Interval Analysis*, L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, Eds. Springer, 2001, pp. 11–43.

[4]CIEDS: French Interdisciplinary Center for Defense and Security.