Bounded Model Checking of Signal Temporal Logic Properties using Syntactic Separation

KYUNGMIN BAE, Pohang University of Science and Technology, South Korea
JIA LEE, Pohang University of Science and Technology, South Korea

Signal temporal logic (STL) is a temporal logic formalism for specifying properties of continuous signals. STL is widely used for analyzing programs in cyber-physical systems (CPS) that interact with physical entities. However, existing methods for analyzing STL properties are incomplete even for bounded signals, and thus cannot guarantee the correctness of CPS programs. This paper presents a new symbolic model checking algorithm for CPS programs that is *refutationally complete* for general STL properties of bounded signals. To address the difficulties of dealing with an infinite state space over a continuous time domain, we first propose a *syntactic separation* of STL, which decomposes an STL formula into an equivalent formula so that each subformula depends only on one of the disjoint segments of a signal. Using the syntactic separation, an STL model checking problem can be reduced to the satisfiability of a first-order logic formula, which is decidable for CPS programs with polynomial dynamics using satisfiability modulo theories (SMT). Unlike the previous methods, our method can verify the correctness of CPS programs for STL properties up to given bounds.

CCS Concepts: • **Theory of computation → Timed and hybrid models**; **Modal and temporal logics**; **Verification by model checking**;

Additional Key Words and Phrases: Signal temporal logic, Bounded model checking, Syntactic separation, Satisfiability modulo theories (SMT)

ACM Reference Format:
Kyungmin Bae and Jia Lee. 2019. Bounded Model Checking of Signal Temporal Logic Properties using Syntactic Separation. *Proc. ACM Program. Lang.* 3, POPL, Article 51 (January 2019), 30 pages. https://doi.org/10.1145/3290364

1 INTRODUCTION

Signal temporal logic (STL) is a temporal logic formalism for specifying linear-time properties of continuous real-valued signals [Maler and Nickovic 2004], defined by extending metric temporal logic (MTL) for real-time properties [Koymans 1990]. Just as linear temporal logic (LTL) is widely used for formally analyzing conventional (discrete) programs, STL is widely used for analyzing programs in cyber-physical systems (CPS) that interact with physical entities exhibiting continuous dynamics. This kind of CPS programs includes automotive, avionics, robotics, and medical software [Dokhanchi et al. 2015; Goldman et al. 2016; Jin et al. 2014; Raman et al. 2015; Roohi et al. 2018]. Since CPS programs are often safety-critical, formal analysis of STL is receiving growing attention, and various techniques have been developed for STL [Annpureddy et al. 2011; Deshmukh et al. 2017; Donzé et al. 2013; Jakšić et al. 2016; Ničković et al. 2018; Roehm et al. 2016].

Authors' addresses: Kyungmin Bae, Department of Computer Science and Engineering, Pohang University of Science and Technology, 77 Cheongam-Ro, Nam-Gu, Pohang, Gyeongbuk, 37673, South Korea, kmbae@postech.ac.kr; Jia Lee, Department of Computer Science and Engineering, Pohang University of Science and Technology, 77 Cheongam-Ro, Nam-Gu, Pohang, Gyeongbuk, 37673, South Korea, cee5539@postech.ac.kr.

2475-1421/2019/1-ART51
https://doi.org/10.1145/3290364

However, model checking of STL properties is still very limited. Due to continuous signals, STL model checking inherently takes into account an infinite number of states. Furthermore, a state of a CPS program changes continuously over an uncountable time domain. For this reason, existing methods for analyzing STL properties are *incomplete*, even for bounded signals. Specifically, monitoring and falsification techniques analyze only a finite number of signals [Annpureddy et al. 2011; Deshmukh et al. 2017; Jakšić et al. 2016; Ničković et al. 2018], and a symbolic model checking technique recently proposed in [Roehm et al. 2016] considers only a finite number of sampled time points. This is in contrast to the case of discrete programs where model checking techniques can typically be complete, at least up to a bound (e.g., [Biere et al. 2003; Cordeiro et al. 2012]).

In this paper we present a symbolic model checking algorithm for STL properties which is complete for bounded signals. Our algorithm is based on a new foundational technique for STL, called *syntactic separation*, proposed in this paper. The syntactic separation decomposes an STL formula into an equivalent formula in which the satisfaction of each subformula only depends on a specified time interval. Using this result, an STL model checking problem can be reduced to the satisfiability of a first-order logic formula, which is decidable if the reachability of the underlying CPS program is decidable. The satisfiability of the resulting formula can be determined using satisfiability modulo theories (SMT) techniques [Biere et al. 2009]. Unlike the previous methods, our method can verify the correctness of CPS programs for STL properties up to given bounds.

Syntactic Separation of STL. Generally, a temporal logic formula φ is called *syntactically separated* if φ is a Boolean combination of formulas, each of which depends only on a disjoint part of the underlying time domain (such as the past, present, or future).[1] Syntactically separating STL formulas is nontrivial, because temporal operators in STL are further constrained by time intervals. Consider an STL formula $\Box_{[0,3]}(x \geq 0 \rightarrow \Diamond_{[0.5,2)}\, x < 0)$, which involves two temporal operators $\Box_{[0,3]}$ and $\Diamond_{[0.5,2)}$. Intuitively, this formula means that "*during the first* 3 *time units, whenever the value of signal* x *is greater than or equal to* 0*, the value of* x *will be less than* 0 *after some time in the interval* $[0.5, 2)$." For this kind of quantitative temporal logics, including STL and MTL, there has been no generic method proposed for separating a formula into disjoint parts [Hunter et al. 2013].

To address this problem, we generalize the syntax of STL by adding extra time constraints. In addition to existing intervals to denote "local" time constraints, the temporal operators of STL are annotated with extra intervals to represent "global" time constraints. For example, by annotating $\Box_{[0,3]}$ and $\Diamond_{[0.5,2)}$ with the interval $[0, 1)$, we obtain the formula $\Box^{[0,1)}_{[0,3]}(x \geq 0 \rightarrow \Diamond^{[0,1)}_{[0.5,2)}\, x < 0)$, which means that "*during the first* 3 *time units, whenever* $x \geq 0$ *at a global time in* $[0, 1)$*, after some time in the interval* $[0.5, 2)$*,* $x < 0$ *will hold at some global time in* $[0, 1)$." This makes it possible to write a formula that only depends on a specified time interval. Adding such global intervals to STL yields a more expressive temporal logic, namely, *STL with global time* (STL-GT).

In this paper we propose a syntactic separation procedure for STL-GT formulas at a chosen time of separation. More precisely, an STL-GT formula φ is syntactically rewritten into an equivalent formula in which every temporal operator is globally restricted, given a time τ, by one of the disjoint time intervals $[0, \tau)$, $\{\tau\}$, or (τ, ∞). As a consequence, each subformula of the resulting formula can depend only on one of the disjoint segments of a signal before τ, at τ, or after τ. We show a number of equivalence laws that can globally separate any STL-GT formulas, including the *Until* operator. We then identify a precise syntactic subclass of STL-GT that includes STL and is closed under the separation operation. This separation procedure can be repeatedly applied to obtain a formula separated at different time points $\tau_1 < \cdots < \tau_n$.

[1] For example, in a discrete time domain, an LTL formula $\Box(p \rightarrow \Diamond q)$ can be rewritten into the syntactically separated formula $(p \rightarrow (q \vee \bigcirc\Diamond q)) \wedge \bigcirc\Box(p \rightarrow \Diamond q)$, using the equivalences $\Box\varphi \equiv \varphi \wedge \bigcirc\Box\varphi$ and $\Diamond\varphi \equiv \varphi \vee \bigcirc\Diamond\varphi$. Observe that p and q depend only on the present, and $\bigcirc\Diamond q$ and $\bigcirc\Box(p \rightarrow \Diamond q)$ depend only on the future.

Translation of STL to First-Order Logic. Using syntactic separation, we present a simple procedure to translate STL formulas into first-order logic formulas for symbolic model checking. Because the semantics of STL is definable in first-order logic, one can immediately translate STL into first-order logic. However, the translation by the semantic definition gives no decision procedure, because the satisfiability of STL is undecidable [Alur et al. 1996]; indeed, formulas with deeply nested quantifiers can be generated by this approach. Instead, we translate STL formulas into a decidable fragment of first-order logic for a signal with *a finite number of variable points*. A signal has a finite number of variable points if the meaning of each proposition changes a finite number of times on the signal, while the value of the signal may continuously change over time.

A key idea underlying our translation procedure is the notion of *full stability*. For a signal with a finite number of variable points, a fully stable formula behaves like a propositional formula in the sense that the truth value of every subformula is fixed to either *true* or *false*. We show that any STL formula can be equivalently rewritten into a fully stable STL-GT formula using the syntactic separation, given variable time points $\tau_1 < \cdots < \tau_n$. It is then straightforward to translate a fully stable STL-GT formula into first-order logic. For fully stable STL-GT formulas, the time constraint of each temporal operator can be expressed as a quantifier-free first-order logic formula in interval arithmetic. The requirement that a signal has given variable points $\tau_1 < \cdots < \tau_n$ can be encoded as a first-order logic formula with universal quantification over time variables. The resulting first-order logic formula can be decidable under reasonable assumptions.

Symbolic Model Checking Algorithm. Based on our translation method, we propose a symbolic model checking algorithm for STL properties of CPS programs. The proposed algorithm constructs a first-order logic formula that is satisfiable if and only if there exists a counterexample from an initial set with a particular number k of variable points. By iteratively incrementing the number k, this algorithm is refutationally complete for bounded signals with *finite variability*, where the number of variable points is finite over a finite period. This condition is typically assumed when analyzing realistic real-time and CPS programs [Ho et al. 2014; Maler and Nickovic 2004; Ouaknine and Worrell 2008]. Our algorithm is related to the notion of bounded model checking [Biere et al. 2003], where the bound is the maximal number of variable points in a continuous signal.

In our algorithm, the behavior of a CPS program is encoded as a first-order logic formula, following SMT-based approaches for reachability analysis of hybrid automata. The semantics of CPS programs can be formalized as hybrid automata [Alur 2015]. Finding a signal with a given number of variable points can be considered as a special case of the reachability problem. The reachability problem can be reduced to the satisfiability of first-order logic formulas [Cimatti et al. 2012b], which is decidable if the continuous dynamics involves only linear functions or polynomials. Together with our translation method, the satisfiability of the generated formula for symbolic model checking can be decided using Z3 [De Moura and Bjørner 2008]. If the continuous dynamics involves transcendental functions, the satisfiability is undecidable. Nevertheless, we can still use a specialized SMT solver based on approximation methods [Gao et al. 2012, 2013a].

Related Work. Separation is one of the foundational techniques in temporal logic. Theoretically, separation has several important consequences [Hodkinson and Reynolds 2005]. Gabbay showed that every LTL formula can be separated into the past, present, or future, and using this result he proved the expressive completeness of LTL [Gabbay 1981], in a much simpler way than Kamp's original proof [Kamp 1968]. A separation plays an important role in formal analysis techniques. Many tableau construction methods use a separation of a formula so that the constraints for the current state can be separated from the constraints for the future states [Clarke et al. 1999; Gerth et al. 1995]. SAT-based model checking [Biere et al. 2003] and rewriting-based monitoring [Havelund and Roşu 2004] use a syntactic separation to translate LTL formulas.

There are two syntactic separation methods for MTL that we are aware of. Hunter *et al.* proposed a separation of MTL to prove the expressive completeness of MTL [Hunter et al. 2013]. Their method separates a formula into a Boolean combination of subformulas that are either bounded or unbounded; unlike our method, a specific time of separation cannot be chosen. Geilen presented an on-the-fly tableau construction for a fragment of MTL, called $\mathrm{MITL}_{\leq}$, based on a separation of $\mathrm{MITL}_{\leq}$ in a logic extended with timers [Geilen 2003]. Geilen's method considers only a fragment of MTL, and how to separate a formula strictly in the extended logic, beyond $\mathrm{MITL}_{\leq}$, is not studied in [Geilen 2003], whereas any STL-GT formula is separable by our method.

STL was first proposed by Maler and Nickovic for runtime monitoring of continuous signals [Maler and Nickovic 2004]. A variety of techniques and tools have been developed for monitoring of continuous signals, including [Deshmukh et al. 2017; Donzé 2010; Donzé et al. 2013; Ho et al. 2014; Jakšić et al. 2016; Ničković et al. 2018; Nickovic and Maler 2007]. These techniques are combined with temporal logic falsification of hybrid systems for finding counterexamples using Monte-Carlo methods [Annpureddy et al. 2011]. As mentioned, monitoring and falsification techniques can analyze only a finite number of bounded signals; they are quite useful for finding counterexamples in practice, but cannot be used to verify STL properties of CPS programs.

Reachability analysis of hybrid automata has been studied for a long time. There are three different approaches in general. Reachable-set computation methods calculate (approximate) sets of reachable states by symbolic constraint solving, e.g., [Althoff 2013; Bak and Duggirala 2017; Chen et al. 2013; Dang and Testylier 2012; Frehse et al. 2011]. Simulation-based methods attempt to obtain approximate sets of reachable states by performing a finite number of simulations and by bloating the simulated trajectories, e.g., [Abbas et al. 2013; Dang and Nahhal 2009; Duggirala et al. 2013; Fan et al. 2016; Girard and Pappas 2006]. SMT-based approaches reduce the reachability problem to the (approximate) satisfiability of first order logic over the real numbers, e.g., [Cimatti et al. 2015; Eggers et al. 2015; Gao et al. 2013b; Ishii et al. 2011; Tiwari 2015]. These methods can verify invariant properties, but cannot verify general STL properties of CPS programs.

Roehm *et al.* recently proposed a symbolic model checking method for STL properties using reachable set computation [Roehm et al. 2016]. Given a finite number of sampled time points, their method reduces the model checking problem of a "sampled time" STL formula into reachable-set computation of bounded signals. But their method is inherently incomplete, because only a finite number of sampled time points is considered. For MTL, Bersaní *et al.* proposed an SMT-based satisfaction checking algorithm by translating MTL into a different temporal logic [Bersani et al. 2015, 2016]. Because their method lacks separation of MTL, the translation is very complex. Also, it is not clear how to use their method for model checking of real-time (CPS) programs.

Summary. Our main contributions are as follows. (1) We present a foundational technique for syntactically separating STL formulas. We define an extension of STL, called STL-GT, and a number of equivalences laws to separate any STL-GT formula. (2) We present a simple procedure to translate STL formulas into a decidable fragment of first-order logic by syntactic separation. This procedure is based on novel ideas of finite variability and full stability. (3) We present a new model checking algorithm for STL that is refutationally complete for bounded signals. This allows verifying STL properties of CPS programs up to given bounds, which was previously not possible.

The rest of this paper is organized as follows. Section 2 provides some background on hybrid automata and STL. Section 3 presents STL-GT, and a syntactic separation procedure for STL-GT. Section 4 explains a translation method from STL formulas into first-order logic formulas using the separation. Section 5 presents a symbolic model checking algorithm for STL properties of CPS programs. Section 6 shows experimental results using a prototype implementation of our algorithm. Finally, Section 7 presents conclusions and discusses future work.

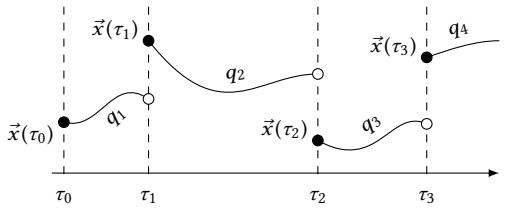

Fig. 1. A trajectory $\vec{x}$ of a hybrid automaton

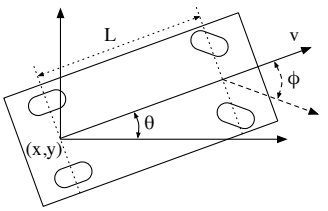

Fig. 2. The motion of an autonomous car

2 PRELIMINARIES ON HYBRID AUTOMATA AND STL

2.1 Hybrid Automata

Hybrid automata [Henzinger 2000] are widely used for formalizing the semantics of CPS programs. In a hybrid automaton H, a set of *modes* Q specifies discrete states, and a set of real-valued *variables* $X = \{x_1, \ldots, x_l\}$ specifies continuous states. That is, a state of H is a pair $\langle q, \vec{v}\rangle$ of a discrete mode $q \in Q$ and a vector $\vec{v} = (v_1, \ldots, v_l) \in \mathbb{R}^l$. There are two kinds of transitions in hybrid automata. A jump condition $jump(q, \vec{v}, q', \vec{v}')$ defines a discrete transition between two states $\langle q, \vec{v}\rangle \longrightarrow \langle q', \vec{v}'\rangle$. A flow condition $\vec{v}_t = flow(q, \vec{v}_0, t)$ defines *trajectories* of X for duration t in mode q, describing continuous changes of X's values from $\vec{v}_0$ to $\vec{v}_t$. An invariant condition $inv(q, \vec{v})$ defines all possible values of X in mode q, and an initial condition $init(q, \vec{v})$ defines a set of initial states in mode q.

Definition 2.1. A *hybrid automaton* is defined as a tuple $H = (Q, X, init, inv, flow, jump)$.

A flow condition of a hybrid automaton is normally written as a system of ordinary differential equations (ODEs). In practice, flow conditions are often restricted to a certain class of real functions. Specifically, a hybrid automaton H is called a linear hybrid automaton if its flow conditions are linear functions, and a polynomial hybrid automaton if its flow conditions are polynomials.

An l-dimensional signal $\vec{x} = (x_1, \ldots, x_l)$ is a function $D \to \mathbb{R}^l$, where its domain $D = \mathrm{dom}(\vec{x})$ is a left-closed interval of nonnegative real numbers with left endpoint $0 \in D$. A signal $\vec{x} : D \to \mathbb{R}^l$ is *bounded* if its time domain is bounded (i.e., $\sup(D) < \infty$). A signal $\vec{x}$ is called a trajectory of a hybrid automaton H, if $\vec{x}$ describes a valid behavior of H over continuous time, as depicted in Fig. 1. To be precise, consider a sequence of time points $0 = \tau_0 \le \tau_1 \le \tau_2 \le \cdots$. The initial condition holds at time τ_0. For each i-th step, the values of X change from $\vec{x}(\tau_{i-1})$ for duration $\tau_i - \tau_{i-1}$ according to the flow condition, while satisfying the invariant condition. A discrete jump happens at time τ_i from the final values of the current step to the starting values $\vec{x}(\tau_i)$ of the next step.

Definition 2.2. For a hybrid automaton H, a signal $\vec{x}$ is a trajectory of H, written $\vec{x} \in H$, if there exist sequences of modes $q_1, q_2, q_3, \ldots$ and of times $0 = \tau_0 \le \tau_1 \le \tau_2 \le \cdots$ such that for $i \ge 1$:

(1) $init(q_1, \vec{x}(\tau_0))$;
(2) $inv(q_i, \vec{x}(t))$ for $t \in [\tau_{i-1}, \tau_i)$;
(3) $\vec{x}(t) = flow(q_i, \vec{x}(\tau_{i-1}), t - \tau_{i-1})$ for $t \in [\tau_{i-1}, \tau_i)$; and
(4) $jump(q_i, \vec{v}_i, q_{i+1}, \vec{x}(\tau_i))$, where $\vec{v}_i = flow(q_i, \vec{x}(\tau_{i-1}), \tau_i - \tau_{i-1})$.

Example 2.3. Consider an autonomous car that is controlled by a CPS program. As illustrated in Fig. 2, the car can be thought of as a rigid body that moves in the plane, where its position is based in the center of the rear axle. Let L be the distance between the front and rear axles. The position (x, y) and the direction θ of the car change according to the speed v and the steering angle ϕ. The motion of the car can be modeled using the ODEs [LaValle 2006]:

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

```
if (goal − θ > c):
    setVelocity (v_low);
    setSteering (α);
else if (goal − θ < −c):
    setVelocity (v_low);
    setSteering (−α);
else:
    setVelocity (v_high);
    setSteering (0);
```

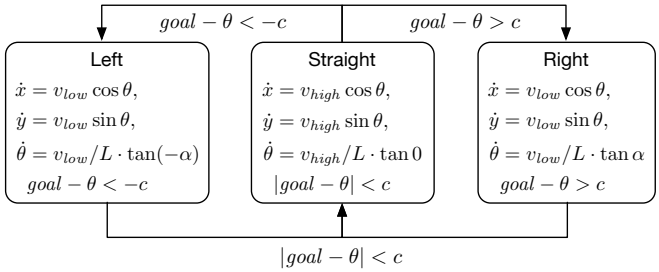

Fig. 3. A CPS program and its hybrid automaton

We consider a simple CPS program to control the direction of the car, adapted from [Alur 2015]. Given a goal direction *goal*, the program determines the velocity θ and the steering angle ϕ based on the difference between *goal* and the current direction θ. If the difference is greater than some threshold $c > 0$, the program changes the velocity and the steering angle so that the vehicle turns left or right at low speed. Otherwise, the program makes the vehicle go straight at high speed.

Fig. 3 shows the simple CPS program, and the hybrid automaton H that specifies its semantics with respect to the physical states. There are three modes `Left`, `Straight`, and `Right`, and three variables (x, y, θ). Each mode has a flow condition specified as ODEs over (x, y, θ), where the velocity and the steering angle are given as constants. The jump condition and the invariant condition are basically given by the program logic. A trajectory of the hybrid automaton H describes the behavior of the physical variables when the program runs on a controller of the car.

2.2 Signal Temporal Logic

Signal temporal logic formulas specify linear-time properties of continuous real-valued signals, such as properties of trajectories for hybrid automata. The syntax of signal temporal logic (STL) is defined as a simple extension of metric temporal logic (MTL).

Definition 2.4 (STL Syntax). The syntax of signal temporal logic (STL) is defined by

$$\varphi := \mathit{true} \mid f(\vec{x}) > 0 \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \,\mathbf{U}_I\, \varphi$$

where $\vec{x} : D \to \mathbb{R}^l$ is a signal, $f : \mathbb{R}^l \to \mathbb{R}$ is a real-valued function, and $I \subseteq D$ is an interval of nonnegative real numbers in the time domain D.

Untimed notations, such as $\mathbf{U}$, are used as shorthand for $\mathbf{U}_{[0,\infty)}$. Intervals in STL formulas are often written using arithmetic expressions; for example, $\mathbf{U}_{=a}$ for $\mathbf{U}_{\{a\}}$ and $\mathbf{U}_{\leq a}$ for $\mathbf{U}_{[0,a]}$. We can define other common Boolean and temporal operators as syntactic abbreviations; e.g.,

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

Example 2.5. For the model of an autonomous car in Example 2.3, we can think about various STL properties over a signal (x, y, θ, v, ϕ). For example:

- $\Box_{[0,60]}(v < 180 \wedge \phi < 60)$: during the first 60 seconds, the speed of the vehicle will always be less than 180 km/h, and the steering angle will always be less than 60°.
- $(\phi > 30 \wedge \phi < 30)\, \mathbf{U}_{\leq 300}\, (x = 100 \wedge y = 100)$: the car will arrive at the position $(100, 100)$ within 300 seconds; until then the steering angle is within the range $(-30°, 30°)$.
- $\Box_{[0,120]}(\phi > 60 \;\rightarrow\; \Diamond_{[2,8)}\, \theta \geq 30)$: during the first 120 seconds, whenever the steering angle is greater than 60°, the direction will exceed 30° sometime within 8 seconds after 2 seconds.
- $\Diamond_{[0,30]}(v > 100 \wedge \Box_{[0,20]}\, v > 100)$: at some time in the first 30 seconds, the speed will go over 100 km/h and stay above 100 km/h for 20 seconds [Dokhanchi et al. 2015].

The semantics of STL is defined as the satisfaction of a formula φ with respect to a signal $\vec{x}$ and a global time $t \in D$. Observe that the interval I in an STL formula $\varphi \mathbf{U}_I \varphi'$ defines the *local temporal context* of its subformula φ', because the satisfaction of $\varphi \mathbf{U}_I^K \varphi'$ at global time t requires the satisfaction of φ' at some "local" time $t' - t$ in the interval I.

Definition 2.6 (STL Semantics). The satisfaction of an STL formula φ at a given time t over a signal $\vec{x}$, denoted by $\vec{x}, t \models \varphi$, is inductively defined as follows:

$$
\begin{aligned}
&\vec{x}, t \models \mathit{true} \\
&\vec{x}, t \models f(\vec{x}) > 0 \quad \text{iff} \quad f(\vec{x}(t)) > 0 \\
&\vec{x}, t \models \neg\varphi \quad \text{iff} \quad \vec{x}, t \not\models \varphi \\
&\vec{x}, t \models \varphi \wedge \varphi' \quad \text{iff} \quad \vec{x}, t \models \varphi \ \text{ and } \ \vec{x}, t \models \varphi \\
&\vec{x}, t \models \varphi \mathbf{U}_I \varphi' \quad \text{iff} \quad (\exists t' \geq t)\ t' - t \in I, \ \ \vec{x}, t' \models \varphi', \ \text{ and } (\forall t'' \in [t, t'])\ \vec{x}, t'' \models \varphi
\end{aligned}
$$

For an STL formula φ, the *future-reach* $fr(\varphi)$ indicates how much of the future is required to determine the satisfaction [Hunter et al. 2013]. Notice that the satisfaction $\vec{x}, t \models \varphi$ is well-defined if the time horizon of $\vec{x}$ includes the future-reach of φ, i.e., $t + fr(\varphi) \in dom(\vec{x})$.

Definition 2.7. Given an STL formula φ, the *future-reach* $fr(\varphi)$ is inductively defined by:

$$
\begin{aligned}
fr(\mathit{true}) = 0, \qquad & fr(\neg\varphi) = fr(\varphi), \quad fr(\varphi \wedge \varphi') = \max(fr(\varphi), fr(\varphi')), \\
fr(f(\vec{x}) > 0) = 0, \qquad & fr(\varphi \mathbf{U}_I \varphi') = \sup(I) + \max(fr(\varphi), fr(\varphi')).
\end{aligned}
$$

The STL model checking problem, denoted by $H, \tau_0 \models \varphi$, is to check whether an STL formula φ is satisfied for every trajectory $\vec{x}$ of a hybrid automaton H at an initial time τ_0 (i.e., $\vec{x}, \tau_0 \models \varphi$). By definition, this is equivalent to finding a counterexample trajectory $\vec{x}$ that satisfies the negated formula $\neg\varphi$. The STL model checking problem is undecidable in general. The reachability problem for hybrid automata is already undecidable [Henzinger et al. 1998], which is a special case of STL model checking. Further, the existing techniques for STL model checking are inherently incomplete, as discussed in Sec. 1, and thus cannot guarantee the correctness for STL properties.

For a signal $\vec{x} : D \to \mathbb{R}^l$, a time point $\tau \in D$ is called a *variable point* if the truth value of a proposition changes at the time τ on the signal $\vec{x}$. For example, the set of variable points with respect to the proposition $y > 2$ for the signal $y = |t - 3|$ is $\{1, 5\}$.

Definition 2.8. Given a signal $\vec{x} : D \to \mathbb{R}^l$, a time point $\tau \in D$ is a *variable point* of $\vec{x}$ with respect to a proposition $p(\vec{x})$ if for some neighborhood $B \ni \tau$, there are different truth values u and v such that $p(\vec{x}(t)) = u$ for every $t \in B \cap [0, \tau)$ and $p(\vec{x}(t)) = v$ for every $t \in B \cap (\tau, \infty)$.

In this paper we consider the STL model checking problem for *bounded signals*. There are two bound parameters: the domain of a signal is bounded by a given time $\tau_{\max} \in \mathbb{R}^+$, and the number of variable points in a signal is bounded by a given number $k \in \mathbb{N}$.

Definition 2.9 (STL Bounded Model Checking). An STL formula φ is satisfied at a time τ_0 on a hybrid automaton H up to bounds $\tau_{\max} > \tau_0 + fr(\varphi)$ and $k \in \mathbb{N}$, denoted by $H, \tau_0 \models_{k, \tau_{\max}} \varphi$, iff $\vec{x}, \tau_0 \models \varphi$ for every trajectory $\vec{x} \in H$ with at most k variable points such that $\sup(dom(\vec{x})) \leq \tau_{\max}$.

3 SYNTACTIC SEPARATION OF STL

We first introduce a syntactic separation procedure of an STL formula so that the satisfaction of each subformula can be exclusively determined by disjoint parts of a signal. We argue that this problem is quite nontrivial for STL, because there are usually overlaps between local time constraints of different subformulas. For this reason, we define a more expressive temporal logic, called STL-GT, and present a separation of STL-GT that meets this requirement.

3.1 Local Separation of STL

Consider the STL formula $\varphi = \Box_{[1,5]}(x > 0 \rightarrow \Diamond_{[1,3]}\, y < 10)$. The satisfaction of the formula φ (at global time 0) depends on the segment of the signal (x, y) over the time interval $[1, 8]$. Our goal is to obtain an equivalent formula in which each subformula depends only on a disjoint segment of a signal, say, before 2, at 2, or after 2. In principle, we can separate the formula φ according to its local temporal contexts, given by the intervals $[1, 5]$ and $[1, 3]$ in φ. The following equivalence rule in [Hunter et al. 2013] can be used to separate this formula:

$$\varphi \mathbf{U}_{[0,\infty)} \varphi' \;\equiv\; \varphi \mathbf{U}_{<\tau} \varphi' \;\vee\; \Big(\Box_{<\tau} \varphi \;\wedge\; \Diamond_{=\tau}((\varphi \;\wedge\; \varphi') \;\vee\; \varphi \mathbf{U}_{>0} \varphi')\Big).$$

However, this kind of "local separation" does not exclusively separate the entire formula, allowing overlaps between different segments. By the above equivalence, the formula φ can be rewritten into the following STL formula that is separated at time 2:

$$\Box_{[1,2)}(x > 0 \rightarrow \Diamond_{[1,3]}\, y < 10) \;\wedge\; \Diamond_{=2}(x > 0 \rightarrow \Diamond_{[1,3]}\, y < 10) \;\wedge\; \Box_{(2,5]}(x > 0 \rightarrow \Diamond_{[1,3]}\, y < 10)$$

Observe that the subformula $\Box_{[1,2)}(x > 0 \rightarrow \Diamond_{[1,3]}\, y < 10)$ depends on the segment over $[1, 5)$, and the subformula $\Box_{(2,5]}(x > 0 \rightarrow \Diamond_{[1,3]}\, y < 10)$ depends on the segment over $(2, 8]$. There is an unintended overlap $(2, 5)$, because the local temporal context $[1, 3]$ is not considered. As a matter of the fact, combining such local temporal contexts in a proper way is very difficult, and thus there has been no satisfactory way for separating STL formulas into disjoint parts.

3.2 STL with Global Time

We consider a simple extension of STL, called *signal temporal logic with global time* (STL-GT), by also adding global time constraints. The only syntactic difference from STL is that temporal operators of STL-GT contain extra *global time intervals* K, besides local time intervals I.

Definition 3.1 (STL-GT Syntax). The syntax of STL with global time (STL-GT) is defined by

$$\varphi := \; \mathit{true} \;\mid\; f(\vec{x}) > 0 \;\mid\; \neg\varphi \;\mid\; \varphi \wedge \varphi \;\mid\; \varphi\, \mathbf{U}_I^K\, \varphi$$

where $\vec{x} : D \to \mathbb{R}^l$ is a signal, $f : \mathbb{R}^l \to \mathbb{R}$ is a real-valued function, and $I, K \subseteq D$ are any intervals of nonnegative real numbers in the time domain D.

Intuitively, $\varphi\, \mathbf{U}_I^K\, \varphi'$ means that φ' will hold, simultaneously, at some "local" time in the interval I and some "global" time in the interval K, and until then φ holds. Similarly, we define other common temporal operators as syntactic abbreviations as follows:

$$\Diamond_I^K \varphi \equiv \mathit{true}\, \mathbf{U}_I^K\, \varphi, \qquad \Box_I^K \varphi \equiv \neg\Diamond_I^K \neg\varphi, \qquad \varphi \mathbf{R}_I^K \varphi' \equiv \neg((\neg\varphi)\mathbf{U}_I^K(\neg\varphi'))$$

The satisfaction of STL-GT formulas takes into account global time intervals as well as local time intervals. The global interval K in $\varphi \mathbf{U}_I^K \varphi'$ defines the *global temporal context* of the subformula φ', because the satisfaction of $\varphi \mathbf{U}_I^K \varphi'$ requires the satisfaction of φ' at some "global" time $t' \in K$. Unlike STL, we can explicitly express the global temporal contexts of subformulas in STL-GT.

Definition 3.2 (STL-GT Semantics). The satisfaction of an STL-GT formula at time t over a signal $\vec{x}$, denoted by $\vec{x}, t \models \varphi$, is inductively defined as follows:

$$\begin{aligned}
&\vec{x}, t \models \mathit{true} \\
&\vec{x}, t \models f(\vec{x}) > 0 \;\text{ iff }\; f(\vec{x}(t)) > 0 \\
&\vec{x}, t \models \neg\varphi \quad\;\; \text{iff }\; \vec{x}, t \not\models \varphi \\
&\vec{x}, t \models \varphi \wedge \varphi' \quad \text{iff }\; \vec{x}, t \models \varphi \;\text{ and }\; \vec{x}, t \models \varphi \\
&\vec{x}, t \models \varphi \mathbf{U}_I^K \varphi' \quad \text{iff }\; (\exists t' \geq t)\; t' \in K,\;\; t' - t \in I,\;\; \vec{x}, t' \models \varphi',\;\; \text{and } (\forall t'' \in [t, t'])\; \vec{x}, t'' \models \varphi
\end{aligned}$$

3.2.1 Expressiveness of STL-GT. Any STL formula can be written in STL-GT. E.g., the STL formula $\Box_{\leq 5}(x > 0 \rightarrow \Diamond_{[1,3]}\, y < 10)$ is written as the STL-GT formula $\Box^{[0,\infty)}_{\leq 5}(x > 0 \rightarrow \Diamond^{[0,\infty)}_{[1,3]}\, y < 10)$. Notice that $\mathbf{U}_I$ and $\mathbf{U}^{[0,\infty)}_I$ have exactly the same meaning by definition. This implies that an STL-GT formula with only global interval $[0,\infty)$ can be equivalently rewritten into an STL formula by replacing each occurrence of $\mathbf{U}^{[0,\infty)}_I$ by $\mathbf{U}_I$, and vice versa. Therefore, STL-GT with only global interval $[0,\infty)$ has the same expressive power as STL.

STL-GT is related to timed propositional temporal logic (TPTL) [Alur and Henzinger 1994]. In the same way that STL extends metric temporal logic, one may define a signal version of TPTL. TPTL formulas can contain many clock variables. For example, the TPTL formula

$$x.(\Box y.(y \leq 5 \rightarrow (p \rightarrow \Diamond z.(z \leq 2 \wedge x > 1 \wedge q))))$$

contains three clock variables x, y, and z. Any STL formula can be written using one clock variable denoting each local time; e.g., the STL formula $\Box_{\leq 5}(x > 0 \rightarrow \Diamond_{[1,3]} y < 10)$ is written as the formula $\Box l.(l \leq 5 \wedge x > 0 \rightarrow \Diamond l.(l \in [1,3] \wedge y < 10))$. Similarly, any STL-GT formula can be written in TPTL using two clock variables, one for local times, and the other for global times.

TPTL is strictly more expressive than MTL without past temporal operators. (MTL with past has the same expressiveness power as TPTL [Hunter et al. 2013].) Bouyer *et al.* showed that the TPTL formula $x.\Diamond(p \wedge x \leq 1 \wedge \Box(x \leq 1 \rightarrow \neg q))$ has no equivalent formula in MTL [Bouyer et al. 2005]. However, it can be written as $\Diamond^{\leq 1}_{[0,\infty)}(p \wedge \Box^{\leq 1}_{[0,\infty)} \neg q)$ using the STL-GT syntax. Since STL is a signal version of MTL, this formula demonstrates that STL-GT is strictly more expressive than STL.

Proposition 3.3 (Expressiveness of STL-GT). *STL-GT is strictly more expressive than STL. STL-GT with only global interval* $[0,\infty)$ *is as expressive as STL.*

Remark. It is worth noting that STL-GT captures a subclass of TPTL that is expressive enough for separation of STL. The full expressiveness of TPTL is not needed in this paper. Moreover, the greater expressiveness of TPTL makes it more difficult to define a separation of TPTL formulas, because TPTL allows writing any quantifier-free constraint over multiple clock variables.

3.3 Separation of STL-GT

We first show a number of equivalences to obtain a separation of STL-GT. Usual equivalence laws for STL, including distributive laws for the *Until* operator, also hold for STL-GT as follows.

Lemma 3.4. *For nonnegative intervals* $I, J, K \subseteq \mathbb{R}^+$*, we have the following equivalences:*

(1) $\varphi \mathbf{U}^K_I(\varphi' \vee \psi') \equiv \varphi \mathbf{U}^K_I \varphi' \vee \varphi \mathbf{U}^K_I \psi'$ *(2)* $(\varphi \wedge \psi)\mathbf{U}^K_I \varphi' \equiv \varphi \mathbf{U}^K_I \varphi' \wedge \psi \mathbf{U}^K_I \varphi'$

(3) $\varphi \mathbf{U}^K_{I \cup J} \varphi' \equiv \varphi \mathbf{U}^K_I \varphi' \vee \varphi \mathbf{U}^K_J \varphi'$ *(4)* $\varphi \mathbf{U}^{K \cup L}_I \varphi' \equiv \varphi \mathbf{U}^K_I \varphi' \vee \varphi \mathbf{U}^L_I \varphi'$

In STL-GT, the zero interval $\{0\}$ can be used to bound its global temporal context. If $\vec{x}, t \models \varphi \mathbf{U}^K_{=0} \varphi'$, the global time t must be in the interval K. From this we have the following equivalences.

Lemma 3.5. *For nonnegative intervals* $I, K, L \subseteq \mathbb{R}^+$*, we have:*

(1) $\varphi \mathbf{U}^K_I(\Diamond^L_{=0} \varphi') \equiv \varphi \mathbf{U}^{K \cap L}_I \varphi'$ *(2)* $\Diamond^K_I(\Diamond^L_{=0} \varphi) \equiv \Diamond^{K \cap L}_I \varphi$ *(3)* $\Box^K_I(\Box^L_{=0} \varphi) \equiv \Box^{K \cap L}_I \varphi$

(4) $(\Box^K_{=0} \varphi)\mathbf{U}^L_I \varphi' \equiv \Diamond^L_I \varphi'$*, if* $\sup(L) \leq \inf(K)$ *and* $L \cap K = \emptyset$

Consider the problem of separating an STL-GT formula $\varphi \mathbf{U}^K_I \varphi'$ at time τ that is earlier than the global interval K. As depicted in Fig. 4, $\varphi \mathbf{U}^K_I \varphi'$ holds if and only if (i) the subformula φ holds until τ, and (ii) the rest of the formula $\varphi \mathbf{U}^K_I \varphi'$ holds after τ. The first condition is written as the STL-GT formula $\Box^{\leq \tau}_{\geq 0} \varphi$, and the second condition can be specified by *restricting* the global temporal context to the interval (τ, ∞). Similarly, we can separate a formula with respect to the local interval I, where the second condition can be specified by *shifting* the local temporal context by time τ.

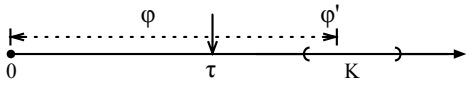

Fig. 4. Separation of $\varphi \mathbf{U}_I^K \varphi'$ at time $\tau \le \inf(K)$

Lemma 3.6. *For a time τ and nonnegative intervals $I, K \subseteq \mathbb{R}^+$, we have:*

(1) $\varphi \mathbf{U}_I^K \varphi' \equiv \Box_{\ge 0}^{\le \tau} \varphi \wedge (\Box_{=0}^{>\tau} \varphi) \mathbf{U}_I^K \varphi'$, *if* $0 \le \tau \le \inf(K)$.[2]

(2) $\varphi \mathbf{U}_I^K \varphi' \equiv \Box_{<\tau}^{\ge 0} \varphi \wedge \Diamond_{=\tau}^{\ge 0} (\varphi \mathbf{U}_{I-\tau}^K \varphi')$, *if* $0 \le \tau \le \inf(I)$.

We now present a separation of an STL-GT formula $\varphi \mathbf{U}_I^K \varphi'$ at an arbitrary time τ, based on the equivalence explained above. Again, we fix either the local interval I or the global interval K, and separate the other interval at time τ. Hence, there are two ways to separate STL-GT formulas:

- *Global separation*: the global interval K is separated into $[0, \tau)$, $\{\tau\}$, and (τ, ∞).
- *Local separation*: the local interval I is separated into $[0, \tau)$, $\{\tau\}$, and (τ, ∞).

Using the global separation of STL-GT, we can syntactically separate the entire STL-GT formula so that the satisfaction of each subformula exclusively depends on disjoint segments of a signal. On the contrary, the local separation of STL-GT has the same limitation as the STL case.

Theorem 3.7 (Separation of $\mathbf{U}_I^K$). *For a time τ:*

(1) $\varphi \mathbf{U}_I^K \varphi' \equiv \varphi \mathbf{U}_I^{K \cap [0,\tau)} \varphi' \vee (\Box_{\ge 0}^{<\tau} \varphi \wedge \Box_{\ge 0}^{=\tau} \varphi \wedge (\Diamond_I^{K \cap \{\tau\}} \varphi' \vee (\Box_{=0}^{>\tau} \varphi) \mathbf{U}_I^{K \cap (\tau,\infty)} \varphi'))$

(2) $\varphi \mathbf{U}_I^K \varphi' \equiv \varphi \mathbf{U}_{I \cap [0,\tau)}^K \varphi' \vee (\Box_{<\tau}^{\ge 0} \varphi \wedge ((\Box_{=\tau}^{\ge 0} \varphi \wedge \Diamond_{I \cap \{\tau\}}^K \varphi') \vee \Diamond_{=\tau}^{\ge 0} (\varphi \mathbf{U}_{I \cap (\tau,\infty) - \tau}^K \varphi')))$

Proof. (1) By applying Lemma 3.4, we have $\varphi \mathbf{U}_I^K \varphi' \equiv \varphi \mathbf{U}_I^{K \cap [0,\tau)} \varphi' \vee \varphi \mathbf{U}_I^{K \cap \{\tau\}} \varphi' \vee \varphi \mathbf{U}_I^{K \cap (\tau,\infty)} \varphi'$. We still need to separate the formulas $\varphi \mathbf{U}_I^{K \cap \{\tau\}} \varphi'$ and $\varphi \mathbf{U}_I^{K \cap (\tau,\infty)} \varphi'$, because they depend on the initial segment of a signal over the interval $[0, \tau)$. For $\varphi \mathbf{U}_I^{K \cap \{\tau\}} \varphi'$, we obtain the equivalence $\varphi \mathbf{U}_I^{K \cap \{\tau\}} \varphi' \equiv \Diamond_I^{K \cap \{\tau\}} \varphi' \wedge \Box_{\ge 0}^{\le \tau} \varphi$ as follows. Since $t' \in K \cap \{\tau\}$ implies $[t, t'] = [t, \tau]$, we have:

$$\begin{aligned} \vec{x}, t \models \varphi \mathbf{U}_I^{K \cap \{\tau\}} \varphi' \text{ iff } & (\exists t' \ge t)\ t' \in K \cap \{\tau\},\ t' - t \in I,\ \vec{x}, t' \models \varphi',\ \text{and } (\forall t'' \in [t, \tau])\ \vec{x}, t'' \models \varphi \\ \text{iff } & \vec{x}, t \models \Diamond_I^{K \cap \{\tau\}} \varphi',\ \text{and } \vec{x}, t \models \Box_{\ge 0}^{\le \tau} \varphi \end{aligned}$$

For $\varphi \mathbf{U}_I^{K \cap (\tau,\infty)} \varphi'$, we obtain the equivalence $\varphi \mathbf{U}_I^{K \cap (\tau,\infty)} \varphi' \equiv \Box_{\ge 0}^{\le \tau} \varphi \wedge (\Box_{=0}^{>\tau} \varphi)\, \mathbf{U}_I^{K \cap (\tau,\infty)} \varphi'$ as follows. If $K \cap (\tau, \infty) \ne \emptyset$, $0 \le \tau \le \inf(K \cap (\tau, \infty))$ and so we apply Lemma 3.6. If $K \cap (\tau, \infty) = \emptyset$, both sides are equivalent to *false*. Hence, by the distributive laws and $\Box_{\le \tau}^{\ge 0} \varphi \equiv \Box_{<\tau}^{\ge 0} \varphi \wedge \Box_{=\tau}^{\ge 0} \varphi$ by Lemma 3.4:

$$\begin{aligned} \varphi \mathbf{U}_I^K \varphi' &\equiv \varphi \mathbf{U}_I^{K \cap [0,\tau)} \varphi' \vee (\Box_{\ge 0}^{\le \tau} \varphi \wedge \Diamond_I^{K \cap \{\tau\}} \varphi') \vee (\Box_{=0}^{\le \tau} \varphi \wedge (\Box_{=0}^{>\tau} \varphi)\, \mathbf{U}_I^{K \cap (\tau,\infty)} \varphi') \\ &\equiv \varphi \mathbf{U}_I^{K \cap [0,\tau)} \varphi' \vee (\Box_{\ge 0}^{<\tau} \varphi \wedge \Box_{\ge 0}^{=\tau} \varphi \wedge (\Diamond_I^{K \cap \{\tau\}} \varphi' \vee (\Box_{=0}^{>\tau} \varphi)\, \mathbf{U}_I^{K \cap (\tau,\infty)} \varphi')) \end{aligned}$$

(2) Similarly, by Lemma 3.4, $\varphi \mathbf{U}_I^K \varphi' \equiv \varphi \mathbf{U}_{I \cap [0,\tau)}^K \varphi' \vee \varphi \mathbf{U}_{I \cap \{\tau\}}^K \varphi' \vee \varphi \mathbf{U}_{I \cap (\tau,\infty)}^K \varphi'$. By definition, we can easily see the equivalence $\varphi \mathbf{U}_{I \cap \{\tau\}}^K \varphi' \equiv \Box_{\le \tau}^{\ge 0} \varphi \wedge \Diamond_{I \cap \{\tau\}}^K \varphi'$. By Lemma 3.6, we obtain the equivalence $\varphi \mathbf{U}_{I \cap (\tau,\infty)}^K \varphi' \equiv \Box_{<\tau}^{\ge 0} \varphi \wedge \Diamond_{=\tau}^{\ge 0} (\varphi\, \mathbf{U}_{I \cap (\tau,\infty) - \tau}^K \varphi')$. Consequently:

$$\begin{aligned} \varphi \mathbf{U}_I^K \varphi' &\equiv \varphi \mathbf{U}_{I \cap [0,\tau)}^K \varphi' \vee (\Box_{\le \tau}^{\ge 0} \varphi \wedge \Diamond_{I \cap \{\tau\}}^K \varphi') \vee (\Box_{<\tau}^{\ge 0} \varphi \wedge \Diamond_{=\tau}^{\ge 0} (\varphi\, \mathbf{U}_{I \cap (\tau,\infty) - \tau}^K \varphi')) \\ &\equiv \varphi \mathbf{U}_{I \cap [0,\tau)}^K \varphi' \vee (\Box_{<\tau}^{\ge 0} \varphi \wedge ((\Box_{=\tau}^{\ge 0} \varphi \wedge \Diamond_{I \cap \{\tau\}}^K \varphi') \vee \Diamond_{=\tau}^{\ge 0} (\varphi\, \mathbf{U}_{I \cap (\tau,\infty) - \tau}^K \varphi'))). \end{aligned}$$

□

[2] Notice that we use the zero interval $\{0\}$ to restrict the global temporal context of the left operand φ to the interval (τ, ∞) in the formula $(\Box_{=0}^{>\tau} \varphi) \mathbf{U}_I^K \varphi'$, by applying Lemma 3.5.

Example 3.8. Consider the STL-GT formula $\varphi = \Box^{\geq 0}_{[1,5]}(x > 0 \rightarrow \Diamond^{\geq 0}_{[1,3]}\, y < 10)$ again. We can separate each subformula at time 2 by the global separation as follows:

$$\Box^{<2}_{[1,5]}(x > 0 \rightarrow \phi) \ \wedge\ \Box^{=2}_{[1,5]}(x > 0 \rightarrow \phi) \ \wedge\ \Box^{>2}_{[1,5]}(x > 0 \rightarrow \phi),$$

where $\phi \equiv \Diamond^{<2}_{[1,3]} y < 10 \vee \Diamond^{=2}_{[1,3]} y < 10 \vee \Diamond^{>2}_{[1,3]} y < 10$. The satisfaction of each subformula exclusively depends on one of the disjoint segments over $[0, 2)$, $\{2\}$, and $(2, \infty)$.

3.4 Global Separation of STL and Restricted Operators

Theorem 3.7 allows separating STL-GT formulas at a given time τ. Because any STL formula can be considered as an STL-GT formula with the global interval $[0, \infty)$ by Proposition 3.3, we can also separate STL formulas by Theorem 3.7. This section shows the interesting result that the global separation of STL needs only a subclass of STL-GT with restricted temporal operators.

Definition 3.9. The restricted *Until* operator $\tilde{\mathbf{U}}^K_I$ is defined as the syntactic abbreviation:

$$\varphi \tilde{\mathbf{U}}^K_I \varphi' \ \equiv\ (\Box^K_{=0}\varphi)\mathbf{U}^K_I\, \varphi'$$

The restricted *Until* operator $\tilde{\mathbf{U}}^K_I$ also bounds the global temporal context of the left operand φ to the interval K. Using $\tilde{\mathbf{U}}^K_I$ instead of $\mathbf{U}^K_I$, we have the subclass of STL-GT, written STL-GT[$\tilde{\mathbf{U}}$]. Other temporal operators can also be defined as syntactic abbreviations in STL-GT[$\tilde{\mathbf{U}}$]:

$$\Diamond^K_I\, \varphi \ \equiv\ \mathit{true}\, \tilde{\mathbf{U}}^K_I \varphi, \qquad \Box^K_I\, \varphi \ \equiv\ \neg\Diamond^K_I\, \neg\varphi, \qquad \varphi \tilde{\mathbf{R}}^K_I \varphi' \ \equiv\ \neg((\neg\varphi)\tilde{\mathbf{U}}^K_I(\neg\varphi'))$$

We can easily see that the operators $\Box^K_I$ and $\Diamond^K_I$ have the same meaning in both STL-GT and STL-GT[$\tilde{\mathbf{U}}$]. Applying Theorem 3.7 to the restricted *Until* operator, any STL-GT[$\tilde{\mathbf{U}}$] formula can be globally separated at an arbitrary time τ, as stated in the following proposition.

Proposition 3.10 (Global Separation of $\tilde{\mathbf{U}}^K_I$). *For a time τ:*

$$\varphi \tilde{\mathbf{U}}^K_I \varphi' \ \equiv\ \varphi \tilde{\mathbf{U}}^{K\cap[0,\tau)}_I \varphi' \vee (\Box^{K\cap[0,\tau)}_{\geq 0}\varphi \wedge \Box^{K\cap\{\tau\}}_{\geq 0}\varphi \wedge (\Diamond^{K\cap\{\tau\}}_I \varphi' \vee \varphi \tilde{\mathbf{U}}^{K\cap(\tau,\infty)}_I \varphi'))$$

Proof. Recall $\varphi \tilde{\mathbf{U}}^K_I \varphi' \equiv (\Box^K_{=0}\varphi)\mathbf{U}^K_I\varphi'$. By Theorem 3.7, $(\Box^K_{=0}\varphi)\mathbf{U}^K_I\varphi'$ is equivalent to:

$$(\Box^K_{=0}\varphi)\mathbf{U}^{K\cap[0,\tau)}_I\, \varphi' \vee (\Box^{<\tau}_{\geq 0}\Box^K_{=0}\varphi \wedge \Box^{=\tau}_{\geq 0}\Box^K_{=0}\varphi \wedge (\Diamond^{K\cap\{\tau\}}_I \varphi' \vee (\Box^{>\tau}_{=0}\Box^K_{=0}\varphi)\, \mathbf{U}^{K\cap(\tau,\infty)}_I\, \varphi'))$$

Using the equivalence $\Box^K_I(\Box^L_{=0}\,\varphi) \equiv \Box^{K\cap L}_I\varphi$ in Lemma 3.5, we can rewrite the formula as:

$$(\Box^K_{=0}\varphi)\mathbf{U}^{K\cap[0,\tau)}_I\, \varphi' \vee (\Box^{K\cap[0,\tau)}_{\geq 0}\varphi \wedge \Box^{K\cap\{\tau\}}_{\geq 0}\varphi \wedge (\Diamond^{K\cap\{\tau\}}_I \varphi' \vee (\Box^{K\cap(\tau,\infty)}_{=0}\varphi)\, \mathbf{U}^{K\cap(\tau,\infty)}_I\, \varphi'))$$

Because $\sup(K \cap [0, \tau)) \leq \inf(K \cap [\tau, \infty))$, by Lemmas 3.4 and 3.5, we have the rewriting steps:

$$\begin{aligned}
(\Box^K_{=0}\varphi)\mathbf{U}^{K\cap[0,\tau)}_I\varphi' \ &\equiv\ (\Box^{K\cap[0,\tau)}_{=0}\varphi \wedge \Box^{K\cap[\tau,\infty)}_{=0}\varphi)\mathbf{U}^{K\cap[0,\tau)}_I\varphi' \\
&\equiv\ (\Box^{K\cap[0,\tau)}_{=0}\varphi)\mathbf{U}^{K\cap[0,\tau)}_I\varphi' \wedge (\Box^{K\cap[\tau,\infty)}_{=0}\varphi)\mathbf{U}^{K\cap[0,\tau)}_I\varphi' \\
&\equiv\ (\Box^{K\cap[0,\tau)}_{=0}\varphi)\mathbf{U}^{K\cap[0,\tau)}_I\varphi' \ \wedge\ \mathit{true}\mathbf{U}^{K\cap[0,\tau)}_I\varphi' \ \equiv\ (\Box^{K\cap[0,\tau)}_{=0}\varphi)\mathbf{U}^{K\cap[0,\tau)}_I\varphi'
\end{aligned}$$

As a consequence, the formula $(\Box^K_{=0}\varphi)\mathbf{U}^K_I\varphi'$ is equivalent to one in the desired form:

$$(\Box^{K\cap[0,\tau)}_{=0}\varphi)\mathbf{U}^{K\cap[0,\tau)}_I\varphi' \vee (\Box^{K\cap[0,\tau)}_{\geq 0}\varphi \wedge \Box^{K\cap\{\tau\}}_{\geq 0}\varphi \wedge (\Diamond^{K\cap\{\tau\}}_I \varphi' \vee (\Box^{K\cap(\tau,\infty)}_{=0}\varphi)\, \mathbf{U}^{K\cap(\tau,\infty)}_I\, \varphi')). \quad \square$$

This proposition implies that STL-GT[$\tilde{\mathbf{U}}$] is precisely related to the global separation of STL. Any STL formula can be written in STL-GT[$\tilde{\mathbf{U}}$] with the global interval $[0, \infty)$, because $\mathbf{U}_I$ and $\tilde{\mathbf{U}}^{[0,\infty)}_I$ also have exactly the same meaning. STL-GT[$\tilde{\mathbf{U}}$] can be considered as the smallest syntactic subclass of STL-GT that includes STL and is closed under the global separation. As a matter of fact, STL-GT[$\tilde{\mathbf{U}}$] is used for symbolic model checking of STL in the rest of the paper.

Example 3.11. Consider the STL formula $\Diamond_{\le 1}(x \le 0 \wedge (y < 5)\mathbf{U}_{\le 2}(x > 0))$. By separating each subformula at time 1 using Proposition 3.10, we obtain the STL-GT[$\tilde{\mathbf{U}}$] formula:

$$\Diamond^{<1}_{\le 1}(x \le 0 \wedge \phi) \wedge \Diamond^{=1}_{\le 1}(x \le 0 \wedge \phi) \wedge \Diamond^{>1}_{\le 1}(x \le 0 \wedge \phi)$$

where $\phi \equiv (y < 5)\tilde{\mathbf{U}}^{<1}_{\le 2}(x > 0) \vee (\Box^{<1}_{\ge 0}(y < 5) \wedge \Box^{=1}_{\ge 0}(y < 5) \wedge (\Diamond^{=1}_{\le 2}(x > 0) \vee (y < 5)\tilde{\mathbf{U}}^{>1}_{\le 2}(x > 0)))$.

4 TRANSLATION OF STL TO FIRST-ORDER LOGIC

This section explains how to translate STL into first-order logic for symbolic model checking. We first define fully stable STL-GT[$\tilde{\mathbf{U}}$] formulas that behave like propositional formulas with respect to a signal with finite variable points. We then present a procedure to translate a fully stable STL-GT[$\tilde{\mathbf{U}}$] formula into a quantifier-free first-order logic formula. We show that any STL formula can be equivalently rewritten into a fully stable STL-GT[$\tilde{\mathbf{U}}$] formula by the syntactic separation.

4.1 Full Stability

Recall that a time point τ is a *variable point* of a signal if the truth value of a proposition changes at the time τ. For a set of propositions AP, when a signal $\vec{x}$ has no variable point in a particular interval I, the truth value of every proposition in AP is clearly fixed in I by definition. Such a stability condition was previously studied for MTL, and has been extended to formulas with respect to timed words [Alur et al. 1996]. Similarly, we define the stability condition for STL-GT formulas.

Definition 4.1. An STL-GT formula φ is called *stable* for an interval I with respect to a signal $\vec{x}$, if for all time points $u, u' \in I$, $\vec{x}, u \models \varphi$ iff $\vec{x}, u' \models \varphi$.

The stability condition is important for STL-GT, as a formula can be treated as a propositional formula if every subformula is stable. Consider the formula $\varphi_1 = \Box^{<9}_{<4}(x > 0 \rightarrow \Diamond^{<9}_{[3,5]}\, x > 0)$, and a signal x with the single variable point 9, say, $x(t) > 0$ for $t \in [0, 9)$. In this case, every subformula is stable, and the satisfaction can be easily determined like a propositional formula as follows:

$$\begin{aligned}
& x, 0 \models \Box^{<9}_{<4}(x > 0 \rightarrow \Diamond^{<9}_{[3,5]}\, x > 0), \\
\text{iff}\ \ & x, 0 \models \Box^{<9}_{<4}(\mathit{true} \rightarrow \Diamond^{<9}_{[3,5]}\, \mathit{true}), && \text{because } x > 0 \text{ is stable for } [0, 9) \\
\text{iff}\ \ & x, 0 \models \Box^{<9}_{<4}(\mathit{true} \rightarrow \mathit{true}), && \text{because } \Diamond^{<9}_{[3,5]}\, \mathit{true} \text{ is stable for } [0, 4) \\
\text{iff}\ \ & x, 0 \models \mathit{true}
\end{aligned}$$

In STL-GT, the truth value of a formula may still vary according to its global temporal context, even if all the propositions are stable. A formula $\varphi \mathbf{U}^K_I \varphi'$ can be vacuously falsified regardless of the satisfaction of its subformulas φ and φ', when the time constraint imposed by the intervals K and I cannot be satisfied. For example, consider another STL-GT formula

$$\varphi_2 = \Box^{<4}_{<4}(x > 0 \rightarrow \Diamond^{<4}_{[3,5]} x > 0),$$

obtained from the above formula φ_1 by replacing the global interval $[0, 9)$ by $[0, 4)$. For the same signal x with the single variable point 9, the subformula $\Diamond^{<4}_{[3,5]} x > 0$ is not stable in the interval $[0, 4)$ anymore. For example, $x, 0 \models \Diamond^{<4}_{[3,5]} x > 0$, but $x, 3 \not\models \Diamond^{<4}_{[3,5]} x > 0$. Specifically, $\vec{x}, t \not\models \Diamond^{<4}_{[3,5]} x > 0$ for any $t \ge 1$, since there exists no $t' \ge t$ such that $t' - t \in [3, 5]$ and $t' \in [0, 4)$ in this case.

This observation motivates our definition of the full stability. It characterizes a sufficient condition for STL-GT[$\tilde{\mathbf{U}}$] that all subformulas become stable with respect to their global temporal contexts. (We will explain how to obtain a fully stable formula using the syntactic separation in Sec. 4.3.) Let us first define some interval operations for nonnegative intervals: $J - I = \{j - i \mid j \in J, i \in I\}$, $J \dot{-} I = (J - I) \cap \mathbb{R}^+$, and $(J \dot{-} I)^{\complement} = \mathbb{R}^+ \setminus (J \dot{-} I)$, for nonnegative intervals $I, J \subseteq \mathbb{R}^+$.

Definition 4.2 (Full Stability). The *full stability* of an STL-GT[$\tilde{\mathbf{U}}$] formula φ for an interval J with respect to a signal $\vec{x}$, written $stable_{\vec{x}}(\varphi, J)$, is inductively defined as follows:

$$
\begin{aligned}
&stable_{\vec{x}}(true, J) \\
&stable_{\vec{x}}(f(\vec{x}) > 0, J) \quad \text{iff} \quad f(\vec{x}) > 0 \text{ is stable for } J \text{ with respect to } \vec{x} \\
&stable_{\vec{x}}(\neg\varphi, J) \quad \text{iff} \quad stable_{\vec{x}}(\varphi, J) \\
&stable_{\vec{x}}(\varphi \wedge \varphi', J) \quad \text{iff} \quad stable_{\vec{x}}(\varphi, J) \text{ and } stable_{\vec{x}}(\varphi', J) \\
&stable_{\vec{x}}(\varphi \tilde{\mathbf{U}}_I^K \varphi', J) \quad \text{iff} \quad stable_{\vec{x}}(\varphi, K),\ stable_{\vec{x}}(\varphi', K), \text{ and } (J \subseteq K \dot{-} I,\ \text{or } J \subseteq (K \dot{-} I)^{\complement})
\end{aligned}
$$

A simple but important idea is to express the satisfiability of the time constraint of an STL-GT[$\tilde{\mathbf{U}}$] formula $\varphi \tilde{\mathbf{U}}_I^K \varphi'$ as a *quantifier-free* expression in interval arithmetic as follows.

Lemma 4.3. *For $t \geq 0$ and nonempty intervals $I, K \in \mathbb{R}^+$, $t \in K \dot{-} I$ iff $(\exists t' \geq t)\ t' \in K$ and $t' - t \in I$.*

Proof. ($\Rightarrow$) By definition, $t = k - i$ for some $k \in K$ and $i \in I$. Let $t' = k$. Observe that $t' \in K$ and $t' - t = i \in I$. ($\Leftarrow$) Since $t - t' \in -I$, $t' \in K$, and $t \geq 0$, we have $t \in K \dot{-} I$. □

For an interval J, if an STL-GT[$\tilde{\mathbf{U}}$] formula φ is fully stable, φ is also stable for J. If any point in J cannot satisfy its time constraint, $\varphi \tilde{\mathbf{U}}_I^K \varphi'$ will be vacuously falsified. If every point in J satisfy the time constraint, the satisfaction of $\varphi \tilde{\mathbf{U}}_I^K \varphi'$ will be entirely determined by its subformulas.

Lemma 4.4. *If an STL-GT[$\tilde{\mathbf{U}}$] formula φ is fully stable for an interval J with respect to a signal $\vec{x}$, then the formula φ is stable for the interval J with respect to the signal $\vec{x}$.*

Proof. The proof is by structural induction on φ. The only nontrivial case is $\varphi \tilde{\mathbf{U}}_I^K \varphi'$, because the other cases are immediate by definition and induction hypothesis. Suppose $stable_{\vec{x}}(\varphi \tilde{\mathbf{U}}_I^K \varphi', J)$ holds. Then, both $stable_{\vec{x}}(\varphi, K)$ and $stable_{\vec{x}}(\varphi', K)$ hold, and either $J \subseteq K \dot{-} I$ or $J \subseteq (K \dot{-} I)^{\complement}$ holds. If $J = \emptyset$, $\varphi \tilde{\mathbf{U}}_I^K \varphi'$ is stable for J by definition. If $I = \emptyset$ or $K = \emptyset$, $\varphi \tilde{\mathbf{U}}_I^K \varphi'$ is *false* and so is stable. Now let us assume $I, J, K \neq \emptyset$. Consider $t_1, t_2 \in J$. For $\ell = 1, 2$, by definition,

$$\vec{x}, t_\ell \models \varphi \tilde{\mathbf{U}}_I^K \varphi' \quad \text{iff} \quad (\exists t'_\ell \geq t_\ell)\ t'_\ell \in K,\ t'_\ell - t_\ell \in I,\ \vec{x}, t'_\ell \models \varphi',\ \text{and } (\forall t'' \in [t_\ell, t'_\ell] \cap K)\ \vec{x}, t'' \models \varphi.$$

There are two cases. When $J \subseteq K \dot{-} I$, since $t_\ell \in J$, by Lemma 4.3, there exists $t'_\ell \geq t_\ell$ such that $t'_\ell \in K$ and $t'_\ell - t_\ell \in I$ for $\ell = 1, 2$. Also, $[t_\ell, t'_\ell] \cap K \neq \emptyset$, because $t'_\ell \geq t_\ell$ and $t'_\ell \in K$. By induction hypothesis, both φ and φ' are stable for the interval K. Therefore,

$$\vec{x}, t'_1 \models \varphi' \ \text{ iff } \ \vec{x}, t'_2 \models \varphi', \quad \text{and} \quad (\forall t'' \in [t_1, t'_1] \cap K)\ \vec{x}, t'' \models \varphi \ \text{ iff } \ (\forall t'' \in [t_2, t'_2] \cap K)\ \vec{x}, t'' \models \varphi.$$

Therefore, $\varphi \tilde{\mathbf{U}}_I^K \varphi'$ is stable for J. When $J \subseteq (K \dot{-} I)^{\complement}$, by Lemma 4.3, there exists no $t'_\ell \geq t_\ell$ such that $t'_\ell \in K$ and $t'_\ell - t_\ell \in I$. Therefore, $\vec{x}, t_\ell \not\models \varphi \tilde{\mathbf{U}}_I^K \varphi'$, for $\ell = 1, 2$. Thus, $\varphi \tilde{\mathbf{U}}_I^K \varphi'$ is stable for J. □

Notice that if a formula $\varphi \tilde{\mathbf{U}}_I^K \varphi'$ is fully stable for an interval J, then both φ and φ' are fully stable for its global temporal context K. As a consequence of Lemma 4.4, if an STL-GT[$\tilde{\mathbf{U}}$] formula is fully stable, then every subformula is stable for its global temporal context. Therefore, Lemma 4.4 provides a sufficient condition for each subformula to be stable for its global temporal context.

Remark. It is crucial in Def. 4.2 that only STL-GT[$\tilde{\mathbf{U}}$] formulas are taken into account. In the proof of Lemma 4.4, to apply the induction hypothesis, it is necessary to restrict the global temporal context of the left operand of $\varphi \tilde{\mathbf{U}}_I^K \varphi'$ to the interval K. For the normal *Until* operator, the global temporal context of φ also includes the initial segment $[0, \inf K]$, in addition to K.

4.2 First-Order Logic Translation of Fully Stable Formulas

It is straightforward to translate a fully stable STL-GT[$\tilde{\mathbf{U}}$] formula into first-order logic. Because each subformula of a fully stable formula is stable for its global temporal context, as mentioned above, the satisfaction of the entire formula can be evaluated just like a propositional formula. Using this fact, we present a simple procedure to translation a fully stable STL-GT[$\tilde{\mathbf{U}}$] formula φ into a quantifier-free first-order logic formula of size $O(|\varphi|)$ as follows.

Definition 4.5 (First-Order Translation). For an STL-GT[$\tilde{\mathbf{U}}$] formula φ and a signal $\vec{x}$, the first-order translation $fotr_{\vec{x}}(\varphi, J)$ for a nonempty interval J is inductively defined by:

$$
\begin{aligned}
fotr_{\vec{x}}(true, J) &= true \\
fotr_{\vec{x}}(f(\vec{x}) > 0, J) &= f(\vec{x}(a)) > 0, \quad \text{for } a \in J \ \ (\text{e.g., } a = (\sup(J) + \inf(J))/2) \\
fotr_{\vec{x}}(\neg\varphi, J) &= \neg\, fotr_{\vec{x}}(\varphi, J) \\
fotr_{\vec{x}}(\varphi \wedge \varphi', J) &= fotr_{\vec{x}}(\varphi, J) \ \wedge \ fotr_{\vec{x}}(\varphi', J) \\
fotr_{\vec{x}}(\varphi \tilde{\mathbf{U}}_I^K \varphi', J) &= J \subseteq K \dot{-} I \ \wedge \ fotr_{\vec{x}}(\varphi, K) \ \wedge \ fotr_{\vec{x}}(\varphi', K)
\end{aligned}
$$

Example 4.6. Consider the formula $\varphi = \Box_{[0,2]}^{(2,4)}(x > 0 \rightarrow \Diamond_{[1,3]}^{<5} y < 5)$, and a signal with one variable point $\{5\}$. The formula φ is fully stable for $[1,2]$, because $(2,4) \subseteq [0,5) \dot{-} [1,3] = [0,4)$ and $[1,2] \subseteq (2,4) \dot{-} [0,2] = (0,4)$. The first-order translation of φ for $[1,2]$ is then:

$$
\begin{aligned}
&fotr_{\vec{x}}(\Box_{[0,2]}^{(2,4)}(x > 0 \rightarrow \Diamond_{[1,3]}^{<5} y < 5),\ [1,2]) \\
&= [1,2] \subseteq (2,4) \dot{-} [0,2] \rightarrow (fotr_{\vec{x}}(x > 0, (2,4)) \rightarrow fotr_{\vec{x}}(\Diamond_{[1,3]}^{<5} y < 5, (2,4))) \\
&= [1,2] \subseteq (2,4) \dot{-} [0,2] \rightarrow (x(3) > 0 \rightarrow ((2,4) \subseteq [0,5) \dot{-} [1,3] \ \wedge \ y(2.5) < 5))
\end{aligned}
$$

An STL-GT[$\tilde{\mathbf{U}}$] formula φ is equisatisfiable to its first-order translation $fotr_{\vec{x}}(\varphi, J)$ over J, provided that φ is fully stable for the interval J with respect to the signal $\vec{x}$. (We will explain how to obtain a fully stable formula in Sec. 4.3 and how to encode the requirements for the signal in Sec. 5.)

Theorem 4.7. *Given an STL-GT[$\tilde{\mathbf{U}}$] formula φ, if φ is fully stable for a nonempty interval J with respect to a signal $\vec{x}$, then* $(\exists t \in J)\ \vec{x}, t \models \varphi \iff fotr_{\vec{x}}(\varphi, J)$.

Proof. The proof is by structural induction on φ. The base cases are immediate by definition.

- $(\varphi = \neg\phi)$: Because $\neg\varphi$ is fully stable for J, the formula φ is fully stable for J. By Lemma 4.4, φ is also stable for $J \neq \emptyset$, and thus $(\exists t \in J)\ \vec{x}, t \models \phi$ iff $(\forall t \in J)\ \vec{x}, t \models \phi$. By induction hypothesis, $(\exists t \in J)\ \vec{x}, t \models \neg\phi$ iff $\neg(\forall t \in J)\ \vec{x}, t \models \phi$ iff $\neg(\exists t \in J)\ \vec{x}, t \models \phi$ iff $\neg fotr_{\vec{x}}(\varphi, J)$.
- $(\varphi = \phi \wedge \phi')$: Because $\phi \wedge \phi'$ is fully stable for J, both ϕ and ϕ' are fully stable for J. First, if $(\exists t \in J)\ \vec{x}, t \models \phi \wedge \phi'$, then $(\exists t \in J)\ \vec{x}, t \models \phi$ and $(\exists t \in J)\ \vec{x}, t \models \phi'$. By induction hypothesis, $fotr_{\vec{x}}(\varphi, J) \wedge fotr_{\vec{x}}(\varphi', J)$. Conversely, if $fotr_{\vec{x}}(\varphi, J) \wedge fotr_{\vec{x}}(\varphi', J)$, by induction hypothesis, $(\exists t \in J)\ \vec{x}, t \models \phi$ and $(\exists t \in J)\ \vec{x}, t \models \phi'$. Since φ and φ' are stable for J, $(\forall t \in J)\ \vec{x}, t \models \phi \wedge \vec{x}, t \models \phi'$.
- $(\varphi = \phi \tilde{\mathbf{U}}_I^K \phi')$: Since $\phi \tilde{\mathbf{U}}_I^K \phi'$ is full stable for J, $stable_{\vec{x}}(\varphi, K)$, $stable_{\vec{x}}(\varphi', K)$, and either $(J \subseteq K \dot{-} I)$ or $(J \subseteq K \dot{-} I)^{\complement}$. Further, by Lemma 4.4, $\phi \tilde{\mathbf{U}}_I^K \phi'$ is stable for J. By definition, $\vec{x}, t \models \phi \tilde{\mathbf{U}}_I^K \phi'$ iff $(\exists t' \geq t)\ t' \in K,\ t' - t \in I,\ \vec{x}, t' \models \phi',$ and $(\forall t'' \in [t, t'] \cap K)\ \vec{x}, t'' \models \phi$.
  - $(\Rightarrow)$ If $(\exists t \in J)\ \vec{x}, t \models \phi \tilde{\mathbf{U}}_I^K \phi'$, since $\phi \tilde{\mathbf{U}}_I^K \phi'$ is stable for $J \neq \emptyset$, $(\forall t \in J)\ \vec{x}, t \models \phi \tilde{\mathbf{U}}_I^K \phi'$. Therefore, from $(\forall t \in J)(\exists t' \geq t)\ t' \in K \wedge t' - t \in I$, by Lemma 4.3, we have $J \subseteq K \dot{-} I$. Since $t' \in K$ and $[t, t'] \cap K \neq \emptyset$, by induction hypothesis, we have $fotr_{\vec{x}}(\phi', K)$ and $fotr_{\vec{x}}(\phi, K)$.
  - $(\Leftarrow)$ If $fotr_{\vec{x}}(\phi \mathbf{U}_I^K \phi', J)$, then $J \subseteq K \dot{-} I$, $fotr_{\vec{x}}(\phi, K)$, and $fotr_{\vec{x}}(\phi', K)$. Since $\emptyset \neq J \subseteq K \dot{-} I$, by Lemma 4.3. $(\exists t \in J)\ (\exists t' \geq t)\ t' \in K \wedge t' - t \in I$. Because ϕ and ϕ' are stable for K, by induction hypothesis, $\vec{x}, t' \models \phi'$ and $(\forall t'' \in [t, t'] \cap K)\ \vec{x}, t'' \models \phi$ hold for any $t' \in K$. □

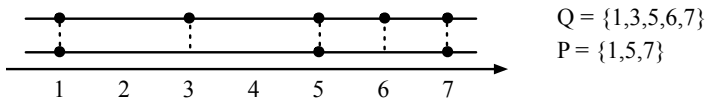

Fig. 5. Two partition $P = \{1, 5, 7\}$ and $Q = \{1, 3, 5, 6, 7\}$, where $P \subseteq Q$

4.3 Full Separation

This section presents a separation procedure to obtain a fully stable STL-GT[$\tilde{\mathbf{U}}$] formula with respect to a signal with finite variable points $\tau_1 < \cdots < \tau_n$. We first introduce some definitions that are necessary to formally define our procedure. For a set of time points $\{\tau_1, \cdots, \tau_n\}$, an interval can be divided into a collection of disjoint subintervals, called a *partition*. For example, $[0, 10)$ can be divided into five subintervals $[0, 1)$, $\{1\}$, $(1, 3)$, $\{3\}$, and $(3, 10)$, by two time points 1 and 3.

Definition 4.8. A finite set of time points $P = \{\tau_1, \ldots, \tau_n\}$ is called a *partition* of an interval D if $\inf(D) \le \tau_1 < \cdots < \tau_n \le \sup(D)$. A partition P divides D into a set of disjoint subintervals:

$$[\![P]\!]_D = \{\{\tau_i\}, (\tau_i, \tau_i + 1) \subseteq D \mid 1 \le i < n\} \cup \{D \cap [0, \tau_1), D \cap (\tau_n, \infty)\}$$

For two partitions P and Q of the same interval D, the partition P is called *finer* than Q if $P \supseteq Q$. A *restriction* $P \restriction E$ by a subinterval $E \subseteq D$ is the partition $P \cap E$.

For two partitions P_1 and P_2 of an interval D, the union $P_1 \cup P_2$ is also a partition of D, which is finer than both of P_1 and P_2. An interval in a finer partition $Q \supseteq P$ is a subset of one in the coarser partition P. For example, consider two partitions $P = \{1, 5, 7\}$ and $Q = \{1, 3, 5, 6, 7\}$ of the interval $[1, \infty)$. As depicted in Fig. 5, any interval in $[\![Q]\!]_{[1,\infty)} = \{\{1\}, (1, 3), \{3\}, (3, 5), \{5\}, (5, 6), \{6\}, (6, 7), \{7\}, (7, \infty)\}$ is a subset of some interval in $[\![P]\!]_{[1,\infty)} = \{\{1\}, (1, 5), \{5\}, (5, 7), \{7\}, (7, \infty)\}$.

Lemma 4.9. *For two partitions P and Q of an interval D:*

(1) if $Q \supseteq P$, for any interval $L \in [\![Q]\!]_D$, there exists $L' \in [\![P]\!]_D$ such that $L \subseteq L'$; and
(2) if $E \subseteq D$, for any interval $L \in [\![P \restriction E]\!]_E$, there exists $L' \in [\![P]\!]_D$ such that $L \subseteq L'$.

We ready to define our full separation procedure. First, we assign to each subformula ϕ a partition $\mathcal{I}(\phi)$. The base partition for propositions is a set of variable points, and a partition of $\varphi \tilde{\mathbf{U}}_I^K \varphi'$ is inductively built using the partitions for their subformulas. The intuition behind the construction is to make $\mathcal{I}(\varphi \tilde{\mathbf{U}}_I^K \varphi')$ *fine enough* to satisfy the full stability condition in Def. 4.2, by adding all the points obtained by subtracting I's endpoints from the partitions for the subformulas.

Definition 4.10 (Partition Construction). For an STL-GT[$\tilde{\mathbf{U}}$] formula φ and a base partition B, a *partition mapping* $\mathcal{I}$ assigns to each subformula ϕ a partition $\mathcal{I}(\phi)$ of $[0, \infty)$, inductively constructed as follows, where $e(I)$ denotes the set of endpoints (e.g., $\inf(I)$ and $\sup(I)$) of an interval I:

$$\begin{aligned}
\mathcal{I}(f(\vec{x}) > 0) &\supseteq B, \\
\mathcal{I}(\neg\varphi) &\supseteq \mathcal{I}(\varphi), \\
\mathcal{I}(\varphi \wedge \varphi') &\supseteq \mathcal{I}(\varphi) \cup \mathcal{I}(\varphi'), \\
\mathcal{I}(\varphi \tilde{\mathbf{U}}_I^K \varphi') &\supseteq \bigcup_{\tau \in P} \{\tau - e \in [0, \infty) \mid e \in e(I)\}, \quad \text{for } P = (\mathcal{I}(\varphi) \cup \mathcal{I}(\varphi') \cup e(K)) \restriction (K \cup e(K)).
\end{aligned}$$

A partition mapping $\mathcal{I}$ is *minimal* if every $\supseteq$ is an equality in the definition. In this case, the size of $\mathcal{I}(\varphi \tilde{\mathbf{U}}_I^K \varphi')$ is at most $2|(\mathcal{I}(\varphi) \cup \mathcal{I}(\varphi') \cup e(K)) \restriction (K \cup e(K))|$. In total, the size of $\mathcal{I}$ is $O(k \cdot 2^{h(\varphi)})$, where k is the size of the base partition B, and $h(\varphi)$ is the height of the formula φ.

Example 4.11. Consider the STL-GT[$\tilde{\mathbf{U}}$] formula $\varphi = \Box^{\geq 0}_{[0,2]}(x > 0 \to \Diamond^{\geq 0}_{[1,3]}\, y < 5)$ and the base partition $\{0, 5, 7\}$. A partition mapping $\mathcal{I}$ is then given by:

$$\mathcal{I}(x > 0) = \mathcal{I}(y < 5) = \{0, 5, 7\} \qquad \mathcal{I}(x > 0 \to \Diamond^{\geq 0}_{[1,3]}\, y < 5) = \{0, 2, 4, 5, 6, 7\}$$
$$\mathcal{I}(\Diamond^{\geq 0}_{[1,3]}\, y < 5) = \{2, 4, 6\} \qquad \mathcal{I}(\Box^{\geq 0}_{[0,2]}(x > 0 \to \Diamond^{\geq 0}_{[1,3]}\, y < 5)) = \{0, 2, 3, 4, 5, 6, 7\}$$

We describe the global separation procedure for $\varphi \tilde{\mathbf{U}}^K_I \varphi'$, given multiple time points $\tau_1 < \cdots < \tau_n$, based on Proposition 3.10. Let $\mathcal{T} = (\tau_1, \ldots, \tau_n)$ be an increasing sequence of time points. We denote the empty sequence by $\varnothing$, and $(\tau_1, \tau_2, \ldots, \tau_n)$ by $(\tau_1, \mathcal{T}')$ with the suffix $\mathcal{T}' = (\tau_2, \ldots, \tau_n)$.

Definition 4.12 (Global Separation Procedure). For an STL-GT[$\tilde{\mathbf{U}}$] formula $\varphi \tilde{\mathbf{U}}^K_I \varphi'$ and a sequence $\mathcal{T}$, the global separation $sep(\varphi \tilde{\mathbf{U}}^K_I \varphi', \mathcal{T})$ is the STL-GT[$\tilde{\mathbf{U}}$] formula inductively defined by:

$$sep(\varphi \tilde{\mathbf{U}}^K_I \varphi', \varnothing) \equiv \varphi \tilde{\mathbf{U}}^K_I \varphi'$$
$$sep(\varphi \tilde{\mathbf{U}}^K_I \varphi', (\tau, \mathcal{T})) \equiv \varphi \tilde{\mathbf{U}}^{K \cap [0,\tau)}_I \varphi' \vee (\Box^{K \cap [0,\tau)}_{\geq 0} \varphi \wedge \Box^{K \cap \{\tau\}}_{\geq 0} \varphi \wedge (\Diamond^{K \cap \{\tau\}}_I \varphi' \vee sep(\varphi \tilde{\mathbf{U}}^{K \cap (\tau,\infty)}_I \varphi', \mathcal{T})))$$

We then apply the global separation procedure to separate each subformula $\varphi \tilde{\mathbf{U}}^K_I \varphi'$ by the partition for its *subformulas*, i.e., $(\mathcal{I}(\varphi) \cup \mathcal{I}(\varphi') \cup e(K)) \restriction (K \cup e(K))$, so that the resulting separated formulas of $\varphi \tilde{\mathbf{U}}^K_I \varphi'$ are fully stable for each interval in the partition $\mathcal{I}(\varphi \tilde{\mathbf{U}}^K_I \varphi')$.

Definition 4.13 (Full Separation Procedure). For an STL-GT[$\tilde{\mathbf{U}}$] formula φ and a partition mapping $\mathcal{I}$, the *full separation* $fsep_{\mathcal{I}}(\varphi)$ is an STL-GT[$\tilde{\mathbf{U}}$] formula inductively defined as follows:

$$\begin{aligned}
fsep_{\mathcal{I}}(true) &\equiv true \\
fsep_{\mathcal{I}}(f(\vec{x}) > 0) &\equiv f(\vec{x}) > 0 \\
fsep_{\mathcal{I}}(\neg\varphi) &\equiv \neg fsep_{\mathcal{I}}(\varphi) \\
fsep_{\mathcal{I}}(\varphi \wedge \varphi') &\equiv fsep_{\mathcal{I}}(\varphi) \wedge fsep_{\mathcal{I}}(\varphi') \\
fsep_{\mathcal{I}}(\varphi \tilde{\mathbf{U}}^K_I \varphi') &\equiv sep\Big(fsep_{\mathcal{I}}(\varphi)\, \tilde{\mathbf{U}}^K_I\, fsep_{\mathcal{I}}(\varphi'),\ (\mathcal{I}(\varphi) \cup \mathcal{I}(\varphi') \cup e(K)) \restriction (K \cup e(K))\Big)
\end{aligned}$$

The size of $fsep_{\mathcal{I}}(\varphi)$ can be linear with respect to the size of a partition mapping $\mathcal{I}$, if subterms are shared as usual. The full separation $fsep_{\mathcal{I}}(\varphi)$ is equivalent to the original formula φ by Theorem 3.7. Moreover, $fsep_{\mathcal{I}}(\varphi)$ is fully stable with respect to a signal with finite variable points.

Theorem 4.14. *Consider an STL-GT[$\tilde{\mathbf{U}}$] formula φ, a signal $\vec{x}$ with finite variable points, and a partition mapping $\mathcal{I}$. The full separation $fsep_{\mathcal{I}}(\varphi)$ is fully stable for any interval $J \in [\![\mathcal{I}(\varphi)]\!]_{[0,\infty)}$, provided that the base partition of $\mathcal{I}$ contains all the variable points of $\vec{x}$.*

Example 4.15. For the STL-GT[$\tilde{\mathbf{U}}$] formula $\varphi = \Box^{\geq 0}_{[0,2]}(x > 0 \to \Diamond^{\geq 0}_{[1,3]}\, y < 5)$ and the partition mapping $\mathcal{I}$ in Example 4.11, we separate $\Diamond^{\geq 0}_{[1,3]}\, y < 5$ by the partition $\mathcal{I}(y < 5) = \{0, 5, 7\}$, and separate φ by $\mathcal{I}(x > 0 \to \Diamond^{\geq 0}_{[1,3]}\, y < 5) = \{0, 2, 4, 5, 6, 7\}$. The full separation $fsep_{\mathcal{I}}(\varphi)$ is then:

$$fsep_{\mathcal{I}}(\varphi) = \bigwedge\nolimits_{I \in [\![\{0,2,4,5,6,7\}]\!]_{[0,\infty)}} \Box^I_{[0,2]}(x > 0 \to \bigvee\nolimits_{J \in [\![\{0.5.7\}]\!]_{[0,\infty)}} \Diamond^J_{[1,3]}\, y < 5).$$

Thanks to Theorem 4.14, the formula $fsep_{\mathcal{I}}(\varphi)$ is fully stable for any interval in $[\![\mathcal{I}(\varphi)]\!]_{[0,\infty)}$ with respect to any signal with the variable points $\{0, 5, 7\}$. By Theorem 3.7, $fsep_{\mathcal{I}}(\varphi) \equiv \varphi$.

4.3.1 Proof of Theorem 4.14. We first state some lemmas that are necessary to prove the theorem. If a formula φ is stable for an interval L, then φ is clearly stable for any subinterval J of L by definition. We can easily see that the same property also holds for the full stability.

Lemma 4.16. *If φ is fully stable for an interval L, then φ is fully stable for any subinterval of L.*

The global separation $sep(\varphi \tilde{\mathbf{U}}_I^K \varphi', \mathcal{T})$ is a Boolean combination of its *top-level subformulas* of the forms: $\varphi \tilde{\mathbf{U}}_I^L \varphi'$, $\Box_{\geq 0}^L \varphi$, $\Box_{\geq 0}^L \varphi$, $\Diamond_I^L \varphi'$, and $\varphi \tilde{\mathbf{U}}_I^L \varphi'$. For the temporal operators of these top-level subformulas, the global intervals are in the partition $\mathcal{T}$ of the interval K.

Lemma 4.17. *Given an STL-GT[$\tilde{\mathbf{U}}$] formula φ and a partition P of K, for a top-level subformula of $sep(\varphi \tilde{\mathbf{U}}_I^K \varphi', \mathcal{T})$, every nonempty global interval of its top-level temporal operator is in $[\![\mathcal{T}]\!]_K$.*

To build a partition $\mathcal{I}(\varphi \tilde{\mathbf{U}}_I^K \varphi')$ in Def. 4.2, we add all the points obtained by subtracting I's endpoints $e(I)$ from the partitions for the subformulas to make $\mathcal{I}(\varphi \tilde{\mathbf{U}}_I^K \varphi')$ fine enough to satisfy the full stability condition. The following proves that this is indeed the case.

Lemma 4.18. *Given intervals $K, I, D \subseteq \mathbb{R}^+$ and a partition P of K that includes K's endpoints, for any interval $L \in [\![\bigcup_{\tau \in P} \{\tau - e \in D \mid e \in e(I)\}]\!]_D$, either $L \subseteq J \dot{-} I$ or $L \subseteq (J \dot{-} I)^\complement$ for each $J \in [\![P]\!]_K$.*

Consider an STL-GT[$\tilde{\mathbf{U}}$] formula φ, a signal $\vec{x}$ with a finite set B of variable points, and a partition mapping $\mathcal{I}$, where the base partition of $\mathcal{I}$ includes B. Theorem 4.14 states that $stable_{\vec{x}}(fsep_{\mathcal{I}}(\varphi), J)$ holds for any interval $J \in [\![\mathcal{I}(\varphi)]\!]_{[0,\infty)}$. The proof is by structural induction on φ as follows.

- ($\varphi = true$): $fsep_{\mathcal{I}}(true) = true$, which is always fully stable by definition.
- ($\varphi = f(\vec{x}) > 0$): For any subinterval of $J \in [\![\mathcal{I}(f(\vec{x}) > 0)]\!]_{[0,\infty)}$, $f(\vec{x}) > 0$ is stable, because $\mathcal{I}(f(\vec{x}) > 0) \supseteq B$ and thus J includes no variable point. Therefore, $stable_{\vec{x}}(f(\vec{x}) > 0, J)$. Since $fsep_{\mathcal{I}}(f(\vec{x}) > 0) = f(\vec{x}) > 0$, $stable_{\vec{x}}(fsep_{\mathcal{I}}(f(\vec{x}) > 0), J)$ holds.
- ($\varphi = \neg\phi$): Consider an interval $J \in [\![\mathcal{I}(\neg\phi)]\!]_{[0,\infty)}$. Because $\mathcal{I}(\neg\phi)$ is finer than $\mathcal{I}(\phi)$ by Def. 4.10, J is a subinterval of some interval $J' \subseteq [\![\mathcal{I}(\phi)]\!]_{[0,\infty)}$ by Lemma 4.9. By induction hypothesis, $stable_{\vec{x}}(fsep_{\mathcal{I}}(\phi), J')$, and by Lemma 4.16, $stable_{\vec{x}}(fsep_{\mathcal{I}}(\phi), J)$ holds. Therefore, by definition:

$$stable_{\vec{x}}(fsep_{\mathcal{I}}(\phi), J) \quad \text{iff} \quad stable_{\vec{x}}(\neg fsep_{\mathcal{I}}(\phi), J) \quad \text{iff} \quad stable_{\vec{x}}(fsep_{\mathcal{I}}(\neg\phi), J).$$

- ($\varphi = \phi_1 \wedge \phi_2$): Consider an interval $J \in [\![\mathcal{I}(\phi_1 \wedge \phi_2)]\!]_{[0,\infty)}$. Similarly, since $\mathcal{I}(\phi_1 \wedge \phi_2)$ is finer than both of $\mathcal{I}(\phi_1)$ and $\mathcal{I}(\phi_2)$, by Lemma 4.9, J is a subinterval of some $J_1 \subseteq [\![\mathcal{I}(\phi_1)]\!]_{[0,\infty)}$ and of some $J_2 \subseteq [\![\mathcal{I}(\phi_2)]\!]_{[0,\infty)}$. By induction hypothesis and Lemma 4.16, $stable_{\vec{x}}(fsep_{\mathcal{I}}(\phi_1), J)$ and $stable_{\vec{x}}(fsep_{\mathcal{I}}(\phi_2), J)$. Therefore, by definition, $stable_{\vec{x}}(fsep_{\mathcal{I}}(\phi_1), J) \wedge stable_{\vec{x}}(fsep_{\mathcal{I}}(\phi_2), J)$ iff $stable_{\vec{x}}(fsep_{\mathcal{I}}(\phi_1) \wedge fsep_{\mathcal{I}}(\phi_2), J)$ iff $stable_{\vec{x}}(fsep_{\mathcal{I}}(\phi_1 \wedge \phi_2), J)$.
- ($\varphi = \phi_1 \tilde{\mathbf{U}}_I^K \phi_2$): Consider an interval $J \in [\![\mathcal{I}(\phi_1 \tilde{\mathbf{U}}_I^K \phi_2)]\!]_{[0,\infty)}$. Let

$$P = (\mathcal{I}(\phi_1) \cup \mathcal{I}(\phi_2) \cup e(K)) \restriction (K \cup e(K)), \qquad Q = \textstyle\bigcup_{\tau \in P} \{\tau - e \in [0,\infty) \mid e \in e(I)\}.$$

Since $\mathcal{I}(\phi_1 \tilde{\mathbf{U}}_I^K \phi_2)$ is finer than the partition Q by Def. 4.10, J is a subinterval of some interval $J' \in [\![Q]\!]_{[0,\infty)}$. It suffices to show $stable_{\vec{x}}(fsep_{\mathcal{I}}(\phi_1 \tilde{\mathbf{U}}_I^K \phi_2), J')$, since then $stable_{\vec{x}}(fsep_{\mathcal{I}}(\phi_1 \tilde{\mathbf{U}}_I^K \phi_2), J)$ by Lemma 4.16. By definition, $fsep_{\mathcal{I}}(\phi_1 \tilde{\mathbf{U}}_I^K \phi_2) = sep(fsep_{\mathcal{I}}(\phi_1) \tilde{\mathbf{U}}_I^K fsep_{\mathcal{I}}(\phi_2), P)$, which is a Boolean combination of top-level subformulas of the forms:

$$fsep_{\mathcal{I}}(\phi_1) \tilde{\mathbf{U}}_I^L fsep_{\mathcal{I}}(\phi_2),\ \Box_{\geq 0}^L fsep_{\mathcal{I}}(\phi_1),\ \Box_{\geq 0}^L fsep_{\mathcal{I}}(\phi_1),\ \Diamond_I^L fsep_{\mathcal{I}}(\phi_2),\ fsep_{\mathcal{I}}(\phi_1) \tilde{\mathbf{U}}_I^L fsep_{\mathcal{I}}(\phi_2)$$

By Lemma 4.17, for each global interval L for these top-level subformulas, $L \in [\![P]\!]_K$. Observe that $L \in [\![P]\!]_{[0,\infty)}$ also holds, since $e(K) \subseteq P$. Because $P = (\mathcal{I}(\phi_1) \cup \mathcal{I}(\phi_2) \cup e(K)) \restriction (K \cup e(K))$ is more restrictive and finer than both of $\mathcal{I}(\phi_1)$ and $\mathcal{I}(\phi_2)$, by induction hypothesis, Lemma 4.9, and Lemma 4.16, $stable_{\vec{x}}(fsep_{\mathcal{I}}(\phi_1), L)$ and $stable_{\vec{x}}(fsep_{\mathcal{I}}(\phi_2), L)$ hold. By Lemma 4.18, for $J' \in [\![Q]\!]_{[0,\infty)}$, either $J' \in L \dot{-} I$ or $J' \in (L \dot{-} I)^\complement$ holds. Therefore, each top-level subformula is fully stable for J' by Def. 4.2, and consequently, $stable_{\vec{x}}(fsep_{\mathcal{I}}(\phi_1 \tilde{\mathbf{U}}_I^K \phi_2), J')$. □

$$\exists q_1,\dots,q_k\ \exists \vec{x}_1^0,\dots,\vec{x}_k^0\ \exists \vec{x}_1^t,\dots,\vec{x}_k^t\ \exists \tau_0,\tau_{k+1}.\quad 0=\tau_0<\tau_1<\cdots<\tau_k<\tau_{k+1}=\tau_{\max}\ \wedge$$

$$init(q_1,\vec{x}_1^0)\wedge \vec{x}_1^t=flow(q_1,\vec{x}_1^0,\tau_1-\tau_0)\quad \wedge\forall t\in[0,\tau_1-\tau_0).\ inv(q_1,flow(q_1,\vec{x}_1^0,t))\quad \wedge$$

$$\bigwedge_{i=2}^{k} jump(q_{i-1},\vec{x}_{i-1}^t,q_i,\vec{x}_i^0)\wedge \vec{x}_i^t=flow(q_i,\vec{x}_i^0,\tau_i-\tau_{i-1})\wedge\forall t\in[0,\tau_i-\tau_{i-1}).\ inv(q_i,flow(q_i,\vec{x}_i^0,t))\wedge$$

$$\bigwedge_{j=1}^{n}\bigwedge_{i=1}^{k+1}(\forall t\in(0,\tau_i-\tau_{i-1}).\ p_j(flow(q_i,\vec{x}_i^0,t)))\vee\neg(\forall t\in(0,\tau_i-\tau_{i-1}).\ p_j(flow(q_i,\vec{x}_i^0,t)))$$

Fig. 6. The encoding $\Psi_H^{k,\tau_{\max}}(\tau_1,\dots,\tau_k)$ of a bounded trajectory with variable points in $\{\tau_1,\dots,\tau_k\}$

5 SYMBOLIC MODEL CHECKING OF STL

In this section we present a symbolic model checking procedure for STL properties. We explain how to encode the STL model checking problem of hybrid automata as first-order logic formulas over the real numbers, up to bound k of variable points, based on our translation method. We then describe our STL model checking algorithm for hybrid automata, which is refutationally complete for bounded signals with finite variability, and discuss its complexity.

5.1 Encoding of STL Model Checking

5.1.1 Time Bound of Signals. We first choose a time bound $\tau_{\max}$ to determine the satisfaction of an STL formula φ. For an STL formula φ and a chosen bound $\tau_{\max}\geq 0$, we can easily obtain the bounded restriction $\varphi|_{\tau_{\max}}$ using the global separation. The formula $\varphi|_{\tau_{\max}}$ is an STL-GT[$\tilde{\mathbf{U}}$] formula where every global interval is bounded by $\tau_{\max}$. Simply, $\varphi|_{\tau_{\max}}$ can be built by separating every subformula at the bound $\tau_{\max}$, and by replacing each subformula that refers to the unknown future, i.e., $(\tau_{\max},\infty)$, by a Boolean constant. If the time bound $\tau_{\max}$ is less than the future-reach $fr(\varphi)$, then $\varphi|_{\tau_{\max}}$ is a bounded under-approximation of φ. If $\tau_{\max}\geq fr(\varphi)$, both φ and $\varphi|_{\tau_{\max}}$ are equivalent.

Lemma 5.1. *For an STL formula φ and $\tau_{\max}\geq 0$, there is an STL-GT[$\tilde{\mathbf{U}}$] formula $\varphi|_{\tau_{\max}}$ such that:*

(1) each global interval in $\varphi|_{\tau_{\max}}$ is bounded by $\tau_{\max}$;
(2) $\vec{x},t\models\varphi|_{\tau_{\max}}\implies\vec{x},t\models\varphi$, for $t\geq 0$; and
(3) $\vec{x},t\models\varphi|_{\tau_{\max}}\iff\vec{x},t\models\varphi$, for $0\leq t<\tau_{\max}-fr(\varphi)$.

5.1.2 Encoding of Signals. In order to encode the existence of a bounded trajectory, we apply SMT-based techniques for reachability of hybrid automata. The reachability of a hybrid automaton H, involving a finite number of jumps, can be reduced to the satisfiability of a first-order logic formula over the real numbers, provided that the *init*, *inv*, *jump*, and *flow* conditions are encoded in first-order logic [Cimatti et al. 2012b]. Nonlinear functions, such as polynomials and solutions of Lipschitz-continuous ODEs, can also be used to specify flow conditions in these approaches [Eggers et al. 2015; Fränzle and Herde 2007; Jovanović and de Moura 2012; Kong et al. 2015].

Given a set of propositions $AP=\{p_1(\vec{x}),\dots.p_n(\vec{x})\}$, finding a signal $\vec{x}\in H$ containing up to k variable points is a special case of reachability. Without loss of generality, assume that jumps can happen at any time without state changes. Consider k first-order variables $\tau_1<\cdots<\tau_k$ with linear order constraints. The k variability of $\vec{x}$ is equivalent to say that every proposition is stable for each interval (τ_i,τ_{i+1}), $0\leq i\leq k$, where $\tau_0=0$ and $\tau_{k+1}=\tau_{\max}$. This condition can easily be encoded in the form of invariant conditions. As a consequence, the problem of finding a trajectory with at most k variable points can be reduced to the reachability up to k mode changes.

$$\mathcal{J}(f(\vec{x}) > 0) = \{\tau_1, \ldots, \tau_k\} \qquad \mathcal{J}(\neg\varphi) = \mathcal{J}(\varphi) \qquad \mathcal{J}(\varphi \wedge \varphi') = \mathcal{J}(\varphi) \cup \mathcal{J}(\varphi')$$

$$\mathcal{J}(\varphi \tilde{\mathbf{U}}_I^K \varphi') = \{w_1, \ldots, w_N\}, \qquad \text{for } P = \mathcal{J}(\varphi) \cup \mathcal{J}(\varphi') \cup e(K), \text{ and } N = |e(I)| \cdot |P|$$

$$\Psi^{\mathcal{J}}_{f(\vec{x})>0} = \mathit{true} \qquad \Psi^{\mathcal{J}}_{\neg\varphi} = \Psi^{\mathcal{J}}_{\varphi} \qquad \Psi^{\mathcal{J}}_{\varphi\wedge\varphi'} = \Psi^{\mathcal{J}}_{\varphi} \wedge \Psi^{\mathcal{J}}_{\varphi'}$$

$$\Psi^{\mathcal{J}}_{\varphi \tilde{\mathbf{U}}_I^K \varphi'} = \Psi^{\mathcal{J}}_{\varphi} \wedge \Psi^{\mathcal{J}}_{\varphi'} \wedge \bigwedge_{i=1}^{N-1} w_i \le w_{i+1}$$
$$\wedge \bigwedge_{\tau\in P} \bigwedge_{e\in e(I)} \Big((\tau \in K \cup e(K) \wedge \tau - e \ge 0) \rightarrow \bigvee_{w\in W} w = \tau - e\Big)$$
$$\wedge \bigwedge_{w\in W} \Big(w = 0 \vee \bigvee_{\tau\in P} \bigvee_{e\in e(I)} (\tau \in K \cup e(K) \wedge \tau - e \ge 0 \wedge w = \tau - e)\Big)$$

Fig. 7. A symbolic partition $\mathcal{J}$ and its partition constraint $\Psi^{\mathcal{J}}_{\varphi}(\tau_1, \ldots, \tau_k)$

Using this idea, the existence of a bounded signal $\vec{x} \in H$ with variable points in the set $\{\tau_1, \ldots, \tau_k\}$ can be encoded as a first-order logic formula $\Psi_H^{k,\tau_{\max}}(\tau_1, \ldots, \tau_k)$ in Fig. 6. This formula encodes the definition of the trajectory in Def. 2.2, together with the stability condition. Each i-th step is in mode q_i, and the values of X begin with $\vec{x}_i^0$ (at time τ_{i-1}) and end with $\vec{x}_i^t$ (at time τ_i). At the beginning of the first step, the initial condition $\mathit{init}(q_1, \vec{x}_1^0)$ holds, and at the beginning of each i-th step, for $2 \le i \le k$, the jump condition holds from the final values $\vec{x}_{i-1}^t$ of the previous step to the starting values $\vec{x}_i^0$ of the current step. The invariant and the stability conditions are encoded as formulas with a universal quantifier over time. The size of $\Psi_H^{k,\tau_{\max}}(\tau_1, \ldots, \tau_k)$ is $O(k)$.

5.1.3 Encoding of Partitions. Consider a set of symbolic variable points $\tau_1 < \cdots < \tau_k$, given by first-order variables. We construct a *symbolic partition mapping* $\mathcal{J}$, where each partition is a set of first-order variables. The constraint for the variables to be a partition mapping for the full separation is encoded as a quantifier-free first-order logic formula. As Def. 4.10 for concrete partitions, the base partition for propositions is given by the set $\{\tau_1, \ldots, \tau_n\}$, and a partition for Boolean connectives is given by $\mathcal{J}(\neg\varphi) = \mathcal{J}(\varphi)$ and $\mathcal{J}(\varphi \wedge \varphi') = \mathcal{J}(\varphi) \cup \mathcal{J}(\varphi')$.

A symbolic partition $\mathcal{J}(\varphi \tilde{\mathbf{U}}_I^K \varphi')$ is defined as a set of *fresh* first-order variables $\{w_1, \ldots, w_N\}$, with linear order constraints $w_1 \le w_2 \le \cdots \le w_N$. The number N is the maximum size of the partition that is determined by the size of the subformulas' partitions $\mathcal{J}(\varphi)$ and $\mathcal{J}(\varphi')$. As explained in Sec. 4.3, the partition of $\varphi \tilde{\mathbf{U}}_I^K \varphi'$ contains all the points obtained by subtracting the local interval I's endpoints from the partition for the subformulas $P = (\mathcal{J}(\varphi) \cup \mathcal{J}(\varphi') \cup e(K)) \restriction (K \cup e(K))$. Therefore, the number N is $|\mathcal{J}(\varphi) \cup \mathcal{J}(\varphi') \cup e(K)|$ multiplied by the number of I's endpoints $|e(I)|$. For example, the interval $[0, 5)$ has two endpoints, but $[1, \infty)$ has one endpoint.

It is straightforward to construct the quantifier-free first-order logic formula $\Psi^{\mathcal{J}}_{\varphi}(\tau_1, \ldots, \tau_k)$ that includes every partition constraint of $\mathcal{J}$. The only nontrivial case is $\mathcal{J}(\varphi \tilde{\mathbf{U}}_I^K \varphi') = \{w_1, \ldots, w_N\}$. The encoding includes the linear order constraints $w_1 \le \cdots \le w_N$, which are non-strict because the size of the partition can be smaller than the maximal case. For the constraint between $\mathcal{J}(\varphi \tilde{\mathbf{U}}_I^K \varphi')$ and the partition for the subformulas P, we encode the following set inclusion relations:

$$\bigcup_{\tau\in P} \{\tau - e \in [0, \infty) \mid e \in e(I)\} \quad \subseteq \quad \mathcal{J}(\phi \tilde{\mathbf{U}}_I^K \phi') \quad \subseteq \quad \{0\} \cup \bigcup_{\tau\in P} \{\tau - e \in [0, \infty) \mid e \in e(I)\},$$

Notice that $\mathcal{J}(\phi \tilde{\mathbf{U}}_I^K \phi')$ may not be minimal, because 0 can be a redundant element. This condition is necessary to deal with the case when the minimal partition is empty.

Given an STL-GT[$\tilde{\mathbf{U}}$] formula φ and a symbolic base partition $\{\tau_1, \ldots, \tau_k\}$, Fig. 7 summarizes how to construct a symbolic partition $\mathcal{J}$ and its partition constraint $\Psi_\varphi^{\mathcal{J}}(\tau_1, \ldots, \tau_k)$. If the size of the base partition is k, the size of the top partition $\mathcal{J}(\varphi)$ is $O(k \cdot 2^{h(\varphi)})$. Because the size of the set inclusion constraint for $\varphi \tilde{\mathbf{U}}_I^K \varphi'$ is $O(2(|\mathcal{J}(\varphi)| + |\mathcal{J}(\varphi')| + 2)^2)$, the size of the top-level constraint is $O(k^2 \cdot 4^{h(\varphi)})$. The accumulated size of the entire formula $\Psi_\varphi^{\mathcal{J}}$ is $O(|\varphi| \cdot k^2 \cdot 4^{h(\varphi)})$.

5.1.4 Encoding of STL Counterexamples. For a hybrid automaton H, a counterexample of an STL formula φ can be encoded as follows, given a number k and a time bound $\tau_{\max}$. First, we build a first-order logic formula $\Psi_H^{k,\tau_{\max}}(\tau_1, \ldots, \tau_k)$ with free variables $\{\tau_1, \ldots, \tau_k\}$. Second, we construct a symbolic partition mapping $\mathcal{J}$ for the negated formula $\neg\varphi|_{\tau_{\max}}$, together with its partition constraint $\Psi_{\neg\varphi|_{\tau_{\max}}}^{\mathcal{J}}(\tau_1, \ldots, \tau_k)$. Third, we obtain the translation $\Psi_{\neg\varphi}^{k,\tau_{\max}}(\tau_0) = \mathit{fotr}_{\vec{x}}(\mathit{fsep}_{\mathcal{J}}(\neg\varphi|_{\tau_{\max}}), \{\tau_0\})$ of the full separation $\mathit{fsep}_{\mathcal{J}}(\neg\varphi|_{\tau_{\max}})$, as explained in Sec. 4. Consider the conjunction:

$$\Psi_{H,\neg\varphi}^{k,\tau_{\max}}(\tau_0) \;=\; \Psi_H^{k,\tau_{\max}} \wedge \Psi_{\neg\varphi}^{k,\tau_{\max}}(\tau_0) \wedge \Psi_{\neg\varphi|_{\tau_{\max}}}^{\mathcal{J}}$$

We claim that this formula is unsatisfiable iff every trajectory of H with at most k variable points satisfies the STL formula φ, up to the time bound $\tau_{\max}$, If $\Psi_{H,\neg\varphi}^{k,\tau_{\max}}(\tau_0)$ is satisfiable, by construction, there exists a bounded trajectory $\vec{x} \in H$ with a time bound $\tau_{\max}$, where its variable points are in $\{\tau_1, \ldots, \tau_n\}$. A concrete partition mapping can be built from the base partition $\{\tau_1, \ldots, \tau_n\}$, and the translation of the full separation of $\neg\varphi|_{\tau_{\max}}$ is satisfiable. By Theorem 3.7, Theorem 4.7, and Theorem 4.14, $\neg\varphi|_{\tau_{\max}}$ is satisfied by the signal $\vec{x}$ at time τ_0. Conversely, suppose that there exists such a counterexample $\vec{x} \in H$ such that $\vec{x}, \tau_0 \models \neg\varphi|_{\tau_{\max}}$. By construction, there exists a set of time points $\{\tau_1, \ldots, \tau_n\}$ that contains all variable points of $\vec{x}$ and satisfies $\Psi_H^{k,\tau_{\max}}$. Based on $\{\tau_1, \ldots, \tau_n\}$, we can construct a partition mapping $\mathcal{I}$ that satisfies $\Psi_{\neg\varphi|_{\tau_{\max}}}^{\mathcal{J}}$. By the above theorems again, the translation $\Psi_{\neg\varphi}^{k,\tau_{\max}}(\tau_0)$ is satisfiable, because $\vec{x}, \tau_0 \models \neg\varphi|_{\tau_{\max}}$. Consequently:

Theorem 5.2. *Given a hybrid automaton H, an STL formula φ, a time bound $\tau_{\max}$, a variability bound k, and an initial time τ_0, $\Psi_{H,\neg\varphi}^{k,\tau_{\max}}(\tau_0)$ is unsatisfiable iff $H, \tau_0 \models_{k,\tau_{\max}} \varphi|_{\tau_{\max}}$ (i.e., $\vec{x}, \tau_0 \models \varphi|_{\tau_{\max}}$ for every trajectory $\vec{x} \in H$ with at most k variable points such that $\sup(\mathit{dom}(\vec{x})) \le \tau_{\max}$).*

5.2 SMT-Based Bounded Model Checking

Our bounded model checking algorithm for STL is summarized in Alg. 1. For a hybrid automaton H, an STL formula φ, a time bound $\tau_{\max}$, and a maximum bound N, the algorithm builds a first-order logic formula that encodes a counterexample of φ up to k variable points, where $0 \le k \le N$. The correctness of the algorithm is given by Theorem 5.2. If the encoding is satisfiable, there exists a counterexample of φ with up to k variable points (i.e., $H, \tau_0 \not\models \varphi|_{\tau_{\max}}$), and the algorithm reports the counterexample using the satisfiable assignment (line 9). If the encoding is not satisfiable, we can guarantee that there is no counterexample of φ up to the bounds $\tau_{\max}$ and k (i.e., $H, \tau_0 \models_{k,\tau_{\max}} \varphi|_{\tau_{\max}}$). We can repeat this procedure by incrementing the number k until $k = N$.

Our algorithm uses an SMT solver to check the satisfiability of the encoding (i.e., `checkSat` in line 8). The universal quantification in the encoding is introduced to express invariant and stability conditions. For *linear and polynomial* hybrid automata, because these conditions can be encoded as quantifier-free formulas [Cimatti et al. 2012a], the entire encoding can be quantifier-free. In this case, the satisfiability is decidable, and an SMT solver, such as Z3 [De Moura and Bjørner 2008], provides a decision procedure for the encoding [Jovanović and de Moura 2012]. If the continuous dynamics involves transcendental functions, such as solutions of ODEs, the satisfiability is undecidable; but there exist various tools based on approximation methods, such as dReal [Gao et al. 2013a].

Algorithm 1: Bounded Model Checking Algorithm of STL

Input: Hybrid automaton H, STL formula φ, time bound $\tau_{\max}$, maximum bound N
Output: True, or a counterexample of φ

1 $\overline{\neg\varphi} \longleftarrow \neg\varphi|_{\tau_{\max}}$; // Lemma. 5.1
2 **for** $k = 0$ **to** N **do**
3 $\quad B \longleftarrow \{\tau_1, \tau_2, \ldots, \tau_k\}$;
4 $\quad \Psi_H \longleftarrow$ encoding of H's trajectory with variable points B ; // Fig. 6
5 $\quad \mathcal{J} \longleftarrow$ symbolic partition for $\overline{\neg\varphi}$ with base partition B ; // Fig. 7
6 $\quad \Psi^{\mathcal{J}} \longleftarrow$ partition constraint for $\mathcal{J}$ with respect to $\overline{\neg\varphi}$; // Fig. 7
7 $\quad \Psi_{\overline{\neg\varphi}} \longleftarrow fotr_{\vec{x}}(fsep_{\mathcal{J}}(\overline{\neg\varphi}), \{0\})$; // Def. 4.5 and Def. 4.13
8 $\quad$ **if** checkSat($\Psi_H \wedge \Psi^{\mathcal{J}} \wedge \Psi_{\overline{\neg\varphi}}$) *is* Sat **then**
9 $\quad\quad$ **return** counterexample(result.*satisfiableAssignment*);
10 **return** True;

Our algorithm is refutationally complete for bounded signals with *finite variability*. A signal has finite variability if there is only a finite number of variable points between any two time points. Consider a time bound $\tau_{\max} \geq 0$ and an STL formula φ. Suppose that there exists a counterexample $\vec{x}$ that has finite variability. Then, in the interval $[0, \tau_{\max})$, the signal $\vec{x}$ has a finite number of variable points, say $n \in \mathbb{N}$. By running our algorithm up to the bound n, a formula that encodes a counterexample of φ with at most n variable points is constructed. By Theorem 5.2, the encoding is satisfiable because $\vec{x}$ is such a counterexample, and a counterexample of φ is reported.

The complexity of the algorithm is determined by two factors: the size of the encoding and the complexity of the underlying decision procedure. Consider a bound k for one iteration. As mentioned, the size of the formula Ψ_H is $O(k)$, the size of the symbolic partition $\mathcal{J}$ is $O(k \cdot 2^{h(\varphi)})$, the size of the partition constraint $\Psi^{\mathcal{J}}$ is $O(|\varphi| \cdot k^2 \cdot 4^{h(\varphi)})$. Because the size of the full separation is linear in both the size of the partition and the size of the formula, the size of $\Psi_{\overline{\neg\varphi}}$ is $O(|\varphi| \cdot k \cdot 2^{h(\varphi)})$, provided that common subterms are shared. In order to share a subformula ϕ, we introduce an extra variable χ_ϕ with constraint $\chi_\phi \leftrightarrow \phi$, and replace each occurrence of ϕ by χ_ϕ. Therefore:

Proposition 5.3. *In Alg. 1, the size of the encoding for k (line 8) is* $O(|\varphi| \cdot k^2 \cdot 4^{h(\varphi)})$.

Checking the satisfiability of SMT constraints is NP-hard [Biere et al. 2009], and the worst-case complexity is often very high. Specifically, typical algorithms for solving polynomial constraints are doubly exponential [Jovanović and de Moura 2012]. This implies that for polynomial hybrid automata, the complexity of SMT-based algorithms, including ours, is also doubly exponential in k. (This high complexity is unavoidable, because the reachability problem is already quite difficult.) Despite that, the computation is often feasible in practice, as shown in Sec. 6.

6 EXPERIMENTAL EVALUATION

To experimentally evaluate our methods, we have developed a prototype tool that implements our STL model checking algorithm of Alg. 1. We have defined a simple API to specify hybrid automata and STL formulas in Python, and implemented functionality to perform STL model checking. In our tool, we use the Z3 solver [De Moura and Bjørner 2008] as a subroutine to check the satisfiability of the generated formulas. Because Z3 can deal with nonlinear real arithmetic [Jovanović and de Moura 2012], STL properties of polynomial hybrid automata can be verified using our tool up to given bounds. We apply the quantifier-free encoding [Cimatti et al. 2012a] to eliminate universal quantification from the encoding. The benchmark models and the prototype implementation are available at https://github.com/cee5539/stlMC/tree/popl2019.

Table 1. Hybrid automata models and their STL formulas

Model	STL Formulas	
Cars	$f_1 : \Box_{[0,100]}(dist > 0.1)$	$f_2 : \Box_{[0,20]}(dist < 6 \rightarrow \Diamond_{[0,15]} acc_2)$
	$f_3 : \Diamond_{[12,20]}(dec_2 \rightarrow \Diamond_{[3,18]} acc_2)$	$f_4 : \Diamond_{[4,40]}(dec_2 \mathbf{U}_{[10,20]}(\lvert x_1 - x_2 \rvert \le 0.5 \wedge \lvert y_1 - y_2 \rvert \le 0.5))$
Thermostats	$f_1 : \Box_{[0,40]}(x_1 \le 21)$	$f_2 : \Box_{[0,20]}((x_2 > 20)\mathbf{R}_{[2,12)}(x_1 \le 10))$
	$f_3 : \Box_{[0,10]}((x_2 > 20)\mathbf{U}_{[0,5]} on_1)$	$f_4 : \Diamond_{[0,20]}(off_1 \wedge off_2 \rightarrow \Box_{[0,5]}(on_1 \vee on_2))$
Watertanks	$f_1 : \Box_{[0,50]}(0 < x_1 \le 9 \wedge 0 < x_2 \le 9)$	$f_2 : \Diamond_{[5,15)}(off_1 \rightarrow \Diamond_{[0,7]}(on_1 \wedge x_1 > 5.5))$
	$f_3 : \Box_{[0,10]}(x_1 < 4.9 \rightarrow \Diamond_{[0,10]}(x_1 \ge 5.1))$	$f_4 : \Box_{[0,20]}((on_1 \wedge on_2) \rightarrow \Diamond_{[0,5]}(off_1 \vee off_2))$
Railroad	$f_1 : \Diamond_{(0,50)}(pos \le 0)$	$f_2 : \Box_{[20,40]}(angle \ge 80 \rightarrow \Diamond_{[1,20]}(pos \le 0))$
	$f_3 : \Box_{[3,50]}(\Diamond_{[5,20]}(angle \ge 80))$	$f_4 : \Box_{[10,60]}(angle \ge 80 \rightarrow \Box_{[20,40]}(angle < 60))$
Batteries	$f_1 : \Box_{(0,20.5)}(\Diamond_{[3,14]}(d_1 \ge 1.4))$	$f_2 : \Diamond_{(5,30]}((live_1 \wedge live_2) \rightarrow \Box_{[7.5,25)}(live_1 \wedge live_2))$
	$f_3 : \Box_{(0,50)}(d_1 > 0.5 \wedge d_2 > 0.5)$	$f_4 : \Box_{(10,50]}((g_1 \ge 0 \vee g_2 \ge 0)\mathbf{U}_{(1,15)}(dead_1 \wedge dead_2))$

Model	$\lvert Q \rvert$	$\lvert X \rvert$	Model	$\lvert Q \rvert$	$\lvert X \rvert$	Model	$\lvert Q \rvert$	$\lvert X \rvert$
Cars (Linear)	6	2	Thermostats	4	2 (linear) / 4 (poly)	Railroad	4	2 (linear) / 4 (poly)
Cars (Poly)	3	17	Watertanks	4	2 (linear) / 4 (poly)	Batteries	6	4 (linear) / 6 (poly)

We have conducted experiments on STL model checking of various polynomial hybrid automata. As summarized in Table 1, five different models are considered: (i) autonomous driving of two cars, (ii) two networked thermostat controllers, (iii) two networked water tank controllers, (iv) a controller for a railroad gate, and (v) a load management controller for two batteries. These examples are adapted from existing benchmarks on nonlinear hybrid systems [Alur 2015; Bae and Gao 2017; Fox et al. 2012; Platzer 2008; Raisch et al. 1999]. For each model, we consider two variants: a polynomial hybrid automaton with nonlinear functions, and a simplified model with only linear functions, where (transcendental) flows are approximated using Taylor series and discretization. Therefore, total 10 different models have been used in the experiments.

We consider four STL properties for each model, including nontrivial formulas with nested temporal operators. For example, the formula $\Box_{(10,50]}((g_1 \ge 0 \vee g_2 \ge 0)\mathbf{U}_{(1,15)}(dead_1 \wedge dead_2))$ includes the *Until* operator inside another temporal operator, and various types of intervals with non-zero left endpoints. We have performed bounded model checking of these STL properties up to $k = 50$ for linear models, and $k = 20$ for polynomial models. For time bounds $\tau_{\max}$, different time bounds are assigned to different models, since $\tau_{\max}$ depends on model parameters. We have measured the size of the encoding and the running time, including SMT solving by Z3, for each case of $(model, formula, k)$. All the experiments in this section were conducted on Intel Xeon 2.6 GHz with 512 GB memory, where we set a timeout of 150 minutes.

The experimental results, summarized in Table 2, show that our model checking algorithm can deal with nontrivial STL properties of complex models. For linear models, all the experiments up to $k = 50$, except for the formulas f_1 and f_3 of Cars, were terminated within 5 minutes. The nonlinear models show more unpredictable results. E.g., for Watertanks, the analysis of the formula f_1 took 17.45 seconds for $k = 8$, but took 16.06 seconds for $k = 12$. This unpredictability is due to the underlying algorithms and heuristics for Z3, which try to find satisfiable assignments for the encoding. It is possible that a satisfiable assignment or a contradiction can be found earlier for a bigger k, depending on branching heuristics, learned clauses, restarting policies, etc. It is worth noting that our method can guarantee the correctness for STL up to given bounds, but existing "incomplete" methods cannot provide any assurance even for $k = 1$.

Table 2. STL bounded model checking

(a) Linear models (up to $k \leq 50$)

		k = 10		k = 20		k = 30		k = 40		k = 50	
		Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)
Cars	f_1	81,481	6.83	251,311	21.65	512,341	45.73	864,571	79.62	1,308,001	128.55
	f_2	132,508	12.75	438,958	60.59	922,208	376.07	1,582,258	475.93	2,419,108	1,085.60
	f_3	23,575	1.97	62,565	5.25	117,555	9.93	188,545	16.14	275,535	24.16
	f_4	139,259	13.19	455,619	60.74	952,379	195.63	1,629,539	639.49	2,487,099	1,582.02
Themo	f_1	18,258	1.62	52,038	4.37	101,818	8.50	167,598	13.97	249,378	21.07
	f_2	91,078	7.73	294,118	24.72	611,958	52.77	1,044,598	92.17	1,592,038	145.25
	f_3	90,526	7.62	293,066	25.06	610,406	52.76	1,042,546	92.38	1,589,486	146.99
	f_4	138,048	11.98	464,268	39.93	981,688	87.71	1,690,308	159.39	2,590,128	275.19
Water	f_1	27,159	2.25	80,249	6.72	160,139	13.49	266,829	22.27	400,319	33.28
	f_2	129,895	10.99	437,265	37.88	925,035	84.57	1,593,205	154.50	2,441,775	249.84
	f_3	127,589	10.67	429,089	37.12	907,389	81.35	1,562,489	146.01	2,394,389	234.13
	f_4	135,166	11.47	458,666	40.47	973,366	87.88	1,679,266	159.17	2,576,366	292.58
Railroad	f_1	16,859	1.47	49,289	4.13	97,719	8.22	162,149	13.87	242,579	20.84
	f_2	127,333	10.89	428,623	38.04	906,713	84.54	1,561,603	153.86	2,393,293	257.33
	f_3	74,237	6.29	237,027	20.48	491,017	43.70	836,207	76.97	1,272,597	120.32
	f_4	127,333	10.83	428,623	37.84	906,713	83.67	1,561,603	150.02	2,393,293	243.20
Battery	f_1	83,621	6.85	255,571	21.13	518,721	42.85	873,071	72.85	1,318,621	110.27
	f_2	135,287	11.24	444,437	37.17	930,387	77.79	1,593,137	133.94	2,432,687	200.94
	f_3	30,529	2.60	79,789	6.62	148,649	12.22	237,109	19.43	345,169	28.48
	f_4	100,733	8.40	313,103	26.06	640,273	53.70	1,082,243	90.54	1,639,013	141.68

(b) Nonlinear models (up to $k \leq 20$)

		k = 4		k = 8		k = 12		k = 16		k = 20	
		Size	Time(s)	Size	Time(s)	Size	Time(s)	Size	Time(s)	Size	Time(s)
Cars	f_1	9,091	0.78	19,759	1.69	32,987	2.76	48,775	4.10	67,123	5.64
	f_2	35,183	2.95	95,307	7.92	183,719	15.34	300,419	25.33	445,407	37.53
	f_3	34,238	2.86	93,606	7.77	181,262	15.17	297,206	25.09	441,438	37.33
	f_4	27,777	2.33	70,581	5.86	131,753	11.01	211,293	17.82	309,201	26.22
Themo	f_1	6,349	14.42	14,613	16.18	25,437	17.08	38,821	24.62	54,765	22.41
	f_2	25,037	16.58	65,437	19.81	124,205	26.78	201,341	-	296,845	-
	f_3	24,785	5.78	64,985	-	123,553	-	200,489	-	295,793	-
	f_4	34771	19.39	96,939	21.85	189,699	29.90	313,051	44.34	466,995	59.61
Water	f_1	9,326	14.52	21,898	17.45	38,758	16.06	59,906	18.40	85,342	20.47
	f_2	33,222	17.23	92,210	2,397.67	180,062	-	296,778	-	442,358	-
	f_3	32,710	17.68	90,646	24.45	176,870	-	291,382	-	434,182	-
	f_4	33,999	2.92	95,551	26.44	187,695	8,684.17	310,431	-	463,759	-
Railroad	f_1	5,469	0.48	13,061	2.68	23,213	13.57	35,925	19.02	51,197	20.28
	f_2	31,811	3.42	89,059	23.90	174,595	32.73	288,419	44.42	430,531	59.78
	f_3	20,727	17.06	53,391	20.33	100,647	25.08	162,495	31.53	238,935	39.44
	f_4	31,811	18.61	89,059	24.41	174,595	32.99	288,419	45.21	430,531	60.43
Battery	f_1	24,601	2.00	60,805	4.90	111,601	9.00	176,989	14.35	256,969	20.87
	f_2	35,035	2.93	95,303	7.66	183,859	14.76	300,703	24.23	445,835	36.01
	f_3	10,755	0.96	23,659	2.02	39,699	3.35	58,875	4.80	81,187	6.69
	f_4	28,789	2.35	72,665	5.89	134,909	11.02	215,521	17.39	314,501	25.66

7 CONCLUSIONS AND FUTURE WORK

In this paper we have presented symbolic techniques for formally verifying STL properties of CPS programs. Previous methods for analyzing STL properties are inherently incomplete, due to the difficulties of dealing with an infinite state space over a continuous time domain. To address this problem, we have proposed a new foundational technique, namely, *syntactic separation* of STL, that allows decomposing a formula so that each subformula depends only on one of the disjoint segments of a time domain. For this purpose, we have defined a more expressive temporal logic, called STL-GT, and studied a number of equivalences rules for separating STL-GT formula. Then, we have developed a separation-based procedure to translate STL into a decidable fragment of first-order logic. Our procedure is based on the notion of *full stability* that characterizes a sufficient condition for STL formulas to behave like propositional formulas. Based on the translation method, we have presented the first symbolic model checking algorithm for STL properties of hybrid automata that is refutationally complete for bounded signals under reasonable assumptions. Using state-of-the-art SMT technology for linear and nonlinear real arithmetic, this allows verifying general STL properties of CPS programs, which was previously not possible.

There are several directions for future research for improving the proposed techniques. Because hybrid automata are infinite-state systems, there exists no completeness threshold for STL bounded model checking; that is, our algorithm cannot verify STL properties of unbounded signals. For reachability analysis of hybrid automata, there exist many techniques to verify invariant properties for unbounded time horizon, such as [Gulwani and Tiwari 2008; Platzer and Clarke 2009; Prajna et al. 2007]. Combining our techniques with these approaches will make it possible to verify STL properties for unbounded time horizon. Other immediate next steps are to optimize the encoding to build a smaller size of the formula for better performance, and to apply our algorithm to hybrid automata with nonlinear ODEs by using a specialized solver for this purpose [Eggers et al. 2015; Gao et al. 2013a]. Besides SMT-based approaches, reachable-set computation and simulation-based methods are widely used for analyzing of hybrid automata, as discussed in Sec. 1. Extending our techniques with these approaches is also one of the important research directions.

A syntactic separation of STL opens a number of possibilities for analyzing continuous-time temporal logics. As mentioned in Sec. 1, separation is widely used in formal analysis techniques for discrete programs, including model checking, monitoring, and tableau construction. Similarly, the syntactic separation techniques for STL also have a wide range of applications for analyzing real-time and CPS programs. For example, the first-order translation procedure in Sec. 4 already gives an online monitoring algorithm for STL formulas, if the construction is optimized to remove redundant computation. The bounded restriction for STL formulas in Lemma 5.1 can be used to identify the future fragment of a formula given by a partial signal, which is usually a key problem to address for online monitoring of STL and MTL [Deshmukh et al. 2017; Ho et al. 2014]. Separation was applied for an on-the-fly tableau construction for $\mathrm{MITL}_{\leq}$, which is a fragment of MTL where every interval has the form $[0, d)$ [Geilen 2003]. Similarly, it can be possible to construct a hybrid automaton for (a decidable fragment) of STL using the syntactic separation.

A PROOFS OF LEMMAS

Lemma 3.4. *For nonnegative intervals $I, J, K \subseteq \mathbb{R}^+$, we have the following equivalences:*

(1) $\varphi \mathbf{U}_I^K(\varphi' \vee \psi') \equiv \varphi \mathbf{U}_I^K \varphi' \vee \varphi \mathbf{U}_I^K \psi'$ *(2)* $(\varphi \wedge \psi)\mathbf{U}_I^K \varphi' \equiv \varphi \mathbf{U}_I^K \varphi' \wedge \psi \mathbf{U}_I^K \varphi'$

(3) $\varphi \mathbf{U}_{I \cup J}^K \varphi' \equiv \varphi \mathbf{U}_I^K \varphi' \vee \varphi \mathbf{U}_J^K \varphi'$ *(4)* $\varphi \mathbf{U}_I^{K \cup L} \varphi' \equiv \varphi \mathbf{U}_I^K \varphi' \vee \varphi \mathbf{U}_I^L \varphi'$

Proof. (1) By definition. $\vec{x}, t \models \varphi \mathbf{U}_I^K(\varphi' \vee \psi')$ iff $(\exists t' \geq t)\ t' \in K, t' - t \in I, \vec{x}, t' \models \varphi'$, and $(\forall t'' \in [t, t'])\ \vec{x}, t'' \models \varphi$. Notice that $\vec{x}, t' \models \varphi' \vee \psi'$ iff $\vec{x}, t' \models \varphi'$ or $\vec{x}, t' \models \psi'$.

(2) Suppose that $\vec{x}, t \models (\varphi \wedge \psi)\mathbf{U}_I^K \varphi'$ holds. Then, $(\exists t' \geq t)\ t' \in K, t' - t \in I, \vec{x}, t' \models \varphi'$, and $(\forall t'' \in [t, t'])\ \vec{x}, t'' \models (\varphi \wedge \psi)$. Observe that using the same t', both $(\forall t'' \in [t, t'])\ \vec{x}, t'' \models \varphi$ and $(\forall t'' \in [t, t'])\ \vec{x}, t'' \models \psi$ hold. Therefore, $\vec{x}, t \models \varphi\mathbf{U}_I^K \varphi' \wedge \psi\mathbf{U}_I^K \varphi'$. Now suppose that $\vec{x}, t \models \varphi\mathbf{U}_I^K \varphi'$ and $\vec{x}, t \models \psi\mathbf{U}_I^K \varphi'$ hold. Then, there exist $t_1', t_2' \geq t$ such that:

$$t_1' \in K,\ t_1' - t \in I,\ \vec{x}, t_1' \models \varphi',\ \text{and}\ (\forall t'' \in [t, t_1'])\ \vec{x}, t'' \models \varphi$$
$$t_2' \in K,\ t_2' - t \in I,\ \vec{x}, t_2' \models \varphi',\ \text{and}\ (\forall t'' \in [t, t_2'])\ \vec{x}, t'' \models \psi$$

For $t' = \min(t_1', t_2')$, $t' \in K$, $t' - t \in I$, $\vec{x}, t' \models \varphi'$, and $(\forall t'' \in [t, t'])\ \vec{x}, t'' \models (\varphi \wedge \psi)$.

(3) $\vec{x}, t \models \varphi\mathbf{U}_{I\cup J}^K \varphi'$ iff $(\exists t' \geq t)\ t' \in K, t' - t \in I \cup J, \vec{x}, t' \models \varphi'$, and $(\forall t'' \in [t, t'])\ \vec{x}, t'' \models \varphi$. Since $t' - t \in I \cup J$ iff $t' - t \in I$ or $t' - t \in J$, we have $\vec{x}, t \models \varphi\mathbf{U}_I^K \varphi' \vee \varphi\mathbf{U}_J^K \varphi'$

(4) $\vec{x}, t \models \varphi\mathbf{U}_I^{K\cup L} \varphi'$ iff $(\exists t' \geq t)\ t' \in K \cup L, t' - t \in I, \vec{x}, t' \models \varphi'$, and $(\forall t'' \in [t, t'])\ \vec{x}, t'' \models \varphi$. Because $t' \in K \cup L$ iff $t' \in K$ or $t' \in L$, we have $\vec{x}, t \models \varphi\mathbf{U}_I^K \varphi' \vee \varphi\mathbf{U}_J^K \varphi'$. □

Lemma 3.5. *For nonnegative intervals $I, K, L \subseteq \mathbb{R}^+$, we have:*

(1) $\varphi\mathbf{U}_I^K(\Diamond_{=0}^L \varphi') \equiv \varphi\mathbf{U}_I^{K\cap L}\varphi'$ *(2)* $\Diamond_I^K(\Diamond_{=0}^L \varphi) \equiv \Diamond_I^{K\cap L}\varphi$ *(3)* $\Box_I^K(\Box_{=0}^L \varphi) \equiv \Box_I^{K\cap L}\varphi$

(4) $(\Box_{=0}^K\varphi)\mathbf{U}_I^L \varphi' \equiv \Diamond_I^L \varphi'$, *if* $\sup(L) \leq \inf(K)$ *and* $L \cap K = \emptyset$

Proof. (1) By definition, $\vec{x}, t \models \varphi\mathbf{U}_I^K(\Diamond_{=0}^L \varphi')$ iff $(\exists t' \geq t)\, t' \in K, t' - t \in I, \vec{x}, t' \models \Diamond_{=0}^L \varphi'$, and $(\forall t'' \in [t, t'])\ \vec{x}, t'' \models \varphi$. Notice $\vec{x}, t' \models \Diamond_{=0}^L \varphi'$ iff $t' \in L$ and $\vec{x}, t' \models \varphi'$. Therefore, $\vec{x}, t \models \varphi\mathbf{U}_I^{K\cap L} \varphi'$.
(2) and (3) are immediate by $\Diamond_I^K \varphi \equiv \mathit{true}\,\mathbf{U}_I^K \varphi$ and $\Box_I^K \varphi \equiv \neg\Diamond_I^K \neg\varphi$.
(4) $\vec{x}, t \models (\Box_{=0}^K\varphi)\mathbf{U}_I^L \varphi'$ iff $(\exists t' \geq t)\, t' \in L, t' - t \in I, \vec{x}, t' \models \varphi'$, and $(\forall t'' \in [t, t'])\ \vec{x}, t'' \models \Box_{=0}^K\varphi$. Notice that by definition, $(\forall t'' \in [t, t'])\ \vec{x}, t'' \models \Box_{=0}^K\varphi$ iff $(\forall t'' \in [t, t'])\ t'' \in K \rightarrow \vec{x}, t'' \models \varphi$. By the assumption, $t' < \inf(K)$, and therefore $t'' \notin K$ for any $t'' \in [t, t']$. This immediately means that $(\forall t'' \in [t, t'])\ t'' \in K \rightarrow \vec{x}, t'' \models \varphi$ is equivalent to *true*. Therefore, $\vec{x}, t \models (\Box_{=0}^K\varphi)\mathbf{U}_I^L \varphi'$ iff $\vec{x}, t \models \mathit{true}\,\mathbf{U}_I^L \varphi'$, where $\mathit{true}\,\mathbf{U}_I^L \varphi' \equiv \Diamond_I^L \varphi'$ by definition. □

Lemma 3.6. *For a time τ and nonnegative intervals $I, K \subseteq \mathbb{R}^+$, we have:*

(1) $\varphi\mathbf{U}_I^K\varphi' \equiv \Box_{\geq 0}^{\leq\tau}\varphi \wedge (\Box_{=0}^{>\tau}\varphi)\mathbf{U}_I^K\varphi'$, *if* $0 \leq \tau \leq \inf(K)$.

(2) $\varphi\mathbf{U}_I^K\varphi' \equiv \Box_{<\tau}^{\geq 0}\varphi \wedge \Diamond_{=\tau}^{\geq 0}(\varphi\mathbf{U}_{I-\tau}^K\varphi')$, *if* $0 \leq \tau \leq \inf(I)$.

Proof. (1) We prove a series of semantic equivalences as follows. By definition, $\vec{x}, t \models \varphi\mathbf{U}_I^K\varphi'$

$$\text{iff}\ \ (\exists t' \geq t)\ t' \in K,\ t' - t \in I,\ \vec{x}, t' \models \varphi',\ \text{and}\ (\forall t'' \in [t, t'])\ \vec{x}, t'' \models \varphi$$

Because either $t'' \in [0, \tau]$ or $t'' \in (\tau, \infty)$ always holds, the statement can be rewritten as:

$$\begin{aligned}\text{iff}\ \ &(\exists t' \geq t)\ t' \in K,\ t' - t \in I,\ \vec{x}, t' \models \varphi',\ \text{and}\\ &(\forall t'' \in [t, t'])\ (t'' \in [0, \tau] \rightarrow \vec{x}, t'' \models \varphi)\ \text{and}\ (t'' \in (\tau, \infty) \rightarrow \vec{x}, t'' \models \varphi)\end{aligned}$$

Observe that $\vec{x}, t'' \models \Box_{=0}^{>\tau}\varphi$ iff $t'' \in (\tau, \infty) \rightarrow \vec{x}, t'' \models \varphi$ for any $t'' \in \mathbb{R}$ by definition. Therefore:

$$\begin{aligned}\text{iff}\ \ &(\exists t' \geq t)\ t' \in K,\ t' - t \in I,\ \vec{x}, t' \models \varphi',\ \text{and}\\ &(\forall t'' \in [t, t'])\ (t'' \in [0, \tau] \rightarrow \vec{x}, t'' \models \varphi)\ \text{and}\ (\forall t'' \in [t, t'])\ \vec{x}, t'' \models \Box_{=0}^{>\tau}\varphi)\end{aligned}$$

Since $\tau \leq \inf(K)$, if $t' \in K$, then $\tau \leq t'$. In this case, $t'' \leq \tau$ implies $t'' \leq t'$. Therefore:

$$\begin{aligned}\text{iff}\ \ &(\exists t' \geq t)\ t' \in K,\ t' - t \in I,\ \vec{x}, t' \models \varphi',\ \text{and}\\ &(\forall t'' \in [t, \infty))\ (t'' \in [0, \tau] \rightarrow \vec{x}, t'' \models \varphi)\ \text{and}\ (\forall t'' \in [t, t'])\ \vec{x}, t'' \models \Box_{=0}^{>\tau}\varphi)\end{aligned}$$

Now the condition $(\forall t'' \in [t,\infty))\ (t'' \in [0,\tau] \to \vec{x}, t'' \models \varphi)$ does not depend on t'. Hence, the above statement can be equivalently rewritten as the following statement:

$$\begin{aligned}\text{iff}\ \ &(\forall t'' \in [t,\infty))\ (t'' \in [0,\tau] \to \vec{x}, t'' \models \varphi),\ \text{and}\\ &(\exists t' \geq t)\ t' \in K,\ t'-t \in I,\ \vec{x}, t' \models \varphi' \text{ and } (\forall t'' \in [t,t'])\ \vec{x}, t'' \models \Box_{=0}^{>\tau}\varphi)\end{aligned}$$

which is exactly by definition $\vec{x}, t \models \Box_{\geq 0}^{\leq\tau}\varphi \wedge (\Box_{=0}^{>\tau}\varphi)\,\mathbf{U}_I^K\,\varphi'$.

(2) Similarly, we show a series of semantic equivalences as follows. By definition, $\vec{x}, t \models \varphi \mathbf{U}_I^K \varphi'$

$$\text{iff}\ \ (\exists t' \geq t)\ t' \in K,\ t'-t \in I,\ \vec{x}, t' \models \varphi',\ \text{and}\ (\forall t'' \in [t,t'])\ \vec{x}, t'' \models \varphi$$

Because $0 \leq \tau \leq \inf(I)$, if $t'-t \in I$, then $t \leq t+\tau \leq t+\inf(I) \leq t'$. By subtracting τ from both sides of $t'-t \in I$, and using $[t,t'] = [t,t+\tau) \cup [t+\tau,t']$, we have:

$$\begin{aligned}\text{iff}\ \ &(\exists t' \geq t)\ t' \in K,\ t'-(t+\tau) \in I-\tau,\ \vec{x}, t' \models \varphi',\ (\forall t'' \in [t,t+\tau))\ \vec{x}, t'' \models \varphi,\\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{and } (\forall t'' \in [t+\tau,t'])\ \vec{x}, t'' \models \varphi\end{aligned}$$

Observe that the condition $(\forall t'' \in [t,t+\tau))\ \vec{x}, t'' \models \varphi$ does not depend on t'. Therefore:

$$\begin{aligned}\text{iff}\ \ &(\forall t'' \in [t,t+\tau))\ \vec{x}, t'' \models \varphi,\ \text{and}\\ &(\exists t' \geq t)\ t' \in K \wedge t'-(t+\tau) \in I-\tau,\ \vec{x}, t' \models \varphi',\ (\forall t'' \in [t+\tau,t'])\ \vec{x}, t'' \models \varphi\end{aligned}$$

Because $0 \leq \tau \leq \inf(I)$, $t'-(t+\tau) \in I-\tau$ implies $t' \geq (t+\tau)$. By introducing a new variable $u = \tau + t$ with an existential quantifier, the above statement can be equivalently rewritten as:

$$\begin{aligned}\text{iff}\ \ &(\forall t'' \in [t,t+\tau))\ \vec{x}, t'' \models \varphi,\ \text{and } (\exists u \geq t)\ u \in [0,\infty),\ u-t=\tau,\ \text{and}\\ &(\exists t' \geq u)\ t' \in K,\ t'-u \in I-\tau,\ \vec{x}, t' \models \varphi',\ (\forall t'' \in [u,t'])\ \vec{x}, t'' \models \varphi\end{aligned}$$

which is exactly by definition $\vec{x}, t \models \Box_{<\tau}^{\geq 0}\varphi \wedge \Diamond_{=\tau}^{\geq 0}(\varphi \mathbf{U}_{I-\tau}^K \varphi')$. □

Lemma 4.17. *Given an STL-GT[$\tilde{\mathbf{U}}$] formula φ and a partition P of K, for a top-level subformula of $sep(\varphi \tilde{\mathbf{U}}_I^K \varphi', \mathcal{T})$, every nonempty global interval of its top-level temporal operator is in $[\![\mathcal{T}]\!]_K$.*

Proof. Let $\mathcal{T} = (\tau_1, \ldots, \tau_N)$. We claim that for each separated subformula of $sep(\varphi \mathbf{U}_I^K \varphi', \mathcal{T})$, the global interval of its top-level temporal operator is one of the following:

$$K \cap [0,\tau_1),\ \ K \cap (\tau_N,\infty),\ \ K \cap \{\tau_i\},\ \ K \cap (\tau_i,\tau_{i+1}),\ \ \text{for } 1 \leq i \leq N$$

For a partition $P = \{\tau_1, \ldots, \tau_N\}$ of K, these intervals are in $[\![P]\!]_K$ by definition. When $N = 1$, it is immediate by definition. Suppose that the claim holds for any increasing sequence of length $N-1$, say, $\mathcal{T}' = (\tau_2, \ldots, \tau_N)$. For $\mathcal{T} = (\tau_1, \mathcal{T}')$, consider the formula $sep(\varphi \mathbf{U}_I^K \varphi', \mathcal{T})$. All the top-level global intervals are by definition, $K \cap [0,\tau_1)$, $K \cap \{\tau_1\}$, and by induction hypothesis:

$$K \cap (\tau_1,\infty) \cap [0,\tau_2),\ K \cap (\tau_1,\infty) \cap (\tau_N,\infty),\ K \cap (\tau_1,\infty) \cap (\tau_i,\tau_{i+1}),\ K \cap (\tau_1,\infty) \cap \{\tau_i\},$$

for $2 \leq i \leq N$. Thus, the top-level global intervals for $sep(\varphi \mathbf{U}_I^K \varphi', \mathcal{T})$ are $K \cap [0,\tau_1)$, $K \cap (\tau_N,\infty)$, $K \cap (\tau_i,\tau_{i+1})$, $K \cap \{\tau_i\}$, for $1 \leq i \leq N$. □

Lemma 4.18. *Given intervals $K, I, D \subseteq \mathbb{R}^+$ and a partition P of K that includes K's endpoints, for any interval $L \in [\![\bigcup_{\tau\in P}\{\tau - e \in D \mid e \in e(I)\}]\!]_D$, either $L \subseteq J \dot{-} I$ or $L \subseteq (J \dot{-} I)^{\complement}$ for each $J \in [\![P]\!]_K$.*

Proof. Let $Q = \bigcup_{\tau\in P}\{\tau - e \in D \mid e \in e(I)\} = \{\mu_1, \mu_2, \ldots, \mu_m\}$, where $\mu_1 < \cdots < \mu_m$. Consider an interval $L \in [\![Q]\!]_D$. Suppose that the lemma does not hold for $J \in [\![P]\!]_K$. For $L \subseteq D$ to intersect both $J \dot{-} I$ and $(J \dot{-} I)^{\complement}$, L must include an endpoint of $J \dot{-} I$ in D, and Q contains all such endpoints. By definition, L has one of the forms $\{\mu_i\}$, (μ_i,μ_{i+1}), $D \cap [0,\mu_1)$, and $D \cap (\mu_m,\infty)$, but only $L = \{\mu_i\}$ can include an endpoint in Q. But $\{\mu_i\} \subseteq J \dot{-} I$ and $\{\mu_i\} \subseteq (J \dot{-} I)^{\complement}$ cannot hold at the same time. □

Lemma 4.9. *For two partitions P and Q of an interval D:*

(1) if $Q \supseteq P$, for any interval $L \in [\![Q]\!]_D$, there exists $L' \in [\![P]\!]_D$ such that $L \subseteq L'$; and
(2) if $E \subseteq D$, for any interval $L \in [\![P \restriction E]\!]_E$, there exists $L' \in [\![P]\!]_D$ such that $L \subseteq L'$.

Proof. (1) Let $Q = \{\tau_1, \ldots, \tau_n\}$, where $\tau_1 < \cdots < \tau_n$. Suppose that the lemma does not hold for some $L \in [\![Q]\!]_D$. The interval L intersects with at least two different intervals in $[\![P]\!]_D$, because otherwise, L is a subset of some interval in $[\![P]\!]_D$. Let $\mu, \mu' \in L$ be these intersect points. There exists an endpoint $\tau \in P$ between μ and μ'. Since L is an interval, $\tau \in L$, and since $Q \supseteq P$, τ is an endpoint in Q. By definition, L has one of the forms: $\{\tau_i\}$, (τ_i, τ_{i+1}), $D \cap [0, \tau_1)$, $D \cap (\tau_m, \infty)$. Thus, to include an endpoint, L has to be $\{\tau\}$. Because τ is an endpoint in P, $\{\tau\} \in [\![P]\!]_D$, which is a contradiction.

(2) Let $P = \{\tau_1, \ldots, \tau_n\}$, where $\tau_1 < \tau_2 < \cdots < \tau_n$. For $1 \le j \le n - m$ and $0 \le m < n$, let $P \restriction E = \{\tau_j, \tau_{j+1}, \ldots, \tau_{j+m}\}$. Consider $L \in [\![P \restriction E]\!]_E$. Clearly, if L has one of the forms $\{\tau_i\}$ and (τ_i, τ_{i+1}), then $L \in [\![P]\!]_D$. There are now two cases: $L = E \cap [0, \tau_j)$ or $L = E \cap (\tau_{j+m}, \infty)$.

- For $L = E \cap [0, \tau_j)$: when $j = 1$, $E \cap [0, \tau_1) \subseteq D \cap [0, \tau_1)$, since $E \subseteq D$. When $j > 1$, since τ_{j-1} is smaller than any element of E (otherwise, $\tau_{j-1} \in E$), $E \cap [0, \tau_j) \subseteq (\tau_{j-1}, \tau_j)$.
- For $L = E \cap (\tau_{j+m}, \infty)$: when $j + m = n$, $E \cap (\tau_n, \infty) \subseteq D \cap (\tau_n, \infty)$. When $j + m < n$, since τ_{j+m+1} is greater than any element of E (otherwise, $\tau_{j+m+1} \in E$), $E \cap (\tau_{j+m}, \infty) \subseteq (\tau_{j+m}, \tau_{j+m+1})$. □

Lemma 5.1. *For an STL formula φ and $\tau_{\max} \ge 0$, there is an STL-GT[$\tilde{\mathbf{U}}$] formula $\varphi|_{\tau_{\max}}$ such that:*

(1) each global interval in $\varphi|_{\tau_{\max}}$ is bounded by $\tau_{\max}$;
(2) $\vec{x}, t \models \varphi|_{\tau_{\max}} \implies \vec{x}, t \models \varphi$, for $t \ge 0$; and
(3) $\vec{x}, t \models \varphi|_{\tau_{\max}} \iff \vec{x}, t \models \varphi$, for $0 \le t < \tau_{\max} - fr(\varphi)$.

Proof. With out loss of generality, we assume that an STL formula φ is in *negation normal form* with the temporal operators $\mathbf{U}_I$ and $\mathbf{R}_I$. By Proposition 3.10, we immediately have:

$$\vec{x}, t \models \phi \tilde{\mathbf{U}}_I^{<\tau_{\max}} \phi' \implies \vec{x}, t \models \phi \tilde{\mathbf{U}}_I^{\ge 0} \phi', \quad \vec{x}, t \models \phi \hat{\mathbf{R}}_I^{<\tau_{\max}} \phi' \wedge \Diamond_{\ge 0}^{<\tau_{\max}} \phi \implies \vec{x}, t \models \phi \mathbf{R}_I^{\ge 0} \phi'.$$

Therefore, we obtain an STL-GT[$\tilde{\mathbf{U}}$] formula $\varphi|_{\tau_{\max}}$ from the formula φ by replacing each subformula $\phi \mathbf{U}_I \phi'$ by $\phi \tilde{\mathbf{U}}_I^{<\tau_{\max}} \phi'$, and each subformula $\phi \mathbf{R}_I \phi'$ by $\phi \hat{\mathbf{R}}_I^{<\tau_{\max}} \phi' \wedge \Diamond_{\ge 0}^{<\tau_{\max}} \phi$, where every global interval is bounded by $\tau_{\max}$. Because φ is in negation normal form, $\vec{x}, t \models \varphi|_{\tau_{\max}}$ implies $\vec{x}, t \models \varphi$. It remains to prove $\vec{x}, t \models \varphi \iff \vec{x}, t \models \varphi|_\tau$, for $0 \le t < \tau - fr(\varphi)$. The proof is by structural induction. The only nontrivial case is $\varphi \mathbf{U}_I \varphi'$. First, because $t < \tau - fr(\varphi \mathbf{U}_I \varphi')$, $t' - t \in I$ implies

$$t' \le t + \sup(I) < \tau + \sup(I) - fr(\varphi \mathbf{U}_I \varphi') = \tau - \max(fr(\varphi), fr(\varphi')).$$

That is, $t' < \tau - fr(\varphi)$ and $t' < \tau - fr(\varphi')$. By induction hypothesis, $\vec{x}, t' \models \varphi$ iff $\vec{x}, t' \models \varphi|_\tau$, and $\vec{x}, t' \models \varphi'$ iff $\vec{x}, t' \models \varphi'|_\tau$. Hence, $\vec{x}, t \models \varphi \mathbf{U}_I \varphi'$ iff $\vec{x}, t \models (\varphi|_\tau) \mathbf{U}_I (\varphi'|_\tau)$ iff, by Proposition 3.10,

$$\vec{x}, t \models (\varphi|_\tau) \tilde{\mathbf{U}}_I^{<\tau} (\varphi'|_\tau) \vee (\Box_{\ge 0}^{<\tau} (\varphi|_\tau) \wedge \Box_{\ge 0}^{=\tau} (\varphi|_\tau) \wedge (\Diamond_I^{=\tau} (\varphi'|_\tau) \vee (\varphi|_\tau) \tilde{\mathbf{U}}_I^{>\tau} (\varphi'|_\tau)))$$

Since $t < \tau - \sup(I)$, the time constraints for $\vec{x}, t \models \Diamond_I^{=\tau} (\varphi'|_\tau)$ and $\vec{x}, t \models (\varphi|_\tau) \tilde{\mathbf{U}}_I^{>\tau} (\varphi'|_\tau)$ cannot be satisfied. Therefore, $\vec{x}, t \models (\varphi|_\tau) \mathbf{U}_I (\varphi'|_\tau)$ iff $\vec{x}, t \models (\varphi|_\tau) \tilde{\mathbf{U}}_I^{<\tau} (\varphi'|_\tau)$ iff $\vec{x}, t \models (\varphi \tilde{\mathbf{U}}_I \varphi')|_\tau$. □

ACKNOWLEDGMENTS

This work was partly supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (No. 2017M3C4A7068175), Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2016R1D1A1B03935275), POSTECH Basic Science Research Institute, and POSTECH Information Research Laboratories.

REFERENCES

Houssam Abbas, Georgios Fainekos, Sriram Sankaranarayanan, Franjo Ivančić, and Aarti Gupta. 2013. Probabilistic temporal logic falsification of cyber-physical systems. *ACM Transactions on Embedded Computing Systems* 12, 2s (2013), 95.

Matthias Althoff. 2013. 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 (HSCC '13)*. ACM, New York, NY, USA, 173–182. https://doi.org/10.1145/2461328.2461358

Rajeev Alur. 2015. *Principles of cyber-physical systems*. The MIT Press.

Rajeev Alur, Tomás Feder, and Thomas A Henzinger. 1996. The benefits of relaxing punctuality. *Journal of the ACM (JACM)* 43, 1 (1996), 116–146.

Rajeev Alur and Thomas A Henzinger. 1994. A really temporal logic. *Journal of the ACM (JACM)* 41, 1 (1994), 181–203.

Yashwanth Annpureddy, Che Liu, Georgios Fainekos, and Sriram Sankaranarayanan. 2011. *S-taliro: a tool for temporal logic falsification for hybrid systems*. Lecture Notes in Computer Science, Vol. 6605. Springer. 254–257 pages.

Kyungmin Bae and Sicun Gao. 2017. Modular SMT-based analysis of nonlinear hybrid systems. In *Proceedings of the 17th Conference on Formal Methods in Computer-Aided Design (FMCAD '17)*. FMCAD, Austin, TX, 180–187. http://dl.acm.org/citation.cfm?id=3168451.3168490

Stanley Bak and Parasara Sridhar Duggirala. 2017. Simulation-equivalent reachability of large linear systems with inputs. In *Computer Aided Verification (Lecture Notes in Computer Science)*, Vol. 10426. Springer, 401–420. https://doi.org/10.1007/978-3-319-63387-9_20

Marcello M Bersani, Matteo Rossi, and Pierluigi San Pietro. 2015. An SMT-based approach to satisfiability checking of MITL. *Information and Computation* 245 (2015), 72–97.

Marcello M Bersani, Matteo Rossi, and Pierluigi San Pietro. 2016. A tool for deciding the satisfiability of continuous-time metric temporal logic. *Acta Informatica* 53, 2 (2016), 171–206.

Armin Biere, Alessandro Cimatti, Edmund M Clarke, Ofer Strichman, and Yunshan Zhu. 2003. Bounded model checking. *Advances in computers* 58 (2003), 117–148.

Armin Biere, Marijn Heule, and Hans van Maaren. 2009. *Handbook of satisfiability*. Vol. 185. IOS press.

Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. 2005. On the expressiveness of TPTL and MTL. In *Proceedings of the 25th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS '05)*. Springer-Verlag, Berlin, Heidelberg, 432–443. https://doi.org/10.1007/11590156_35

Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. 2013. Flow*: an analyzer for non-linear hybrid systems. In *Computer Aided Verification*, Natasha Sharygina and Helmut Veith (Eds.). Springer, Berlin, Heidelberg, 258–263.

Alessandro Cimatti, Alberto Griggio, Sergio Mover, and Stefano Tonetta. 2015. 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. Springer, 52–67.

Alessandro Cimatti, Sergio Mover, and Stefano Tonetta. 2012a. A quantifier-free SMT encoding of non-linear hybrid automata. In *Formal Methods in Computer-Aided Design (FMCAD)*. IEEE, 187–195.

Alessandro Cimatti, Sergio Mover, and Stefano Tonetta. 2012b. SMT-based verification of hybrid systems. In *AAAI Conference on Artificial Intelligence*. AAAI. https://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5072

Edmund M Clarke, Orna Grumberg, and Doron Peled. 1999. *Model checking*. MIT press.

Lucas Cordeiro, Bernd Fischer, and Joao Marques-Silva. 2012. SMT-based bounded model checking for embedded ANSI-C software. *IEEE Transactions on Software Engineering* 38, 4 (2012), 957–974.

Thao Dang and Tarik Nahhal. 2009. Coverage-guided test generation for continuous and hybrid systems. *Formal Methods in System Design* 34, 2 (2009), 183–213.

Thao Dang and Romain Testylier. 2012. Reachability analysis for polynomial dynamical systems using the Bernstein expansion. *Reliable Computing* 17, 2 (2012), 128–152.

Leonardo De Moura and Nikolaj Bjørner. 2008. 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 (TACAS'08/ETAPS'08)*. Springer-Verlag, Berlin, Heidelberg, 337–340. http://dl.acm.org/citation.cfm?id=1792734.1792766

Jyotirmoy V Deshmukh, Alexandre Donzé, Shromona Ghosh, Xiaoqing Jin, Garvit Juniwal, and Sanjit A Seshia. 2017. Robust online monitoring of signal temporal logic. *Formal Methods in System Design* 51, 1 (2017), 5–30.

Adel Dokhanchi, Bardh Hoxha, and Georgios Fainekos. 2015. Metric interval temporal logic specification elicitation and debugging. In *Proceedings of the 2015 ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE '15)*. IEEE Computer Society, Washington, DC, USA, 70–79. https://doi.org/10.1109/MEMCOD.2015.7340472

Alexandre Donzé. 2010. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Proceedings of the 22nd International Conference on Computer Aided Verification (CAV'10)*. Springer-Verlag, Berlin, Heidelberg, 167–170. https://doi.org/10.1007/978-3-642-14295-6_17

Alexandre Donzé, Thomas Ferrère, and Oded Maler. 2013. Efficient robust monitoring for STL. In *Computer Aided Verification*, Natasha Sharygina and Helmut Veith (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 264–279.

Parasara Sridhar Duggirala, Sayan Mitra, and Mahesh Viswanathan. 2013. Verification of annotated models from executions. In *Proceedings of the Eleventh ACM International Conference on Embedded Software (EMSOFT '13)*. IEEE Press, Piscataway, NJ, USA, Article 26, 10 pages. http://dl.acm.org/citation.cfm?id=2555754.2555780

Andreas Eggers, Nacim Ramdani, Nedialko S Nedialkov, and Martin Fränzle. 2015. Improving the SAT modulo ODE approach to hybrid systems analysis by combining different enclosure methods. *Software & Systems Modeling* 14, 1 (2015), 121–148.

Chuchu Fan, Bolun Qi, Sayan Mitra, Mahesh Viswanathan, and Parasara Sridhar Duggirala. 2016. Automatic reachability analysis for nonlinear hybrid models with C2E2. In *Computer Aided Verification*, Swarat Chaudhuri and Azadeh Farzan (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 531–538.

Maria Fox, Derek Long, and Daniele Magazzeni. 2012. Plan-based policies for efficient multiple battery load management. *Journal of Artificial Intelligence Research* 44 (2012), 335–382.

Martin Fränzle and Christian Herde. 2007. HySAT: An efficient proof engine for bounded model checking of hybrid systems. *Formal Methods in System Design* 30, 3 (2007), 179–198. https://doi.org/10.1007/s10703-006-0031-0

Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. 2011. SpaceEx: scalable verification of hybrid systems. In *Computer Aided Verification*, Ganesh Gopalakrishnan and Shaz Qadeer (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 379–395.

Dov M. Gabbay. 1981. *Expressive functional completeness in tense logic.* Springer Netherlands, Dordrecht, 91–117. https://doi.org/10.1007/978-94-009-8384-7_4

Sicun Gao, Jeremy Avigad, and Edmund M. Clarke. 2012. δ-completee decision procedures for satisfiability over the reals. In *Proceedings of the 6th International Joint Conference on Automated Reasoning (IJCAR'12)*. Springer-Verlag, Berlin, Heidelberg, 286–300. https://doi.org/10.1007/978-3-642-31365-3_23

Sicun Gao, Soonho Kong, and Edmund M. Clarke. 2013a. dReal: an SMT solver for nonlinear theories over the reals. In *Proceedings of the 24th International Conference on Automated Deduction (CADE'13)*. Springer-Verlag, Berlin, Heidelberg, 208–214. https://doi.org/10.1007/978-3-642-38574-2_14

Sicun Gao, Soonho Kong, and Edmund M. Clarke. 2013b. Satisfiability modulo ODEs. In *Formal Methods in Computer-Aided Design*. IEEE, 105–112.

Marc Geilen. 2003. An improved on-the-fly tableau construction for a real-time temporal logic. In *Computer Aided Verification*. Springer Berlin Heidelberg, 394–406.

Rob Gerth, Doron Peled, Moshe Y Vardi, and Pierre Wolper. 1995. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification, Testing and Verification XV*. Springer, 3–18.

Antoine Girard and George J. Pappas. 2006. Verification using simulation. In *Proceedings of the 9th International Conference on Hybrid Systems: Computation and Control (HSCC'06)*. Springer-Verlag, Berlin, Heidelberg, 272–286. https://doi.org/10.1007/11730637_22

Robert P. Goldman, Daniel Bryce, Michael J. Pelican, David J. Musliner, and Kyungmin Bae. 2016. A hybrid architecture for correct-by-construction hybrid planning and control. In *Proceedings of the 8th International Symposium on NASA Formal Methods - Volume 9690 (NFM 2016)*. Springer-Verlag New York, Inc., New York, NY, USA, 388–394. https://doi.org/10.1007/978-3-319-40648-0_29

Sumit Gulwani and Ashish Tiwari. 2008. Constraint-based approach for analysis of hybrid systems. In *Computer Aided Verification*, Aarti Gupta and Sharad Malik (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 190–203.

Klaus Havelund and Grigore Roşu. 2004. Efficient monitoring of safety properties. *International Journal on Software Tools for Technology Transfer (STTT)* 6, 2 (2004), 158–173.

ThomasA. Henzinger. 2000. The theory of hybrid automata. In *Verification of Digital and Hybrid Systems (NATO ASI Series)*, Vol. 170. Springer, 265–292. https://doi.org/10.1007/978-3-642-59615-5_13

Thomas A Henzinger, Peter W Kopke, Anuj Puri, and Pravin Varaiya. 1998. What's decidable about hybrid automata? *Journal of computer and system sciences* 57, 1 (1998), 94–124.

Hsi-Ming Ho, Joël Ouaknine, and James Worrell. 2014. Online monitoring of metric temporal logic. In *Runtime Verification*, Borzoo Bonakdarpour and Scott A. Smolka (Eds.). Springer, Berlin, Heidelberg, 178–192.

Ian M. Hodkinson and Mark Reynolds. 2005. Separation - past, present, and future. In *We Will Show Them! Essays in Honour of Dov Gabbay, Volume Two*. College Publications, 117–142.

Paul Hunter, Joel Ouaknine, and James Worrell. 2013. Expressive completeness for metric temporal logic. In *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '13)*. IEEE Computer Society, Washington, DC, USA, 349–357. https://doi.org/10.1109/LICS.2013.41

Daisuke Ishii, Kazunori Ueda, and Hiroshi Hosobe. 2011. An interval-based SAT modulo ODE solver for model checking nonlinear hybrid systems. *Int. J. Softw. Tools Technol. Transf.* 13, 5 (Oct. 2011), 449–461. https://doi.org/10.1007/s10009-011-0193-y

Stefan Jakšić, Ezio Bartocci, Radu Grosu, and Dejan Ničković. 2016. Quantitative monitoring of STL with edit distance. In *Runtime Verification*, Yliès Falcone and César Sánchez (Eds.). Springer-Verlag, Berlin, Heidelberg, 201–218.

Xiaoqing Jin, Jyotirmoy V. Deshmukh, James Kapinski, Koichi Ueda, and Ken Butts. 2014. Powertrain control verification benchmark. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control (HSCC '14)*. ACM, New York, NY, USA, 253–262. https://doi.org/10.1145/2562059.2562140

Dejan Jovanović and Leonardo de Moura. 2012. Solving non-linear arithmetic. In *Proceedings of the 6th International Joint Conference on Automated Reasoning (IJCAR'12)*. Springer-Verlag, Berlin, Heidelberg, 339–354. https://doi.org/10.1007/978-3-642-31365-3_27

Johan Anthony Wilem Kamp. 1968. *Tense logic and the theory of linear order*. Ph.D. Dissertation. University of California, Los Angeles.

Soonho Kong, Sicun Gao, Wei Chen, and Edmund Clarke. 2015. dReach: δ-reachability analysis for hybrid systems. In *Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems - Volume 9035*. Springer-Verlag, Berlin, Heidelberg, 200–205. https://doi.org/10.1007/978-3-662-46681-0_15

Ron Koymans. 1990. Specifying real-time properties with metric temporal logic. *Real-time systems* 2, 4 (1990), 255–299.

Steven M LaValle. 2006. *Planning algorithms*. Cambridge university press.

Oded Maler and Dejan Nickovic. 2004. 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. Springer, 152–166.

Dejan Ničković, Olivier Lebeltel, Oded Maler, Thomas Ferrère, and Dogan Ulus. 2018. AMT 2.0: qualitative and quantitative trace analysis with extended signal temporal logic. In *Tools and Algorithms for the Construction and Analysis of Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 303–319.

Dejan Nickovic and Oded Maler. 2007. AMT: a property-based monitoring tool for analog systems. In *Formal Modeling and Analysis of Timed Systems*, Jean-François Raskin and P. S. Thiagarajan (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 304–319.

Joël Ouaknine and James Worrell. 2008. Some recent results in metric temporal logic. In *Proceedings of the 6th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS '08)*. Springer-Verlag, Berlin, Heidelberg, 1–13. https://doi.org/10.1007/978-3-540-85778-5_1

André Platzer. 2008. Differential dynamic logic for hybrid systems. *Journal of Automated Reasoning* 41, 2 (2008), 143–189.

André Platzer and Edmund M Clarke. 2009. Computing differential invariants of hybrid systems as fixedpoints. *Formal Methods in System Design* 35, 1 (2009), 98–120.

Stephen Prajna, Ali Jadbabaie, and George J Pappas. 2007. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Trans. Automat. Control* 52, 8 (2007), 1415–1428.

Jörg Raisch, Eberhard Klein, Christian Meder, Alexander Itigin, and Siu O'Young. 1999. Approximating automata and discrete control for continuous systems — two examples from process control. In *Hybrid systems V*. Springer, 279–303.

Vasumathi Raman, Alexandre Donzé, Dorsa Sadigh, Richard M. Murray, and Sanjit A. Seshia. 2015. Reactive synthesis from signal temporal logic specifications. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control (HSCC '15)*. ACM, New York, NY, USA, 239–248. https://doi.org/10.1145/2728606.2728628

Hendrik Roehm, Jens Oehlerking, Thomas Heinz, and Matthias Althoff. 2016. STL model checking of continuous and hybrid systems. In *International Symposium on Automated Technology for Verification and Analysis*. Springer, 412–427.

Nima Roohi, Ramneet Kaur, James Weimer, Oleg Sokolsky, and Insup Lee. 2018. Parameter invariant monitoring for signal temporal logic. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (Part of CPS Week) (HSCC '18)*. ACM, New York, NY, USA, 187–196. https://doi.org/10.1145/3178126.3178140

Ashish Tiwari. 2015. Time-aware abstractions in HybridSal. In *Computer Aided Verification (Lecture Notes in Computer Science)*, Vol. 9206. Springer, 504–510.