Efficient SMT-Based Model Checking for Signal Temporal Logic


Jia Lee
POSTECH
Pohang, South Korea
cee5539@postech.ac.kr

Geunyeol Yu
POSTECH
Pohang, South Korea
rgyen@postech.ac.kr

Kyungmin Bae
POSTECH
Pohang, South Korea
kmbae@postech.ac.kr

2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE) | 978-1-6654-0337-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ASE51524.2021.9678719

***Abstract*—Signal temporal logic (STL) is widely used to specify and analyze properties of cyber-physical systems with continuous behaviors. However, STL model checking is still quite limited, as existing STL model checking methods are either incomplete or very inefficient. This paper presents a new SMT-based model checking algorithm for verifying STL properties of cyber-physical systems. We propose a novel translation technique to reduce the STL bounded model checking problem to the satisfiability of a first-order logic formula over reals, which can be solved using state-of-the-art SMT solvers. Our algorithm is based on a new theoretical result, presented in this paper, to build a small but complete discretization of continuous signals, which preserves the bounded satisfiability of STL. Our translation method allows an efficient STL model checking algorithm that is refutationally complete for bounded signals, and that is much more scalable than the previous refutationally complete algorithm.**

***Index Terms*—Signal temporal logic, model checking, SMT**

I. Introduction

Many safety-critical systems, such as cars and airplanes, interact with physical entities, and therefore are hybrid systems exhibiting both discrete and continuous behaviors. The safety requirements of such cyber-physical systems (CPSs) often involve continuously changing states of physical environments. *Signal temporal logic* (STL) is a temporal logic formalism to specify linear-time properties of continuous signals [1]. STL is widely used for specifying and analyzing (safety) requirements of safety-critical CPSs, including automotive, avionics, robotics, and medical systems [2]–[8].

There are many approaches for analyzing STL properties. STL monitoring [9]–[12] checks whether a signal satisfies an STL property, and STL falsification [13], [14] tries to find a counterexample of an STL property by (randomly) generating signals. These methods are based on concrete sampling of signals and cannot be used to guarantee correctness. On the other hand, the reachability analysis of hybrid automata, such as [15]–[19], can guarantee the correctness of invariants, and some STL formulas can be encoded using reachability [20].

However, model checking of general STL properties is still very limited, as existing STL model checking algorithms are incomplete or inefficient. Recently, two STL model checking techniques have been proposed. The method proposed in [21] samples a finite number of concrete time points to perform the reachability analysis over those points, and therefore the correctness cannot be guaranteed. The work [22] provides a refutationally complete model checking procedure for STL, but the algorithm is not scalable in practice.

A common challenge in existing approaches is to *discretize continuous signals* to reason about temporal operators. Since model checking of hybrid systems involves an infinite number of states that continuously change over an uncountable time domain, signal discretization has been studied for effectively analyzing temporal logic properties (e.g, LTL [23], [24]) of hybrid systems. The important problem is obtaining *complete discretization* of a signal to preserve the correctness of model checking, while maintaining the size of discretization as small as possible for the performance of the analysis.

This paper proposes a new SMT-based model checking algorithm for verifying STL properties of hybrid systems that is refutationally complete with respect to bounded signals. Our algorithm is based on a new theoretical result to build *small but complete* discretized signals for the bounded satisfiability of STL. Our algorithm combines the advantages of the previous methods [21], [22]: it considers a finite number of sampled time points, but which is complete for STL model checking without any loss of information.

We consider bounded signals with finite variable points (excluding Zeno behaviors), which is typically assumed when analyzing real-time and hybrid systems [1], [23]–[26]. For a signal with finite variable points, the satisfaction of a temporal logic formula must depend on a finite number of sampled time points. There exist signal discretization methods proposed for different temporal logics [23], [24], [27], but they cannot be directly applicable to STL. The main difficulty comes from the timed until operator $\mathbf{U}_I$, which often results in too "fine" discretized signals that prevent efficient model checking. This paper presents a novel technique to build a complete but reasonably coarse discretized signals for STL (Sec. IV).

A complete discretization of signals allows reducing the bounded satisfaction problem of STL into the satisfiability of a first-order formula. It is well known that temporal logics for discrete systems, such as LTL and MITL, can be analyzed for each subformula in a modular way [28]. However, an effective translation of STL is nontrivial, because temporal operators in STL involves arbitrary nonnegative intervals of the real

This work was partially supported by the National Research Foundation of Korea (NRF) grants funded by the Korea government (MSIT) (No. 2021R1A5A1021944 and No. 2019R1C1C1002386).

978-1-6654-0337-5/21/$31.00 ©2021 IEEE
DOI 10.1109/ASE51524.2021.00039

numbers. We leverage syntactic separation of STL [22] to translate the bounded satisfaction of STL into a quantifier-free first-order formula in linear real arithmetic (Sec. V).

Based on our translation method, we present a new bounded STL model checking algorithm (Sec. VI). Our algorithm builds a first-order formula that is satisfiable if and only if there exists a counterexample signal up to a bound N on the size of a discretized signal. To encode the existence of signals with finite variable points, we apply SMT-based approaches for the reachability of hybrid automata [18], [29]. The algorithm is refutationally complete for bounded signals with finite number of variable points. The experimental results show that the proposed algorithm can greatly outperform the previous refutationally complete algorithm [22] (Sec. VII).

Our main contributions can be summarized as follows: (1) We present an efficient technique to obtain small but complete discretization of continuous signals; (2) We present a modular method to encode the bounded STL satisfiability in SMT; (3) We present a new SMT-based STL model checking algorithm that is refutationally complete for bounded signal but much more scalable than the previous algorithm [22]. Due to space limitations, we sometimes include only proof sketches for some lemmas; the complete proofs of the lemmas in the paper can be found in the supplementary material [30].

II. Preliminaries on Signal Temporal Logic

A. Hybrid Automata

Hybrid automata [31] are state machines with continuous variables. Many verification techniques and tools for hybrid systems use hybrid automata as their modeling formalisms [15]–[19], and CPS models can often be translated into hybrid automata for verification [32]–[34]. Therefore, we also use hybrid automata as the modeling formalism in this paper.

In a hybrid automaton H, discrete states are given by a set of *modes* Q, and continuous states are specified by a finite set of real-valued *variables* X. A state of H is a pair $\langle q, \vec{v}\rangle$ of a mode $q \in Q$ and a vector $\vec{v} \in \mathbb{R}^l$, where $|X| = l$. An *initial condition* $init(q, \vec{v})$ defines a set of initial states. An *invariant condition* $inv(q, \vec{v})$ defines a set of valid states. A *jump condition* $jump(\langle q, \vec{v}\rangle, \langle q', \vec{v'}\rangle)$ defines a "discrete" transition between states $\langle q, \vec{v}\rangle \to \langle q', \vec{v'}\rangle$. A *flow condition* $\langle q, \vec{v_t}\rangle = flow(\langle q, \vec{v_0}\rangle, t)$ defines a continuous evolution of X's values from $\vec{v_0}$ to $\vec{v_t}$ over time t in mode q. Consequently, a *hybrid automaton* is $H = (Q, X, init, inv, flow, jump)$.

A *signal* σ represents a continuous execution of a hybrid automaton H, given by a function $\sigma : [0, \tau) \to Q \times \mathbb{R}^l$ over time domain $\mathrm{dom}(\sigma) = [0, \tau)$, $\tau > 0$. The value $\sigma(t)$ denotes the state of H at time t. A signal σ is *bounded* if its domain is bounded (i.e., $\tau < \infty$). A signal σ is a *trajectory* of H, written $\sigma \in H$, if σ describes a valid behavior of H: i.e, as depicted in Fig. 1, there exists a sequence of times $0 = t_0 < t_1 < t_2 < \cdots$ such that: (i) $\sigma(t_0)$ is an initial state; (ii) for each time interval $[t_i, t_{i+1})$, H's state evolves from $\sigma(t_i)$ according to the flow condition, satisfying the invariant condition but not changing the mode; and (iii) at each time point t_i, $i > 0$, a discrete transition takes places by the jump condition.

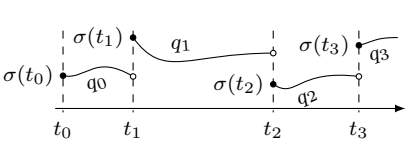

Fig. 1. A trajectory σ of a hybrid automaton

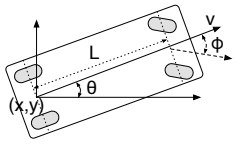

Fig. 2. A simple car

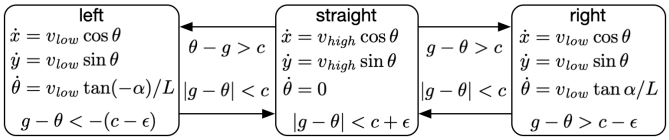

Fig. 3. A hybrid automaton for a simple car

Example 1. *Figure 3 shows a hybrid automaton for a simple autonomous car [22], [35]. The position (x, y) and direction θ depend on its speed v and steering angle ϕ (see Fig. 2). The car dynamics is modeled as the ordinary differential equations:*

$$\dot{x} = v\cos\theta, \qquad \dot{y} = v\sin\theta, \qquad \dot{\theta} = v/L \cdot \tan\phi,$$

*where L is the distance between the front and rear axles [36]. There are three modes (*left*,* straight*, and* right*) of assigning different control values to v and ϕ. Initially, the car can be in any of the three modes. When the difference between θ and a goal direction g is greater than some threshold $c > 0$, the car turns left or right at low speed v_{low} with steering angle α. Otherwise, the car goes straight at high speed v_{high}.*

B. Signal Temporal Logic

Properties of hybrid automata trajectories can be specified in signal temporal logic (STL) [1]. The syntax of STL over a set of state propositions Π is defined by:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi\, \mathbf{U}_I\, \varphi$$

where $p \in \Pi$, and $I \subseteq \mathbb{R}_{\geq 0}$ is an interval of nonnegative real numbers. Other operators can be derived by equivalences: e.g.,

$$\Diamond_I\, \varphi \equiv \top\, \mathbf{U}_I\, \varphi, \qquad \Box_I\, \varphi \equiv \neg\, \Diamond_I\, \neg\varphi.$$

The set of all subformulas of φ is denoted by $\mathrm{sub}(\varphi)$, and the set of propositions occurring in φ is denoted by $\mathrm{prop}(\varphi)$.

The meaning of a proposition p is specified as a function $p : Q \times \mathbb{R}^l \to \mathbb{B}$ assigning to each state $\langle q, \vec{v}\rangle$ of H a Boolean value in $\mathbb{B}$. Examples of state propositions include relational expressions of the form $f(\vec{x}) > 0$ over variables X, where $f : \mathbb{R}^l \to \mathbb{R}$ is a real-valued function. We assume that state propositions are definable in first-order real arithmetic.

Example 2. *For the hybrid automaton in Example 1, consider the following STL formulas:*

- $\Diamond_{[0,30]}(v > 100 \wedge \Box_{[0,20]}\, v > 100)$*: at some time in the first* 30 *seconds, v will go over* $100\,\mathrm{km/h}$ *and stay above* $100\,\mathrm{km/h}$ *for* 20 *seconds [6].*
- $\Box_{[0,\infty)}(\theta > 15 \to (\theta < 20)\, \mathbf{U}_{[2,3)}\, toLeft)$*: whenever θ is greater than* $15°$*, the mode will be left after sometime within* $[2, 3)$*; until then θ is less than* $20°$*.*

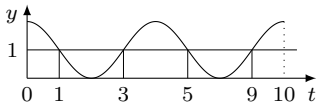

Fig. 4. Signal $y = \cos(\pi t/2) + 1$

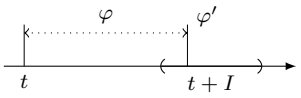

Fig. 5. Satisfaction of $\varphi \, \mathbf{U}_I \, \varphi'$

The complement of an interval I is denoted by $I^{\complement}$ (that is, $I^{\complement} = \mathbb{R} \setminus I$). For two intervals I and J, the Minkowski sum $\{i+j \mid i \in I, j \in J\}$ is denoted by $I+J$, and the Minkowski difference $\{i-j \mid i \in I, j \in J\}$ is denoted by $I-J$. (E.g., $[a,b]+[c,d] = [a+c, b+d]$, and $(a,b)-(c,d) = (a-d, b-c)$.) For a singular interval $\{t\}$, the sets $\{t\}+I$ and $\{t\}-I$ are often written as $t+I$ and $t-I$, respectively. The following property for interval operations is particularly useful for STL.

Lemma 1. *[22] For two intervals $I, K \subseteq \mathbb{R}_{\geq 0}$, $t \in K - I$ iff there exists $t' \geq t$ such that $t' \in K$ and $t' \in t + I$.*

The semantics of STL is defined as the satisfaction relation of a formula φ with respect to a signal σ and a time t. To properly deal with bounded signals, a bound τ is explicitly considered. When $\tau = \infty$, our definition is exactly the same as the standard (unbounded) semantics of STL in [1].

Definition 1. *The* satisfaction *of an STL formula φ at time t over a signal σ up to a bound τ, denoted by $\sigma, t \models_\tau \varphi$, where $\mathrm{dom}(\sigma) \subseteq [0, \tau)$, is defined by:*

- $\sigma, t \models_\tau p$ *iff* $t < \tau$ *and* $p(\sigma(t)) = \top$
- $\sigma, t \models_\tau \neg\varphi$ *iff* $\sigma, t \not\models_\tau \varphi$
- $\sigma, t \models_\tau \varphi \wedge \varphi'$ *iff* $\sigma, t \models_\tau \varphi$ *and* $\sigma, t \models_\tau \varphi'$
- $\sigma, t \models_\tau \varphi \, \mathbf{U}_I \, \varphi'$ *iff* $(\exists t' \geq t)\ t' < \tau, t' \in t + I, \sigma, t' \models_\tau \varphi'$, *and* $(\forall t'' \in [t, t'])\ \sigma, t'' \models_\tau \varphi$

C. Variable Points and Stability

The value of a signal σ changes continuously over time, but the truth of a proposition p changes discontinuously on the signal σ. A time point t is called a variable point for p if the truth value of p changes at time t. For example, consider a signal $y = \cos(\pi t/2) + 1$ over $[0, 10)$ in Fig. 4. There are four variable points $\{1, 3, 5, 9\}$ for proposition $y > 1$.

An STL formula φ is *stable* for a signal σ in an interval J, written $stable^{\sigma}_{J}(\varphi)$, if the truth of φ does not change over J (i.e., for any $u, u' \in J$, $\sigma, u \models_\tau \varphi$ iff $\sigma, u' \models_\tau \varphi$).

Definition 2. *A time t is a* variable point *of a signal σ for φ, if φ is not stable for σ in each open interval $J \subseteq \mathrm{dom}(\sigma)$ with $t \in J$. A time t is a variable point of σ for a set of formulas Γ, if t is a variable point of σ for* some *formula $\varphi \in \Gamma$.*

The behavior of real-time and hybrid systems is associated with sequences of disjoint time intervals [22]–[24], [27]. Let $[N]$ denote the set $\{1, \ldots, N\}$ for a positive integer N.

Definition 3. *A* partition *of an interval D is a family of nonempty disjoint intervals $\mathcal{P} = \{J_i\}_{i \in [N]}$, where $|\mathcal{P}| = N$, such that (i) each interval J_i is either open or singular, (ii) $\bigcup_{i=1}^{N} J_i = D$, and (iii) $\sup(J_i) \leq \inf(J_k)$ for any $i < k$.*

Consider two partitions $\mathcal{P} = \{J_i\}_{i \in [N]}$ and $\mathcal{Q} = \{K_j\}_{j \in [M]}$ of the same interval. $\mathcal{Q}$ is called *finer* than $\mathcal{P}$, written $\mathcal{Q} \sqsubseteq \mathcal{P}$, if each interval $K_j \in \mathcal{Q}$ is a subset of some interval $J_i \in \mathcal{P}$. A finer partition $\mathcal{Q}$ is called a *refinement* of a coarser partition $\mathcal{P}$ [23], [24]. The coarsest partition that is finer than both $\mathcal{P}$ and $\mathcal{Q}$ is denoted by $\mathcal{P} \sqcap \mathcal{Q}$, which always exists [22].

Example 3. *Consider three partitions of the interval $[0, 5)$: $\mathcal{P} = \{\{0\}, (0,4), \{4\}, (4,5)\}$, $\mathcal{Q} = \{\{0\}, (0,1), \{1\}, (1,5)\}$, and $\mathcal{R} = \{\{0\}, (0,1), \{1\}, (1,4), \{4\}, (4,5)\}$. Notice that $e(\mathcal{P}) = \{0, 4, 5\}$, $e(\mathcal{Q}) = \{0, 1, 5\}$, and $e(\mathcal{R}) = \{0, 1, 4, 5\}$. Then, $\mathcal{R}$ is finer than both of $\mathcal{P}$ and $\mathcal{Q}$, and $\mathcal{R} = \mathcal{P} \sqcap \mathcal{Q}$. But neither $\mathcal{P} \sqsubseteq \mathcal{Q}$ nor $\mathcal{Q} \sqsubseteq \mathcal{P}$ holds.*

A formula φ is called *stable* for a signal σ in $\mathcal{P} = \{J_i\}_{i \in [N]}$ (or, $\mathcal{P}$ is stable for φ with respect to σ), written $stable^{\sigma}_{\mathcal{P}}(\varphi)$, if φ is stable for σ in each interval $J_i \in \mathcal{P}$. When $stable^{\sigma}_{\mathcal{P}}(\varphi)$, $\sigma, J_i \models_\tau \varphi$ indicates that φ is true in J_i, and $\sigma, J_i \not\models_\tau \varphi$ indicates that φ is false in J_i. A formula that is stable in $\mathcal{P}$ is stable in a finer partition $\mathcal{Q} \sqsubseteq \mathcal{P}$ [23], [24].

Let $e(I)$ denote the set of endpoints of an interval I; e.g., $e([1,2)) = \{1,2\}$ and $e((1,\infty)) = \{1\}$. Let $e(\mathcal{P})$ denote the set of all endpoints of the intervals in a partition $\mathcal{P}$. Variable points are then related to stable partitions as follows.

Proposition 1. *[23] For a signal σ and a partition $\mathcal{P}$ of an interval $D \subseteq \mathrm{dom}(\sigma)$, $stable^{\sigma}_{\mathcal{P}}(\varphi)$ iff every variable point t of σ in D for φ is an endpoint in $\mathcal{P}$ (i.e., $t \in e(\mathcal{P})$).*

In many real-time temporal logics, the stability of a formula is related to the stability of its subformulas [23], [24]. The cases for negation and conjunction are stated as follows.

Proposition 2. *[23] For a signal σ and a partition $\mathcal{P}$ of an interval $D \subseteq \mathrm{dom}(\sigma)$: (1) $stable^{\sigma}_{\mathcal{P}}(\varphi)$ iff $stable^{\sigma}_{\mathcal{P}}(\neg\varphi)$. (2) If $stable^{\sigma}_{\mathcal{P}}(\varphi)$ and $stable^{\sigma}_{\mathcal{P}}(\varphi')$, then $stable^{\sigma}_{\mathcal{P}}(\varphi \wedge \varphi')$.*

The case of $\varphi \, \mathbf{U}_I \, \varphi'$ is not straightforward. Even if φ and φ' are stable for a signal σ in $\mathcal{P}$, $\varphi \, \mathbf{U}_I \, \varphi'$ need not be stable in $\mathcal{P}$. As depicted in Fig. 5, the satisfaction of $\varphi \, \mathbf{U}_I \, \varphi'$ at time t depends on the satisfaction of φ' in $t+I$ and the satisfaction of φ in $[t, \inf(t+I)]$. There are several methods to build a refinement of $\mathcal{P}$ by taking I into account [22], [27].

Suppose that there is a finite number of variable points of a signal σ for the set of propositions in an STL formula φ. By the above results [22]–[24], there exists a partition $\mathcal{P}$ such that every subformula of φ is stable for σ in $\mathcal{P}$. In this case, we call φ *fully stable* for σ in $\mathcal{P}$. There is also a finite number of variable points of σ for the set of subformulas $\mathrm{sub}(\varphi)$.

III. Overview of Bounded STL Model Checking

A. Problem Statement

The STL model checking problem is to check whether every trajectory of a hybrid automaton H satisfies an STL formula φ. This problem is in general undecidable; the reachability problem of hybrid automata—a special case of STL model checking—is already undecidable [37]. An appropriate notion of bounds is needed to cope with this undecidability.

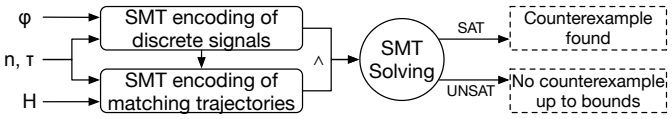

Fig. 6. An SMT-based bounded STL model checking framework

Bounded STL model checking restricts the search for a counterexample of a formula φ to *bounded trajectories* with finite variable points [22]. There are two bound parameters: τ for the time domain, and n for the number of variable points for the set of subformulas $\mathrm{sub}(\varphi)$. That is, the truth of each $\phi \in \mathrm{sub}(\varphi)$ can change at most n times in $[0, \tau)$.

Definition 4 (Bounded STL Model Checking Problem)**.** *An STL formula φ is satisfied at time t in a hybrid automaton H up to bounds $\tau > 0$ and $n \in \mathbb{N}$, denoted by $H, t \models^n_\tau \varphi$, iff $\sigma, t \models_\tau \varphi$ for every trajectory $\sigma \in H$ with at most n variable points with respect to* $\mathrm{sub}(\varphi)$*, where* $\mathrm{dom}(\sigma) = [0, \tau)$.

B. SMT-Based Bounded STL Model Checking

The bounded STL model checking problem can be solved by discretizing signals with full stability [22]. When an STL formula φ is fully stable for a signal σ in a partition $\mathcal{P}$ (i.e., $stable^\sigma_\mathcal{P}(\phi)$ for each $\phi \in \mathrm{sub}(\varphi)$), σ can be *discretized* as the truth values of the subformulas $\mathrm{sub}(\varphi)$ in each interval in $\mathcal{P}$. This observation leads to the concept of discrete signals.

Definition 5. *A* discrete signal *for a formula φ is $\zeta_\varphi = (\mathcal{P}, \theta)$, where $\mathcal{P}$ is a partition and $\theta : \mathrm{sub}(\varphi) \times |\mathcal{P}| \to \mathbb{B}$ assigns a truth value to subformula $\phi \in \mathrm{sub}(\varphi)$ and interval $J_i \in \mathcal{P}$.*

Intuitively, a discrete signal ζ_φ involves an infinite number of "matching" continuous signals. A continuous signal σ *matches* a discrete signal $\zeta_\varphi = (\mathcal{P}, \theta)$, written $\sigma \prec \zeta_\varphi$, if: (i) $\mathcal{P}$ is a partition of $\mathrm{dom}(\sigma)$, (ii) φ is fully stable for σ in $\mathcal{P}$, and (iii) $\theta(\phi, i) = \top$ iff $\sigma, J_i \models \phi$, for each subformula $\phi \in \mathrm{sub}(\varphi)$ and interval $J_i \in \mathcal{P}$.

The bounded STL model checking problem $H, t \models^n_\tau \varphi$ is equivalent to finding a bounded trajectory $\sigma \in H$ that satisfies the negated formula $\neg\varphi$ (i.e., $\sigma, t \models^n_\tau \neg\varphi$). This problem can be decomposed into the problems of finding a discrete signal $\zeta_{\neg\varphi} = (\mathcal{P}, \theta)$, where $\theta(\neg\varphi, i) = \top$ and $t \in J_i$, and finding a bounded trajectory $\sigma \in H$ that matches $\zeta_{\neg\varphi}$ [22].

An SMT-based bounded STL model checking framework, proposed in [22], is shown in Fig. 6. The existence of discrete signals and matching trajectories is encoded in SMT, provided the reachability of H can be encoded in SMT. This procedure is refutationally complete; if a bounded counterexample to φ exists, then it is guaranteed to be found for some bounds.

C. Key Challenge and Our Approach

One of the major challenges in SMT-based bounded STL model checking is to find efficient SMT encodings. Indeed, the previous technique [22] often produces very complex and huge SMT encodings that conventional SMT solvers cannot solve within a reasonable time. The contribution of this paper is to propose much more efficient SMT encodings.

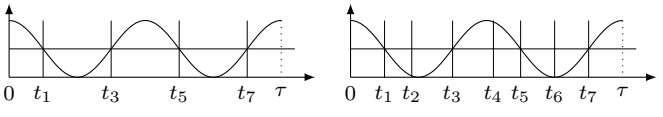

Fig. 7. Two partitions $\mathcal{P}_1$ (left) and $\mathcal{P}_2$ (right), where $\mathcal{P}_1 \sqsupseteq \mathcal{P}_2$

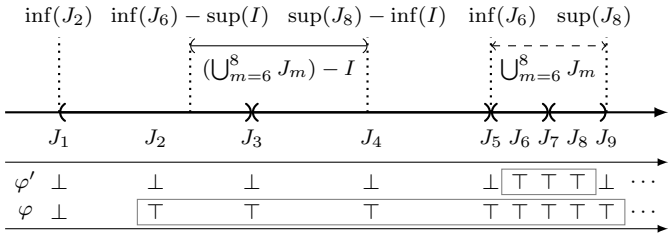

Fig. 8. An STL formula $\varphi\,\mathbf{U}_I\,\varphi'$, and a stable partition $\mathcal{P}$ for φ and φ'

The inefficiency is due to redundant endpoints in discrete signals. E.g., suppose both $\mathcal{P}_1$ and $\mathcal{P}_2$ are fully stable for φ in Fig. 7. The corresponding discrete signals are equivalent in terms of the satisfaction of φ, but $\mathcal{P}_2$ contains redundant endpoints (t_2, t_4, and t_6). A fully stable partition built for φ in [22] usually has many redundant endpoints.

Another problem is that a bound parameter of the previous algorithm [22] is the number of variable points *for the set of propositions* $\mathrm{prop}(\varphi)$. A signal σ with k variable points for $\mathrm{prop}(\varphi)$ can have $O(k \cdot 2^{d(\varphi)})$ variable points for $\mathrm{sub}(\varphi)$ [9], where $d(\varphi)$ is the nesting depth of $\mathbf{U}_I$. Therefore, the previous algorithm [22] may generate unnecessarily large encodings.

We address this challenge in a twofold way. (1) We develop an efficient technique to build fully stable partitions for STL that are much smaller than those produced by the previous approaches [22], [27]. (2) We develop a flexible technique to encode discrete signals; its bound parameter is the number of variable points for the set of subformulas $\mathrm{sub}(\varphi)$, and the size of the resulting encoding is linear in the size of φ.

IV. Partition Construction for STL

This section presents a simple method to build a partition that is fully stable for an STL formula φ. As mentioned, there are a couple of methods that can be adapted to build fully stable partitions for STL formulas [22], [27], but they result in too fine refinements in general. Section IV-A presents a precise condition for the stability of $\varphi\,\mathbf{U}_I\,\varphi'$. Using this condition, Section IV-B explains how to construct a reasonably coarse partition for $\varphi\,\mathbf{U}_I\,\varphi'$. Finally, Section IV-C explains how to build fully stable refinements for general STL formulas.

A. Stability Conditions for $\varphi\,\mathbf{U}_I\,\varphi'$

Figure 8 illustrates the main intuition behind our method. Consider a partition $\mathcal{P} = \{J_i\}_{i \in [N]}$, where both φ and φ' are stable in $\mathcal{P}$. Suppose φ is true in $J_2, \ldots, J_9$, and φ' is true in $J_6, \ldots, J_8$. By definition, $\varphi\,\mathbf{U}_I\,\varphi'$ is true at time t, if there exists $t' \geq t$ such that $t' \in t + I$ and $t' \in \bigcup_{m=6}^{8} J_m$. By Lemma 1, this condition is equivalent to $t \in (\bigcup_{m=6}^{8} J_m) - I$. That is, $\varphi\,\mathbf{U}_I\,\varphi'$ is stably true in $(\bigcup_{m=6}^{8} J_m) - I$.

The important observation from Fig. 8 is that the satisfaction of $\varphi \,\mathbf{U}_I\, \varphi'$ can be expressed using a partition $\mathcal{P}$ for the subformulas φ and φ'. If $\varphi \,\mathbf{U}_I\, \varphi'$ is true at time $t \in J_n$, there exists a subsequent interval J_m, $m \geq n$, such that φ' is true in J_m, and φ is true in all intermediate intervals $J_n, J_{n+1}, \ldots, J_m$. The relation between such indices n and m is denoted by $U_{\sigma,\mathcal{P}}^{\varphi,\varphi'}$ (in short, $U_{\mathcal{P}}$). The following lemma then follows from Lemma 1 and the definition of $U_{\mathcal{P}}$.

Lemma 2. *Suppose φ and φ' are stable for a signal σ in a partition $\mathcal{P} = \{J_i\}_{i\in[N]}$. Let $(n,m) \in U_{\sigma,\mathcal{P}}^{\varphi,\varphi'}$ iff*

$$\sigma, J_m \models_\tau \varphi' \quad \textit{and} \quad \sigma, J_i \models_\tau \varphi \ \textit{for any}\ n \leq i \leq m.$$

For $t \in J_n$, $\sigma, t \models_\tau \varphi \,\mathbf{U}_I\, \varphi'$ iff $\exists (n,m) \in U_{\mathcal{P}}.\ t \in J_m - I$.

The above lemma exactly characterizes the set of time points in which $\varphi \,\mathbf{U}_I\, \varphi'$ is true. This gives a necessary and sufficient condition for $\varphi \,\mathbf{U}_I\, \varphi'$ to be stable in a refinement interval K below, denoted by $nscd_{\sigma,\mathcal{P}}^{\varphi \mathbf{U}_I \varphi'}(K,n)$ (in short, $nscd_{\mathcal{P}}(K,n)$). Let $U_{\mathcal{P}}(n)$ denote the image set of n under the relation $U_{\mathcal{P}}$ (i.e., $U_{\mathcal{P}}(n) = \{m \mid (n,m) \in U_{\mathcal{P}}\}$).

Lemma 3. *Suppose φ and φ' are stable for a signal σ in $\mathcal{P} = \{J_i\}_{i\in[N]}$. Let $nscd_{\sigma,\mathcal{P}}^{\varphi \mathbf{U}_I \varphi'}(K,n)$ denote the condition*

$$K \subseteq \textstyle\bigcup_{m\in U_{\mathcal{P}}(n)} J_m - I \ \ \textit{or}\ \ K \subseteq (\bigcup_{m\in U_{\mathcal{P}}(n)} J_m - I)^{\complement}.$$

For an interval $K \subseteq J_n$, $stable_K^\sigma(\varphi \,\mathbf{U}_I\, \varphi')$ iff $nscd_{\mathcal{P}}(K,n)$.

Proof. ($\Leftarrow$) Let $t, t' \in K$. Suppose $\sigma, t \models_\tau \varphi \,\mathbf{U}_I\, \varphi'$. By Lemma 2, $t \in J_{m_1} - I$ for some $(n, m_1) \in U_{\mathcal{P}}$. This implies $K \subseteq \bigcup_{m\in U_{\mathcal{P}}(n)} J_m - I$. Since $t' \in K$, $t' \in J_{m_2} - I$ for some $m_2 \in U_{\mathcal{P}}(n)$. By Lemma 2, $\sigma, t' \models_\tau \varphi \,\mathbf{U}_I\, \varphi'$.
($\Rightarrow$) Let $t \in \bigcup_{m\in U_{\mathcal{P}}(n)} J_m - I$ and $t' \in (\bigcup_{m\in U_{\mathcal{P}}(n)} J_m - I)^{\complement}$ for $t, t' \in K$. Then, $t \in J_{m_1} - I$ for some $m_1 \in U_{\mathcal{P}}(n)$, but $t' \notin J_{m_2} - I$ for any $m_2 \in U_{\mathcal{P}}(n)$. By Lemma 2, we have $\sigma, t \models_\tau \varphi \,\mathbf{U}_I\, \varphi'$ and $\sigma, t' \not\models_\tau \varphi \,\mathbf{U}_I\, \varphi'$. □

Example 4. *Let $\mathcal{P} = \{\{0\}, (0,3), \{3\}, (3,6), \{6\}, (6,8)\}$. Consider a formula $p \,\mathbf{U}_{(1,3)}\, q$, and a signal σ such that:*

i	1	2	3	4	5	6
J_i	$\{0\}$	$(0,3)$	$\{3\}$	$(3,6)$	$\{6\}$	$(6,8)$
p	$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$
q	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\top$

For J_2, observe $\bigcup_{m\in U_{\mathcal{P}}(2)}(J_m - I) = (-3,2) \cup (3,7)$, where $U_{\mathcal{P}}(2) = \{2,3,6\}$. Therefore, $p \,\mathbf{U}_{(1,3)}\, q$ is stably true in the subinterval $(0,2) \subseteq J_2$, and stably false in $[2,3] \subseteq J_2$.

B. Stable Refinements for $\varphi \,\mathbf{U}_I\, \varphi'$

Variable points in a stable partition $\mathcal{P}$ involve the rising and falling edges of a Boolean truth signal for φ [28]. An interval $J_i \in \mathcal{P}$ has a *rising edge*, written $i \in \uparrow_{\mathcal{P}} \varphi$, if φ is true in J_i but false in J_{i-1} (or there is no previous interval). Similarly, $J_i \in \mathcal{P}$ has a *falling edge*, written $i \in \downarrow_{\mathcal{P}} \varphi$, if φ is true in J_i but false in J_{i+1} (or there is no next interval).

Definition 6. *Given a partition $\mathcal{P} = \{J_i\}_{i\in[N]}$ and $i \in [N]$: (1) $i \in \uparrow_{\mathcal{P}} \varphi$ iff $\sigma, J_i \models_\tau \varphi$, and $i = 1$ or $\sigma, J_{i-1} \not\models_\tau \varphi$; and (2) $i \in \downarrow_{\mathcal{P}} \varphi$ iff $\sigma, J_i \models_\tau \varphi$, and $i = N$ or $\sigma, J_{i+1} \not\models_\tau \varphi$.*

Consider Fig. 8. Since $\{6,7,8\} \subseteq U_{\mathcal{P}}(n)$, $2 \leq n \leq 4$, by Lemma 2, if $6 \leq m \leq 8$ and $t \in J_n$, then $t \in J_m - I$ implies $\sigma, t \models_\tau \varphi \,\mathbf{U}_I\, \varphi'$. Hence, $(\bigcup_{m=6}^{8} J_m) - I$ does not contain variable points. For $\bigcup_{m=6}^{8} J_m$, its endpoints are variable points for $\{\varphi, \varphi'\}$, the left endpoint $\inf(J_6)$ is a rising edge for φ', and the right endpoint $\sup(J_8)$ is a falling edge for φ'. Accordingly, the left and right endpoints of $(\bigcup_{6\leq m\leq 8} J_m) - I$ are, respectively, $\inf(J_6) - \sup(I)$ and $\sup(J_8) - \inf(I)$. This suggests the definition of variable point candidates.

Definition 7. *For a formula $\varphi \,\mathbf{U}_I\, \varphi'$, a partition $\mathcal{P}$, and a signal σ, where φ and φ' are stable for σ in $\mathcal{P}$, the set of* variable point candidates *$C_{\sigma,\mathcal{P}}^{\varphi \mathbf{U}_I \varphi'} \subseteq \mathbb{R}_{\geq 0}$ (in short, $C_{\mathcal{P}}^{\mathbf{U}}$) is:*

$$\{v \mid m \in \uparrow_{\mathcal{P}} \varphi \cup \uparrow_{\mathcal{P}} \varphi', v = \inf(J_m) - \sup(I), cd(m,v)\}$$
$$\cup \{v \mid m \in \downarrow_{\mathcal{P}} \varphi \cup \downarrow_{\mathcal{P}} \varphi', v = \sup(J_m) - \inf(I), cd(m,v)\},$$

where $cd(m,v) \equiv \exists n.\ (n,m) \in U_{\mathcal{P}} \ \wedge\ \inf(J_n) \leq v$. The until-refinement *of a partition $\mathcal{P}$ for $\varphi \,\mathbf{U}_I\, \varphi'$, denoted by $\mathcal{R}^\sigma_{\varphi \mathbf{U}_I \varphi'}[\mathcal{P}]$ (in short, $\mathcal{R}_{\mathbf{U}}[\mathcal{P}]$), is $\mathcal{R}_{\mathbf{U}}[\mathcal{P}] = \mathcal{P} \sqcap \mathcal{Q}$, where $\mathcal{Q}$ is the partition with the endpoints $C_{\mathcal{P}}^{\mathbf{U}} \cup \{0, \tau\}$.*

Not all endpoints of $\mathcal{P}$ are considered for finding variable point candidates in $C_{\mathcal{P}}^{\mathbf{U}}$; only variable points for $\{\varphi, \varphi'\}$ are needed. E.g., in the above example, neither $\inf(J_7) - \sup(I)$ nor $\sup(J_7) - \inf(I)$ is a candidate in $C_{\mathcal{P}}^{\mathbf{U}}$. If $e(C_{\mathcal{P}}^{\mathbf{U}}) \subseteq e(\mathcal{P})$, then $\mathcal{P}$'s until-refinement is just $\mathcal{P}$. This motivates the concept of maximally stable partitions with respect to a signal σ.

Definition 8. *A partition $\mathcal{P}$ is* maximally stable *for φ, if any partition that is strictly coarser than $\mathcal{P}$ is not stable for φ.*

It follows from Proposition 1 that for a stable partition $\mathcal{P}$, there exists a unique maximally stable partition, denoted by $\max_\varphi^\sigma(\mathcal{P})$, that is coarser than $\mathcal{P}$. Simply put, $\max_\varphi^\sigma(\mathcal{P})$ is the partition with variable points as the only endpoints.

Lemma 4. *Suppose φ is stable for a signal σ in a partition of domain D. Let $\mathcal{O}$ be the partition such that $e(\mathcal{O}) \setminus e(D)$ is the set of variable points for φ in D. Then, $\max_\varphi^\sigma(\mathcal{P}) = \mathcal{O}$.*

A variable point for $\{\varphi, \varphi'\}$ in $\mathcal{P}$ is either an endpoint of $\max_\varphi^\sigma(\mathcal{P})$ or an endpoint of $\max_{\varphi'}^\sigma(\mathcal{P})$. The partition $\max_\varphi^\sigma(\mathcal{P}) \sqcap \max_{\varphi'}^\sigma(\mathcal{P})$ is the coarsest partition that contains all these variable points. Indeed, variable point candidates $C_{\mathcal{P}}^{\mathbf{U}}$ can be obtained using $\max_\varphi^\sigma(\mathcal{P}) \sqcap \max_{\varphi'}^\sigma(\mathcal{P})$ as follows.

Lemma 5. *Given $\varphi \,\mathbf{U}_I\, \varphi'$, suppose φ and φ' are stable for σ in a partition $\mathcal{P}$ of $[0,\tau)$. let $\mathcal{O}$ be $\max_\varphi^\sigma(\mathcal{P}) \sqcap \max_{\varphi'}^\sigma(\mathcal{P})$, where $\mathcal{O} = \{L_i\}_{i\in[N]}$. For any $(n,m) \in U_{\mathcal{O}}$, we have:*

$$\inf(L_n) \leq \inf(L_m) - \sup(I) \implies \inf(L_m) - \sup(I) \in C_{\mathcal{P}}^{\mathbf{U}}$$
$$\inf(L_n) \leq \sup(L_m) - \inf(I) \implies \sup(L_m) - \inf(I) \in C_{\mathcal{P}}^{\mathbf{U}}$$

Example 5. *In Fig. 8, $\mathcal{O} = \max_\varphi^\sigma(\mathcal{P}) \sqcap \max_{\varphi'}^\sigma(\mathcal{P})$ contains intervals $L_1 = J_1$, $L_2 = \bigcup_{m=2}^{5} J_m$, and $L_3 = \bigcup_{m=6}^{8} J_m$. The left endpoint of L_2 is $\inf(J_2)$, and the endpoints of $L_3 - I$ are $\inf(J_6) - \sup(I)$ and $\sup(J_8) - \inf(I)$. Since $(2,3) \in U_{\mathcal{O}}$ and $\inf(J_2) < \inf(J_6) - \sup(I), \sup(J_8) - \inf(I)$, by Lemma 5, $\inf(J_6) - \sup(I)$ and $\sup(J_8) - \inf(I) \in e(\mathcal{R}_{\mathbf{U}}[\mathcal{P}])$.*

The stability of a formula $\varphi \,\mathbf{U}_I\, \varphi'$ in the until-refinement $\mathcal{R}_{\mathbf{U}}[\mathcal{P}]$ follows from Lemma 3 and the lemma below.

Lemma 6. *Let $\mathcal{O} = \max^{\sigma}_{\varphi}(\mathcal{P}) \sqcap \max^{\sigma}_{\varphi'}(\mathcal{P})$, where φ and φ' are stable for σ in $\mathcal{P}$. For any intervals $K_j \in \mathcal{R}_{\mathbf{U}}[\mathcal{P}]$ and $L_k \in \mathcal{O}$, $K_j \subseteq L_k$ implies $nscd_{\mathcal{O}}(K_j, k)$.*

Proof Sketch. Let $\mathcal{O} = \{L_k\}_{k\in[N]}$. It follows from Lemma 5 that for *each* $K_j \in \mathcal{R}_{\mathbf{U}}[\mathcal{P}]$, if $K_j \subseteq L_n$ and $(n, m) \in U_{\mathcal{O}}$, then either $K_j \subseteq L_m - I$ or $K_j \subseteq (L_m - I)^{\complement}$ holds. Then, for $K_j \in \mathcal{R}_{\mathbf{U}}[\mathcal{P}]$ and $L_k \in \mathcal{O}$, where $K_j \subseteq L_k$, by the above claim, we can easily see that $nscd_{\mathcal{O}}(K_j, n)$ holds. □

C. Fully Stable Refinements for STL

We present how to obtain a fully stable refinement for a formula φ, given a partition that is stable for each proposition in $\mathrm{prop}(\varphi)$. We inductively build a fully stable refinement for each subformula. For the negation and conjunction cases, we apply the existing methods [23], [24] (Proposition 2), and for the $\mathbf{U}_I$ case, we apply the until-refinement in Def. 7.

Definition 9. *For an STL formula φ, a signal σ, and a partition $\mathcal{P}$, where each proposition $p \in \mathrm{prop}(\varphi)$ is stable for σ in $\mathcal{P}$, the* φ-refinement $\mathcal{R}^{\sigma}_{\mathcal{P}}(\varphi)$ *is defined by:*

$$\mathcal{R}^{\sigma}_{\mathcal{P}}(p) = \mathcal{P} \qquad \mathcal{R}^{\sigma}_{\mathcal{P}}(\phi \wedge \phi') = \mathcal{R}^{\sigma}_{\mathcal{P}}(\phi) \sqcap \mathcal{R}^{\sigma}_{\mathcal{P}}(\phi')$$
$$\mathcal{R}^{\sigma}_{\mathcal{P}}(\neg\phi) = \mathcal{R}^{\sigma}_{\mathcal{P}}(\phi) \quad \mathcal{R}^{\sigma}_{\mathcal{P}}(\phi \,\mathbf{U}_I\, \phi') = \mathcal{R}_{\mathbf{U}}[\mathcal{R}^{\sigma}_{\mathcal{P}}(\phi) \sqcap \mathcal{R}^{\sigma}_{\mathcal{P}}(\phi')]$$

Theorem 1. *An STL formula φ is fully stable for a signal σ in $\mathcal{R}^{\sigma}_{\mathcal{P}}(\varphi)$, where each $p \in \mathrm{prop}(\varphi)$ is stable for σ in $\mathcal{P}$.*

Proof. We proceed by structural induction on φ. The cases of $\varphi = p$ and $\varphi = \neg\phi$ immediately follow from the definitions, induction hypothesis, and Proposition 2.

- ($\varphi = \phi \wedge \phi'$): Let $\mathcal{Q} = \mathcal{R}^{\sigma}_{\mathcal{P}}(\phi)$ and $\mathcal{Q}' = \mathcal{R}^{\sigma}_{\mathcal{P}}(\phi')$. By induction hypothesis, $stable^{\sigma}_{\mathcal{Q}}(\phi)$ and $stable^{\sigma}_{\mathcal{Q}'}(\phi')$ hold. By the monotonicity of stability with respect to $\sqsupseteq$, $stable^{\sigma}_{\mathcal{Q}\sqcap\mathcal{Q}'}(\phi)$ and $stable^{\sigma}_{\mathcal{Q}\sqcap\mathcal{Q}'}(\phi')$. By Proposition 2, $stable^{\sigma}_{\mathcal{Q}\sqcap\mathcal{Q}'}(\phi \wedge \phi')$.
- ($\varphi = \phi \,\mathbf{U}_I\, \phi'$): Let $\mathcal{Q} = \mathcal{R}^{\sigma}_{\mathcal{P}}(\phi) \sqcap \mathcal{R}^{\sigma}_{\mathcal{P}}(\phi')$. As shown in the above case, ϕ and ϕ' are fully stable for σ in $\mathcal{Q}$. Since $\mathcal{R}_{\mathbf{U}}[\mathcal{Q}] \sqsubseteq \mathcal{Q}$, ϕ and ϕ' are also fully stable for σ in $\mathcal{R}_{\mathbf{U}}[\mathcal{Q}]$. Let $\mathcal{O} = \max^{\sigma}_{\varphi}(\mathcal{Q}) \sqcap \max^{\sigma}_{\varphi'}(\mathcal{Q})$. Consider any interval K_j in $\mathcal{R}_{\mathbf{U}}[\mathcal{Q}]$. Since $\mathcal{R}_{\mathbf{U}}[\mathcal{Q}] \sqsubseteq \mathcal{O}$, there exists an interval L_k in $\mathcal{O}$ such that $K_j \subseteq L_k$. By Lemma 6, $nscd_{\mathcal{O}}(K_j, k)$ holds. Thus, by Lemma 3, $\varphi \,\mathbf{U}_I\, \varphi'$ is stable for σ in K_j. As a consequence, $\varphi \,\mathbf{U}_I\, \varphi'$ is fully stable for σ in $\mathcal{R}_{\mathbf{U}}[\mathcal{Q}]$. □

Example 6. *Let $\mathcal{P} = \{\{0\}, (0,6), \{6\}, (6,8)\}$. Consider an STL formula $\varphi = \Diamond_{[1,\infty)}(p \,\mathbf{U}_{(1,3)}\, q)$ and a signal σ such that:*

$\mathcal{P}$	$\{0\}$	$(0,6)$	$\{6\}$	$(6,8)$
p	$\top$	$\top$	$\top$	$\top$
q	$\bot$	$\bot$	$\top$	$\top$

Since $\{6\} \in \mathcal{P}$ has a rising edge and $(6,8) \in \mathcal{P}$ has a falling edge, $C^{p\,\mathbf{U}_{(1,3)}\,q}_{\sigma,\mathcal{P}}$ is $\{3,7\}$. Thus, $\mathcal{R}^{\sigma}_{p\,\mathbf{U}_{(1,3)}\,q}[\mathcal{P}] = \mathcal{Q}$, where, $\mathcal{Q} = \{\{0\}, (0,3), \{3\}, (3,6), \{6\}, (6,7), \{7\}, (7,8)\}$. Then:

$\mathcal{Q}$	$\{0\}$	$(0,3)$	$\{3\}$	$(3,6)$	$\{6\}$	$(6,7)$	$\{7\}$	$(7,8)$
$p\,\mathbf{U}_{(1,3)}\,q$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\top$	$\bot$	$\bot$
φ	$\top$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$

Similarly, since $(3,6) \in \mathcal{Q}$ has a rising edge and $(6,7) \in \mathcal{Q}$ has a falling edge, $C^{\sigma,\mathcal{Q}}_{\varphi} = \{6\}$, and therefore $\mathcal{R}^{\sigma}_{\varphi}[\mathcal{Q}] = \mathcal{Q}$. Consequently, $\mathcal{R}^{\sigma}_{\mathcal{P}}(\varphi)$, the φ-refinement of $\mathcal{P}$, is $\mathcal{Q}$.

The size of the until-refinement $\mathcal{R}^{\sigma}_{\phi\,\mathbf{U}_I\,\phi'}[\mathcal{P}]$ is at most $3N$ for a partition $\mathcal{P} = \{J_i\}_{i\in[N]}$, because each interval $J_i - I$ can introduce two fresh endpoints. Example 7 below exhibits the worst case. As a result, the size of the φ-refinement $\mathcal{R}^{\sigma}_{\mathcal{P}}(\varphi)$ is at most $O(k \cdot t(\varphi) \cdot 2^{d(\varphi)})$, where $t(\varphi)$ is the number of $\mathbf{U}_I$ and $d(\varphi)$ is the nesting depth of $\mathbf{U}_I$, given k the size of a base partition for propositions.

Example 7. *Consider $\varphi \,\mathbf{U}_{(0.1,0.2)}\, \varphi'$ and $\mathcal{P} = \{J_i\}_{i\in[2n]}$. Let $J_{2k-1} = \{k-1\}$, $J_{2k} = (k-1, k)$, and for $1 \le k \le n$: $\sigma, J_{2k-1} \cup J_{2k} \models_\tau \varphi$, $\sigma, J_{2k-1} \not\models_\tau \varphi'$, and $\sigma, J_{2k} \models_\tau \varphi'$. Each time point k, $1 \le k \le n$, is then a variable point, and $(2k-1, 2k) \in U_{\mathcal{P}}$. Therefore, $C^{\mathbf{U}}_{\mathcal{P}} = \bigcup_{k=1}^{n}\{k - 0.1, k - 0.2\}$, where all endpoints of $\mathcal{P}$ contributes to the size of $C^{\mathbf{U}}_{\mathcal{P}}$. Because $C^{\mathbf{U}}_{\mathcal{P}} \cap e(\mathcal{P}) = \emptyset$, $|e(\mathcal{R}_{\mathbf{U}}[\mathcal{P}])| = 3n$.*

The work [9] gave an example of an STL formula φ where a signal has $O(2^{d(\varphi)})$ variable points for the set of subformulas $\mathrm{sub}(\varphi)$. The worst-case exponential size of $\mathcal{R}^{\sigma}_{\mathcal{P}}(\varphi)$ is not avoidable. The refinement construction in [22] also has the same upper bound $O(k \cdot t(\varphi) \cdot 2^{d(\varphi)})$. But our φ-refinement minimizes "redundant" variable points to reduce the partition size, as mentioned in Sec. III-C, whereas the existing methods [22], [27] do not apply such a reduction.

V. Encoding of Bounded Satisfiability for STL

The bounded STL satisfiability problem is to check whether there exists a bounded signal that satisfies φ and has a finite number of variable points. This section presents a symbolic method to encode the bounded satisfaction of STL in a decidable fragment of first-order logic using Theorem 1. We symbolically represent discrete signals using logical variables (Sec. V-A), and identify constraints on those variables imposed by the STL semantics for a discrete signal (Sec. V-B) and by the full stability of a partition (Sec. V-C).

A. Symbolic Representation with Full Stability

For an STL formula φ, we consider the problem of finding a discrete signal $\zeta_\varphi = (\mathcal{P}, \theta)$ with at least one matching signal (see Sec. III-B). We symbolically encode this problem as a first-order formula so that there exists such a discrete signal ζ_φ iff the encoding is satisfiable. Discrete signals are symbolically represented using first-order variables as follows.

Definition 10. *For an STL formula φ and $n \in \mathbb{N}$, a* symbolic discrete signal *is $\varsigma_{\varphi,n} = (\{\gamma_j\}_{j\in[n-1]}, \{\chi^i_\phi\}_{i\in[2n],\phi\in\mathrm{sub}(\varphi)})$, where each γ_j is a real variable and χ^i_ϕ is a Boolean variable.*

Intuitively, $\varsigma_{\varphi,n}$ involves a symbolic partition $\{\mathfrak{J}_i\}_{i\in[2n]}$ such that $\mathfrak{J}_{2j-1} = \{\gamma_{j-1}\}$ and $\mathfrak{J}_{2j} = (\gamma_{j-1}, \gamma_j)$ for each $j \in [n]$, where $\gamma_0 = 0$ and $\gamma_n = \tau$. The variables for the left and right endpoints of $\mathfrak{J}_i$ are denoted by $l(\mathfrak{J}_i)$ and $r(\mathfrak{J}_i)$, respectively. Each Boolean variable χ^i_ϕ, represents the truth value of subformula ϕ in symbolic interval $\mathfrak{J}_i$.

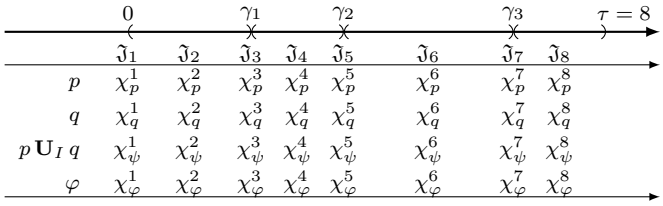

Fig. 9. A symbolic discrete signal $\varsigma_{\varphi,4}$ for $\varphi = \Diamond_J(p\,\mathbf{U}_I\,q)$

An assignment α of the variables in a symbolic discrete signal $\varsigma_{\varphi,n}$ gives a concrete discrete signal $\alpha(\varsigma_{\varphi,n}) = (\mathcal{P}, \theta)$, called an *instance* of $\varsigma_{\varphi,n}$, where $\mathcal{P} = \{\alpha(\mathfrak{J}_i)\}_{i\in[2n]}$ is a concrete partition, and θ is a concrete truth mapping with $\theta(\phi, i) = \alpha(\chi_\phi^i)$ for each $\phi \in \mathrm{sub}(\varphi)$ and $i \in [2n]$.

Example 8. *Figure 9 illustrates a symbolic discrete signal $\varsigma_{\varphi,4}$ for $\varphi = \Diamond_J(p\,\mathbf{U}_I\,q)$ and $n = 4$. The truth of each subformula ϕ in the i-th symbolic interval $\mathfrak{J}_i$, $1 \le i \le 8$, is represented as Boolean variable χ_ϕ^i. For an assignment α with $\gamma_1 \mapsto 1$ and $\gamma_2 \mapsto 3$, $\alpha(\mathfrak{J}_3) = \{1\}$ and $\alpha(\mathfrak{J}_4) = (1, 3)$.*

B. Encoding of Discrete Signals for STL

In STL, the truth of a non-atomic formula φ is determined by the truth values of φ's subformulas. For a symbolic discrete signal $\varsigma_{\varphi,n}$, variable χ_φ^i for φ and variables for its subformulas must satisfy certain constraints imposed by the semantics of STL. We can easily see the constraints $\chi_{\neg\varphi}^i = \neg\chi_\varphi^i$ for negation and $\chi_{\varphi\wedge\varphi'}^i = \chi_\varphi^i \wedge \chi_{\varphi'}^i$ for conjunction.

To identify constraints for $\varphi\,\mathbf{U}_I\,\varphi'$, we need to obtain an equivalent formula, where the satisfaction of each subformula depends only on a single interval in a partition $\mathcal{P}$. For this purpose, we consider a "global-time" temporal operator $\mathbf{U}_I^K$ with an extra global time interval K. The satisfaction of $\varphi\,\mathbf{U}_I^K\,\varphi'$ is bounded by the global interval K as follows.

Definition 11. *[22] For a signal σ and a time point t, the satisfaction of $\varphi\,\mathbf{U}_I^K\,\varphi'$ is defined by:*

$$\sigma, t \models_\tau \varphi\,\mathbf{U}_I^K\,\varphi' \iff (\exists t' \ge t)\ t' \in [0,\tau) \cap K,\ t' \in t + I,\ \sigma, t' \models_\tau \varphi',\ (\forall t'' \in [t, t'] \cap K)\ \sigma, t'' \models_\tau \varphi$$

Using global time constraints, we can apply the *syntactic separation* method [22] to syntactically decompose a formula $\varphi\,\mathbf{U}_I^K\,\varphi'$ into a Boolean combination of the subformulas φ and φ' that depend only on disjoint segments of the global interval K as follows. We can repeatedly apply this procedure to obtain a formula separated at multiple time points.

Proposition 3. *[22] For two successive intervals K_1 and K_2, $\varphi\,\mathbf{U}_I^{K_1\cup K_2}\,\varphi' \equiv \varphi\,\mathbf{U}_I^{K_1}\,\varphi' \vee (\Box_{[0,\infty)}^{K_1}\,\varphi \wedge \varphi\,\mathbf{U}_I^{K_2}\,\varphi')$, where $K_1 \cup K_2$ is an interval, $K_1 \cap K_2 = \emptyset$, and $\sup(K_1) = \inf(K_2)$.*

Now consider a partition $\mathcal{P} = \{J_i\}_{i\in[N]}$ of domain $[0,\tau)$, where φ and φ' are stable for a signal σ in $\mathcal{P}$. Let $v_i \in J_i$, for each $i \in [N]$, be a time sample from the i-th interval J_i, and $L_n = \bigcup_{n\le j\le N} J_n$, for each $n \in [N]$, be the n-th suffix interval. We then have the following separation law.

Lemma 7. *For any indices $i, n \in [N]$ with $1 \le i \le n \le N$: $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{L_n}\,\varphi'$ iff ($v_i \in J_n - I$ and $\sigma, v_n \models_\tau \varphi \wedge \varphi'$) or ($\sigma, v_n \models_\tau \varphi$ and $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{L_{n+1}}\,\varphi'$).*

Proof Sketch. By Proposition 3, $\varphi\,\mathbf{U}_I^{L_n}\,\varphi'$ is equivalent to $\varphi\,\mathbf{U}_I^{J_n}\,\varphi' \vee (\Box_{[0,\infty)}^{J_n}\,\varphi \wedge \varphi\,\mathbf{U}_I^{L_{n+1}}\,\varphi')$. By Lemma 1 and the stability of J_n, (1) $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{J_n}\,\varphi'$ iff $v_i \in J_n - I$ and $\sigma, v_n \models_\tau \varphi\wedge\varphi'$, and (2) $\sigma, v_i \models_\tau \Box_{[0,\infty)}^{J_n}\,\varphi$ iff $\sigma, v_n \models_\tau \varphi$. □

The truth values of φ and φ' at a time sample $v_i \in J_i$ represent the truth values of φ and φ' in the entire interval J_i, provided φ and φ' are stable for a signal σ in J_i. Since $\sup(L_i) = \tau$, $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I\,\varphi'$ iff $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{L_i}\,\varphi'$. By repeatedly applying the above separation law, the satisfaction of $\varphi\,\mathbf{U}_I^{L_i}\,\varphi'$ at $v_i \in J_i$ can be expressed using the satisfaction of φ and φ' at time samples $v_i, v_{i+1}, \ldots, v_N$.

This allows us to define the constraints between a variable $\chi_{\varphi\,\mathbf{U}_I\,\varphi'}^i$ and variables χ_φ^n and $\chi_{\varphi'}^n$, $1 \le i \le n \le N$. Let a term u_i represent a time sample from symbolic interval $\mathfrak{J}_i$. A constraint $\rho_{\phi\,\mathbf{U}_I\,\phi'}^i(n)$—which corresponds to the satisfaction of $\phi\,\mathbf{U}_I^{L_n}\,\phi'$ at time sample u_i—is then inductively defined as follows. As explained above, $\rho_{\phi\,\mathbf{U}_I\,\phi'}^i(i)$ corresponds to the satisfaction of $\varphi\,\mathbf{U}_I\,\varphi'$ at time sample u_i.

Definition 12. *For $1 \le i \le n \le N$, $u_i = (l(\mathfrak{J}_i) + r(\mathfrak{J}_i))/2$, $\rho_{\phi\,\mathbf{U}_I\,\phi'}^i(N+1) = \bot$, and $\rho_{\phi\,\mathbf{U}_I\,\phi'}^i(n)$ is the first-order formula*

$$(u_i \in \mathfrak{J}_n - I \wedge \chi_\phi^n \wedge \chi_{\phi'}^n) \vee (\chi_\phi^n \wedge \rho_{\phi\,\mathbf{U}_I\,\phi'}^i(n+1)).$$

The resulting constraint for a symbolic discrete signal $\varsigma_{\varphi,n}$ is a quantifier-free first-order formula $\Psi_{\mathsf{STL}}[\varsigma_{\varphi,n}]$ below. We denote by $\alpha(\Psi_{\mathsf{STL}}[\varsigma_{\varphi,n}])$, for an assignment α, the formula obtained by replacing each variable x by $\alpha(x)$.

Definition 13. *For $\varsigma_{\varphi,n} = (\{\gamma_j\}_{j\in[n-1]}, \{\chi_\phi^i\}_{i\in[2n],\phi\in\mathrm{sub}(\varphi)})$, $\Psi_{\mathsf{STL}}[\varsigma_{\varphi,n}]$ is the conjunction of the following equalities:*

$$\chi_{\neg\phi}^i = \neg\chi_\phi^i, \quad \chi_{\phi\wedge\phi'}^i = \chi_\phi^i \wedge \chi_{\phi'}^i, \quad \chi_{\phi\,\mathbf{U}_I\,\phi'}^i = \rho_{\phi\,\mathbf{U}_I\,\phi'}^i(i)$$

Example 9. *Consider $\varsigma_{\varphi,3}$ with $\mathfrak{J}_4 = (\gamma_1, \gamma_2)$, $\mathfrak{J}_5 = \{\gamma_2\}$, and $\mathfrak{J}_6 = (\gamma_2, \tau)$. Then, $\chi_{p\,\mathbf{U}_I\,q}^4 = \rho_{p\,\mathbf{U}_I\,q}^4(4)$, $\rho_{p\,\mathbf{U}_I\,q}^4(7) = \bot$, and $\rho_{p\,\mathbf{U}_I\,q}^4(n)$, for $4 \le n < 7$, is the quantifier-free formula*

$$((\gamma_1 + \gamma_2)/2 \in \mathfrak{J}_n - I \wedge \chi_p^n \wedge \chi_q^n) \vee (\chi_p^n \wedge \rho_{p\,\mathbf{U}_I\,q}^4(n+1)).$$

Consider an assignment α for a symbolic discrete signal $\varsigma_{\varphi,n}$. Suppose that the STL formula φ is fully stable for a signal σ in the corresponding partition $\{\alpha(\mathfrak{J}_i)\}_{i\in[2n]}$, where the signal σ "matches" the assignment α for each proposition $p \in \mathrm{prop}(\varphi)$ (i.e., $p(\sigma(t)) = \alpha(\chi_p^i)$ for any $t \in \alpha(\mathfrak{J}_i)$ in each interval $\alpha(\mathfrak{J}_i)$). Then, $\Psi_{\mathsf{STL}}[\varsigma_{\varphi,n}]$ exactly captures the constraints by the semantics of STL as follows.

Lemma 8. *Suppose that φ is fully stable for a signal σ in $\{\alpha(\mathfrak{J}_i)\}_{i\in[2n]}$, and $(\forall t \in \alpha(\mathfrak{J}_i))\ p(\sigma(t)) = \alpha(\chi_p^i)$ for each $p \in \mathrm{prop}(\varphi)$. The following are equivalent: (i) $\alpha(\Psi_{\mathsf{STL}}[\varsigma_{\varphi,n}])$ is true. (ii) $\alpha(\chi_\phi^i) = \top$ iff $\sigma, \alpha(\mathfrak{J}_i) \models_\tau \phi$ for each $\phi \in \mathrm{sub}(\varphi)$.*

Proof Sketch. The proof is by structural induction on φ. The cases of p, $\neg\phi$, and $\phi \wedge \phi'$ follow from the definitions, conditions $\chi_{\neg\phi}^i = \neg\chi_\phi^i$ and $\chi_{\phi\wedge\phi'}^i = \chi_\phi^i \wedge \chi_{\phi'}^i$, and induction

$$
\begin{aligned}
&cand_{\phi\,\mathbf{U}_I\,\phi'} \equiv \bigwedge_{m=1}^{2n} \begin{bmatrix} [(\uparrow_\phi^m \vee \uparrow_{\phi'}^m) \to cd_{\phi,\phi'}^m(l(\mathfrak{J}_m) - \sup(I))] \\ \wedge [(\downarrow_\phi^m \vee \downarrow_{\phi'}^m) \to cd_{\phi,\phi'}^m(r(\mathfrak{J}_m) - \inf(I))] \end{bmatrix} \\
&cd_{\phi,\phi'}^m(v) \equiv \textstyle\bigwedge_{n=1}^m ((U_{\phi,\phi'}^{n,m} \wedge l(\mathfrak{J}_n) \le v) \to endp_{n,m}^v) \\
&\quad U_{\phi,\phi'}^{n,m} \equiv \chi_\phi^n \wedge U_{\phi,\phi'}^{n+1,m} \qquad \uparrow_\phi^m \equiv \chi_\phi^m \wedge \neg\chi_\phi^{m-1} \\
&\quad U_{\phi,\phi'}^{m,m} \equiv \chi_\phi^m \wedge \chi_{\phi'}^m \qquad \uparrow_\phi^1 \equiv \chi_\phi^1 \\
&\quad endp_{n,m}^v \equiv endp_{n,n}^v \vee endp_{n+1,m}^v \qquad \downarrow_\phi^m \equiv \chi_\phi^m \wedge \neg\chi_\phi^{m+1} \\
&\quad endp_{m,m}^v \equiv v = l(\mathfrak{J}_m) \vee v = r(\mathfrak{J}_m) \qquad \downarrow_\phi^{2n} \equiv \chi_\phi^{2n}
\end{aligned}
$$

Fig. 10. Auxiliary predicates for $\Psi_{\mathsf{PAR}}[\varsigma_{\varphi,n}]$

hypothesis. The case of $\phi\,\mathbf{U}_I\,\phi'$ follows from the full stability of $\{\alpha(\mathfrak{J}_i)\}_{i\in[2n]}$, induction hypothesis, and Lemma 7. □

C. Encoding of Fully Stable Partitions

Figure 10 shows several auxiliary predicates to encode the "building blocks" of the until-refinement condition in Def. 7. The predicates $U_{\phi,\phi'}^{n,m}$, $\uparrow_\phi^m$, and $\downarrow_\phi^m$ represent $(n,m) \in U_\mathcal{P}$, $m \in \uparrow_\mathcal{P} \phi$, and $m \in \downarrow_\mathcal{P} \phi$, respectively (cf. Lemma 2 and Def. 6). The predicate $endp_{n,m}^v$ represents that v is an endpoint of one of $\mathfrak{J}_i$, $n \le i \le m$, and $cand_{\phi\,\mathbf{U}_I\,\phi'}$ represents that all candidates in $C_\mathcal{P}^\mathbf{U}$ are endpoints in $\{\mathfrak{J}_i\}_{i\in[2n]}$.

The φ-refinement condition in Def. 9 can then be encoded as a quantifier-free formula using those auxiliary predicates:

Definition 14. *For an STL formula φ and a symbolic discrete signal $\varsigma_{\varphi,n}$, $\Psi_{\mathsf{PAR}}[\varsigma_{\varphi,n}]$ is the conjunction of the conditions $cand_{\phi\,\mathbf{U}_I\,\phi'}$ for each subformula $\phi\,\mathbf{U}_I\,\phi' \in \mathrm{sub}(\varphi)$.*

Example 10. *Consider $\varsigma_{p\,\mathbf{U}_{(1,3)}\,q,1} = (\emptyset, \{\chi_p^i, \chi_q^i, \chi_{p\,\mathbf{U}_I\,q}^i\}_{i\in2})$ for $p\,\mathbf{U}_{(1,3)}\,q$. Then, $cand_{p\,\mathbf{U}_{(1,3)}\,q}$ is the first-order formula:*

$$
\begin{aligned}
&((\uparrow_p^1 \vee \uparrow_q^1) \to cd_{p,q}^1(-3)) \wedge ((\downarrow_p^1 \vee \downarrow_q^1) \to cd_{p,q}^1(-1)) \\
\wedge\, &((\uparrow_p^2 \vee \uparrow_q^2) \to cd_{p,q}^2(-3)) \wedge ((\downarrow_p^2 \vee \downarrow_q^2) \to cd_{p,q}^2(\tau-1))
\end{aligned}
$$

because $\gamma_0 = 0$, $\gamma_1 = \tau$, $\mathfrak{J}_1 = \{0\}$, and $\mathfrak{J}_2 = (0,\tau)$. We show below the encodings for some auxiliary predicates:

$$
\begin{aligned}
cd_{p,q}^1(-3) &\equiv ((U_{p,q}^{1,1} \wedge 0 \le -3) \to endp_{1,1}^{-3}) \\
cd_{p,q}^2(-3) &\equiv \textstyle\bigwedge_{n=1}^2 ((U_{p,q}^{n,2} \wedge l(\mathfrak{J}_n) \le -3) \to endp_{n,2}^{-3}) \\
cd_{p,q}^2(\tau-1) &\equiv ((U_{p,q}^{1,2} \wedge 0 \le \tau-1) \to endp_{1,2}^{\tau-1}) \\
&\quad \wedge ((U_{p,q}^{2,2} \wedge 0 \le \tau-1) \to endp_{2,2}^{\tau-1}) \\
endp_{1,2}^{\tau-1} &\equiv endp_{1,1}^{\tau-1} \vee endp_{2,2}^{\tau-1} \\
endp_{2,2}^{\tau-1} &\equiv (\tau-1 = 0 \vee \tau-1 = \tau) \\
\uparrow_p^1 \equiv \chi_p^1 \quad &\downarrow_p^1 \equiv \chi_p^1 \wedge \neg\chi_p^2 \quad U_{p,q}^{1,2} \equiv \chi_p^1 \wedge U_{p,q}^{2,2} \\
\uparrow_p^2 \equiv \chi_p^2 \wedge \neg\chi_p^1 \quad &\downarrow_p^2 \equiv \chi_p^2 \qquad\quad U_{p,q}^{2,2} \equiv \chi_p^2 \wedge \chi_q^2
\end{aligned}
$$

Consider an assignment α for a symbolic discrete signal $\varsigma_{\varphi,n}$. $\Psi_{\mathsf{PAR}}[\varsigma_{\varphi,n}]$ asserts that $\mathcal{P} = \{\alpha(\mathfrak{J}_i)\}_{i\in[2n]}$ includes all variable point candidates for all subformulas $\phi\,\mathbf{U}_I\,\phi'$. Therefore, the φ-refinement of $\mathcal{P}$ is $\mathcal{P}$ itself, and Theorem 1 implies that φ is fully stable for a signal σ in $\mathcal{P}$, provided the signal σ matches α for each proposition in φ.

Lemma 9. *Suppose $(\forall t \in \alpha(\mathfrak{J}_i))\ p(\sigma(t)) = \alpha(\chi_p^i)$ for each $p \in \mathrm{prop}(\varphi)$. The following are equivalent: (i) $\mathcal{R}_\mathcal{P}^\sigma(\varphi) = \mathcal{P}$ for $\mathcal{P} = \{\alpha(\mathfrak{J}_i)\}_{i\in[2n]}$, and $\alpha(\chi_\phi^i) = \top$ iff $\sigma, \alpha(\mathfrak{J}_i) \models_\tau \phi$ for each $\phi \in \mathrm{sub}(\varphi)$. (ii) $\alpha(\Psi_{\mathsf{PAR}}[\varsigma_{\varphi,n}] \wedge \Psi_{\mathsf{STL}}[\varsigma_{\varphi,n}])$ is true.*

Proof Sketch. $\Psi_{\mathsf{PAR}}[\varsigma_{\varphi,n}]$ enforces $\mathcal{R}_\mathcal{P}^\sigma(\phi) = \mathcal{P}$ for each subformula $\phi \in \mathrm{sub}(\varphi)$, where $\mathcal{P} = \{\alpha(\mathfrak{J}_i)\}_{i\in[2n]}$. The lemma then easily follows by structural induction on φ from the correctness of $\Psi_{\mathsf{PAR}}[\varsigma_{\varphi,n}]$, Lemma 8, and Theorem 1. □

The remaining question is how to encode the existence of matching signals. For bounded satisfiability, we only need to consider *piecewise-constant* signals, which can be represented as a sequence of variables $\vec{\nu}_1, \vec{\nu}_2, \ldots, \vec{\nu}_{2n}$. Let

$$
\begin{aligned}
\Phi_{\mathsf{SIG}}[\varsigma_{\varphi,n}] \;\equiv\; &(\exists \vec{\nu}_1, \ldots, \vec{\nu}_{2n}) \textstyle\bigwedge_{p\in\mathrm{prop}(\varphi)} \bigwedge_{i=1}^{2n} (\chi_p^i = p(\vec{\nu}_i)) \\
&\wedge\, 0 \le \gamma_1 \le \cdots \le \gamma_{n-1} < \tau,
\end{aligned}
$$

where $\chi_p^i = p(\vec{\nu}_i)$ asserts that p holds in the i-th interval. The bounded satisfiability of φ can be reduced to the satisfiability of a first-order encoding as follows.

Theorem 2. *An STL formula φ is satisfiable by a bounded signal with a finite set of variable points iff for some number $n \in \mathbb{N}$, $\chi_\varphi^1 \wedge \Psi_{\mathsf{PAR}}[\varsigma_{\varphi,n}] \wedge \Psi_{\mathsf{STL}}[\varsigma_{\varphi,n}] \wedge \Phi_{\mathsf{SIG}}[\varsigma_{\varphi,n}]$ is satisfiable.*

Proof. ($\Rightarrow$) Suppose $\sigma, 0 \models_\tau \varphi$ for a signal σ with a finite set T of variable points for $\mathrm{sub}(\varphi)$. For the partition $\mathcal{P}$ with $e(\mathcal{P}) = T \cup \{0,\tau\}$, consider the φ-refinement $\mathcal{R}_\mathcal{P}^\sigma(\varphi)$, where $|\mathcal{R}_\mathcal{P}^\sigma(\varphi)| = 2n$. Let v_i be an element of each i-th interval of $\mathcal{R}_\mathcal{P}^\sigma(\varphi)$, and α be an assignment with $\{\alpha(\mathfrak{J}_i)\}_{i\in[2n]} = \mathcal{R}_\mathcal{P}^\sigma(\varphi)$, $\alpha(\vec{\nu}_i) = \sigma(v_i)$, and $\alpha(\chi_\phi^i) = \top$ iff $\sigma, v_i \models_\tau \phi$, for $i \in [2n]$ and $\phi \in \mathrm{sub}(\varphi)$. By Lemma 9, α satisfies the encoding. ($\Leftarrow$) Suppose the encoding is satisfiable by an assignment α. Let σ be a piecewise-constant signal with $\forall t \in \alpha(\mathfrak{J}_i).\ \sigma(t) = \alpha(\vec{\nu}_i)$. Then, for each $p \in \mathrm{prop}(\varphi)$, $(\forall t \in \alpha(\mathfrak{J}_i))\ \alpha(\chi_p^i) = p(\sigma(t))$. It follows from Lemma 9 that $\alpha(\chi_\varphi^1) = \top$ iff $\sigma, 0 \models_\tau \varphi$. □

VI. SMT-Based Model Checking Algorithm

We now consider the bounded model checking problem of an STL formula φ for a hybrid automaton H, stated in Def. 4. This problem is equivalent to finding a bounded trajectory σ of H that satisfies the negated formula $\neg\varphi$. Thanks to Lemma 9, the only remaining challenge is to encode the existence of matching bounded trajectories of H.

A. Encoding of Matching Bounded Trajectories

We follow an SMT-based approach for the reachability of hybrid automata [18], [29] to encode boundary trajectories of H. The reachability of H can be reduced to the satisfiability of an SMT formula, provided *init*, *inv*, *flow*, and *jump* conditions can be encoded in SMT [18]. Finding a bounded trajectory σ of H with at most k variable points for $\mathrm{prop}(\varphi)$ can be considered as a special case of reachability [22].

Figure 11 shows the first-order encoding $\Phi_{\mathsf{TRJ}}^H[\varsigma_{\varphi,n}]$ for a matching trajectory $\sigma \in H$. An initial state s_0^0 satisfies the *init* condition, and there are $n-1$ time points γ_j, $1 \le j \le n-1$, at which discrete jumps *can* occur from state s_{j-1}^t to s_j^0. The value of σ evolves from state s_j^0 to s_j^t according to the *flow* condition, satisfying *inv* (indicated by $cnt(s_j^0, s_j^t)$). The truth of each proposition p is stably χ_p^i in each interval $\mathfrak{J}_i$ (indicated

$$
\begin{aligned}
\Phi^H_{\mathsf{TRJ}}[\varsigma_{\varphi,n}] \equiv\ & (\exists s^0_0,\dots,s^t_{n-1})\ 0=\gamma_0\le\cdots\le\gamma_{n-1}<\gamma_n=\tau \\
& \wedge\ init(s^0_0)\ \wedge\ \textstyle\bigwedge_{j=1}^{n-1}(jump(s^t_{j-1},s^0_j)\vee s^t_{j-1}=s^0_j) \\
& \wedge\ \textstyle\bigwedge_{j=0}^{n-1} cnt(s^0_j,s^t_j)\ \wedge\ \bigwedge_{p\in\mathrm{prop}(\varphi)}\bigwedge_{i=1}^{2n} match^i_p \\
cnt(s^0_j,s^t_j) \equiv\ & s^t_j = flow(s^0_j,\gamma_{j+1}-\gamma_j) \\
& \wedge\ \forall u\in[0,\gamma_{j+1}-\gamma_j).\ inv(flow(s^0_j,u)) \\
match^{2j+1}_p \equiv\ & (p(s^0_j)=\chi^{2j+1}_p) \\
match^{2j+2}_p \equiv\ & ((\forall u\in(0,\gamma_{j+1}-\gamma_j).\ p(flow(s^0_j,u)))\wedge\chi^{2j+2}_p) \\
& \vee\ ((\forall u\in(0,\gamma_{j+1}-\gamma_j).\ \neg p(flow(s^0_j,u)))\wedge\neg\chi^{2j+2}_p)
\end{aligned}
$$

Fig. 11. Encoding $\Phi^H_{\mathsf{TRJ}}[\varsigma_{\varphi,n}]$ of matching trajectories (adapted from [22])

by $match^i_p$). The equality $s^t_{j-1}=s^0_j$ allows the truth of a proposition to change without jumps. By construction:

Lemma 10. *$\Phi^H_{\mathsf{TRJ}}[\varsigma_{\varphi,n}]$ is satisfiable by an assignment α iff there exists a trajectory $\sigma\in H$, with at most $n-1$ jumps, where $(\forall t\in\alpha(\mathfrak{J}_i))\ p(\sigma(t))=\alpha(\chi^i_p)$ for each $p\in\mathrm{prop}(\varphi)$.*

Finally, in a similar way to the case of bounded satisfiability (Theorem 2), the bounded model checking problem of an STL formula φ for a hybrid automaton H can be reduced to the satisfiability of a first-order encoding as follows.

Theorem 3. *$H,0\not\models^n_\tau\varphi$ holds iff for some number $N\ge n$, $\neg\chi^1_\varphi\wedge\Psi_{\mathsf{PAR}}[\varsigma_{\varphi,N}]\wedge\Psi_{\mathsf{STL}}[\varsigma_{\varphi,N}]\wedge\Phi^H_{\mathsf{TRJ}}[\varsigma_{\varphi,N}]$ is satisfiable.*

B. Algorithm

Our model checking algorithm is shown in Alg. 1. Ψ_{PAR} and Ψ_{STL} are in linear arithmetic. For polynomial *flow* conditions, Φ^H_{TRJ} is in nonlinear real arithmetic, supported by SMT solvers [38]–[40]. For *flow* conditions with transcendental functions, the satisfiability of Φ^H_{TRJ} can be undecidable, but there are approximate solvers [41], [42]. Our algorithm is refutationally complete for bounded trajectories with finite variable points, assuming an oracle for the satisfiability of the encoding.

Algorithm 1: Bounded STL Model Checking

Input: Hybrid automaton H, formula φ, bounds τ and N

```
for n = 1 to N do
    Φ^H_TRJ ⟵ encoding of H's trajectories (Fig. 11);
    foreach φ ∈ sub(φ) do
        Ψ_STL ⟵ add conditions for χ^i_φ, i ∈ [2n] (Def. 13);
        Ψ_PAR ⟵ add full stability conditions for φ (Def. 14);
    if checkSat(¬χ^1_φ ∧ Ψ_PAR ∧ Ψ_STL ∧ Φ^H_TRJ) is Sat then
        return counterexample(result.satAssignment);
return True;
```

For a bound n, the entire encoding size is $O(n|H|+n^2|\varphi|)$. The size of Φ^H_{TRJ} is $O(n|H|)$. The size of Ψ_{STL} is $O(n^2|\varphi|)$, as there is a condition for each subformula and each i-th step, $i\in[2n]$, where the size of the condition for $\phi\,\mathbf{U}_I\,\phi'$ is $O(i)$. For Ψ_{PAR}, all predicates in Fig. 10 can be encoded as formulas of size $O(n^2|\varphi|)$, using a typical encoding method to introduce extra Boolean variables for predicate instances [43].

As mentioned in Sec. III-C, a bound parameter k of the previous algorithm [22] denotes the number of variable points for $\mathrm{prop}(\varphi)$, and the size of the encoding is $O(k\cdot t(\varphi)\cdot 2^{d(\varphi)})$, with $t(\varphi)$ the number of $\mathbf{U}_I$ and $d(\varphi)$ the nesting depth of $\mathbf{U}_I$. Because the size of a φ-refinement is $O(k\cdot t(\varphi)\cdot 2^{d(\varphi)})$, in principle, our algorithm needs a larger bound than the previous algorithm. However, in practice, the size of a φ-refinement can be much smaller than in the worst case (see Sec. VII).

C. Handling Universal Quantification

The encoding Φ^H_{TRJ} includes universal quantification over time (e.g., $cnt(s^0_j,s^t_j)$ and $match^i_p$), as in typical SMT-based methods for the reachability of hybrid automata [18], [29]. Several SMT solvers, including Z3 [38] and Yices2 [39], provide limited support for these $\exists\forall$-formulas, but require high computational complexity. Our algorithm can be combined with two methods to remove universal quantification.

1) Quantifier-Free Encoding [44]: For polynomial hybrid automata, such $\exists\forall$-conditions can be encoded as quantifier-free formulas. When a polynomial f is monotonic on an interval $[a,b]$ and a predicate p is convex, $\forall t\in[a,b].\ p(f(t))$ can be reduced to $p(f(a))\wedge p(f(b))$. Since the monotonicity of f can be expressed as a convex invariant of the derivative $\dot f$, we can obtain a quantifier-free formula by applying this repeatedly.

2) Invariant Simplification: For invariant STL properties of the form $p\to\Box_I q$, the model checking problem is equivalent to the reachability. Finding a counterexample of $p\to\Box_I q$ can be reduced to checking whether there exists a state satisfying $\neg q$ that is reachable within the time interval I from an initial state satisfying p. Therefore, universal quantifiers in $match^i_p$ can be safely replaced by existential quantifiers.

VII. Experimental Evaluation

We have implemented our algorithm of Alg. 1, along with simplification methods in Sec. VI-C, using Yices2 [39] as the underlying SMT solver. This section experimentally evaluates our technique by addressing the following research questions:

RQ1 How effective is our algorithm compared to the previous algorithm [22] for bounded STL model checking?

RQ2 How effective is our encoding compared to the previous encoding [22] for bounded STL satisfiability checking?

RQ3 How effective is our approach compared to reachability analysis techniques for invariant STL properties?

We have run all experiments on Intel Xeon 2.8GHz with 256 GB memory. More details on the benchmark models and experiments can be found in the supplementary material [30].

A. RQ1: STL Model Checking of Hybrid Automata

We consider the following hybrid automata models (adapted from [22], [35], [45], [46]): autonomous cars (Car), thermostat (The) and water tank (Wat) systems, load management of batteries (Bat), and railroad gate controllers (Rai). We use two variants of continuous dynamics: linear dynamics, and polynomial dynamics of degree 2. For each model, we use four STL formulas: $\varphi^d_\top$ holds, and $\varphi^d_\bot$ has a counterexample for $d=1,2$; and φ^b_1 contains one temporal operator, and φ^b_2 contains two nested temporal operators for $b=\top,\bot$ [30].

TABLE I
BOUNDED MODEL CHECKING OF STL (TIME IS IN SECONDS, AND '-' DENOTES A TIMEOUT)

Dynamics	Model	Method	No counterexample						Counterexample found at n							
			$\varphi^1_\top$			$\varphi^2_\top$			$\varphi^1_\bot$				$\varphi^2_\bot$			
			|Enc|	|ς|	Time	|Enc|	|ς|	Time	|Enc|	|ς|	Time	n	|Enc|	|ς|	Time	n
Linear (N = 50)	Car	New	8.1	50	112	14.7	50	1,455	1.7	9	1.4	9	3.2	14	9.15	14
		[22]	67.2	154	-	604	466	-	5.5	48	2.23	8	-	-	-	-
	The	New	8.1	50	3,391	14.7	50	5,734	1.9	10	1.9	10	1.2	4	1.2	4
		[22]	67.2	154	-	604	466	-	7.7	58	1.7	10	9.6	64	11.2	4
	Wat	New	8.5	50	326	15.2	50	779	2.2	11	6.9	11	2.9	12	36.4	12
		[22]	67.6	154	-	316	366	-	7.7	58	92.4	10	-	-	-	-
	Bat	New	12.8	50	140	18.6	50	242	1.6	6	0.7	6	2.5	8	70.5	8
		[22]	72.3	154	395	608	466	1,135	2.4	19	0.8	5	13.6	80	249	4
	Rai	New	6.5	50	74.7	13.2	50	152	1.0	6	0.1	6	0.9	5	0.1	5
		[22]	65.6	154	54.8	1,086	678	419	2.8	33	0.1	5	5.7	44	1.6	4
Polynomial (N = 20)	Car	New	5.7	20	56.8	7.2	20	4.5	2.3	7	2.3	7	2.8	8	7.5	8
		[22]	16.2	64	-	113	196	15.2	5.5	43	5.2	7	-	-	-	-
	The	New	3.3	20	4.4	4.8	20	23.1	0.9	4	0.5	4	0.9	4	0.9	4
		[22]	13.8	64	-	110	196	-	1.6	16	1.2	4	9.0	52	1,087	4
	Wat	New	3.5	20	28.4	5.0	20	4.8	0.8	4	0.2	4	1.2	5	2.4	5
		[22]	14.0	64	-	111	196	15.0	1.5	16	0.5	4	-	-	-	-
	Bat	New	5.1	20	14.1	6.2	20	6.4	2.1	5	1.7	5	1.5	5	0.6	5
		[22]	15.7	64	73.0	112	196	232	3.0	16	1.1	4	13.6	80	197	4
	Rai	New	3.0	20	26.7	4.5	20	78.0	1.1	6	0.5	6	1.0	5	0.6	5
		[22]	13.6	64	199	201	288	-	2.9	33	0.7	5	-	-	-	-

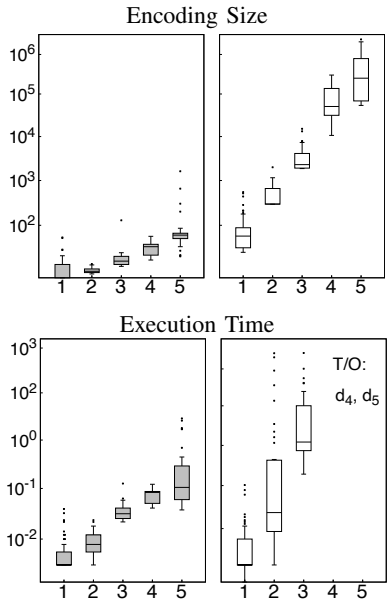

Fig. 12. Bounded Satisfiability of STL (New: filled boxes, [22]: empty boxes)

We have performed bounded model checking of these STL properties, up to bound $N = 50$ for linear models, and $N = 20$ for polynomial models. We assign different time bounds τ to different models [30], since τ depends on different model parameters. We measure the size of the SMT encoding, the size of the symbolic discrete signal, and the cumulative execution times to run STL model checking, including SMT solving by the underlying solver. We set a timeout of 120 minutes.

The experimental results are summarized in Table I. |Enc| denotes encoding sizes (in thousands), given by the number of connectives in the encoding. $|\varsigma|$ denotes the size of symbolic discrete signal ς, given by the number of real variables. For the cases with counterexamples, the table also shows the smallest bound $n \leq N$ for which a counterexample is found.

As discussed in Sec. VI-B, bounds have different meanings in our algorithm and the previous algorithm [22]. For our algorithm, a bound corresponds to the size of a fully stable φ-refinement, and therefore $|\varsigma|$ is the same as the bound. For [22], a bound denotes the number of variable points for the set of propositions, and thus $|\varsigma|$ is usually larger than the bound. This can result in huge differences in the encoding size.

As shown in Table I, our algorithm significantly outperforms the previous algorithm [22] in most cases. In particular, our algorithm shows much better performance for complex STL formulas with nested temporal operators ($\varphi^2_\top$ and $\varphi^2_\bot$). The performance improvement is due to the much compact size of the encoding by our method. E.g., for $\varphi^2_\top$ and linear cases, our algorithm produces 20–80 times smaller encodings.

Since different notions of bound n are used, the previous algorithm often found counterexamples of $\varphi^1_\bot$ and $\varphi^2_\bot$ at a smaller bound. Thanks to our partition construction method, the differences are often very minor (0 or 1). The only exceptional case is Bat with linear dynamics for $\varphi^2_\bot$: a counterexample was found at $k = 4$ by the previous algorithm but at $k = 8$ by our algorithm. Even in this case, our algorithm performed much better than the previous one.

B. RQ2: Bounded STL Satisfiability Checking

To evaluate the effectiveness of our encoding for Ψ_{PAR} and Ψ_{STL}, We have performed bounded STL satisfiability checking using Theorem 2. We randomly generate 250 STL formulas of different sizes and complexity [30]: specifically, there are 50 formulas for each nesting depth $1 \leq d \leq 5$. We measure the execution time and the size of generated SMT formulas, up to bound $N = 20$ with a timeout of 30 minutes.

Figure 12 summarizes the experimental results in box plots. The x-axis shows nesting depth d, and the y-axis shows encoding sizes and execution times (seconds), accordingly, in a log scale. Filled boxes indicated our encoding method, and empty boxes indicates the previous method [22]. Outliers are plotted as individual points in the box plots.

As seen, our method produced much smaller encodings that can be solved in a much shorter time. This improvement tends to be more apparent for more complex formulas: for example, for $d = 5$, the previous method mostly timed out, whereas our method only needed less than 3 seconds on average.

TABLE II
ANALYZING INVARIANT PROPERTIES

Model		Linear ($N = 50$)					Polynomial ($N = 10$)			
		ONew	New	HC	SE	FS	ONew	New	SE	FS
Car	$I_\top$	17.1	133	0.57	3.62	16.4	1.93	2.21	825	75.1
	$I_\bot$	0.01	0.02	0.02	3.72	1.41	1.53	2.92	803	9.23
Therm	$I_\top$	4.52	37.1	4.21	2.23	18.2	1.44	2.92	3.12	5.82
	$I_\bot$	0.01	0.02	0.03	2.24	1.41	0.02	0.21	3.01	1.42
Water	$I_\top$	3.75	27.1	8.72	1.72	56.1	8.51	5.92	0.01	0.08
	$I_\bot$	0.01	0.02	0.04	1.71	1.91	0.42	114	0.01	0.14
Battry	$I_\top$	4.41	24.0	1.82	0.07	43.1	1.33	1.41	0.01	20.1
	$I_\bot$	0.23	0.52	0.02	0.06	6.42	0.21	6.91	0.01	4.53
Rail	$I_\top$	1.91	32.0	0.69	0.64	2.01	0.82	1.52	0.23	2.81
	$I_\bot$	0.01	0.02	0.02	0.63	5.11	0.03	0.41	0.24	2.31

C. RQ3: Comparison with Reachability Analysis Methods

We compare our approach with three reachability analysis tools for hybrid automata: HyComp [18] (HC), SpaceEx [15] (SE), and Flow* [16] (FS). We consider two versions of our algorithm: one with invariant simplification (ONew) in Sec. VI-C, and one without invariant simplification (New). For each model, we consider two invariant properties that can be handled by those tools: $I_\top$ holds, and $I_\bot$ does not hold.

We measure the execution times for analyzing the invariant properties with a timeout of 15 minutes. Since each tool has a different notion of bounds, we use the number of jumps N and the maximum time horizon τ as the common bound parameters (which are "encoded" in the models). For SpaceEx, we use PHAVer for linear models, and STC for polynomial models. For Flow*, we use adaptive steps and TM orders 1 (for linear) and 2 (for polynomial). We use BMC for HyComp.

The experimental results are summarized in Table II, with execution times in seconds. The results for polynomial models do not include HyComp, which does not support nonlinear polynomial dynamics. For $I_\top$, the table shows the time taken to prove the absence of counterexamples. For $I_\bot$, the table shows the time taken to find a counterexample.

We observe that our algorithm with invariant simplification (ONew) has comparable performance to the other tools for both invariant properties $I_\top$ and $I_\bot$. The implementation without invariant simplification (New) often shows slower performance, especially for $I_\top$. This problem is caused by stability conditions $match_p^i$ in Φ^H_{TRJ}, which can be avoided in ONew when analyzing simple invariant properties.

D. Threats to Validity

Two major threats to internal validity are (1) whether the performance improvements are due to the proposed partition construction and encoding methods, and (2) whether different notions of bounds have unintended effects on the evaluation of the performance. To address these threats, we applied exactly the same simplification methods in Sec. VI-C for the experiments in Sec. VII-A and Sec. VII-B, and evaluated the performance based on the smallest bound for which a counterexample is found for the experiment in Sec. VII-A.

One threat to external validity is the possible bias of the benchmark models and formulas used in the experiments. We tried to reduce this threat by using various linear/polynomial hybrid automata models, and using a variety of STL properties with different sizes, complexity, truth values, etc. Another threat is that our algorithm is compared only to the most related previous work for general STL formulas; comparing with the other STL model checking algorithm [21] is difficult, because there is no implementation available.

VIII. RELATED WORK

The most closely related work is the paper [22] which proposes the refutationally complete bounded STL model checking algorithm. As discussed in Sec. III-C, the algorithm [22] does not take into account reducing the size of discrete signals, and thus often produces a very complex encoding. In contrast, our method uses a precise condition for discretizing signals and produces a much simpler encoding.

Another STL model checking method [21] uses reachable set computation, instead of SMT solving. Unlike our method, the algorithm [21] is incomplete, since only finite *concrete* sampled time points are considered, and thus the correctness cannot be guaranteed. In contrast, our method *symbolically* encodes discrete signals without any loss of information.

There are a couple of methods for verifying LTL properties of hybrid systems using signal discretization: [23] proposes a deductive system for LTL, and [24] presents a requirement validation method for an extension of LTL. Only *untimed* temporal operators are considered in [23], [24]. We address the challenge of optimally discretizing signals involving $\mathbf{U}_I$.

A discretization of signals with $\mathbf{U}_I$ has been studied for MITL in the context of real-time systems. A refinement for $\varphi\, \mathbf{U}_I\, \varphi'$ can be built using multiples of I's endpoints [27]. A signal is further discretized using a unit duration based on a relative interval width [28]. These methods produce very fine refinements, which are not scalable for hybrid systems.

Our work is related to reachability analysis techniques for hybrid automata. They can guarantee the correctness, but only deal with invariant properties. There are roughly two kinds of methods: reachable-set computation [15], [16], [47]–[50], and SMT-based techniques [18], [29], [42], [51]–[53].

IX. CONCLUDING REMARKS

We have presented a new SMT-based model checking algorithm for STL properties of hybrid systems. Our algorithm is based on a new theoretical result to build *small but complete* discretized signals for the bounded satisfiability of STL. We have developed a modular translation method to encode the bounded satisfiability of STL in SMT. We have leveraged our translation technique to obtain an efficient SMT-based STL model checking algorithm that is still refutationally complete for bounded signals, but that produces much simpler encodings and results in much better performance.

Currently, our method can only be applied to SMT-based approaches. We should therefore investigate refutationally complete STL model checking algorithms that can be integrated with reachable-set computation methods.

REFERENCES

[1] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Proc. FORMATS*, ser. LNCS, vol. 3253. Springer, 2004, pp. 152–166.

[2] X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. Butts, "Powertrain control verification benchmark," in *Proc. HSCC*. ACM, 2014.

[3] R. P. Goldman, D. Bryce, M. J. Pelican, D. J. Musliner, and K. Bae, "A hybrid architecture for correct-by-construction hybrid planning and control," in *Proc. NFM*, ser. LNCS, vol. 9690. Springer, 2016.

[4] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Proc. HSCC*. ACM, 2015, pp. 239–248.

[5] N. Roohi, R. Kaur, J. Weimer, O. Sokolsky, and I. Lee, "Parameter invariant monitoring for signal temporal logic," in *Proc. HSCC*. ACM, 2018, pp. 187–196.

[6] A. Dokhanchi, B. Hoxha, and G. Fainekos, "Metric interval temporal logic specification elicitation and debugging," in *Proc. MEMOCODE*. IEEE, 2015, pp. 70–79.

[7] H. Roehm, T. Heinz, and E. C. Mayer, "STLInspector: STL validation with guarantees," in *Proc. CAV*, ser. LNCS, vol. 10426. Springer, 2017, pp. 225–232.

[8] Z. Xu, C. Belta, and A. Julius, "Temporal logic inference with prior information: An application to robot arm movements," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 141–146, 2015.

[9] A. Donzé, T. Ferrère, and O. Maler, "Efficient robust monitoring for STL," in *Proc. CAV*, ser. LNCS, vol. 8044. Springer, 2013.

[10] J. V. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia, "Robust online monitoring of signal temporal logic," *Form. Methods Syst. Des.*, vol. 51, no. 1, pp. 5–30, 2017.

[11] S. Jakšić, E. Bartocci, R. Grosu, and D. Ničković, "Quantitative monitoring of STL with edit distance," *Form. Methods Syst. Des.*, vol. 53, no. 1, pp. 83–112, 2018.

[12] D. Ničković, O. Lebeltel, O. Maler, T. Ferrère, and D. Ulus, "Amt 2.0: qualitative and quantitative trace analysis with extended signal temporal logic," in *Proc. TACAS*, vol. 10806. Springer, 2018, pp. 303–319.

[13] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-taliro: A tool for temporal logic falsification for hybrid systems," in *Proc. TACAS*, ser. LNCS, vol. 6605. Springer, 2011, pp. 254–257.

[14] Z. Zhang, D. Lyu, P. Arcaini, L. Ma, I. Hasuo, and J. Zhao, "Effective hybrid system falsification using monte carlo tree search guided by qb-robustness," in *Proc. CAV*, ser. LNCS, vol. 12759. Springer, 2021, pp. 595–618.

[15] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *Proc. CAV*, ser. LNCS, vol. 6806. Springer, 2011, pp. 379–395.

[16] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *Proc. CAV*, ser. LNCS, vol. 8044. Springer, 2013, pp. 258–263.

[17] S. Bak and P. S. Duggirala, "Hylaa: A tool for computing simulation-equivalent reachability for linear systems," in *Proc. HSCC*. ACM, 2017, pp. 173–178.

[18] A. Cimatti, A. Griggio, S. Mover, and S. Tonetta, "HyComp: an SMT-based model checker for hybrid systems," in *Proc. TACAS*, ser. LNCS, vol. 9035. Springer, 2015, pp. 52–67.

[19] S. Kong, S. Gao, W. Chen, and E. M. Clarke, "dReach: δ-reachability analysis for hybrid systems," in *Proc. TACAS*, ser. LNCS, vol. 7898. Springer, 2015, pp. 200–205.

[20] P. S. Duggirala, C. Fan, S. Mitra, and M. Viswanathan, "Meeting a powertrain verification challenge," in *Proc. CAV*, ser. LNCS, vol. 9206. Springer, 2015, pp. 536–543.

[21] H. Roehm, J. Oehlerking, T. Heinz, and M. Althoff, "STL model checking of continuous and hybrid systems," in *Proc. ATVA*, ser. LNCS, vol. 9938. Springer, 2016, pp. 412–427.

[22] K. Bae and J. Lee, "Bounded model checking of signal temporal logic properties using syntactic separation," *Proc. ACM Program. Lang.*, vol. 3, POPL, no. 51, pp. 1–30, 2019.

[23] L. De Alfaro and Z. Manna, "Verification in continuous time by discrete reasoning," in *Proc. AMAST*, ser. LNCS, vol. 936. Springer, 1995, pp. 292–306.

[24] A. Cimatti, M. Roveri, and S. Tonetta, "Requirements validation for hybrid systems," in *Proc. CAV*, ser. LNCS, vol. 5643. Springer, 2009, pp. 188–203.

[25] J. Ouaknine and J. Worrell, "Some recent results in metric temporal logic," in *Proc. FORMATS*, ser. LNCS, vol. 5215. Springer, 2008.

[26] H.-M. Ho, J. Ouaknine, and J. Worrell, "Online monitoring of metric temporal logic," in *Proc. RV*, ser. LNCS, vol. 8734. Springer, 2014, pp. 178–192.

[27] R. Alur, T. Feder, and T. A. Henzinger, "The benefits of relaxing punctuality," *JACM*, vol. 43, no. 1, pp. 116–146, 1996.

[28] T. Ferrère, O. Maler, D. Ničković, and A. Pnueli, "From real-time logic to timed automata," *JACM*, vol. 66, no. 3, pp. 1–31, 2019.

[29] S. Gao, S. Kong, and E. M. Clarke, "Satisfiability modulo ODEs," in *Proc. FMCAD*. IEEE, 2013, pp. 105–112.

[30] J. Lee, G. Yu, and K. Bae, "Supplementary material: proofs of lemmas and benchmarks," https://github.com/stlmc/stlmc/tree/ase2021.

[31] T. Henzinger, "The theory of hybrid automata," in *Verification of Digital and Hybrid Systems*, ser. NATO ASI Series. Springer, 2000, vol. 170, pp. 265–292.

[32] S. Minopoli and G. Frehse, "SL2SX translator: from simulink to spaceex models," in *Proc. HSCC*. ACM, 2016, pp. 93–98.

[33] S. Bak, O. A. Beg, S. Bogomolov, T. T. Johnson, L. V. Nguyen, and C. Schilling, "Hybrid automata: from verification to implementation," *STTT*, vol. 21, no. 1, pp. 87–104, 2019.

[34] S. Ran, J. Lin, Y. Wu, J. Zhang, and Y. Xu, "Converting Ptolemy II models to SpaceEx for applied verification," in *Proc. ICA3PP*, ser. LNCS, vol. 8630. Springer, 2014, pp. 669–683.

[35] R. Alur, *Principles of cyber-physical systems*. The MIT Press, 2015.

[36] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[37] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" *Journal of computer and system sciences*, vol. 57, no. 1, pp. 94–124, 1998.

[38] L. De Moura and N. Bjørner, "Z3: an efficient SMT solver," in *Proc. TACAS*, ser. LNCS, vol. 4963. Springer, 2008, pp. 337–340.

[39] B. Dutertre, "Yices 2.2," in *Proc. CAV*, ser. LNCS, A. Biere and R. Bloem, Eds., vol. 8559. Springer, 2014, pp. 737–744.

[40] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, and C. Tinelli, "CVC4," in *Proc. CAV*, ser. LNCS, vol. 6806. Springer, 2011, pp. 171–177.

[41] S. Gao, S. Kong, and E. M. Clarke, "dReal: An SMT solver for nonlinear theories over the reals," in *Proc. CADE*, ser. LNCS, vol. 7898. Springer, 2013, pp. 208–214.

[42] A. Eggers, N. Ramdani, N. S. Nedialkov, and M. Fränzle, "Improving the SAT modulo ODE approach to hybrid systems analysis by combining different enclosure methods," *SoSyM*, vol. 14, no. 1, pp. 121–148, 2015.

[43] S. D. Prestwich, "CNF encodings." *Handbook of satisfiability*, vol. 185, pp. 75–97, 2009.

[44] A. Cimatti, S. Mover, and S. Tonetta, "Quantifier-free encoding of invariants for hybrid systems," *Form. Methods Syst. Des.*, vol. 45, no. 2, pp. 165–188, 2014.

[45] J. Raisch, E. Klein, C. Meder, A. Itigin, and S. O'Young, "Approximating automata and discrete control for continuous systems — two examples from process control," in *Hybrid systems V*, ser. LNCS, vol. 1567. Springer, 1999, pp. 279–303.

[46] M. Fox, D. Long, and D. Magazzeni, "Plan-based policies for efficient multiple battery load management," *JAIR*, vol. 44, pp. 335–382, 2012.

[47] M. Althoff, "Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets," in *Proc. HSCC*. ACM, 2013, pp. 173–182.

[48] T. Dang and R. Testylier, "Reachability analysis for polynomial dynamical systems using the bernstein expansion," *Reliable Computing*, vol. 17, no. 2, pp. 128–152, 2012.

[49] S. Bak and P. S. Duggirala, "Simulation-equivalent reachability of large linear systems with inputs," in *Proc. CAV*, ser. LNCS, vol. 10426. Springer, 2017, pp. 401–420.

[50] C. Fan, B. Qi, S. Mitra, M. Viswanathan, and P. S. Duggirala, "Automatic reachability analysis for nonlinear hybrid models with C2E2," in *Proc. CAV*, ser. LNCS, vol. 9779. Springer, 2016, pp. 531–538.

[51] K. Bae and S. Gao, "Modular SMT-based analysis of nonlinear hybrid systems," in *Proc. FMCAD*. IEEE, 2017, pp. 180–187.

[52] D. Ishii, K. Ueda, and H. Hosobe, "An interval-based SAT modulo ODE solver for model checking nonlinear hybrid systems," *STTT*, vol. 13, no. 5, pp. 449–461, 2011.

[53] A. Tiwari, "Time-aware abstractions in HybridSal," in *Proc. CAV*, ser. LNCS, vol. 9206. Springer, 2015, pp. 504–510.