Efficient SMT-Based Model Checking for Signal Temporal Logic

Jia Lee, Geunyeol Yu, and Kyungmin Bae

Pohang University of Science and Technology
{cee5539,rgyen,kmbae}@postech.ac.kr

Abstract. We present a new SMT-based model checking algorithm for verifying signal temporal logic (STL) properties of hybrid automata. STL is widely used for analyzing continuous and hybrid systems. However, STL model checking is still quite limited; in particular, existing STL model checking methods are either incomplete or very inefficient. This paper proposes a novel translation method to reduce the STL bounded model checking problem for hybrid automata to the satisfiability of a first-order logic formula over the real numbers, which can be solved using state-of-the-art SMT solvers for (nonlinear) real arithmetic. Our modular 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.

1 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 often involve continuously changing states of physical entities. *Signal temporal logic* (STL) is a temporal logic formalism to specify linear-time properties of continuous signals [39]. STL is widely used for specifying and analyzing (safety) requirements of continuous and hybrid systems [16, 30, 37, 44, 46].

There are many approaches for analyzing STL properties of hybrid systems. STL monitoring [15, 18, 36, 40] checks whether a signal satisfies an STL property, and STL falsification [6] 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 the correctness. The reachability analysis of hybrid automata, such as [9, 11, 12, 26] can guarantee the correctness of invariants, and some STL formulas can be encoded using reachability [19].

However, model checking of general STL properties is very limited. Existing STL model checking algorithms are incomplete or very inefficient. Recently, two model checking methods have been proposed for STL. The work [45] considers a finite number of sampled time points and performs the reachability analysis over those points, and so the correctness cannot be guaranteed. Our previous work [8] provides a refutationally complete model checking procedure for STL, but the algorithm is rather theoretical and not scalable in practice.

In this paper we propose a new SMT-based model checking algorithm for verifying STL properties of hybrid automata that is refutationally complete with respect to bounded signals. Our algorithm is based on a discretization of signals, proposed in this paper, which is *complete* for the bounded satisfiability of STL. Using (symbolically) discretized signals, we present a modular translation of the STL bounded model checking problem to the satisfiability of a first-order logic formula over the real numbers. Our algorithm combines the advantages of the previous methods [8,45]: it considers a finite number of sampled time points, but which is complete without any loss of information for STL model checking.

In this paper, we consider bounded signals with finite variable points (i.e., excluding Zeno behaviors), which is typically assumed when analyzing real-time and hybrid systems [33,39,41] Intuitively, for a signal with finite variable points, the satisfaction of an STL formula must depend on a finite number of sampled time points. To the best of our knowledge, no complete discretization method has been proposed for STL. This paper proposes such a complete discretization of signals with respect to the bounded satisfaction of STL (Section 3).

A complete discretization of signals allows reducing the bounded satisfaction of STL into the satisfiability of a first-order logic 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 [24]. However, an effective translation of STL is nontrivial, because temporal operators in STL involves arbitrary nonnegative intervals of the real numbers. We leverage a syntactic separation of STL [8] to translate the bounded satisfaction of STL into a quantifier-free first-order logic formula in linear real arithmetic. Our translation method exploits a complete discretization of signal to generate an encoding in a modular way (Section 4).

Based on our modular translation method, we present an efficient bounded model checking algorithm for STL properties of hybrid automata (Section 5). Our algorithm builds a first-order logic formula that is satisfiable if and only if there exists a counterexample trajectory of a hybrid automaton H 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 [12, 28]. The algorithm is refutationally complete for bounded signals with finite number of variable points, by incrementing a bound n. We have implemented our algorithm in the STLMC STL bounded model checker [2] for hybrid automata. The experimental results shows that the proposed algorithm can greatly outperform the previous algorithm [8] (Section 6).

Our main contributions can be summarized as follows: (1) We present a new theoretical technique to completely characterize the bounded satisfaction of STL formulas using a discretization of a continuous signal; (2) We present a modular method to encode the bounded satisfiability of STL in a decidable fragment of first-order logic; (3) We present a new SMT-based model checking algorithm for STL that is refutationlly complete for bounded signal but that is much more scalable than the previous algorithm [8]. The complete proofs of the lemmas in the paper can be found in Appendix A. The benchmark models, and the experimental results are available in [1].

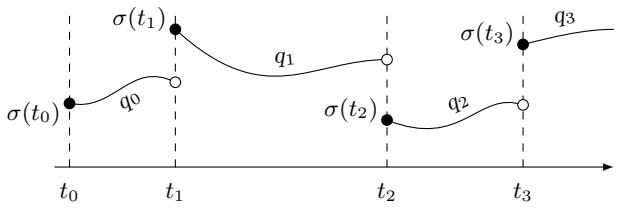


Fig. 1. A trajectory σ of a hybrid automaton

2 Background: Bounded Model Checking of STL

2.1 Hybrid Automata

Hybrid automata [32] are state machines with continuous variables. Discrete states are specified by a set of *modes* Q, and continuous states are specified by a finite set of real-valued *variables* X. A state 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 *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.[1] A *jump condition* $jump(\langle q, \vec{v} \rangle, \langle q', \vec{v'} \rangle)$ defines a discrete transition between states $\langle q, \vec{v} \rangle \to \langle q', \vec{v'} \rangle$.

Definition 1. *A* hybrid automaton *is a tuple* $H = (Q, X, init, inv, flow, jump)$.

A continuous "execution" of a hybrid automaton H over a time domain $[0, \tau)$, for $\tau > 0$, can be given by a function $\sigma : [0, \tau) \to Q \times \mathbb{R}^l$, called a *signal*, where $\sigma(t)$ denotes the state of H at time $0 \leq t < \tau$. A signal σ is called *bounded* if its time domain $[0, \tau)$ is bounded (i.e., $\tau < \infty$). A *trajectory* of a hybrid automaton H is a signal that describes a valid continuous behavior of H, e.g., 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 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 t_i, $i > 0$, a discrete jump takes places. Formally:

Definition 2. *Given a hybrid automaton* H*, a signal* $\sigma : [0, \tau) \to Q \times \mathbb{R}^l$ *is a* trajectory *of* H*, written* $\sigma \in H$*, if for a (potentially infinite) sequence of times* $0 = t_0 < t_1 < t_2 < t_3 < \cdots < \tau_N = \tau$*, the following conditions hold:*

(i) $init(\sigma(t_0))$*;*
(ii) $\sigma(u) = flow(\sigma(t_i), u - t_i)$ *and* $inv(\sigma(u))$ *for* $u \in [t_i, t_{i+1})$*; and*
(iii) $jump(s^t_{i-1}, \sigma(t_i))$ *and* $inv(s^t_{i-1})$*, where* $s^t_{i-1} = flow(\sigma(t_{i-1}), t_i - t_{i-1})$.

[1] A flow condition *flow* is usually defined using continuous real functions or ordinary differential equations (ODEs).

Example 1. Figure 3 shows a hybrid automaton for a simple autonomous car [8]. The position (x, y) and direction θ of the car depend on its speed v and steering angle ϕ. The car dynamics can be 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 [38]. There are three modes of assigning different control values to v and ϕ. 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}.

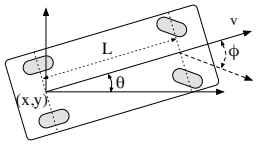


Fig. 2. A simple car

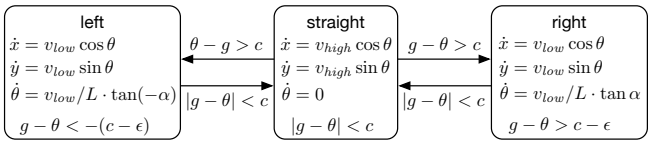


Fig. 3. A hybrid automaton for a simple car

2.2 Signal Temporal Logic

For hybrid automata, a state proposition represents a function $p : Q \times \mathbb{R}^l \to \mathbb{B}$ that assigns to each state $\langle q, \vec{v}\rangle$ 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.

Properties of hybrid automata trajectories can be specified in signal temporal logic (STL). 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 $p \in \Pi$ is a state proposition, and $I \subseteq \mathbb{R}_{\geq 0}$ is an interval of nonnegative real numbers.[2] Untimed notations, such as $\Diamond$, $\Box$, and $\mathbf{U}$, are used as shorthand for $\Box_{[0,\infty)}$, $\Box_{[0,\infty)}$, and $\mathbf{U}_{[0,\infty)}$. respectively. The set of state propositions occurring in a formula φ is denoted by $\mathrm{Props}(\varphi)$.

An interval I is often shortened using arithmetic expressions: for example, $\mathbf{U}_{=a} \equiv \mathbf{U}_{\{a\}}$, $\mathbf{U}_{<a} \equiv \mathbf{U}_{[0,a)}$, and $\mathbf{U}_{\geq a} \equiv \mathbf{U}_{[a,\infty)}$. Other logical and temporal operators can be derived using equivalences: for example,

$$\varphi \vee \varphi' \equiv \neg(\neg\varphi \wedge \neg\varphi'), \quad \varphi \to \varphi' \equiv \neg\varphi \vee \varphi',$$
$$\Diamond_I\, \varphi \equiv \top\, \mathbf{U}_I\, \varphi, \qquad \Box_I\, \varphi \equiv \neg\, \Diamond_I\, \neg\varphi, \quad \varphi\, \mathbf{R}_I\, \varphi' \equiv \neg((\neg\varphi)\, \mathbf{U}_I (\neg\varphi')).$$

[2] In the literature [18, 39, 45], mode-related propositions are usually not considered, and I is restricted to a non-singular closed interval. In this paper, we consider a slightly generalized version of STL that does not have such restrictions.

Example 2. For a hybrid automaton in Example 1, consider the 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 km/h and stay above 100 km/h for 20 seconds [16].
- $\Box_{\geq 0}(\theta > 15 \rightarrow (\theta < 20)\, \mathbf{U}_{[2,3)}\, \mathit{toLeft})$: whenever θ is greater than 15°, the mode will be *left* after sometime within $[2, 3)$; until then θ is less than 20°.

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 $I - J$. For example, $[a, b] + [c, d] = [a + c, b + d]$ and $(a, b) - (c, d) = (a - d, b - c)$. The sets $\{t\} + I$ and $\{t\} - I$ are often written as $t + I$ and $t - I$, respectively.

The semantics of STL is defined as the satisfaction relation of a formula φ with respect to a signal σ and a time $t \in \mathrm{dom}(\sigma)$. 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.

Definition 3. *The* satisfaction *of an STL formula φ at time t over a signal σ up to a bound τ, where* $\mathrm{dom}(\sigma) \supseteq [0, \tau)$*, denoted by $\sigma, t \models_\tau \varphi$, is defined by:*

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

The *future-reach* $fr(\varphi)$ represents the future time required to determine the satisfaction of φ [34]. We can easily see that the bounded semantics of STL is equivalent to the unbounded semantics for any bound $\tau > fr(\varphi)$.

Lemma 1. *The* future-reach $fr(\varphi)$ *of an STL formula φ is defined by:*

$$
\begin{array}{ll}
fr(p) = 0 & fr(\varphi \wedge \varphi') = \max(fr(\varphi), fr(\varphi')) \\
fr(\neg\varphi) = fr(\varphi) & fr(\varphi\, \mathbf{U}_I\, \varphi') = \sup(I) + \max(fr(\varphi), fr(\varphi')).
\end{array}
$$

For any bound $\tau > t + fr(\varphi)$, $\sigma, t \models_\tau \varphi$ iff $\sigma, t \models_\infty \varphi$, where $\mathrm{dom}(\sigma) \supseteq [0, \tau)$.

2.3 Bounded Model Checking of STL

The STL model checking problem is to determine if every trajectory of a hybrid automaton H satisfies an STL formula φ. The STL model checking problem is undecidable in general; indeed, the reachability problem of hybrid automata, which is a special case of STL model checking, is already undecidable [31].

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 $t \in \mathrm{dom}(\sigma)$ is called a variable point if the truth value of p changes at time t.

Definition 4. *A time $t \in \mathrm{dom}(\sigma)$ is a* variable point *of a signal σ with respect to a set of propositions Π, if there exists no open interval $J \subseteq \mathrm{dom}(\sigma)$ with $t \in J$ such that for each proposition $p \in \Pi$, $(\forall u, u' \in J)\ p(\sigma(u)) = p(\sigma(u'))$.*

Example 3. Consider a signal $y = \cos(\pi t/2) + 1$ with $\mathrm{dom}(y) = [0, 10)$ in Fig. 4. There are four variable points $\{1, 3, 5, 9\}$ with respect to proposition $y > 1$.

Bounded STL model checking [8] restricts the search for a counterexample to bounded trajectories with a finite number of variable points. There are two bound parameters for a trajectory: a bound τ for the time domain, and a bound k for the number of variable points in the time domain $[0, \tau)$.

Definition 5. *Given a hybrid automaton H, an STL formula φ is satisfied up to bounds $\tau > 0$ and $k \in \mathbb{N}$ at time $t < \tau$, denoted by*

$$H, t \models^k_\tau \varphi,$$

iff $\sigma, t \models_\tau \varphi$ holds for every trajectory $\sigma \in H$, where $\mathrm{dom}(\sigma) \supseteq [0, \tau)$, with at most k variable points in the interval $[0, \tau)$ with respect to $\mathrm{Props}(\varphi)$.

The bounded STL model checking problem is equivalent to finding a bounded trajectory $\sigma \in H$ that satisfies the negated formula $\neg\varphi$. By Proposition 1, if the future-reach of φ is bounded (i.e., $fr(\varphi) < \infty$), then we only need to consider a time bound $\tau > t + fr(\varphi)$ to determine $H, t \models^k_\infty \varphi$.

3 Discretization of Signals for STL

The time domain of a signal with finite variable points can be divided into a finite set of disjoint stable intervals in which the truth value of each proposition does not change. The satisfaction of an STL formula is completely determined by such a discretized version of a signal. This section presents how a signal can be discretized using a stable partition of a time domain for STL.

3.1 Stable Formulas

By definition, a non-variable point $t \in \mathrm{dom}(\sigma)$ involves an open neighborhood $J \ni t$ in which the truth value of every proposition $p \in \Pi$ is "stable" in the interval J (that is, $p(\sigma(u)) = p(\sigma(u'))$ for any time points $u, u' \in J$). The notion of stability can also be defined for arbitrary STL formulas.

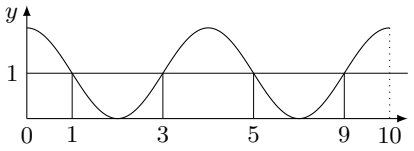


Fig. 4. Variable points of signal y.

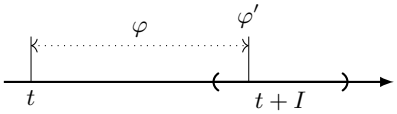


Fig. 5. The satisfaction of $\varphi\, \mathbf{U}_I\, \varphi'$

Definition 6. *Given a signal σ, an STL formula φ is* stable *for an interval $J \subseteq \mathrm{dom}(\sigma)$, if for any time points $u, u' \in J$, $\sigma, u \models_\tau \varphi \iff \sigma, u' \models_\tau \varphi$.*

The stability and non-variability of a proposition do not always coincide. For example, consider the signal y in Example 3. The proposition $y > 1$ is stable for the closed interval $[1, 3]$, but there are two variable points $1, 3 \in [1, 3]$. The following lemma states the relationship between stability and non-variability.

Lemma 2. *Given a signal σ, a proposition p, and an interval $J \subseteq \mathrm{dom}(\sigma)$: (1) If J is open and p is stable for J, then there is no variable point in J. (2) If there is no variable point in J, then p is stable for J.*

The stability of an STL formula φ is related to the stability of its subformulas. The cases for negation and conjunction are stated in the following lemma. Note that the converse of (2) in the lemma does not hold in general: for example, $p \wedge \neg p$ is stably false, regardless of the stability of p.

Lemma 3. *For a signal σ and an interval $J \subseteq \mathrm{dom}(\sigma)$: (1) φ is stable for J iff $\neg\varphi$ is stable for J. (2) If φ and φ' are stable for J, then $\varphi \wedge \varphi'$ is stable for J.*

The case of $\mathbf{U}_I$ is not straightforward. As depicted in Fig. 5, the satisfaction of $\varphi\,\mathbf{U}_I\,\varphi'$ at time t depends on the satisfaction of φ in the "prefix" interval $[t, \inf(t + I)]$. Even if φ and φ' are stable for the interval $t + I$, φ' need not be stable for $[t, \inf(t + I)]$. To account for such prefix intervals, we consider the stability with respect to a sequence of disjoint intervals. Let $[N] = \{1, \ldots, N\}$.

Definition 7. *A* partition *of an interval J is a family of nonempty disjoint intervals $\mathcal{P} = \{J_i\}_{i\in[N]}$ such that $\bigcup_{i=1}^{N} J_i = J$ and $\sup(J_i) \leq \inf(J_k)$ for $i < k$.*

For two partitions $\mathcal{P} = \{J_i\}_{i\in[N]}$ and $\mathcal{Q} = \{K_j\}_{j\in[M]}$ of the same interval J, $\mathcal{P}$ is called *finer* than $\mathcal{Q}$, written $\mathcal{P} \sqsubseteq \mathcal{Q}$, if each J_i is a subset of some K_j. An STL formula φ is stable for a partition $\mathcal{P}$, if φ is stable for each interval in $\mathcal{P}$. A stable formula for one partition is obviously stable for finer partitions.

Corollary 1. *If an STL formula φ is stable for a partition $\mathcal{P}$ of an interval J, then φ is stable for any partition $\mathcal{Q} \sqsubseteq \mathcal{P}$ of J that is finer than $\mathcal{P}$.*

Suppose that φ and φ' are stable for a partition $\mathcal{P}$ of time domain $[0, \tau)$. If $\varphi\,\mathbf{U}_I\,\varphi'$ is satisfied at time t in some interval K, then φ' must be satisfied at some time $t' \in t + I$, which is an element of some interval in $\mathcal{P}$. To be stable for K, given any time $t \in K$, it should be possible to choose t' from the same interval J_k. This requirement is formally characterized in the following lemma.

Lemma 4. *An STL formula $\varphi\,\mathbf{U}_I\,\varphi'$ is stable for a partition $\mathcal{Q} = \{K_j\}_{j\in[M]}$ of time domain $[0, \tau)$, if: (i) φ and φ' are stable for a partition $\mathcal{P} = \{J_i\}_{i\in[N]}$, (ii) either $K_j \subseteq J_i - I$ or $K_j \subseteq (J_i - I)^{\complement}$ for any K_j and J_i, and (iii) $\mathcal{Q} \sqsubseteq \mathcal{P}$.*

Example 4. Consider a formula $q\,\mathbf{U}_{(1,3)}\,p$, two partitions $\mathcal{P} = \{[0,5],(5,8)\}$ and $\mathcal{Q} = \{[0,2],(2,4),[4,7),[7,8)\}$, and a signal σ with $\mathrm{dom}(\sigma) = [0,8)$ such that

$$\forall t \in [0,8).\ p(\sigma(t)) = \top, \quad \forall t \in [0,5].\ q(\sigma(t)) = \bot, \quad \forall t \in (5,8).\ q(\sigma(t)) = \top$$

The partitions $\mathcal{P}$ and $\mathcal{Q}$ satisfy Conditions (i) and (ii), but not Condition (iii). Observe that $q\,\mathbf{U}_{(1,3)}\,p$ is not stable for $\mathcal{P}$ and $\mathcal{Q}$ (e.g. $\sigma, 4.5 \not\models_8 q\,\mathbf{U}_{(1,3)}\,p$, $\sigma, 6 \models_8 q\,\mathbf{U}_{(1,3)}\,p$, and $\sigma, 7.5 \not\models_8 q\,\mathbf{U}_{(1,3)}\,p$), but is stable for the partition $\{[0,2],(2,4),[4,5],(5,7),[7,8)\}$, which also satisfies Condition (iii). This example shows that all three conditions are needed for the stability of $\varphi\,\mathbf{U}_I\,\varphi'$.[3]

3.2 Partition Construction

Consider a signal σ with finite variable points $t_1, \ldots, t_n \in [0,\tau)$ with respect to a set of propositions Π. Each proposition $p \in \Pi$ is clearly stable for each point interval $\{t_i\}$, and, by Lemma 2, is stable for each open interval (t_i, t_{i+1}). In other words, each proposition $p \in \Pi$ is stable for the partition induced by T [8].

Definition 8. *The partition induced by a set of times T is $\mathcal{P}_\tau(T) = \{J_i\}_{i\in[2n]}$ such that $J_i = \{t_k\}$ for $i = 2k+1$, and $J_i = (t_k, t_{k+1})$ for $i = 2k$, where $T \cup \{0,\tau\} = \{t_0, t_1, \ldots, t_{n+1}\}$ and $0 = t_0 < t_1 < \ldots < t_{n+1} = \tau$.*

Corollary 2. *Given a signal σ with a finite set of variable points T for a set of proposition Π, each proposition $p \in \Pi$ is stable for the partition $\mathcal{P}_\tau(T)$.*

We first summarize some properties of induced partitions $\mathcal{P}_\tau(T)$ [8]. For two finite sets of times T and T', if T is a subset of T', then $\mathcal{P}_\tau(T') \sqsubseteq \mathcal{P}_\tau(T)$; by Corollary 1, if φ is stable for $\mathcal{P}_\tau(T)$, then φ is stable for $\mathcal{P}_\tau(T')$. Let $e(I)$ denote the set of endpoints of an interval I.[4] For the set T_I obtained by subtracting I's endpoints from each element of T, $\mathcal{P}_\tau(T_I)$ satisfies Condition (ii) in Lemma 4.

Lemma 5. *[8, Lemma 4.18] For a finite set $T \subseteq [0,\tau)$ and an interval I, let $T_I = \bigcup_{u\in T\cup\{\tau\}}\{u - v \mid v \in e(I)\} \cap [0,\tau)$. For partitions $\mathcal{P}_\tau(T) = \{J_i\}_{i\in[N]}$ and $\mathcal{P}_\tau(T_I) = \{K_j\}_{j\in[M]}$, $K_j \subseteq J_i - I$ or $K_j \subseteq (J_i - I)^\complement$ holds for any K_j and J_i.*

Suppose that an STL formula φ is stable for an induced partition $\mathcal{P}_\tau(T)$. For any strict subset $U \subset T$, if φ is not stable for $\mathcal{P}_\tau(U)$, then T is called *minimal* with respect to the stability of φ. When T is not minimal, there always exists a minimal subset of T, denoted by $min^\sigma_\varphi(T)$, as stated in the following lemma.

Lemma 6. *Given a signal σ, an STL formula φ, and a finite set $T \subseteq [0,\tau)$, if φ is stable for $\mathcal{P}_\tau(T)$, there exists a minimal subset $min^\sigma_\varphi(T) \subseteq T$ such that for any strict subset $U \subset min^\sigma_\varphi(T)$, φ is not stable for $\mathcal{P}_\tau(U)$.*

[3] This differs from the condition proposed for STL with global time (STL-GT) in [8].
[4] E.g., $e([1,2)) = \{1,2\}$, $e((1,\infty)) = \{1\}$, and $e([0,\infty)) = \{0\}$.

For a signal σ with a finite set of variable points T, we can build a set of times $\mathcal{T}^\sigma_\varphi(T) \subseteq [0,\tau)$ where each subformula ψ of an STL formula φ is stable for the partition induced by $\mathcal{T}^\sigma_\varphi(T)$. We build a set of times with respect to the stability of each subformula $\psi \in \mathrm{sub}(\varphi)$, and take the union of those sets.[5]

Definition 9. *For an STL formula φ and a signal σ with a finite set of variable points $T \subseteq [0,\tau)$ with respect to* $\mathrm{Props}(\varphi)$, $\mathcal{T}^\sigma_\varphi(T) = \bigcup_{\psi\in\mathrm{sub}(\varphi)} \mathcal{I}(\psi)$ *such that*

$$\mathcal{I}(p) = T \qquad \mathcal{I}(\neg\phi) = \mathcal{I}(\phi) \qquad \mathcal{I}(\phi \wedge \phi') = \mathcal{I}(\phi) \cup \mathcal{I}(\phi')$$
$$\mathcal{I}(\phi\, \mathbf{U}_I\, \phi') = \mathcal{I}(\phi) \cup \mathcal{I}(\phi') \cup \left(\bigcup_{u\in U} \{u - v \mid v \in e(I)\} \cap [0,\tau)\right)$$

where $U = \mathit{min}^\sigma_\phi(\mathcal{I}(\phi)) \cup \mathit{min}^\sigma_{\phi'}(\mathcal{I}(\phi')) \cup \{\tau\}$.[6]

The size of $\mathcal{T}^\sigma_\varphi(T)$ is at most $O(k \cdot t(\varphi) \cdot 2^{d(\varphi)})$, where $k = |T|$, $t(\varphi)$ is the number of $\mathbf{U}_I$, and $d(\varphi)$ is the nesting depth of $\mathbf{U}_I$. However, because of the reduction with $\mathit{min}^\sigma_\varphi$, the size of $\mathcal{T}^\sigma_\varphi(T)$ can be much smaller than in the worst case.

By construction, each subformula ψ of an STL formula φ is stable for the partition $\mathcal{P}_\tau(\mathcal{I}(\psi))$. Each proposition p is stable for $\mathcal{I}(p)$, because T contains all variable points in $[0,\tau)$. By Lemma 3 and induction hypothesis, $\neg\varphi$ is stable for $\mathcal{P}_\tau(\mathcal{I}(\neg\varphi))$, and $\varphi \wedge \varphi'$ is stable for $\mathcal{P}_\tau(\mathcal{I}(\varphi \wedge \varphi'))$. By Lemma 5, $\mathcal{P}_\tau(U)$ satisfies Condition (ii) in Lemma 4, and therefore, by induction hypothesis, $\varphi\, \mathbf{U}_I\, \varphi'$ is stable for $\mathcal{P}_\tau(\mathcal{I}(\varphi\, \mathbf{U}_I\, \varphi'))$. Finally, $\mathcal{P}_\tau(\mathcal{T}^\sigma_\varphi(T))$ is finer than each $\mathcal{P}_\tau(\mathcal{I}(\psi))$, and thus each subformula ψ is stable for $\mathcal{P}_\tau(\mathcal{T}^\sigma_\varphi(T))$. Consequently:

Theorem 1. *For an STL formula φ, let σ be a signal with a finite set of variable points $T \subseteq [0,\tau)$ in time domain $[0,\tau)$ with respect to* $\mathrm{Props}(\varphi)$. *Each subformula $\psi \in \mathrm{sub}(\varphi)$ is stable for the partition $\mathcal{P}_\tau(\mathcal{T}^\sigma_\varphi(T))$ induced by $\mathcal{T}^\sigma_\varphi(T)$.*

Example 5. Consider a signal σ with variable points $T = \{5\}$ in Example 4. For an STL formula $\varphi = \Diamond_{\geq 1}(q\, \mathbf{U}_{(1,3)}\, p)$, $\mathcal{T}^\sigma_\varphi(T) = \{2,4,5,6,7\}$, because

$$\mathcal{I}(p) = \mathcal{I}(q) = \{5\} \quad \mathcal{I}(q\, \mathbf{U}_{(1,3)}\, p) = \{2,4,5,7\} \quad \mathcal{I}(\Diamond_{\geq 1}(q\, \mathbf{U}_{(1,3)}\, p)) = \{4,5,6,7\}$$

By Theorem 1, each subformula of φ is stable for $\mathcal{P}_8(\{2,4,5,6,7\})$. The reduction with $\mathit{min}^\sigma_\varphi$ in Def. 9 can have a significant effect: e.g., without reduction, we obtain $\{1,2,3,4,5,6,7\}$ for this example.

4 Modular Translation of STL

This section presents a modular method to encode the bounded satisfaction of STL in a decidable fragment of first-order logic. Unlike the encoding in [8], a first-order logic formula is built *separately* for each subformula, using stable partition construction in Sec. 3. This provides a much simpler procedure—that also produces a much smaller formula—than the non-modular translation [8].

[5] The partition construction in [8] returns a partition mapping for STL-GT, assigning to each subformula a different partition. In contrast, we build a *single* partition for STL, reduced using $\mathit{min}^\sigma_\varphi$, where every subformula is stable.

[6] Not every $\mathcal{I}(\psi)$ for $\psi \in \mathrm{sub}(\varphi)$ is minimized, since an effective SMT encoding is not known. In contrast, the condition for U can easily be encoded in SMT (see Sec. 4.2).

4.1 Separation with Complete Sampling

By Theorem 1, for an STL formula φ and a signal σ with finite variable points, there exists a partition $\mathcal{P} = \{J_i\}_{i\in[N]}$ for which every subformula of φ is stable. We can sample a time point $v_i \in J_i$ from each interval J_i in the partition $\mathcal{P}$; for example, $v_i = t_i$ for $J_i = \{t_i\}$, and $v_i = (t_i + t_{i+1})/2$ for $J_i = (t_i, t_{i+1})$. Let χ^i_ψ denote the truth value of each subformula ψ at each sampled time point v_i.

Definition 10. *A* time sample *of a partition $\mathcal{P} = \{J_i\}_{i\in[N]}$ is a family of time points $\mathcal{S}_\mathcal{P} = \{v_i\}_{i\in[N]}$ such that $v_i \in J_i$ for each $i \in [N]$. The sampled values from a signal σ is denoted by $\sigma[\mathcal{S}_\mathcal{P}] = \{\sigma(v_i)\}_{i\in[N]}$. For an STL formula ψ, $\chi^i_\psi \in \mathbb{B}$ is a Boolean value such that $\chi^i_\psi = \top$ iff $\sigma, v_i \models_\tau \psi$.*

Intuitively, for an STL formula ψ and a sampled time v_i, the truth value χ^i_ψ can be determined using the truth values for ψ's subformulas. By definition, $\chi^i_p = p(\sigma(v_i))$ for proposition p. We can easily see that $\chi^i_{\neg\varphi} = \neg\chi^i_\varphi$ for $\neg\varphi$, and $\chi^i_{\varphi\wedge\varphi'} = \chi^i_\varphi \wedge \chi^i_{\varphi'}$ for $\varphi \wedge \varphi'$. However, it is nontrivial to identify such relationship for $\varphi\,\mathbf{U}_I\,\varphi'$, since we need to decompose the satisfaction of $\varphi\,\mathbf{U}_I\,\varphi'$ into the satisfaction of its subformulas φ and φ' over each interval in $\mathcal{P}$.

For this purpose, we use a "global-time" temporal operator $\mathbf{U}^K_I$ with an extra global time interval K [8]. Other global-time operators can be derived; e.g., $\Diamond^K_I\,\varphi \equiv \top\,\mathbf{U}^K_I\,\varphi$ and $\Box^K_I\,\varphi \equiv \neg\Diamond^K_I\neg\varphi$. By definition, $\sigma, t \models_\tau \varphi\,\mathbf{U}^K_I\,\varphi'$ iff[7] $(\exists t' \geq t)\ t' \in [0,\tau) \cap K,\ t' \in t + I,\ \sigma, t' \models_\tau \varphi'$, and $(\forall t'' \in [t, t'] \cap K)\ \sigma, t'' \models_\tau \varphi$. We can syntactically separate $\varphi\,\mathbf{U}^K_I\,\varphi'$ into an equivalent formula where the satisfaction of each subformula only depends on disjoint segments of K.

Lemma 7. *[8, Proposition 3.10] For intervals K_1 and K_2, where $K_1 \cap K_2 = \emptyset$ and $\sup(K_1) = \inf(K_2)$, $\varphi\,\mathbf{U}^{K_1\cup K_2}_I\,\varphi' \equiv \varphi\,\mathbf{U}^{K_1}_I\,\varphi' \vee (\Box^{K_1}_{\geq 0}\,\varphi \wedge \varphi\,\mathbf{U}^{K_2}_I\,\varphi')$.*

Using syntactic separation, the satisfaction of $\varphi\,\mathbf{U}_I\,\varphi'$ at time $v_i \in J_i$ can be reduced to the satisfaction of φ and φ' at times $v_i, \ldots, v_N$. For $L_k = \bigcup_{j=k}^N J_k$, consider the satisfaction of $\varphi\,\mathbf{U}^{L_k}_I\,\varphi'$ at time $v_i \in J_i$. Because $L_k = J_k \cup L_{k+1}$, and φ and φ' are stable for J_k, we obtain the following separation laws:

Lemma 8. *For a partition $\mathcal{P} = \{J_i\}_{i\in[N]}$ and its time sample $\{v_i\}_{i\in[N]}$, where φ and φ' are stable for $\mathcal{P}$, let $L_k = \bigcup_{j=k}^N J_k$ for $k \in [N]$. (1) $\sigma, v_i \models_\tau \varphi\,\mathbf{U}^{L_k}_I\,\varphi'$ iff $\sigma, v_i \models_\tau \varphi\,\mathbf{U}^{J_k}_I\,\varphi' \vee (\Box^{J_k}_{\geq 0}\,\varphi \wedge \varphi\,\mathbf{U}^{L_{k+1}}_I\,\varphi')$. (2) $\sigma, v_i \models_\tau \varphi\,\mathbf{U}^{J_k}_I\,\varphi'$ iff $v_i \in J_k - I$, $\sigma, v_k \models_\tau \varphi$, and $\sigma, v_k \models_\tau \varphi'$. (3) $\sigma, v_i \models_\tau \Box^{J_k}_{\geq 0}\,\varphi$ iff $\sigma, v_k \models_\tau \varphi$.*

Because $v_i \in L_i$ and $\sup(L_i) = \tau$, $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I\,\varphi'$ iff $\sigma, v_i \models_\tau \varphi\,\mathbf{U}^{L_i}_I\,\varphi'$. By repeatedly applying the separation laws in the above lemma, we can obtain a condition consisting of $v_i \in J_k - I$, $\sigma, v_k \models_\tau \varphi$, and $\sigma, v_k \models_\tau \varphi'$, for $i \leq k \leq N$. This allows us to determine the value of $\chi^i_{\varphi\,\mathbf{U}_I\,\varphi'}$ using the values of χ^k_φ and $\chi^k_{\varphi'}$ for its subformulas, together with extra time constraints $v_i \in J_k - I$.

[7] $\mathbf{U}^K_I$ in this paper is equivalent to the restricted *Until* operator $\tilde{\mathbf{U}}^K_I$ in [8].

Definition 11. *Given an STL formula φ and a partition $\mathcal{P} = \{J_i\}_{i\in[N]}$, for $\mathcal{S}_\mathcal{P} = \{v_i\}_{i\in[N]}$ and $\mathcal{X} = \{\chi^i_\psi \mid \psi \in \mathrm{sub}(\varphi), i \in [N]\}$, let*

$$\Gamma_\Pi(\sigma, \mathcal{S}_\mathcal{P}, \mathcal{X}) = \{\chi^i_p = p(\sigma(v_i)) \mid i \in [N]\},$$

and $\Gamma_\varphi(\mathcal{P}, \mathcal{X})$ be a set of the following conditions over $\mathcal{X}$:

$$\chi^i_{\neg\phi} = \neg\chi^i_\phi, \qquad \chi^i_{\phi\wedge\phi'} = \chi^i_\phi \wedge \chi^i_{\phi'}, \qquad \chi^i_{\phi\,\mathbf{U}_I\,\phi'} = tr_i(\phi\,\mathbf{U}_I\,\phi', i),$$

where $tr_i(\phi\,\mathbf{U}_I\,\phi', k) = (v_i \in J_k - I \wedge \chi^k_\phi \wedge \chi^k_{\phi'}) \vee (\chi^k_\phi \wedge tr_i(\phi\,\mathbf{U}_I\,\phi', k+1))$ for each $i \le k \le N$, and $tr_i(\phi\,\mathbf{U}_I\,\phi', N+1) = \bot$.

The following lemma shows that when every subformula of an STL formula φ is stable for a partition $\mathcal{P}$, the set of conditions $\Gamma_\varphi(\mathcal{P}, \mathcal{X}) \cup \Gamma_\Pi(\sigma, \mathcal{S}_\mathcal{P}, \mathcal{X})$ can "completely" determine the bounded satisfaction of φ over the signal σ. In this sense, $\mathcal{P}$ has a *complete sample* $\{\sigma(v_i)\}_{i\in[N]}$ with respect to φ and σ.

Lemma 9. *Suppose that each $\psi \in \mathrm{sub}(\varphi)$ of an STL formula φ is stable for a partition $\mathcal{P} = \{J_i\}_{i\in[N]}$. The following conditions are equivalent: (i) $\chi^i_\psi = \top$ iff $(\forall t \in J_i)\ \sigma, t \models_\tau \psi$. (ii) every condition in $\Gamma_\varphi(\mathcal{P}, \mathcal{X}) \cup \Gamma_\Pi(\sigma, \mathcal{S}_\mathcal{P}, \mathcal{X})$ holds.*

Example 6. Consider an STL formula $\varphi = \Diamond_{\ge 1}(q\,\mathbf{U}_{(1,3)}\,p)$, and the partition $\mathcal{P}_8(\mathcal{T}^\sigma_\varphi(T))$ induced by $\mathcal{T}^\sigma_\varphi(T) = \{2, 4, 5, 6, 7\}$ in Example 5. The value of χ^i_ψ for each subformula $\psi \in \mathrm{sub}(\varphi)$ and interval J_i in $\mathcal{P}_8(\mathcal{T}^\sigma_\varphi(T))$ is as follows:

Subformula	$\{0\}$	$(0,2)$	$\{2\}$	$(2,4)$	$\{4\}$	$(4,5)$	$\{5\}$	$(5,6)$	$\{6\}$	$(6,7)$	$\{7\}$	$(7,8)$
p	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
q	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$
$q\,\mathbf{U}_{(1,3)}\,p$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\top$	$\bot$	$\bot$
$\Diamond_{\ge 1}(q\,\mathbf{U}_{(1,3)}\,p)$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$

By Lemma 9, $\Gamma_\varphi(\mathcal{P}, \mathcal{X}) \cup \Gamma_\Pi(\sigma, \mathcal{S}_\mathcal{P}, \mathcal{X})$ holds. E.g., for $\psi = q\,\mathbf{U}_{(1,3)}\,p$ and a time sample $v_{10} = 6.5$ from the sixth interval $(6, 7)$, we obtain the condition $\chi^{10}_{q\,\mathbf{U}_{(1,3)}\,p} = (6.5 \in (6,7) - (1,3) \wedge \chi^{10}_q \wedge \chi^{10}_p) \vee (\chi^{10}_q \wedge ((6.5 \in \{7\} - (1,3) \wedge \chi^{11}_q \wedge \chi^{11}_p) \vee (\chi^{11}_q \wedge (6.5 \in (7,8) - (1,3) \wedge \chi^{12}_q \wedge \chi^{12}_p))))$. Both sides of the condition are $\top$, since $\chi^k_p = \chi^k_q = \top$ for $10 \le k \le 12$, and $6.5 \in (7,8) - (1,3) = (4,7)$.

4.2 Encoding of Bounded Satisfiability

Recall that every subformula of an STL formula φ is stable for the partition induced by a set $\mathcal{T}^\sigma_\varphi(T)$ with a finite set of variable points T. For any set of times $V \subseteq [0, \tau)$, if $V \supseteq \mathcal{T}^\sigma_\varphi(T)$, then by Corollary 1, each subformula of φ is stable for the partition $\mathcal{P}_\tau(V)$ induced by V. The following definition logically characterizes the condition $V \supseteq \mathcal{T}^\sigma_\varphi(T)$ as a set of formulas $\Delta_\varphi(V)$.

Definition 12. *For an STL formula φ and a set $V \cup \{0, \tau\} = \{t_0, \ldots, t_{n+1}\}$, where $0 = t_0 < t_1 < \ldots < t_{n+1} = \tau$, $\Delta_\varphi(V)$ is a collection of the following conditions over V: for each $\phi_1\,\mathbf{U}_I\,\phi_2 \in \mathrm{sub}(\varphi)$, $k \in [n+1]$, and $v \in e(I)$:*

$$(k = n+1 \vee \textstyle\bigvee_{j=1,2}(\chi^{2k}_{\phi_j} \neq \chi^{2k+2}_{\phi_j})) \to (v = 0 \vee t_k - v < 0 \vee \textstyle\bigvee_{j=1}^{k-1}(t_k - v = t_j))$$

For a finite set of variable points T, let V be the set $\mathcal{T}^{\sigma}_{\varphi}(T)$ in Def. 9. By Theorem 1, each subformula of φ is stable for $\mathcal{P}_\tau(V)$, and thus by Lemma 9, the conditions $\mathit{\Gamma}_{\varphi}(\mathcal{P}_\tau(V), \mathcal{X}) \cup \mathit{\Gamma}_{\mathit{\Pi}}(\sigma, \mathcal{S}_{\mathcal{P}_\tau(V)}, \mathcal{X})$ hold. Because $\mathit{\Delta}_{\varphi}(V)$ encodes the requirements for $\mathcal{T}^{\sigma}_{\varphi}(T)$ in Def. 9, the conditions $\mathit{\Delta}_{\varphi}(V)$ also hold. Let $\mathit{\Omega}_{\varphi}(V, \mathcal{X}, \sigma) = \mathit{\Delta}_{\varphi}(V) \cup \mathit{\Gamma}_{\varphi}(\mathcal{P}, \mathcal{X}) \cup \mathit{\Gamma}_{\mathit{\Pi}}(\sigma, \mathcal{S}_{\mathcal{P}_\tau(V)}, \mathcal{X})$.

Lemma 10. *For a formula φ and a finite set $V \subseteq [0, \tau)$, if $V = \mathcal{T}^{\sigma}_{\varphi}(T)$ for a set of variable points T, then every condition in $\mathit{\Omega}_{\varphi}(V, \mathcal{X}, \sigma)$ holds.*

As expected, for any set $V \subseteq [0, \tau)$ satisfying the conditions $\mathit{\Omega}_{\varphi}(V, \mathcal{X}, \sigma)$, every subformula of φ is stable for the partition $\mathcal{P}_\tau(V)$ induced by V, provided that V includes every variable point with respect to the set of propositions $\mathrm{Props}(\varphi)$. The conditions $\mathit{\Delta}_{\varphi}(V)$ enforce that V includes any "unstable" points for the subformulas of $\phi_1 \,\mathbf{U}_I\, \phi_2$, using the partition construction in Lemma 5.

Lemma 11. *If every condition in $\mathit{\Omega}_{\varphi}(V, \mathcal{X}, \sigma)$ holds, then each $\psi \in \mathrm{sub}(\varphi)$ is stable for $\mathcal{P}_\tau(V)$, provided that each $p \in \mathrm{Props}(\varphi)$ is stable for $\mathcal{P}_\tau(V)$.*

The converse of Lemma 11 does not hold in general. For example, consider an STL formula $\varphi = \Diamond_{\geq 1}(q \,\mathbf{U}_{(1,3)}\, p)$ in Example 6. Let $V = \{5, 6, 7\}$. As shown in the table in Example 6, every subformula of φ is stable for $\mathcal{P}_\tau(V)$. However, $\mathit{\Delta}_{\varphi}(V)$ does not hold; in particular, $5 \in V$ and the truth value of q changes at 5, but $5 - v \notin V$ for any endpoint $v \in e((1, 3))$ for subformula $q \,\mathbf{U}_{(1,3)}\, p$.

When a finite set V satisfies the conditions $\mathit{\Omega}_{\varphi}(V, \mathcal{X}, \sigma)$, any finite set $V' \supseteq V$ satisfies $\mathit{\Omega}_{\varphi}(V', \mathcal{X}, \sigma)$. By Lemma 11, every subformula of φ is stable for $\mathcal{P}_\tau(V)$, and, since $\mathcal{P}_\tau(V')$ is finer than $\mathcal{P}_\tau(V)$, is also stable for $\mathcal{P}_\tau(V')$. This implies that $\mathit{\Gamma}_{\varphi}(\mathcal{P}_\tau(V'), \mathcal{X}) \cup \mathit{\Gamma}_{\mathit{\Pi}}(\sigma, \mathcal{S}_{\mathcal{P}_\tau(V')}, \mathcal{X})$ holds. Since the truth of each subformula of φ does not change at any time $t \in V' \setminus V$, $\mathit{\Delta}_{\varphi}(V')$ also holds.

Corollary 3. *Suppose each $p \in \mathrm{Props}(\varphi)$ is stable for $\mathcal{P}_\tau(V)$. If $\mathit{\Omega}_{\varphi}(V, \mathcal{X}, \sigma)$ holds, then for any finite set $V' \subseteq [0, \tau)$ with $V' \supseteq V$, $\mathit{\Omega}_{\varphi}(V', \mathcal{X}, \sigma)$ holds,*

Together with Lemma 9, the bounded satisfaction of an STL formula φ can be reduced to the satisfiability of the conditions $\mathit{\Omega}_{\varphi}(V, \mathcal{X}, \sigma)$. If $\sigma, 0 \models_\tau \varphi$ for a signal σ with a finite set of variable points T, then by Lemma 10, for $V = \mathcal{T}^{\sigma}_{\varphi}(T)$, $\mathit{\Omega}_{\varphi}(V, \mathcal{X}, \sigma)$ holds, and so by Lemma 9, $\chi^1_{\varphi} = \top$ holds. Conversely, suppose that $\{\chi^1_{\varphi}\} \cup \mathit{\Delta}_{\varphi}(V) \cup \mathit{\Gamma}_{\varphi}(\mathcal{P}, \mathcal{X})$ is satisfied. For the partition $\mathcal{P}_\tau(V)$ induced by V, let $\mathcal{S}_{\mathcal{P}_\tau(V)} = \{v_i\}_{i \in [N]}$. Let σ be any signal satisfying the equalities $\chi^i_p = p(\sigma(v_i))$ in $\mathit{\Gamma}_{\mathit{\Pi}}(\sigma, \mathcal{S}_{\mathcal{P}_\tau(V)}, \mathcal{X})$ such that each proposition $p \in \mathrm{Props}(\varphi)$ is stable for $\mathcal{P}_\tau(V)$ (such as, a piecewise-constant signal). By Lemma 11 an Lemma 9, $\chi^1_{\varphi} = \top$ iff $(\forall t \in J_1)\ \sigma, t \models_\tau \varphi$, where $J_1 = \{0\}$. Consequently:

Theorem 2. *There exists a signal σ with a finite set of variable points T such that $\sigma, 0 \models_\tau \varphi$ iff for some finite set V and signal σ, $\{\chi^1_{\varphi}\} \cup \mathit{\Delta}_{\varphi}(V) \cup \mathit{\Gamma}_{\varphi}(\mathcal{P}, \mathcal{X})$ and $\mathit{\Gamma}_{\mathit{\Pi}}(\sigma, \mathcal{S}_{\mathcal{P}_\tau(V)}, \mathcal{X})$ are satisfied, where $|V| \geq |\mathcal{T}^{\sigma}_{\varphi}(T)|$.*

$$
\begin{aligned}
&(\exists s_0^0,\ldots,s_n^0)\ (\exists s_0^t,\ldots,s_n^t) \quad 0=t_0<t_1<\cdots<t_n<t_{n+1}=\tau \ \wedge \\
&\mathit{init}(s_0^0)\wedge s_0^t=\mathit{flow}(s_0^0,t_1-t_0) \quad \wedge\forall u\in[0,t_1-t_0).\ \mathit{inv}(\mathit{flow}(s_0^0,u)) \quad \wedge \\
&\textstyle\bigwedge_{i=1}^{n}\ \mathit{jump}(s_{i-1}^t,s_i^0)\wedge s_i^t=\mathit{flow}(s_i^0,t_{i+1}-t_i)\wedge\forall u\in[0,t_{i+1}-t_i).\ \mathit{inv}(\mathit{flow}(s_i^0,u))\wedge \\
&\bigwedge_{p\in\Pi}\bigwedge_{i=0}^{n}(p(s_i^0)=\chi_p^{2i+1})\wedge\begin{bmatrix}((\forall u\in(0,t_{i+1}-t_i).\ p(\mathit{flow}(s_i^0,u)))\wedge\chi_p^{2i+2})\vee\\((\forall u\in(0,t_{i+1}-t_i).\ \neg p(\mathit{flow}(s_i^0,u)))\wedge\neg\chi_p^{2i+2})\end{bmatrix}
\end{aligned}
$$

Fig. 6. Encoding $\Psi_H^{n,\tau}(V,\mathcal{X})$ of bounded trajectories.

5 SMT-Based Model Checking Algorithm

This section presents an SMT-based bounded model checking algorithm for STL properties of hybrid automata. Like the previous model checking algorithm for STL in [8], our model checking algorithm is *refutationally complete* for bounded trajectories. Thanks to our modular translation method, our algorithm produces a much simpler encoding than one generated by the previous algorithm [8].

5.1 SMT Encoding of Bounded STL Model Checking

We apply an SMT-based approach for the reachability of hybrid automata [12,28] to encode the existence of boundary trajectories. The reachability problem of hybrid automata can be reduced to the satisfiability of a first-order logic formula, when *init*, *inv*, *flow*, and *jump* conditions can be encoded in first-order logic [12]. Finding a bounded trajectory σ of a hybrid automaton H with at most n variable points can be considered as a special case of reachability [8].

Consider two sets of first-order variables $V\cup\{0,\tau\}=\{t_0,\ldots,t_{n+1}\}$ and $\{\chi_p^i\mid p\in\Pi, i\in[2n+2]\}\subseteq\mathcal{X}$, given a set of propositions Π. Figure 6 shows the first-order logic encoding $\Psi_H^{n,\tau}(V,\mathcal{X})$ for the existence of a bounded trajectory σ, with variable points in V, that satisfies the conditions in $\Gamma_\Pi(\sigma,\mathcal{S}_{\mathcal{P}_\tau(V)},\mathcal{X})$. In Fig. 6, the first three lines represent the definition of trajectories in Def. 2, and the last line encodes that each proposition $p\in\Pi$ is stable for the partition induced by V, together with $\Gamma_\Pi(\sigma,\mathcal{S}_{\mathcal{P}_\tau(V)},\mathcal{X})$.

Consider a hybrid automaton H, an STL formula φ, and a bound n for V. A counterexample of an STL formula φ can be encoded as the formula: [8]

$$\Psi_{H,\neg\varphi}^{n,\tau}\ =\ \Psi_H^{n,\tau}(V,\mathcal{X})\wedge\chi_{\neg\varphi}^1\wedge\bigwedge\Delta_{\neg\varphi}(V)\cup\Gamma_{\neg\varphi}(\mathcal{P},\mathcal{X}).$$

If $\Psi_{H,\neg\varphi}^{n,\tau}$ is satisfiable, by construction, there exists a bounded trajectory $\sigma\in H$ with variable points in V and satisfies the conditions $\Gamma_\Pi(\sigma,\mathcal{S}_{\mathcal{P}_\tau(V)},\mathcal{X})$. Thus, by Theorem 2, $\sigma,0\models_\tau\neg\varphi$ holds. Conversely, if there exists a counterexample $\sigma\in H$, with a finite set T of variable points, such that $\sigma,0\models\neg\varphi$. By Theorem 2, there are a finite set V and a signal σ such that $\{\chi_{\neg\varphi}^1\}\cup\Delta_{\neg\varphi}(V)\cup\Gamma_{\neg\varphi}(\mathcal{P},\mathcal{X})$ and $\Gamma_\Pi(\sigma,\mathcal{S}_{\mathcal{P}_\tau(V)},\mathcal{X})$, are satisfied, where $|V|\geq|\mathcal{T}_{\neg\varphi}^{\sigma}(T)|$. Therefore:

[8] For $\mathcal{S}_\mathcal{P}=\{v_i\}_{i\in[N]}$, $v_i=t_i$ for $J_i=\{t_i\}$, and $v_i=(t_i+t_{i+1})/2$ for $J_i=(t_i,t_{i+1})$.

Theorem 3. *For a hybrid automaton H, an STL formula φ, and a bound n, $\Psi^{n,\tau}_{H,\neg\varphi}$ is satisfiable iff $H, 0 \not\models^{k}_{\tau} \varphi$ in such a way that there exists a counterexample with a finite set of variable points T, where $k = |T|$ and $n \geq |\mathcal{T}^{\sigma}_{\neg\varphi}(T)|$.*

5.2 STL Model Checking Algorithm

For bound $n \leq M$, our algorithm in Alg. 1 builds a first-order logic formula to encode a counterexample of φ, which is built for each subformula in a modular way (Line 3). If the encoding is satisfiable, then a counterexample is reported (Line 7). If not, then there is no counterexample of φ up to bounds τ and n. By Theorem 3, our algorithm is refutationally complete for bounded trajectories with a finite number of variable points.

Algorithm 1: Bounded Model Checking Algorithm of STL

Input: Hybrid automaton H, STL formula φ, time bound τ, max bound M
Output: True, or a counterexample of φ

```
1 for n = 1 to M do
2 |  Ψ_H^{n,τ} ⟵ encoding of H's trajectories in Fig. 6;
3 |  foreach ψ ∈ sub(¬φ) do
4 |  |  Γ ⟵ equality condition for χ^i_ψ, i ∈ [2n], in Def. 11;
5 |  |  Δ ⟵ partition condition for ψ in Def. 12;
6 |  if checkSat(Ψ_H ∧ χ^1_{¬φ} ∧ Γ ∧ Δ) is Sat then
7 |  |  return counterexample(result.satisfiableAssignment);
8 return True;
```

We use an SMT solver to check the satisfiability of the encoding (Line 6). $\Delta_{\neg\varphi}(V) \cup \Gamma_{\neg\varphi}(\mathcal{P}, \mathcal{X})$ contains first-order logic formulas in linear real arithmetic. When *flow* conditions are polynomials, $\Psi^{n,\tau}_{H}$ is a first-order formula in nonlinear real arithmetic, supported by SMT solvers such as Z3 [14] and Yices2 [21]. When *flow* conditions involve transcendental functions, the satisfiability of $\Psi^{n,\tau}_{H}$ is in general undecidable, but there are several approximate solvers such as [22, 27].

For a bound n, the size of the entire encoding $\Psi^{n,\tau}_{H,\neg\varphi}$ is $O(|H| \cdot n + |\varphi| \cdot n^2)$. The size of $\Psi^{n,\tau}_{H}$ is $O(|H| \cdot n)$. For $\Delta_{\neg\varphi}$ and $\Gamma_{\neg\varphi}$, the encoding size is $O(|\varphi| \cdot n^2)$, because there is a condition for each subformula $\psi \in \mathrm{sub}(\varphi)$ and for each i-th step, $1 \leq i \leq 2n + 2$, where the size of the condition for $\psi \,\mathbf{U}_I\, \psi'$ is $O(i)$.

Unlike the previous algorithm in [8], a bound parameter n does not directly represent the number of variable points, but the size of a stable partition $\mathcal{T}^{\sigma}_{\neg\varphi}(T)$ for variable points T. As mentioned, the size of $\mathcal{T}^{\sigma}_{\neg\varphi}(T)$ is $O(k \cdot t(\varphi) \cdot 2^{d(\varphi)})$, where $k = |T|$, $t(\varphi)$ is the number of $\mathbf{U}_I$, and $d(\varphi)$ is the nesting depth of $\mathbf{U}_I$. However, n and k are actually incomparable, because the size of $\mathcal{T}^{\sigma}_{\neg\varphi}(T)$ can be much smaller than in the worst case, thanks to the reduction with min^{σ}_{φ}.

6 Experimental Evaluation

To experimentally evaluate the effectiveness of the proposed algorithm, we have implemented our algorithm in the STLMC bounded model checker [2]. We have compared the performance of the new algorithm with the previous algorithm [8] on: (i) bounded model checking of STL properties for hybrid automata; and (ii) bounded satisfiability checking of STL formulas. We have run all experiments on a 16-core 32-thread Intel Xeon 2.8GHz with 256 GB memory. The benchmark models and the experimental results are available in [1].

6.1 STL Model Checking for Hybrid Automata

We consider a number of hybrid automata models, adapted from [5,7,25,42,43]: networked thermostat and water thank systems, autonomous driving of cars, railroad gate controllers, turning an airplain, and load management of batteries. We consider three variants of continuous dynamics: linear dynamics, polynomial dynamics of degree 2, and nonlinear dynamics by (nonlinear) ordinary differential equations. For each model, we consider four different STL formulas of different sizes and complexity. The formulas f_1 and f_2 contain one temporal operator, and f_2 and f_3 contain two nested temporal operators. Examples of STL formulas for the linear and polynomial cases are shown in the following table:

Model	STL Formulas	
Car	$f_1 : \Diamond_{[10,30]}(x_2 - x_1 < 20)$ $f_2 : \Box_{[0,100]}(x_2 \geq x_1)$	$f_3 : \Diamond_{(0,20]}(\Box_{[0,5]}(x_2 - x_1 > 10))$ $f_4 : \Box_{[0,60]}((x_2 - x_1 < 2) \rightarrow \Diamond_{[0,10)}(v_1 \leq 30))$
Thermo	$f_1 : \Diamond_{[0,40]}(x_2 > 24)$ $f_2 : \Box_{[10,30]}(x_1 < 22)$	$f_3 : \Box_{[0,10]}(off_1 \mathbf{U}_{[0,15)}(x_1 < 20))$ $f_4 : \Diamond_{(5,30)}((x_2 > 19)\mathbf{R}_{[0,30]} on_2)$
Water	$f_1 : off_1 \mathbf{U}_{[0,30)}(x_1 < 4)$ $f_2 : \Box_{[0,50]}(x_1 > 2)$	$f_3 : \Box_{[5,30]}(\Diamond_{(0,15)}(x_2 > 6))$ $f_4 : \Diamond_{[0,40]}((x_1 > 3)\mathbf{R}_{[0,40)} on_2)$
Railroad	$f_1 : \Diamond_{[0,20]}(tx < -5)$ $f_2 : \Box_{[0,50)}(tx > 60)$	$f_3 : (bx < 10)\mathbf{U}_{[10,40]}(\Diamond_{[0,20)}(tx > 10))$ $f_4 : \Box_{[0,40]}((bx > 80) \rightarrow \Diamond_{[0,20)}(tx > 10))$
Battery	$f_1 : \Diamond_{[0,100]}(d_1 > 0.2)$ $f_2 : (g_2 > 0.5)\mathbf{R}_{[0,40]}(dead_2)$	$f_3 : \Diamond_{(10,50]}((g_2 \geq 0)\mathbf{U}_{(1,15)}(d_1 < 0.1))$ $f_4 : \Box_{[0,40]}((d_1 > 1) \rightarrow \Diamond_{[0,20)}(g_2 < 6))$
Airplane	$f_1 : (\phi > 0)\mathbf{R}_{[10,20]}(p < 0)$ $f_2 : \Diamond_{[0,100)}(xAIL \geq 0)$	$f_3 : \Box_{[5,35]}((p > 0)\mathbf{U}_{[2,8]}(\beta < 1)))$ $f_4 : \Diamond_{[10,40]}((\beta < -0.2) \rightarrow \Diamond_{[5,10)}(p < 0))$

We have performed model checking of these STL properties up to bound $n = 50$ for linear models, $n = 20$ for polynomial models, and $n = 5$ for nonlinear models with ODEs. We assign different time bounds τ to different models with different continuous dynamics, since τ depends on different model parameters. For each case (H, φ, n), given by a model H, an STL formula φ, and a bound n, we measure the execution times for running STL model checking, including SMT solving by the underlying solver by the new algorithm and the previous algorithm. We use Yices2 [21] and Z3 [14] for linear and polynomial models, and dReal3 [27] for nonlinear ODE models. The timeout is 180 minutes.

The experimental results are summarized in Fig. 7 (for the models with linear and polynomial dynamics) and Fig. 8 (for the models with ODE dynamics). The x-axis shows step bounds n, and the y-axis shows execution times (seconds) in a log scale. Solid lines denote the proposed algorithm, and the dashed lines denote the previous algorithm. Empty shapes indicate that a counterexample is found, and filled shapes indicates that no counterexample is found. The title of each graph has the form *model* (*dynamics*/*formulas*), where for *dynamics*, L, P, and N denote, respectively, linear, polynomial, and (nonlinear) ODE dynamics. Figures 9, 10, and 11 also show the experimental results in tables. In the tables, '-' denotes timeout, CF indicates that a counterexample was found, and NC indicates that no counterexample was found.

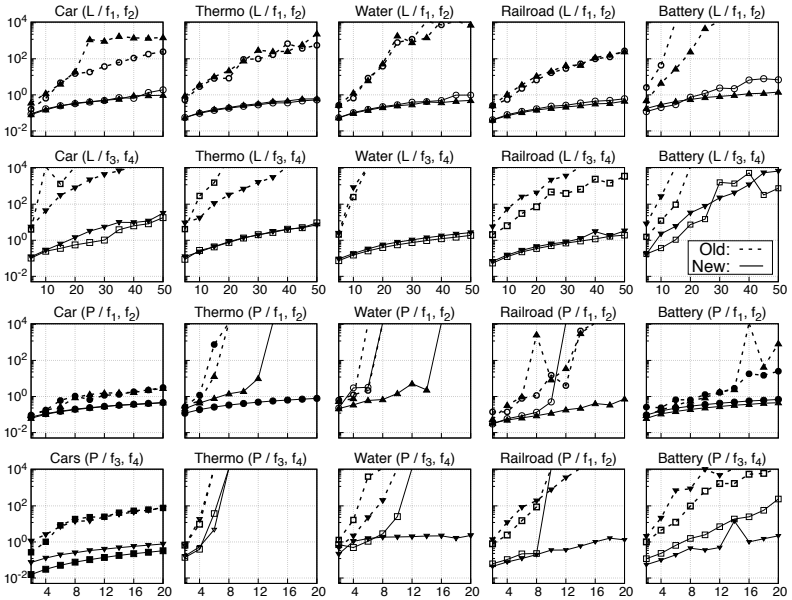
As shown in Figures 7 and 8, our algorithm significantly outperforms the previous algorithm [8] for various SMT solvers. In particular, our algorithm shows much better performance than previous one for complex formulas with nested temporal operators (i.e., f_3 and f_4). For example, consider Water with polynomial dynamics for formula f_4. The new algorithms took only 2.27 seconds for bound $n = 20$ using Yices2, whereas the previous algorithm timed out already for bound $n = 10$ using both of Yices2 and Z3. The performance improvement is due to the much compact size of the encoding provided by our method. E.g., in Fig. 10, for the same case of Water at bound $n = 20$, the size of the encoding by our algorithm is $68,794$, whereas the size by the previous algorithm is $275,555$.

6.2 Bounded Satisfiability Checking of STL

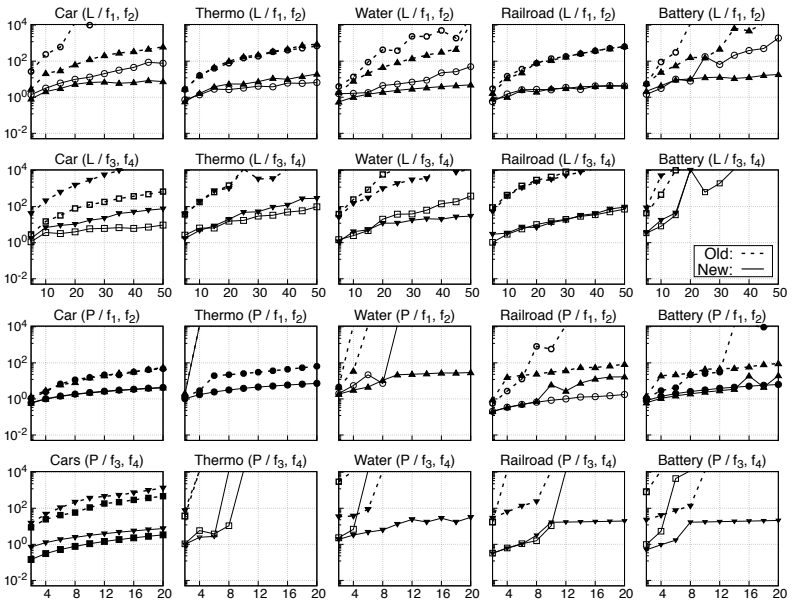
To evaluate the effectiveness of our encoding method, compared to the previous encoding used in [8], we have performed bounded satisfiability checking of STL formulas, up to bound $n = 20$. Note that this experiment does not involve hybrid automata. We consider 30 STL formulas of different sizes and complexity. Specifically, there are 10 formulas for each nesting depth $d = 1, 2, 3$. We use Yices2 and Z3 for the underlying SMT solver. We measure the execution time for checking the bounded satisfiability (including SMT solving), and the size of generated SMT formulas. The timeout is 30 minutes.

Figure 12 summarizes the experimental results. The first and second lines represent execution time with Yices2 and Z3 respectively. The third line represents encoding sizes. The x-axis shows bound n, and the y-axis shows execution times (seconds) and encoding sizes, accordingly, in a log scale. The graph title $d(f)$ denotes the nesting depth. In the graphs, filled boxes indicates our encoding method, and empty boxes indicates the previous method [8]. Figure 13 also shows the experimental results in tables, including the average and standard deviation for the 10 formulas of each nesting depth.

As expected, the encoding generated by our modular method can much easily solved, which is due to the much smaller size of the encoding. This improvement tends to be more apparent for more complex formulas and bigger bounds. E.g., for nesting depth 3 and $k = 20$, using Yices2, the execution time for the new algorithm is 0.77 seconds in average, whereas the execution time for the old algorithm is 1,771.3 seconds in average (8 out of 10 formulas timed out).



(a) Uzing Yices2



(b) Using Z3

Fig. 7. Bounded model checking of STL, where $\bigcirc = f_1$, $\bigtriangleup = f_2$, $\Box = f_3$, and $\bigtriangledown = f_4$.

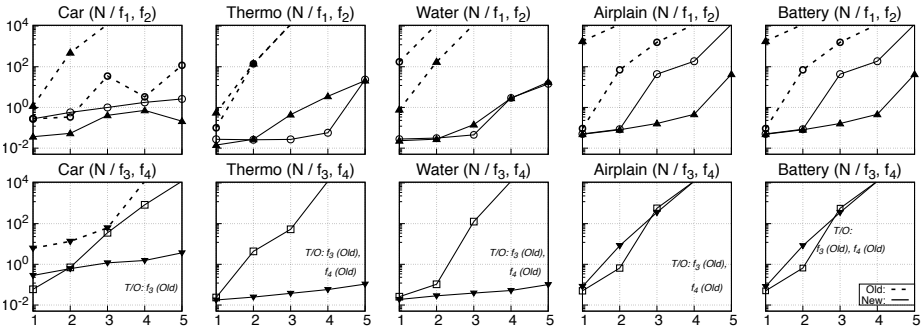


Fig. 8. Bounded model checking of STL (ODE).

7 Related Work

The most closely related work is our previous work [8] that proposes the first refutationally complete bounded model checking algorithm for STL. As briefly discussed in Sec. 1, the algorithm [8] is based on a non-modular translation, which produces a much more complex encoding than the modular translation proposed in this paper. In particular, the non-modular translation in [8] is based on an interval analysis of time domain to construct a "fully stable" STL-GT formula, whose size is quite big and is often a source of the complexity for the encoding. In contrast, our modular translation takes advantage of a complete discretization of signals, and therefore produces a much simpler formula.

Another STL model checking method has been proposed in [45]. Unlike our method, the algorithm in [45] is based on reachable set computation, instead of SMT solving. For a finite number of *concrete* sampled time points, their method reduces the model checking problem of "sampled time" STL formula into reachable-set computation of bounded signals. As mentioned, this technique is incomplete, because only a finite number of sampled time points is considered, and thus the correctness cannot be guaranteed. In contrast, our method is based on a complete discretization of signals, represented as symbolic variables, without any loss of information with respect to STL bounded model checking.

Besides model checking, there are various techniques and tools for analyzing STL properties. STL monitoring and falsification, e.g., [3, 6, 15, 17, 18, 36, 40], checks if concrete signals satisfy STL requirements. They are quite effective for "testing" concrete systems, but cannot guarantee the correctness of the system, because these methods are based on concrete sampling of signals. On the other hand, the reachability analysis of hybrid automata can guarantee the correctness, but only for invariant properties. There are two kinds of techniques for the reachability analysis of hybrid automata: reachable-set computation, such as [4,10,11,13,20,23,26,29], and SMT-based approaches, such as [7,12,22,28,35,47].

		Alg.	Solver	k = 10			k = 20			k = 30			k = 40			k = 50		
				Time (s)	Size (K)	Result	Time (s)	Size (K)	Result	Time (s)	Size (K)	Result	Time (s)	Size (K)	Result	Time (s)	Size (K)	Result
Cars	f_1	Old	Yices2	0.64	23.03	CF	15.02	60.52	CF	37.05	113.0	CF	108.9	180.5	CF	245.4	263.0	CF
			Z3	238.5			-		-	-		-	-		-	-		-
		New	Yices2	0.17	13.78	CF	0.33	27.07	CF	0.48	40.36	CF	0.64	53.65	CF	1.89	66.94	CF
			Z3	3.17			9.83			20.32			45.06			76.53		
	f_2	Old	Yices2	1.20	23.06	NC	18.90	60.56	NC	951.4	113.1	NC	1437.0	180.6	NC	1498.3	263.1	NC
			Z3	20.87			63.06			205.6			339.92			617.58		
		New	Yices2	0.15	13.85	NC	0.34	27.2	NC	0.50	40.55	NC	0.90	53.9	NC	0.95	67.25	NC
			Z3	2.05			5.27			7.01			6.51			7.40		
	f_3	Old	Yices2	-	72.34	-	-	223.9	-	-	456.8	-	-	771.1	-	-	1167	-
			Z3	15.0		CF	79.28		CF	181.49		CF	372.13		CF	679.5		CF
		New	Yices2	0.3	20.95	CF	0.55	55.02	CF	1.02	102.7	CF	6.32	164	CF	18.01	238.8	CF
			Z3	3.7			3.97			6.17			6.17			9.42		
	f_4	Old	Yices2	44.0	120.5	NC	815.8	397.6	NC	4446.3	833.6	NC	-	1428	-	-	2257	-
			Z3	211.2			1563.5			5777.4			-		-	-		-
		New	Yices2	0.3	22.75	NC	1.42	58.21	NC	5.42	107.1	NC	9.68	169.3	NC	31.87	245.0	NC
			Z3	6.9			10.56			22.65			50.36			75.33		
Thermo	f_1	Old	Yices2	2.8	16.54	CF	8.32	47.63	CF	98.71	93.72	CF	690.18	154.8	CF	560.1	230.9	CF
			Z3	16.2			77.15			187.23			406.33			694.8		
		New	Yices2	0.1	7.57	CF	0.18	14.75	CF	0.28	21.93	CF	0.36	29.11	CF	0.53	36.29	CF
			Z3	1.4			2.67			4.13			6.16			6.61		
	f_2	Old	Yices2	3.5	16.51	NC	21.68	47.57	NC	297.70	93.63	NC	256.41	154.7	NC	2406.2	230.8	NC
			Z3	15.8			98.89			223.84			465.86			875.4		
		New	Yices2	0.1	7.62	NC	7615.0	14.84	NC	0.32	22.06	NC	0.46	29.28	NC	0.60	36.5	NC
			Z3	1.6			5.47			7.33			9.96			19.30		
	f_3	Old	Yices2	286.8	81.77	CF	-	269.8	-	-	566.2	-	-	970.0	-	-	1484	-
			Z3	180.2			1480.8		CF	-		-	-		-	-		-
		New	Yices2	0.3	20.48	CF	0.76	65.36	CF	2.04	135	CF	4.08	229.5	CF	9.22	348.8	CF
			Z3	6.6			15.96			30.98			52.57			102.80		
	f_4	Old	Yices2	17.2	82.67	NC	326.3	272.8	NC	1633.7	572.5	NC	-	981.8	-	-	1501	-
			Z3	186.8			1042.3			3407.8			-		-	-		-
		New	Yices2	0.2	20.00	NC	0.83	64.41	NC	1.95	133.6	NC	4.02	227.6	NC	7.18	346.4	NC
			Z3	4.9			20.35			52.71			152.31			294.4		
Water	f_1	Old	Yices2	0.7	19.49	CF	38.54	58.13	CF	1215.7	116.4	CF	7547.3	194.2	CF	-	291.7	-
			Z3	13.4			482.4			2385.0			5046.8			-		-
		New	Yices2	0.1	8.84	CF	0.24	18.49	CF	0.42	29.34	CF	0.52	41.39	CF	1.06	54.64	CF
			Z3	1.7			4.50			6.97			22.72			50.00		
	f_2	Old	Yices2	1.3	16.86	NC	52.86	48.26	NC	795.3	94.68	NC	-	156.1	-	7426.0	232.5	NC
			Z3	7.9			45.59			138.8			318.9		NC	-		-
		New	Yices2	0.1	7.53	NC	0.21	14.67	NC	0.30	21.81	NC	0.39	28.95	NC	0.51	36.09	NC
			Z3	1.0			1.93			2.89			3.90			4.87		
	f_3	Old	Yices2	248.7	66.14	CF	-	211.1	-	-	437.1	-	-	744.1	-	-	1132	-
			Z3	248.9			6077.1		CF	-		-	-		-	-		-
		New	Yices2	0.2	14.56	CF	0.52	42.13	CF	1.08	83.1	CF	1.97	137.5	CF	3.16	205.2	CF
			Z3	2.5			20.00			39.53			144.82			383.01		
	f_4	Old	Yices2	835.8	82.64	NC	-	272.3	-	-	571.1	-	-	979.3	-	-	1497	-
			Z3	151.6			1012.3		NC	3056.6		NC	-		-	-		-
		New	Yices2	0.2	20.26	NC	0.83	65.33	NC	1.71	135.6	NC	3.31	231.1	NC	5.02	351.7	NC
			Z3	4.2			12.49			19.62			21.46			34.00		
Railroad	f_1	Old	Yices2	0.6	14.95	CF	7.06	44.51	CF	28.98	89.07	CF	99.39	148.6	CF	275.2	223.2	CF
			Z3	14.7			77.01			182.9			373.8			642.1		
		New	Yices2	0.1	7.01	CF	0.18	14.89	CF	0.29	23.97	CF	0.49	34.25	CF	0.70	45.73	CF
			Z3	1.5			2.71			3.53			4.22			4.29		
	f_2	Old	Yices2	1.1	15.02	NC	11.09	44.65	NC	43.46	89.28	NC	117.5	148.9	NC	246.7	223.5	NC
			Z3	10.8			79.19			172.5			379.9			663.4		
		New	Yices2	0.1	5.69	NC	0.15	11.05	NC	0.22	16.41	NC	0.33	21.77	NC	0.47	27.13	NC
			Z3	1.0			1.95			3.09			4.02			4.21		
	f_3	Old	Yices2	6.7	113.7	CF	73.02	384.5	CF	396.4	814.9	CF	2439.8	1615	CF	3548.8	2514	CF
			Z3	399.4			2398.1			5301.5			-		-	-		-
		New	Yices2	0.2	12.86	CF	0.43	38.98	CF	0.97	78.7	CF	1.82	132	CF	3.13	198.9	CF
			Z3	3.0			10.51			20.08			36.62			73.56		
	f_4	Old	Yices2	54.3	112.3	NC	436.03	381.2	NC	3720.8	809.2	NC	-	1396	-	-	2208	-
			Z3	426.6			2890.0			8300.8			-		-	-		-
		New	Yices2	0.2	14.35	NC	0.54	41.63	NC	1.09	82.31	NC	3.67	136.4	NC	4.55	203.9	NC
			Z3	3.5			7.12			18.70			42.94			95.78		
Battery	f_1	Old	Yices2	43.3	25.10	CF	-	64.50	-	-	118.9	-	-	188.3	-	-	272.7	-
			Z3	89.8			-		-	-		-	-		-	-		-
		New	Yices2	0.2	15.68	CF	0.73	30.72	CF	2.37	45.76	CF	6.60	60.8	CF	6.80	75.84	CF
			Z3	3.1			7.99			65.72			387.9			1902.5		
	f_2	Old	Yices2	4.5	30.18	NC	242.9	78.98	NC	-	148.6	-	-	237.0	-	-	345.0	-
			Z3	24.5			162.0			468.16		NC	6610.4		NC	-		-
		New	Yices2	0.3	18.64	NC	0.55	36.33	NC	0.82	54.02	NC	1.15	71.71	NC	1.42	89.4	NC
			Z3	26.7			610.1			9925.9			-		-	-		-
	f_3	Old	Yices2	-	92.2	-	-	289.9	-	-	595.5	-	-	1009	-	-	1531	-
			Z3	457.0		CF	-		-	-		-	-		-	-		-
		New	Yices2	0.4	29.70	CF	7.38	82.58	CF	1622.4	159.5	CF	5424.3	260.3	CF	764.83	385.2	CF
			Z3	8.2			-		-	1946.0			-		-	-		-
	f_4	Old	Yices2	210.7	123.0	NC	-	402.3	-	-	840.6	-	-	1438	-	-	2654	-
			Z3	5191.4			-		-	-		-	-		-	-		-
		New	Yices2	2.3	24.44	NC	32.91	60.85	NC	226.0	110.1	NC	1207.0	172.1	NC	6845.9	246.9	NC
			Z3	17.8			10527			-		-	-		-	-		-

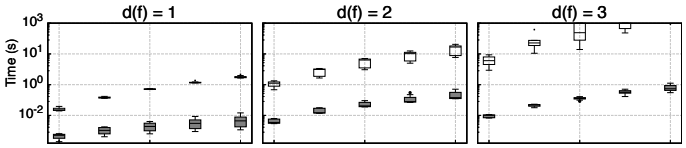
Fig. 9. Bounded model checking of STL (linear dynamics).

		Alg.	Solver	k = 2			k = 6			k = 10			k = 14			k = 20		
				Time (s)	Size (K)	Result	Time (s)	Size (K)	Result	Time (s)	Size (K)	Result	Time (s)	Size (K)	Result	Time (s)	Size (K)	Result
Cars	f_1	Old	Yices2	0.09	2.75	NC	0.60	60.52	NC	0.66	17.7	NC	1.30	28.70	NC	3.16	49.78	NC
			Z3	1.1			6.74			15.63			21.29			46.93		
		New	Yices2	0.07	2.06	NC	0.15	5.22	NC	0.24	8.39	NC	0.35	11.57	NC	0.48	16.32	NC
			Z3	0.61			1.43			2.26			3.07			4.31		
	f_2	Old	Yices2	0.08	2.76	NC	0.27	9.00	NC	1.29	17.64	NC	1.62	28.68	NC	2.82	49.74	NC
			Z3	1.34			6.06			14.8			26.93			54.61		
		New	Yices2	0.06	2.07	NC	0.14	5.25	NC	0.23	8.43	NC	0.33	11.61	NC	0.45	16.38	NC
			Z3	0.58			1.36			2.14			2.92			4.10		
	f_3	Old	Yices2	0.27	7.72	NC	8.32	29.10	NC	23.56	63.50	NC	41.80	110.9	NC	76.88	206.5	NC
			Z3	8.7			41.54			110.3			218.3			467.2		
		New	Yices2	0.02	1.46	NC	0.05	5.70	NC	0.10	12.11	NC	0.17	20.70	NC	0.32	37.66	NC
			Z3	0.1			0.52			1.08			1.83			3.36		
	f_4	Old	Yices2	0.1	2.76	NC	0.6	9.01	NC	0.7	17.66	NC	1.30	28.70	NC	3.16	49.78	NC
			Z3	15.0			107.8			365.4			533.9			1335.9		
		New	Yices2	0.1	2.71	NC	0.19	8.57	NC	0.33	16.58	NC	0.48	26.73	NC	0.76	45.97	NC
			Z3	0.7			1.84			3.15			4.73			7.50		
Thermo	f_1	Old	Yices2	0.26	3.34	NC	760.0	10.44	NC	-	19.94	-	-	31.84	-	-	54.19	-
			Z3	1.26			19.39			25.57		NC	36.18		NC	64.49		NC
		New	Yices2	0.12	2.65	NC	0.26	6.69	NC	0.41	10.73	NC	0.56	14.77	NC	0.79	20.83	NC
			Z3	1.03			2.43			3.84			5.16			7.27		
	f_2	Old	Yices2	0.26	3.72	NC	13.02	10.98	NC	-	20.65	-	-	32.71	-	-	55.31	-
			Z3	2.1			-		-	-		-	-		-	-		-
		New	Yices2	0.21	3.02	NC	0.78	7.20	NC	1.92	11.39	NC	-	15.57	-	-	21.85	-
			Z3	1.72			-		-	-		-	-		-	-		-
	f_3	Old	Yices2	0.67	9.88	CF	-	38.13	-	-	83.72	-	-	146.65	-	-	273.6	-
			Z3	37.09			-		-	-		-	-		-	-		-
		New	Yices2	0.14	2.89	CF	37.55	10.79	CF	-	22.85	-	-	39.07	-	-	71.20	-
			Z3	1.07			3.8			-		-	-		-	-		-
	f_4	Old	Yices2	0.63	9.96	CF	-	38.48	-	-	84.54	-	-	148.1	-	-	276.4	-
			Z3	76.34			-		-	-		-	-		-	-		-
		New	Yices2	0.16	2.77	CF	4.83	10.37	CF	-	22.03	-	-	37.75	-	-	68.96	-
			Z3	1.00			2.77			-		-	-		-	-		-
Water	f_1	Old	Yices2	0.63	3.60	CF	2.17	11.10	CF	-	21.74	-	-	35.52	-	-	62.06	-
			Z3	4.50			-		-	-		-	-		-	-		-
		New	Yices2	0.23	2.89	CF	3.27	6.89	CF	-	11.09	-	-	15.48	-	-	22.42	-
			Z3	2.00			21.67			-		-	-		-	-		-
	f_2	Old	Yices2	0.54	3.38	NC	-	9.87	-	-	18.76	-	-	30.05	-	-	51.49	-
			Z3	4.51			-		-	-		-	-		-	-		-
		New	Yices2	0.22	2.69	NC	0.61	6.12	NC	1.47	9.55	NC	2.33	12.98	NC	-	18.13	-
			Z3	1.83			4.53			21.79			24.89			28.89		NC
	f_3	Old	Yices2	1.38	8.96	CF	4163.6	31.69	CF	-	67.39	-	-	116.0	-	-	213.3	-
			Z3	2966.1			-		-	-		-	-		-	-		-
		New	Yices2	0.71	2.66	CF	1.10	8.22	CF	25.50	15.93	CF	-	25.78	-	-	44.58	-
			Z3	2.32			-		-	-		-	-		-	-		-
	f_4	Old	Yices2	0.53	10.36	NC	22.48	38.72	NC	-	84.55	-	-	147.8	-	-	275.6	-
			Z3	34.17			87.60			-		-	-		-	-		-
		New	Yices2	0.22	3.22	NC	1.61	10.74	NC	2.04	22.28	NC	2.13	37.86	NC	2.27	68.79	NC
			Z3	1.88			4.74			13.07			17.22			31.42		
Railroad	f_1	Old	Yices2	0.14	2.24	CF	0.78	7.73	CF	15.22	15.61	CF	4339.2	25.90	CF	-	45.82	-
			Z3	0.56			12.54			581.4			-		-	-		-
		New	Yices2	0.03	1.55	CF	0.08	3.98	CF	0.50	6.40	CF	-	8.826	-	-	12.46	-
			Z3	0.20			0.50			0.84			1.27		CF	1.75		CF
	f_2	Old	Yices2	0.05	2.23	NC	0.96	7.70	NC	8.45	15.56	NC	3128.3	25.83	NC	-	45.72	-
			Z3	0.91			20.08			30.46			55.37			82.64		NC
		New	Yices2	0.03	1.54	NC	0.07	3.95	NC	0.12	6.35	NC	0.23	8.76	NC	0.73	12.36	NC
			Z3	0.20			0.48			5.99			7.51			16.68		
	f_3	Old	Yices2	0.79	12.04	CF	15.13	50.41	CF	-	114.3	-	-	203.8	-	-	385.8	-
			Z3	17.47			-		-	-		-	-		-	-		-
		New	Yices2	0.04	2.33	CF	0.17	7.69	CF	-	15.23	-	-	24.94	-	-	43.59	-
			Z3	0.33			1.08			10.74		CF	-		-	-		-
	f_4	Old	Yices2	1.36	11.7	NC	83.51	49.51	NC	795.6	112.8	NC	-	201.5	-	-	382.3	-
			Z3	28.92			134.3			-		-	-		-	-		-
		New	Yices2	0.04	2.32	NC	0.12	7.59	NC	0.35	15.01	NC	0.58	24.58	NC	1.30	42.94	NC
			Z3	0.32			1.03			16.70			17.68			19.60		
Battery	f_1	Old	Yices2	0.26	5.07	NC	0.66	15.48	NC	1.28	28.28	NC	2.86	43.49	NC	25.25	70.79	NC
			Z3	1.09			4.17			25.40			-		-	-		-
		New	Yices2	0.09	4.38	NC	0.22	11.73	NC	0.36	19.07	NC	0.49	26.42	NC	0.68	37.43	NC
			Z3	0.80			1.96			3.14			4.43			6.28		
	f_2	Old	Yices2	0.09	5.26	NC	0.32	16.52	NC	0.85	30.93	NC	2.42	48.48	NC	842.0	80.69	NC
			Z3	1.4			22.2			43.97			49.22			90.66		
		New	Yices2	0.06	4.35	NC	0.15	11.66	NC	0.24	18.98	NC	0.33	26.30	NC	0.45	37.27	NC
			Z3	0.61			1.47			2.39			3.43			19.67		
	f_3	Old	Yices2	1.01	12.60	CF	12.33	45.43	CF	658.1	95.53	CF	1736.9	162.9	CF	-	296.4	-
			Z3	825.4			-		-	-		-	-		-	-		-
		New	Yices2	0.12	5.44	CF	0.64	17.49	CF	2.5	33.48	CF	18.6	53.41	CF	239.7	90.67	CF
			Z3	0.98			4439.6			-		-	-		-	-		-
	f_4	Old	Yices2	2.08	13.83	NC	707.3	55.70	NC	10687	123.0	NC	-	215.8	-	-	402.6	-
			Z3	22.6			75.90			-		-	-		-	-		-
		New	Yices2	0.05	4.47	NC	0.19	13.78	NC	0.35	25.23	NC	14.42	38.82	NC	2.18	63.23	NC
			Z3	0.49			1.70			17.45			18.80			21.09		

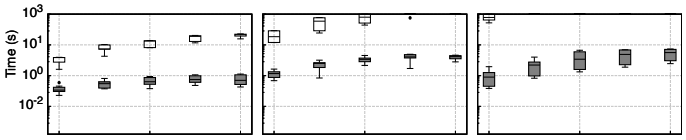
Fig. 10. Bounded model checking of STL (polynomial dynamics).

		Alg.	k = 1			k = 2			k = 3			k = 4			k = 5		
			Time (s)	Size (K)	Result	Time (s)	Size (K)	Result	Time (s)	Size (K)	Result	Time (s)	Size (K)	Result	Time (s)	Size (K)	Result
Cars	f_1	Old	0.28	1.02	CF	0.34	1.87	CF	34.53	2.87	CF	3.32	4.02	CF	117.6	5.32	CF
		New	0.28	0.72	CF	0.57	1.18	CF	1.00	1.63	CF	1.77	2.08	CF	2.61	2.53	CF
	f_2	Old	1.19	1.16	NC	488.0	2.16	NC	-	3.35	-	-	4.73	-	-	6.31	-
		New	0.04	0.81	NC	0.06	1.32	NC	0.41	1.82	NC	0.72	2.33	NC	0.21	2.83	NC
	f_3	Old	6.19	4.52	CF	13.4	7.83	CF	61.6	11.94	CF	-	16.87	-	-	22.61	-
		New	0.06	0.83	CF	0.71	1.52	CF	35.49	2.34	CF	836.5	3.30	CF	-	-	-
	f_4	Old	-	4.97	-	-	8.96	-	-	14.03	-	-	20.20	-	-	27.45	-
		New	0.29	0.93	NC	0.61	1.80	NC	1.18	2.92	NC	1.55	4.29	NC	3.66	5.91	NC
Thermo	f_1	Old	0.10	0.57	CF	138.5	1.15	CF	-	1.88	-	-	2.76	-	-	3.78	-
		New	0.03	0.26	CF	0.03	0.46	CF	0.03	0.64	CF	0.06	0.82	CF	23.3	1.00	CF
	f_2	Old	0.55	0.65	NC	142.9	1.34	NC	-	2.23	-	-	3.31	-	-	4.59	-
		New	0.02	0.30	NC	0.03	0.50	NC	0.44	0.70	NC	3.52	0.90	NC	20.8	1.11	NC
	f_3	Old	-	4.45	-	-	8.14	-	-	12.91	-	-	18.76	-	-	25.71	-
		New	0.03	0.47	CF	4.32	1.10	CF	52.48	1.99	CF	-	3.15	-	-	4.57	-
	f_4	Old	-	5.24	-	-	10.17	-	-	16.68	-	-	24.79	-	-	34.48	-
		New	0.02	0.45	NC	0.03	0.95	NC	0.04	1.58	NC	0.06	2.37	NC	0.11	3.29	NC
Water	f_1	Old	174.0	0.57	CF	-	1.15	-	-	1.87	-	-	2.74	-	-	3.76	-
		New	0.03	0.28	CF	0.03	0.46	CF	0.05	0.64	CF	2.94	0.82	CF	14.29	1.00	CF
	f_2	Old	0.78	0.65	NC	172.2	1.34	NC	-	2.22	-	-	3.30	-	-	4.57	-
		New	0.03	0.36	NC	0.03	0.63	NC	0.15	0.90	NC	2.83	1.19	NC	17.54	1.50	NC
	f_3	Old	-	4.49	-	-	8.21	-	-	13.01	-	-	18.91	-	-	25.91	-
		New	0.03	0.47	CF	0.11	1.11	CF	125.1	2.01	CF	-	3.18	-	-	4.61	-
	f_4	Old	-	5.26	-	-	10.21	-	-	16.75	-	-	24.88	-	-	34.61	-
		New	0.02	0.39	NC	0.03	0.82	NC	0.04	1.39	NC	0.06	2.09	NC	0.11	2.92	NC
Airplain	f_1	Old	0.10	2.60	CF	70.4	4.77	CF	1578.1	7.13	CF	-	9.69	-	-	12.45	-
		New	0.05	2.25	CF	0.09	3.93	CF	43.10	5.61	CF	185.5	7.29	CF	-	8.96	-
	f_2	Old	1809.2	2.46	NC	-	4.48	-	-	6.64	-	-	8.96	-	-	11.43	-
		New	0.05	2.16	NC	0.08	3.79	NC	0.17	5.41	NC	0.47	7.04	NC	42.56	8.67	NC
	f_3	Old	-	6.43	-	-	11.61	-	-	17.87	-	-	25.21	-	-	33.65	-
		New	0.06	2.36	CF	0.66	4.41	CF	569.8	6.70	CF	-	9.25	-	-	12.04	-
	f_4	Old	-	7.24	-	-	13.69	-	-	21.72	-	-	31.35	-	-	42.58	-
		New	0.09	2.34	NC	8.23	4.24	NC	352.7	6.27	NC	-	8.44	-	-	10.75	-
Battery	f_1	Old	-	1.65	-	-	3.06	-	-	4.61	-	-	6.31	-	-	8.17	-
		New	0.05	1.36	CF	0.07	2.37	CF	0.14	3.38	CF	0.17	4.39	CF	15.61	5.41	CF
	f_2	Old	0.06	1.66	NC	0.12	3.07	NC	-	4.62	-	-	6.33	-	-	8.18	-
		New	0.03	1.36	NC	0.05	2.38	NC	0.09	3.39	NC	2.27	4.41	NC	37.52	5.42	NC
	f_3	Old	-	5.15	-	-	9.00	-	-	13.67	-	-	19.14	-	-	25.42	-
		New	0.04	1.46	CF	0.08	2.71	CF	-	4.09	-	-	5.60	-	-	7.25	-
	f_4	Old	-	5.60	-	-	10.14	-	-	15.75	-	-	22.45	-	-	30.22	-
		New	0.04	1.55	NC	214.2	2.98	NC	6188.4	4.65	NC	-	6.57	-	-	8.73	-

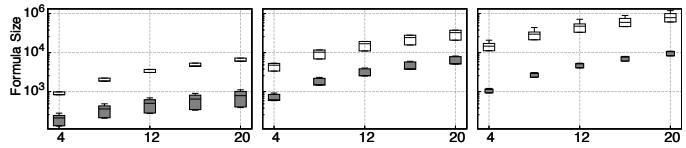
Fig. 11. Bounded model checking of STL (ODE dynamics).



(a) Execution time (Using Yices2)



(b) Execution time (Using Z3)



(c) Encoding size

Fig. 12. Bounded satisfiability of STL

Nesting depth	Alg.	k = 4			k = 8			k = 12			k = 16			k = 20		
		AVG (s)	STDEV	# t.o.	AVG (s)	STDEV	# t.o.	AVG (s)	STDEV	# t.o.	AVG (s)	STDEV	# t.o.	AVG (s)	STDEV	# t.o.
1	**Old**	0.06	0.01	0	0.24	0.02	0	0.66	0.02	0	1.48	0.14	0	2.90	0.42	0
	New	0.002	0.001	0	0.005	0.002	0	0.008	0.004	0	0.012	0.007	0	0.016	0.011	0
2	**Old**	1.27	0.39	0	5.77	2.17	0	17.44	7.9	0	39.75	19.21	0	83.06	42.66	0
	New	0.014	0.004	0	0.049	0.015	0	0.108	0.035	0	0.210	0.103	0	0.34	0.16	0
3	**Old**	21.1	11.4	0	380.1	546.5	1	819.1	721.1	3	1407.3	478.5	5	1771.3	61.3	8
	New	0.026	0.004	0	0.094	0.012	0	0.21	0.033	0	0.46	0.11	0	0.77	0.28	0

(a) Execution time (Using Yices2)

Nesting depth	Alg.	k = 4			k = 8			k = 12			k = 16			k = 20		
		AVG (s)	STDEV	# t.o.	AVG (s)	STDEV	# t.o.	AVG (s)	STDEV	# t.o.	AVG (s)	STDEV	# t.o.	AVG (s)	STDEV	# t.o.
1	**Old**	7.89	2.79	0	33.00	12.00	0	57.13	21.31	0	110.16	33.89	0	146.13	23.61	0
	New	0.24	0.11	0	0.45	0.19	0	0.61	0.27	0	0.77	0.33	0	0.76	0.42	0
2	**Old**	150.0	85.57	0	729.3	440.6	0	1201.8	564.6	3	1669.6	275.9	8	1790.0	31.6	9
	New	1.42	0.61	0	4.38	1.85	0	8.13	2.88	0	10.33	3.87	0	10.41	2.26	0
3	**Old**	1293.3	475.0	4	1800.0	0.0	10	1800.0	0.0	10	1800.0	0.0	10	1800.0	0.0	10
	New	1.19	1.04	0	4.19	3.43	0	10.62	8.81	0	13.46	9.18	0	16.31	9.39	0

(b) Execution time (Using Z3)

Nesting depth	Alg.	k = 4		k = 8		k = 12		k = 16		k = 20	
		AVG (s)	STDEV	AVG (s)	STDEV	AVG (s)	STDEV	AVG (s)	STDEV	AVG (s)	STDEV
1	**Old**	2301.1	278.4	7225.1	933.5	14843.5	1968.5	25156.3	3383.6	38163.5	5178.8
	New	290.6	117.9	616.6	281.7	990.6	494.6	1412.6	757.2	1882.6	1069.8
2	**Old**	22277.9	5848.9	65615.5	19600.3	131865.1	41315.3	221026.7	70994.1	333100.3	108636.5
	New	1673.7	469.1	6153.7	1900.6	13456.1	4298.7	23580.9	7663.3	36528.1	11994.4
3	**Old**	122265.9	41375.3	330179.5	128883.9	639821.1	264438.7	958379.2	359015.0	1423197.0	543698.8
	New	2994.8	467.9	11234.8	1897.3	24738.8	4293.1	43506.8	7655.1	67538.8	11983.3

(c) Encoding size

Fig. 13. Bounded satisfiability of STL

8 Concluding Remarks

We have presented a new SMT-based bounded model checking algorithm for STL properties of hybrid automata. Our algorithm is based on a new theoretical foundation to completely characterize the bounded satisfaction of STL formulas using a discretization of continuous signals. Based on complete discretization of signals, we have developed a modular translation method to encode the bounded satisfiability of STL in a decidable fragment of first-order logic, built separately for each subformula in a modular way. We have leveraged our translation method to obtain an efficient SMT-based bounded model checking algorithm for STL that is still refutationlly complete for bounded signals, but that produces much simpler encodings and results in much better performance. We demonstrated the efficiency of our algorithm on a number of hybrid automata models with different continuous dynamics, and showed that our algorithm significantly outperforms the previous refutationlly complete algorithm proposed in [8].

References

1. Supplementary material: proofs of lemmas, benchmarks, and implementation, https://stlmc.github.io/tacas2021
2. stlMC: STL bounded model checker for hybrid automata, https://stlmc.github.io
3. Abbas, H., Fainekos, G., Sankaranarayanan, S., Ivančić, F., Gupta, A.: Probabilistic temporal logic falsification of cyber-physical systems. ACM Transactions on Embedded Computing Systems **12**(2s), 95 (2013)
4. Althoff, M.: Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In: Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control. pp. 173–182. HSCC '13, ACM, New York, NY, USA (2013). https://doi.org/10.1145/2461328.2461358, http://doi.acm.org/10.1145/2461328.2461358
5. Alur, R.: Principles of cyber-physical systems. The MIT Press (2015)
6. Annpureddy, Y., Liu, C., Fainekos, G., Sankaranarayanan, S.: S-taliro: A tool for temporal logic falsification for hybrid systems, Lecture Notes in Computer Science, vol. 6605. Springer (2011)
7. Bae, K., Gao, S.: Modular SMT-based analysis of nonlinear hybrid systems. In: Proc. FMCAD. pp. 180–187. IEEE (2017)
8. Bae, K., Lee, J.: Bounded model checking of signal temporal logic properties using syntactic separation. Proceedings of the ACM on Programming Languages **3**(POPL), 1–30 (2019)
9. Bak, S., Duggirala, P.S.: Hylaa: A tool for computing simulation-equivalent reachability for linear systems. In: Proc. HSCC. pp. 173–178 (2017)
10. Bak, S., Duggirala, P.S.: Simulation-equivalent reachability of large linear systems with inputs. In: CAV. pp. 401–420. Springer (2017)
11. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: An analyzer for non-linear hybrid systems. In: Proc. CAV. pp. 258–263. Springer (2013)
12. Cimatti, A., Griggio, A., Mover, S., Tonetta, S.: HyComp: an SMT-based model checker for hybrid systems. In: Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science, vol. 9035, pp. 52–67. Springer (2015)

13. Dang, T., Testylier, R.: Reachability analysis for polynomial dynamical systems using the bernstein expansion. Reliable Computing **17**(2), 128–152 (2012)
14. De Moura, L., Bjørner, N.: Z3: an efficient smt solver. In: Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. TACAS'08/ETAPS'08, Springer-Verlag, Berlin, Heidelberg (2008), http://dl.acm.org/citation.cfm?id=1792734.1792766
15. Deshmukh, J.V., Donzé, A., Ghosh, S., Jin, X., Juniwal, G., Seshia, S.A.: Robust online monitoring of signal temporal logic. Formal Methods in System Design **51**(1), 5–30 (2017)
16. Dokhanchi, A., Hoxha, B., Fainekos, G.: Metric interval temporal logic specification elicitation and debugging. In: Proceedings of the 2015 ACM/IEEE International Conference on Formal Methods and Models for Codesign. pp. 70–79. MEMOCODE '15, IEEE Computer Society, Washington, DC, USA (2015). https://doi.org/10.1109/MEMCOD.2015.7340472, https://doi.org/10.1109/MEMCOD.2015.7340472
17. Donzé, A.: Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: Proc. CAV. pp. 167–170. Springer (2010)
18. Donzé, A., Ferrère, T., Maler, O.: Efficient robust monitoring for stl. In: Sharygina, N., Veith, H. (eds.) Computer Aided Verification. pp. 264–279. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
19. Duggirala, P.S., Fan, C., Mitra, S., Viswanathan, M.: Meeting a powertrain verification challenge. In: International Conference on Computer Aided Verification. pp. 536–543. Springer (2015)
20. Duggirala, P.S., Mitra, S., Viswanathan, M.: Verification of annotated models from executions. In: Proceedings of the Eleventh ACM International Conference on Embedded Software. pp. 26:1–26:10. EMSOFT '13, IEEE Press, Piscataway, NJ, USA (2013), http://dl.acm.org/citation.cfm?id=2555754.2555780
21. Dutertre, B.: Yices 2.2. In: Biere, A., Bloem, R. (eds.) CAV. LNCS, vol. 8559, pp. 737–744. Springer (July 2014)
22. Eggers, A., Ramdani, N., Nedialkov, N.S., Fränzle, M.: Improving the sat modulo ode approach to hybrid systems analysis by combining different enclosure methods. Software & Systems Modeling **14**(1), 121–148 (2015)
23. Fan, C., Qi, B., Mitra, S., Viswanathan, M., Duggirala, P.S.: Automatic reachability analysis for nonlinear hybrid models with c2e2. In: Chaudhuri, S., Farzan, A. (eds.) Computer Aided Verification. pp. 531–538. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
24. Ferrère, T., Maler, O., Ničković, D., Pnueli, A.: From real-time logic to timed automata. Journal of the ACM (JACM) **66**(3), 1–31 (2019)
25. Fox, M., Long, D., Magazzeni, D.: Plan-based policies for efficient multiple battery load management. Journal of Artificial Intelligence Research **44**, 335–382 (2012)
26. Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: SpaceEx: Scalable verification of hybrid systems. In: Proc. CAV. Lecture Notes in Computer Science, vol. 6806. Springer (2011)
27. Gao, S., Kong, S., Clarke, E.M.: dReal: An SMT solver for nonlinear theories over the reals. In: Proc. CADE. Lecture Notes in Computer Science, vol. 7898. Springer (2013)
28. Gao, S., Kong, S., Clarke, E.M.: Satisfiability modulo ODEs. In: Proc. FMCAD. IEEE (2013)
29. Girard, A., Pappas, G.J.: Verification using simulation. In: Proc. HSCC, Lecture Notes in Computer Science, vol. 3927, pp. 272–286. Springer (2006)

30. Goldman, R.P., Bryce, D., Pelican, M.J., Musliner, D.J., Bae, K.: A hybrid architecture for correct-by-construction hybrid planning and control. In: Proceedings of the 8th International Symposium on NASA Formal Methods - Volume 9690. pp. 388–394. NFM 2016, Springer-Verlag New York, Inc., New York, NY, USA (2016)
31. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? Journal of computer and system sciences **57**(1), 94–124 (1998)
32. Henzinger, T.: The theory of hybrid automata. In: Verification of Digital and Hybrid Systems, NATO ASI Series, vol. 170, pp. 265–292. Springer (2000)
33. Ho, H.M., Ouaknine, J., Worrell, J.: Online monitoring of metric temporal logic. In: Bonakdarpour, B., Smolka, S.A. (eds.) Runtime Verification. pp. 178–192. Springer, Berlin, Heidelberg (2014)
34. Hunter, P., Ouaknine, J., Worrell, J.: Expressive completeness for metric temporal logic. In: Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science. pp. 349–357. LICS '13, IEEE Computer Society, Washington, DC, USA (2013). https://doi.org/10.1109/LICS.2013.41, http://dx.doi.org/10.1109/LICS.2013.41
35. Ishii, D., Ueda, K., Hosobe, H.: An interval-based sat modulo ode solver for model checking nonlinear hybrid systems. Int. J. Softw. Tools Technol. Transf. **13**(5), 449–461 (Oct 2011)
36. Jakšić, S., Bartocci, E., Grosu, R., Ničković, D.: Quantitative monitoring of stl with edit distance. In: Falcone, Y., Sánchez, C. (eds.) Runtime Verification. pp. 201–218. Springer-Verlag, Berlin, Heidelberg (2016)
37. Jin, X., Deshmukh, J.V., Kapinski, J., Ueda, K., Butts, K.: Powertrain control verification benchmark. In: Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control. pp. 253–262. HSCC '14, ACM, New York, NY, USA (2014). https://doi.org/10.1145/2562059.2562140, http://doi.acm.org/10.1145/2562059.2562140
38. LaValle, S.M.: Planning algorithms. Cambridge university press (2006)
39. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Lecture Notes in Computer Science, vol. 3253, pp. 152–166. Springer (2004)
40. Ničković, D., Lebeltel, O., Maler, O., Ferrère, T., Ulus, D.: Amt 2.0: qualitative and quantitative trace analysis with extended signal temporal logic. In: Tools and Algorithms for the Construction and Analysis of Systems. pp. 303–319. Springer Berlin Heidelberg, Berlin, Heidelberg (2018)
41. Ouaknine, J., Worrell, J.: Some recent results in metric temporal logic. In: Proceedings of the 6th International Conference on Formal Modeling and Analysis of Timed Systems. pp. 1–13. FORMATS '08, Springer-Verlag, Berlin, Heidelberg (2008)
42. Platzer, A.: Differential dynamic logic for hybrid systems. Journal of Automated Reasoning **41**(2), 143–189 (2008)
43. Raisch, J., Klein, E., Meder, C., Itigin, A., O'Young, S.: Approximating automata and discrete control for continuous systems — two examples from process control. In: Hybrid systems V. pp. 279–303. Springer (1999)
44. Raman, V., Donzé, A., Sadigh, D., Murray, R.M., Seshia, S.A.: Reactive synthesis from signal temporal logic specifications. In: Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control. pp. 239–248. HSCC '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2728606.2728628, http://doi.acm.org/10.1145/2728606.2728628

45. Roehm, H., Oehlerking, J., Heinz, T., Althoff, M.: Stl model checking of continuous and hybrid systems. In: International Symposium on Automated Technology for Verification and Analysis. pp. 412–427. Springer (2016)
46. Roohi, N., Kaur, R., Weimer, J., Sokolsky, O., Lee, I.: Parameter invariant monitoring for signal temporal logic. In: Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (Part of CPS Week). pp. 187–196. HSCC '18, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3178126.3178140, http://doi.acm.org/10.1145/3178126.3178140
47. Tiwari, A.: Time-aware abstractions in hybridsal. In: Computer Aided Verification. Lecture Notes in Computer Science, vol. 9206, pp. 504–510. Springer (2015)

A Proofs of Lemmas

Lemma 1. *The* future-reach $fr(\varphi)$ *of an STL formula* φ *is defined by:*

$$\begin{aligned} fr(p) &= 0 & fr(\varphi \wedge \varphi') &= \max(fr(\varphi), fr(\varphi')) \\ fr(\neg\varphi) &= fr(\varphi) & fr(\varphi \,\mathbf{U}_I\, \varphi') &= \sup(I) + \max(fr(\varphi), fr(\varphi')). \end{aligned}$$

For any bound $\tau > t + fr(\varphi)$, $\sigma, t \models_\tau \varphi$ *iff* $\sigma, t \models_\infty \varphi$, *where* $\mathrm{dom}(\sigma) \supseteq [0, \tau)$.

Proof. The proof is by structural induction on φ.
– $(\varphi = p)$: $\sigma, t \models_\tau p$ iff $t < \tau$ and $p(\sigma(t)) = \mathit{true}$ iff, because $t = t + fr(p) < \tau$, $t < \infty$ and $p(\sigma(t)) = \mathit{true}$ iff $\sigma, t \models_\infty p$.
– $(\varphi = \neg\phi)$: $\sigma, t \models_\tau \neg\phi$ iff $\sigma, t \not\models_\tau \phi$ iff, because $\tau > t + fr(\neg\phi) = t + fr(\phi)$, by induction hypothesis, iff $\sigma, t \not\models_\infty \phi$ iff $\sigma, t \models_\infty \neg\phi$.
– $(\varphi = \phi \wedge \phi')$: Since $\tau > t + fr(\phi \wedge \phi') = t + \max(fr(\phi), fr(\phi'))$, $\tau > t + fr(\phi)$ and $\tau > t + fr(\phi')$. Therefore, $\sigma, t \models_\tau \phi \wedge \phi'$ iff $\sigma, t \models_\tau \phi$ and $\sigma, t \models_\tau \phi'$ iff, by induction hypothesis, $\sigma, t \models_\infty \phi$ and $\sigma, t \models_\infty \phi'$ iff $\sigma, t \models_\infty \phi \wedge \phi'$.
– $(\varphi = \phi \,\mathbf{U}_I\, \phi')$: Since $\tau > t + fr(\phi \,\mathbf{U}_I\, \phi') = t + \sup(I) + \max(fr(\phi), fr(\phi'))$, for any $u \le t + \sup(I)$, $\tau > u + fr(\phi)$ and $\tau > u + fr(\phi')$. Let $t' - t \in I$. Clearly, $t' \le t + \sup(I)$, and for $t'' \in [t, t']$, $t'' \le t + \sup(I)$. Therefore, $\sigma, t \models_\tau \phi \,\mathbf{U}_I\, \phi'$ iff $(\exists t' \ge t)\ \ t' < \tau$, $t' - t \in I$, $\sigma, t' \models_\tau \phi'$, $(\forall t'' \in [t, t'])\ \sigma, t'' \models_\tau \phi$ iff, by the above observation and induction hypothesis, $(\exists t' \ge t)\ \ t' < \infty$, $t' - t \in I$, $\sigma, t' \models_\infty \phi'$, $(\forall t'' \in [t, t'])\ \sigma, t'' \models_\infty \phi$ iff $\sigma, t \models_\infty \phi \,\mathbf{U}_I\, \phi'$. □

Lemma 2. *Given a signal* σ, *a proposition* p, *and an interval* $J \subseteq \mathrm{dom}(\sigma)$*: (1) If* J *is open and* p *is stable for* J, *then there is no variable point in* J. *(2) If there is no variable point in* J, *then* p *is stable for* J.

Proof. (1) Each time point $t \in J$ is not a variable point, because t has a stable open neighborhood in J, namely, J itself.
(2) Suppose that p is not stable for J. Then, there exist different time points $u < u'$ in J such that $p(\sigma(u)) \neq p(\sigma(u'))$. Let

$$K_u = \{t \in J \mid p(\sigma(t)) = p(\sigma(u))\}.$$

Observe that $u \le \sup(K_u) \le u'$. Therefore, $\sup(K_u) \in J$. Furthermore, for any $t \in J$ with $t > \sup(K_u)$, $p(\sigma(t)) = p(\sigma(u'))$, but any open neighborhood of $\sup(K_u)$ intersects with K_u. Therefore, $\sup(K_u)$ is not a variable point. □

Lemma 3. *For a signal* σ *and an interval* $J \subseteq \mathrm{dom}(\sigma)$*: (1)* φ *is stable for* J *iff* $\neg\varphi$ *is stable for* J. *(2) If* φ *and* φ' *are stable for* J, *then* $\varphi \wedge \varphi'$ *is stable for* J.

Proof. (1) φ is stable for J, iff $\sigma, u \models_\tau \varphi \iff \sigma, u' \models_\tau \varphi$ for any $u, u' \in J$, iff $\sigma, u \not\models_\tau \varphi \iff \sigma, u' \not\models_\tau \varphi$ for any $u, u' \in J$, iff $\sigma, u \models_\tau \neg\varphi \iff \sigma, u' \models_\tau \neg\varphi$ for any $u, u' \in J$, iff $\neg\varphi$ is stable for J.
(2) Suppose that φ and φ' are stable for J. Then, $\sigma, u \models_\tau \varphi \iff \sigma, u' \models_\tau \varphi$, and $\sigma, u \models_\tau \varphi' \iff \sigma, u' \models_\tau \varphi'$, for any $u, u' \in J$. Therefore, $\sigma, u \models_\tau \varphi$ and $\sigma, u \models_\tau \varphi'$ iff $\sigma, u' \models_\tau \varphi$ and $\sigma, u' \models_\tau \varphi'$. Consequently, $\sigma, u \models_\tau \varphi \wedge \varphi'$ iff $\sigma, u' \models_\tau \varphi \wedge \varphi'$, for any $u, u' \in J$. □

Lemma 12. *[8, Lemma 4.3] For intervals I and J of nonnegative real numbers, $t \in J - I$ iff $(\exists t' \geq t)\ t' \in J$ and $t' \in t + I$.*

Proof. ($\Rightarrow$) By definition, $t \in J - I$ iff $t = j - i$ for some $j \in J$ and $i \in I$, where $i, j \geq 0$. Observe that $j \geq t$ and $j = t + i \in t + I$.
($\Leftarrow$) Since $t' \in t + I$, $t' = t + i$ for some $i \in I$. Since $t' \in J$, $t = t' - i \in J - I$. □

Lemma 13. *For intervals K, J, and I, if either $K \subseteq J - I$ or $K \subseteq (J - I)^{\complement}$ holds, then for any $u_1, u_2 \in K$, $u_1 \in J - I$ iff $u_2 \in J - I$.*

Proof. Suppose $u_1 \in J - I$. Since $u_1 \in K$, $K \cap (J - I) \neq \emptyset$. Because either $K \subseteq J - I$ or $K \subseteq (J - I)^{\complement}$ holds, we must have $K \subseteq J - I$. As a consequence, $u_2 \in K \subseteq J - I$. The other direction is exactly the same. □

Lemma 4. *An STL formula $\varphi\, \mathbf{U}_I\, \varphi'$ is stable for a partition $\mathcal{Q} = \{K_j\}_{j \in [M]}$ of time domain $[0, \tau)$, if: (i) φ and φ' are stable for a partition $\mathcal{P} = \{J_i\}_{i \in [N]}$, (ii) either $K_j \subseteq J_i - I$ or $K_j \subseteq (J_i - I)^{\complement}$ for any K_j and J_i, and (iii) $\mathcal{Q} \sqsubseteq \mathcal{P}$.*

Proof. Let $u_1, u_2 \in K_j$. By definition, $\sigma, u_1 \models_\tau \varphi\, \mathbf{U}_I\, \varphi'$ iff $(\exists t'_1 \geq u_1)\ t'_1 < \tau$, $t'_1 \in u_1 + I$, $\sigma, t'_1 \models_\tau \varphi'$, and $(\forall t''_1 \in [u_1, t'_1])\ \sigma, t''_1 \models_\tau \varphi$. Suppose $t'_1 \in J_i$. Then:

$$\begin{aligned} & (\exists t'_1 \geq u_1)\ t'_1 < \tau,\ t'_1 \in u_1 + I, \text{ and } t'_1 \in J_i \\ \text{iff } & (\exists t'_1 \geq u_1)\ t'_1 \in J_i \cap [0, \tau) \text{ and } t'_1 \in u_1 + I \\ \text{iff } & u_1 \in (J_i \cap [0, \tau)) - I \text{ iff } u_1 \in J_i - I, \end{aligned}$$

because of Lemma 12 and $J_i \subseteq [0, \tau)$. By Lemma 13, $u_2 \in J_i - I$. Similarly, $u_2 \in J_i - I$ iff $(\exists t'_2 \geq u_2)\ t'_2 < \tau$, $t'_2 \in u_2 + I$, and $t'_2 \in J_i$. Thus, $t'_1, t'_2 \in J_i$. Since φ and φ' are stable for $\mathcal{P}$, $\sigma, t'_1 \models_\tau \varphi'$ iff $\sigma, t'_2 \models_\tau \varphi'$. Since $\mathcal{Q} \sqsubseteq \mathcal{P}$, u_1 and u_2 are elements of the same interval in $\mathcal{P}$. Thus, $(\forall t''_1 \in [u_1, t'_1])\ \sigma, t''_1 \models_\tau \varphi$ iff $(\forall t''_2 \in [u_2, t'_2])\ \sigma, t''_2 \models_\tau \varphi$. Therefore, $\sigma, u_1 \models_\tau \varphi\, \mathbf{U}_I\, \varphi'$ iff $\sigma, u_2 \models_\tau \varphi\, \mathbf{U}_I\, \varphi'$. □

Lemma 5. *[8, Lemma 4.18] For a finite set $T \subseteq [0, \tau)$ and an interval I, let $T_I = \bigcup_{u \in T \cup \{\tau\}} \{u - v \mid v \in e(I)\} \cap [0, \tau)$. For partitions $\mathcal{P}_\tau(T) = \{J_i\}_{i \in [N]}$ and $\mathcal{P}_\tau(T_I) = \{K_j\}_{j \in [M]}$, $K_j \subseteq J_i - I$ or $K_j \subseteq (J_i - I)^{\complement}$ holds for any K_j and J_i.*

Proof. Let $T_I \cup \{0, \tau\} = \{u_0, \ldots, u_{m+1}\}$, where $0 = u_0 < \cdots < u_m < u_{m+1} = \tau$. Suppose that the lemma does not hold for K_j in $\mathcal{P}_\tau(T_I)$ and J_i in $\mathcal{P}_\tau(T)$; that is, K_j intersects both $J_i - I$ and $(J_i - I)^{\complement}$. Then, K_j must include an endpoint of $J_i - I$. Observe that $T_I \cup \{0, \tau\}$ contains all such endpoints. By definition, K_j is either $\{u_i\}$ or (u_i, u_{i+1}), and only $\{u_i\}$ can include an endpoint in $T_I \cup \{0, \tau\}$. However, $\{u_i\} \subseteq J_i - I$ and $\{u_i\} \subseteq (J_i - I)^{\complement}$ cannot hold at the same time. □

Lemma 6. *Given a signal σ, an STL formula φ, and a finite set $T \subseteq [0, \tau)$, if φ is stable for $\mathcal{P}_\tau(T)$, there exists a minimal subset $min^\sigma_\varphi(T) \subseteq T$ such that for any strict subset $U \subset min^\sigma_\varphi(T)$, φ is not stable for $\mathcal{P}_\tau(U)$.*

Proof. Let $T \cup \{0, \tau\} = \{u_0, \ldots, u_{m+1}\}$, $0 = u_0 < \cdots < u_m < u_{m+1} = \tau$. Let $min^\sigma_\varphi(T) = \{u_i \in T \mid \sigma, u_i \models_\tau \varphi$ is not equal to $\sigma, v_i \models_\tau \varphi$ or $\sigma, v_{i-1} \models_\tau \varphi\}$, where $v_i = (u_i + u_{i+1})/2$, $0 \leq i \leq m$. Observe that φ is stable for $\mathcal{P}_\tau(min^\sigma_\varphi(T))$ and for any strict subset $U \subset min^\sigma_\varphi(T)$, φ cannot be stable for $\mathcal{P}_\tau(U)$. □

Lemma 7. *[8, Proposition 3.10] For intervals K_1 and K_2, where $K_1 \cap K_2 = \emptyset$ and $\sup(K_1) = \inf(K_2)$, $\varphi\,\mathbf{U}_I^{K_1 \cup K_2}\,\varphi' \equiv \varphi\,\mathbf{U}_I^{K_1}\,\varphi' \vee (\Box_{\geq 0}^{K_1}\,\varphi \wedge \varphi\,\mathbf{U}_I^{K_2}\,\varphi')$.*

Proof. Let σ be a signal, $t \geq 0$, and $K = K_1 \cup K_2$. Then, $\sigma, t \models_\tau \varphi\,\mathbf{U}_I^K\,\varphi'$ iff $(\exists t' \geq t)\ t' \in [0,\tau) \cap K, t' \in t + I, \sigma, t' \models_\tau \varphi', (\forall t'' \in [t,t'] \cap K)\ \sigma, t'' \models_\tau \varphi$. Because $t' \in [0,\tau) \cap K$ iff $t' \in [0,\tau) \cap (K_1 \cup K_2)$ iff either $t' \in [0,\tau) \cap K_1$ or $t' \in [0,\tau) \cap K_2$, the statement can be equivalently rewritten as:

$$(\exists t' \geq t)\ t' \in [0,\tau) \cap K_1, t' \in t + I, \sigma, t' \models_\tau \varphi', (\forall t'' \in [t,t'] \cap K)\ \sigma, t'' \models_\tau \varphi$$
$$\text{or } (\exists t' \geq t)\ t' \in [0,\tau) \cap K_2, t' \in t + I, \sigma, t' \models_\tau \varphi', (\forall t'' \in [t,t'] \cap K)\ \sigma, t'' \models_\tau \varphi$$

When $t' \in K_1$, because $t' \leq \inf(K_2)$ and $K_1 \cap K_2 = \emptyset$, $[t,t'] \cap K = [t,t'] \cap K_1$. When $t' \in K_2$, because $\sup(K_1) \leq t' < \tau$, $[t,t'] \cap K = ([t,t'] \cap K_2) \cup ([t,\tau) \cap K_1)$. Therefore, the above statement can be equivalently rewritten as:

$$(\exists t' \geq t)\ t' \in [0,\tau) \cap K_1, t' \in t + I, \sigma, t' \models_\tau \varphi', (\forall t'' \in [t,t'] \cap K_1)\ \sigma, t'' \models_\tau \varphi$$
$$\text{or } (\exists t' \geq t)\ t' \in [0,\tau) \cap K_2, t' \in t + I, \sigma, t' \models_\tau \varphi', (\forall t'' \in [t,t'] \cap K_2)\ \sigma, t'' \models_\tau \varphi,$$
$$(\forall t'' \in [t,\tau) \cap K_1)\ \sigma, t'' \models_\tau \varphi$$

Observe that $(\forall t'' \in [t,\tau) \cap K_1)\ \sigma, t'' \models_\tau \varphi$ iff $\sigma, t \models_\tau \Box_{\geq 0}^{K_1}\,\varphi$. Therefore, the above is equivalent to: $\sigma, t \models_\tau \varphi\,\mathbf{U}_I^{K_1}\,\varphi'$ or $(\sigma, t \models_\tau \varphi\,\mathbf{U}_I^{K_2}\,\varphi'$ and $\sigma, t \models_\tau \Box_{\geq 0}^{K_1}\,\varphi)$. Consequently, $\sigma, t \models_\tau \varphi\,\mathbf{U}_I^K\,\varphi'$ iff $\sigma, t \models_\tau \varphi\,\mathbf{U}_I^{K_1}\,\varphi' \vee (\Box_{\geq 0}^{K_1}\,\varphi \wedge \varphi\,\mathbf{U}_I^{K_2}\,\varphi')$. □

Lemma 8. *For a partition $\mathcal{P} = \{J_i\}_{i \in [N]}$ and its time sample $\{v_i\}_{i \in [N]}$, where φ and φ' are stable for $\mathcal{P}$, let $L_k = \bigcup_{j=k}^N J_k$ for $k \in [N]$. (1) $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{L_k}\,\varphi'$ iff $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{J_k}\,\varphi' \vee (\Box_{\geq 0}^{J_k}\,\varphi \wedge \varphi\,\mathbf{U}_I^{L_{k+1}}\,\varphi')$. (2) $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{J_k}\,\varphi'$ iff $v_i \in J_k - I$, $\sigma, v_k \models_\tau \varphi$, and $\sigma, v_k \models_\tau \varphi'$. (3) $\sigma, v_i \models_\tau \Box_{\geq 0}^{J_k}\,\varphi$ iff $\sigma, v_k \models_\tau \varphi$.*

Proof. (1) Because $L_k = J_k \cup L_{k+1}$ for $k \in [N]$, by Lemma 7, $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{L_k}\,\varphi'$ iff $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{J_k}\,\varphi' \vee (\Box_{\geq 0}^{J_k}\,\varphi \wedge \varphi\,\mathbf{U}_I^{L_{k+1}}\,\varphi')$.
(2) $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{J_k}\,\varphi'$ iff $(\exists t' \geq v_i)\ t' \in [0,\tau) \cap J_k, t' \in v_i + I, \sigma, t' \models_\tau \varphi'$, $(\forall t'' \in [v_i,t'] \cap K)\ \sigma, t'' \models_\tau \varphi$ iff, because $J_k \subseteq [0,\tau)$, and because φ and φ' are stable for J_k, by Lemma 12, $v_i \in J_k - I$, $\sigma, v_k \models_\tau \varphi$, and $\sigma, v_k \models_\tau \varphi'$.
(3) $\sigma, v_i \models_\tau \Box_{\geq 0}^{J_k}\,\varphi$ iff $\neg((\exists t' \geq v_i)\ t' \in [0,\tau) \cap J_k, t' \in v_i + [0,\infty), \sigma, t' \models_\tau \neg\varphi)$ iff, since φ is stable for J_k and $v_i \in [0,\tau) \cap J_k - [0,\infty)$, by Lemma 12, $\sigma, v_k \models_\tau \varphi$. □

Lemma 14. *Suppose that φ and φ' are stable for $\mathcal{P} = \{J_i\}_{i \in [N]}$, $\chi_\varphi^k = \top$ iff $\sigma, v_k \models_\tau \varphi$, $\chi_{\varphi'}^k = \top$ iff $\sigma, v_k \models_\tau \varphi'$, and $L_k = \bigcup_{j=k}^N J_k$, for each $k \in [N]$. For $\mathcal{P}$'s time sample $\{v_i\}_{i \in [N]}$, for $i \leq k \leq N$, $tr_i(\varphi\,\mathbf{U}_I\,\varphi', k) = \top$ iff $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{L_k}\,\varphi'$.*

Proof. The proof is by induction on $N - k$. When $k = N$, $tr_i(\varphi\,\mathbf{U}_I\,\varphi', N) = \top$ iff $(v_i \in J_N - I \wedge \chi_\varphi^N \wedge \chi_{\varphi'}^N) \vee (\chi_\varphi^N \wedge \bot)$ iff $v_i \in J_N - I \wedge (\sigma, v_N \models_\tau \varphi) \wedge (\sigma, v_N \models_\tau \varphi')$ iff, by Lemma 8, $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{J_N}\,\varphi'$ iff $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{L_N}\,\varphi'$. When $k = n + 1$, $tr_i(\varphi\,\mathbf{U}_I\,\varphi', n) = \top$ iff $(v_i \in J_n - I \wedge \chi_\varphi^n \wedge \chi_{\varphi'}^n) \vee (\chi_\varphi^n \wedge tr_i(\varphi\,\mathbf{U}_I\,\varphi', n+1))$ iff, by Lemma 8 and induction hypothesis, $(v_i \in J_n - I \wedge \sigma, v_n \models_\tau \varphi \wedge \sigma, v_n \models_\tau \varphi')$ $\vee\ (\sigma, v_n \models_\tau \varphi \wedge (\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{L_{n+1}}\,\varphi'))$ iff, by Lemma 8, $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{J_n}\,\varphi' \vee$ $(\sigma, v_i \models_\tau \Box_{\geq 0}^{J_n}\,\varphi \wedge (\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{L_{n+1}}\,\varphi'))$ iff, by Lemma 8, $\sigma, v_i \models_\tau \varphi\,\mathbf{U}_I^{L_n}\,\varphi'$. □

Lemma 9. *Suppose that each $\psi \in \mathrm{sub}(\varphi)$ of an STL formula φ is stable for a partition $\mathcal{P} = \{J_i\}_{i\in[N]}$. The following conditions are equivalent: (i) $\chi^i_\psi = \top$ iff $(\forall t \in J_i)\ \sigma, t \models_\tau \psi$. (ii) every condition in $\Gamma_\varphi(\mathcal{P}, \mathcal{X}) \cup \Gamma_\Pi(\sigma, \mathcal{S}_\mathcal{P}, \mathcal{X})$ holds.*

Proof. Let $\{v_i\}_{i\in[N]}$ be a time sample of $\mathcal{P}$, and $L_k = \bigcup_{j=k}^N J_k$ for each $k \in [N]$. First, $\sigma, v_i \models_\tau \psi$ iff $(\forall t \in J_i)\ \sigma, t \models_\tau \psi$, because each subformula $\psi \in \mathrm{sub}(\varphi)$ is stable for $\mathcal{P}$. The proof is by structural induction on φ.
– $(\varphi = p)$: $(\Rightarrow)$ $\chi^i_p = \top$ iff $\sigma, v_i \models_\tau p$ iff $v_i < \tau \wedge p(\sigma(v_i)) = \top$ iff, because $v_i < \tau$, $p(\sigma(v_i)) = \top$. Therefore, the condition $\chi^i_p = p(\sigma(v_i))$ holds. $(\Leftarrow)$ Suppose $\chi^i_p = p(\sigma(v_i))$ holds. Then, $\chi^i_p = \top$ iff $p(\sigma(v_i)) = \top$ iff $(\forall t \in J_i)\ \sigma, t \models_\tau p$.
– $(\varphi = \neg\phi)$: $(\Rightarrow)$ $\chi^i_{\neg\phi} = \top$ iff $\sigma, v_i \models_\tau \neg\phi$ iff $\sigma, v_i \not\models_\tau \phi$ iff, by induction hypothesis, $\chi^i_\phi = \bot$. Thus, $\chi^i_{\neg\phi} = \neg\chi^i_\phi$. $(\Leftarrow)$ Suppose $\chi^i_{\neg\phi} = \neg\chi^i_\phi$. Then, $\chi^i_{\neg\phi} = \top$ iff $\chi^i_\phi = \bot$ iff, by induction hypothesis, $\sigma, v_i \not\models_\tau \phi$ iff $(\forall t \in J_i)\ \sigma, t \models_\tau \neg\phi$.
– $(\varphi = \phi \wedge \phi')$: $(\Rightarrow)$ $\chi^i_{\phi\wedge\phi'} = \top$ iff $\sigma, v_i \models_\tau \phi \wedge \phi'$ iff $\sigma, v_i \models_\tau \phi$ and $\sigma, v_i \models_\tau \phi'$ iff, by induction hypothesis, $\chi^i_\phi = \top$ and $\chi^i_{\phi'} = \top$. Therefore, $\chi^i_{\phi\wedge\phi'} = \chi^i_\phi \wedge \chi^i_{\phi'}$. $(\Leftarrow)$ Suppose $\chi^i_{\phi\wedge\phi'} = \chi^i_\phi \wedge \chi^i_{\phi'}$. Then, $\chi^i_{\phi\wedge\phi'} = \top$ iff $\chi^i_\phi \wedge \chi^i_{\phi'} = \top$ iff, by induction hypothesis, $\sigma, v_i \models_\tau \phi$ and $\sigma, v_i \models_\tau \phi'$ iff $\sigma, v_i \models_\tau \phi \wedge \phi'$ iff $(\forall t \in J_i)\ \sigma, t \models_\tau \phi \wedge \phi'$.
– $(\varphi = \phi\,\mathbf{U}_I\,\phi')$: $(\Rightarrow)$ $\chi^i_{\phi\,\mathbf{U}_I\,\phi'} = \top$ iff $\sigma, v_i \models_\tau \phi\,\mathbf{U}_I\,\phi'$ iff, because $v_i \in L_i$, $\sigma, v_i \models_\tau \phi\,\mathbf{U}^{L_i}_I\,\phi'$ iff, by induction hypothesis and Lemma 14, $tr_i(\phi\,\mathbf{U}_I\,\phi', i) = \top$. Thus, $\chi^i_{\phi\,\mathbf{U}_I\,\phi'} = tr_i(\phi\,\mathbf{U}_I\,\phi', i)$ holds. $(\Leftarrow)$ Suppose $\chi^i_{\phi\,\mathbf{U}_I\,\phi'} = tr_i(\phi\,\mathbf{U}_I\,\phi', i)$ holds. Then, $\chi^i_{\phi\,\mathbf{U}_I\,\phi'} = \top$ iff $tr_i(\phi\,\mathbf{U}_I\,\phi', i) = \top$ iff, by induction hypothesis and Lemma 14, $\sigma, v_i \models_\tau \phi\,\mathbf{U}^{L_i}_I\,\phi'$ iff $\sigma, v_i \models_\tau \phi\,\mathbf{U}_I\,\phi'$ iff $(\forall t \in J_i)\ \sigma, t \models_\tau \phi\,\mathbf{U}_I\,\phi'$. □

Lemma 10. *For a formula φ and a finite set $V \subseteq [0, \tau)$, if $V = \mathcal{T}^\sigma_\varphi(T)$ for a set of variable points T, then every condition in $\Omega_\varphi(V, \mathcal{X}, \sigma)$ holds.*

Proof. By Theorem 1, each subformula of φ is stable for $\mathcal{T}^\sigma_\varphi(T)$. By Lemma 9, $\Gamma_\varphi(\mathcal{P}_\tau(V), \sigma[\mathcal{S}_{\mathcal{P}_\tau(V)}]))$ holds for $V = \mathcal{T}^\sigma_\varphi(T)$, and $\chi^i_\psi = \top$ iff $(\forall t \in J_i)\ \sigma, t \models_\tau \psi$ for each $\psi \in \mathrm{sub}(\varphi)$. Suppose $k = n+1 \vee \bigvee_{j=1,2}(\chi^{2k}_{\phi_j} \neq \chi^{2k+2}_{\phi_j})$. Then, $t_k = \tau$ or the truth of ϕ_j changes at time t_k; i.e., for $U = min^\sigma_\phi(\mathcal{I}(\phi)) \cup min^\sigma_{\phi'}(\mathcal{I}(\phi')) \cup \{\tau\}$ in Def. 9, $t_k \in U$. By construction, $\bigcup_{u\in U}\{u - v \mid v \in e(I)\} \cap [0, \tau) \subseteq V$. Thus, the condition $v = 0 \vee t_k - v < 0 \vee \bigvee_{j=1}^{k-1}(t_k - v = t_j)$ holds. □

Lemma 11. *If every condition in $\Omega_\varphi(V, \mathcal{X}, \sigma)$ holds, then each $\psi \in \mathrm{sub}(\varphi)$ is stable for $\mathcal{P}_\tau(V)$, provided that each $p \in \mathrm{Props}(\varphi)$ is stable for $\mathcal{P}_\tau(V)$.*

Proof. By structural induction on φ. The case of $\varphi = p$ is by assumption. The cases of $\varphi = \neg\phi$ and $\varphi = \phi \wedge \phi'$ are immediate by induction hypothesis. Consider $\varphi = \phi\,\mathbf{U}_I\,\phi'$. By assumption, the condition $(k = n+1 \vee \bigvee_{j=1,2}(\chi^{2k}_{\phi_j} \neq \chi^{2k+2}_{\phi_j})) \to (v = 0 \vee t_k - v < 0 \vee \bigvee_{j=1}^{k-1}(t_k - v = t_j))$ in $\Delta_\varphi(V)$ holds for each $t_k \in V \cup \{0, \tau\}$ and $v \in e(I)$. By induction hypothesis, any subformula of ϕ and ϕ' is stable for $\mathcal{P}_\tau(V)$. By Lemma 9, $\chi^k_{\phi_j} = \top$ iff $(\forall t \in J_k)\ \sigma, t \models_\tau \phi_j$ for $j = 1, 2$. Thus, the premise of the above condition holds if either $t_k = \tau$ or the truth value of ϕ_j changes at time t_k, and in that case, $t_k - v \in V$, provided $t_k - v \geq 0$. Therefore, by Lemma 4, Lemma 5, and Corollary 1, $\phi\,\mathbf{U}_I\,\phi'$ is stable for $\mathcal{P}_\tau(V)$. □