STL Model Checking of Continuous and Hybrid Systems

Hendrik Roehm[1], Jens Oehlerking[1], Thomas Heinz[1], and Matthias Althoff[2]

[1] Robert Bosch GmbH, Corporate Research, Renningen, Germany
{hendrik.roehm, jens.oehlerking, thomas.heinz2}@de.bosch.com
[2] Department of Informatics, Technische Universität München, Germany
althoff@in.tum.de

Abstract. Signal Temporal Logic (STL) is a formalism for reasoning about temporal properties of continuous-time traces of hybrid systems. Previous work on this subject mostly focuses on robust satisfaction of an STL formula for a particular trace. In contrast, we present a method solving the problem of formally verifying an STL formula for continuous and hybrid system models, which exhibit uncountably many traces. We consider an abstraction of a model as an evolution of reachable sets. Through leveraging the representation of the abstraction, the continuous-time verification problem is reduced to a discrete-time problem. For the given abstraction, the reduction to discrete-time and our decision procedure are sound and complete for finitely represented reach sequences and sampled time STL formulas. Our method does not rely on a special representation of reachable sets and thus any reachability analysis tool can be used to generate the reachable sets. The benefit of the method is illustrated on an example from the context of automated driving.

Keywords: Model Checking, Reachability Analysis, Hybrid Systems, Temporal Logic, Continuous Time.

1 Introduction

In recent years, the functionality and complexity of products, production processes, and software has been increasing. Furthermore, the interaction between the physical parts of a system (mechanics, thermodynamics, sensors, actuators, and others) and its computational elements is becoming tighter and is organized over large networks, which has resulted in so-called *cyber-physical systems* [21, 14]. Due to their advanced capabilities, newly developed cyber-physical systems often fulfill safety-critical tasks that were previously only entrusted to humans; see, e. g., automated road vehicles, surgical robots, automatic operation of smart grids, and collaborative human-robot manufacturing [22, 18]. The aforementioned trends drastically increase the demand for formal verification methods of hybrid (mixed discrete/continuous) systems.

Hybrid systems contain the interplay of discrete and continuous dynamics and therefore are inherently difficult to verify formally [18, 23]. As a result, most

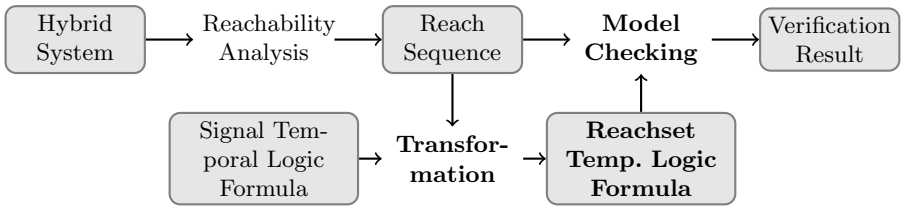


Fig. 1. Structure of the proposed model checking method. Bold parts are novel.

hybrid system researchers have focused on solving reach-problems and reach-avoid-problems: for all possible initial states and all possible disturbances, the system has to avoid forbidden regions while reaching a goal set [6, 13]. There are several tools for reach-avoid problems, which compute sets of reachable states over time and check for intersection of these sets with forbidden regions [2, 12]. More complicated formal specifications based on temporal logics, such as computation tree logic (CTL), linear temporal logic (LTL), or μ-calculus, have mostly been applied to the verification of purely discrete systems or timed automata [9, 7, 4]. For hybrid systems, a continuous-time and real-valued version of such temporal logics, called Signal Temporal Logic (STL), has been proposed as a formal specification language [16]. However, STL has mainly not been used for verification of hybrid systems, but for checking single traces only, e. g., for runtime monitoring and for test generation [15, 26, 27, 17, 1]. Therefore, there is a demand for formal verification techniques which are able to verify a temporal (STL) property for all (infinitely many) possible traces of a hybrid system.

In this work, we propose a new idea to verify specifications in STL for a hybrid system. Given a hybrid system S and an STL property φ, we propose the following steps to formally verify φ on S, as shown in Fig. 1:

1. A new *reachset temporal logic (RTL)* is defined (Sec. 3). The semantics of RTL is directly defined on the reach sequence, which corresponds to an infinite set of traces. A *reach sequence* is a function mapping time to the set of states reachable from a set of initial states and uncertain inputs. Therefore, with RTL, we are able to reason about infinitely many traces with a finite representation, in contrast to STL, which cannot be used to directly verify an infinite set of traces by simply evaluating the STL formula.
2. A *transformation* from sampled time STL to RTL is defined (Sec. 4). We prove that this transformation is sound and complete with respect to finitely represented reach sequences and give a sound transformation from general STL to sampled time STL. Therefore, we are able to translate the STL verification problem on traces to an RTL verification problem on reach sequences.
3. A *model checking* algorithm is introduced to formally verify an STL property on a reach sequence using the transformation from STL to RTL and the semantics of RTL (Sec. 5).

Our theory does not rely on a special representation of reach sequences. Since there exist many reachability analysis tools, such as Cora [2], SpaceEx [12], and C2E2 [10], which can compute reach sequences, our approach is broadly applicable. We show the benefits of our model checking method on an example from the domain of automated driving (Sec. 6).

Note that we are working with overapproximations of exact reach sequences, because reachability analysis for hybrid systems is undecidable in general [20]. There are already some model checking techniques for temporal properties related to LTL: in order to be able to verify an uncountable set of possible traces, one can translate a temporal logic called Hybrid LTL (HyLTL) to a (Büchi) monitor automaton [8]. After parallel composition of the monitor automaton with the hybrid system to be verified, the verification problem reduces to finding a loop in the reach sequence. A problem of the HyLTL model checking approach is that to the best of our knowledge, there is no proof for the soundness of the verification result for the proposed method using overapproximative methods and bounded time horizons, which are common for reachability analysis tools due to undecidability. Another drawback of the HyLTL approach is that parallel composition drastically complicates the hybrid automaton and the reachability analysis so that the composition typically becomes so large that it is infeasible to analyze. With our method, temporal properties can be verified without changing the hybrid automaton, see the example in Sec. 6.

There are several works that also present approaches for model checking of hybrid systems that restrict to discrete time traces [24, 11]. However, these works typically give no formal guarantees for the satisfaction on the continuous time traces, either because they sample the time, or one has to make additional assumptions about the behavior between the sampling points. In contrast, we formally reason about the continuous time traces.

2 Preliminaries

Linked to our model checking method are hybrid systems and Signal Temporal Logic, which are shortly introduced in the following.

2.1 Hybrid systems

Our methods are defined on a sequence of reachable sets of states and thus are invariant to the modeling formalism that describes the evolution of a hybrid system. However, in order to describe how hybrid traces and reach sequences are generated, without loss of generality we use *hybrid automata* as a well-established modeling formalism [19]. In the following, we introduce hybrid automata in a non-formal way. Because the dynamics of real systems are typically not known exactly, we propose including non-deterministic behavior. Components of a hybrid automaton are visualized in Fig. 2 together with a possible reach sequence. Informally, the semantics of a hybrid automaton is as follows: The combined discrete and continuous trace $\xi(t) = (v(t), x(t))$ starts from (v^0, x^0) and $x(t) \in \mathbb{R}^n$

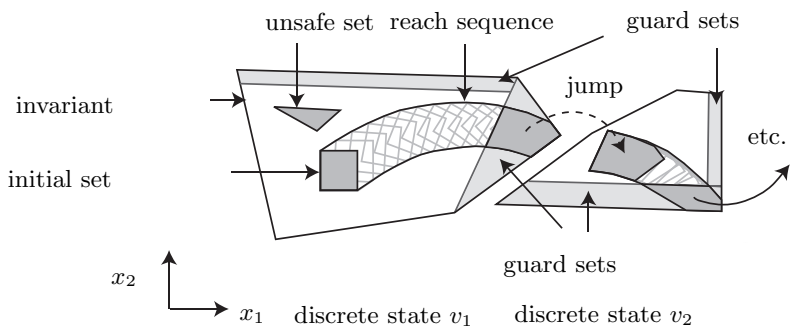


Fig. 2. Illustration of the evolution of a reachable set of a hybrid automaton.

changes according to a differential inclusion $\dot{x}(t) \in \mathcal{H}(v(t), x(t), u(t))$ [25], where $\mathcal{H}(v, x, u)$ is a set of values based on the discrete state $v(t) \in \{v_1, v_2, \ldots, v_p\}$, the continuous state $x(t)$, and the input $u(t) \in \mathbb{R}^m$, such that the differential inclusion models many possible solutions as opposed to ordinary differential equations. If the continuous state is within a *guard set*, the corresponding transition to a new discrete state can be taken. It has to be taken if the state would leave an *invariant*, which is the region in which the differential inclusion of the current discrete state is defined. After the discrete transition is taken (in zero time), the continuous state is updated according to a *jump function*, which models possible instantaneous changes of the continuous state. For ease of presentation, we assume that a hybrid automaton is non-Zeno and non-blocking.

A *trace* $\xi : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ of the hybrid automaton S is of the form

$$\xi(t) = \begin{cases} \xi_0(t), \text{ for } t \in [0, t_1) \\ \xi_1(t), \text{ for } t \in [t_1, t_2) \\ \ldots \end{cases}$$

where $\xi_i : [t_i, t_{i+1}) \to \mathbb{R}^n$ are the evolutions between discrete transitions. The set of all traces of a system S is denoted by $\mathit{Traces}(S)$. In contrast to discrete systems, one cannot generate a tree of possible traces for a system with continuous state variables, since its number of traces is uncountably large. Thus, algorithms for computing reach sequences of systems involving continuous states do not preserve traces anymore, but only store the set of values for points in time and time intervals. A function $\mathbf{R} : \mathbb{R}_{\geq 0} \to \mathcal{P}(\mathbb{R}^n)$ mapping to the power set $\mathcal{P}(\mathbb{R}^n)$ is called a *reach sequence* of S, iff

$$\forall t \in \mathbb{R}_{\geq 0} : \{\xi(t) \mid \xi \in \mathit{Traces}(S)\} \subseteq \mathbf{R}(t) \tag{1}$$

holds. The reach sequence is called exact, iff (1) holds with '$\subseteq$' replaced by '='. An evaluation $\mathbf{R}(t)$ for one point in time t is called a *reachable set*. Typically, other papers use the terminology reachable set only. However, in our work the distinction between reach sequence and reachable set is important for rigorous formulation and understandability. Due to undecidability, exact reachable sets

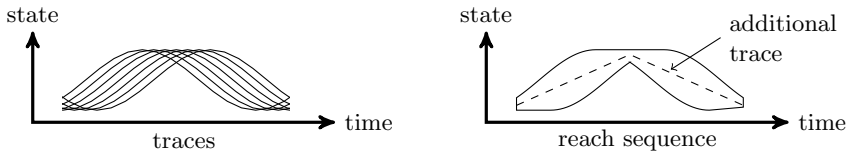


Fig. 3. The reach sequence of a set of traces has potentially additional traces.

typically cannot be obtained for hybrid systems. The set of traces corresponding to $\mathbf{R}$ is defined as

$$\mathcal{C}(\mathbf{R}) = \{\xi \mid \forall t \geq 0 : \xi(t) \in \mathbf{R}(t)\}$$

and contains the set of traces $\mathit{Traces}(S)$ and potentially additional traces (even if $\mathbf{R}$ is exact), as visualized in Fig. 3.

Remark 1. To reduce this conservatism, reachable sets can be split resulting in a tree structure of reach sequence segments. For instance, Cora [2] uses reachable set splitting for accuracy reasons resulting in multiple branches with reach sequences that progress independently. Every path of the tree from the root to a leaf represents one reach sequence. While we focus on one reach sequence in this paper, the results can also be applied to the more general case by considering all reach sequences that can be generated from the tree.

Reachability analysis tools such as Cora can compute (overapproximative) reachable sets R_i for points at time t_i and reachable sets $\overline{R}_i$ for time intervals $[t_i, t_{i+1}]$. We call reachable sequences of the form

$$\mathbf{R} = (t_0, R_0)\ ((t_0, t_1), \overline{R}_0)\ (t_1, R_1)\ ((t_1, t_2), \overline{R}_1)\ \ldots\ ((t_m, t_{m+1}), \overline{R}_m), \tag{2}$$

finitely represented reach sequences, where R_i and $\overline{R}_i$ are sets of states, $t_0 = 0$, $t_{m+1} = \infty$, and define

$$\mathbf{R}(t) = R_i, \text{ iff } t = t_i \quad \text{and} \quad \mathbf{R}(t) = \overline{R}_i, \text{ iff } t \in (t_i, t_{i+1}). \tag{3}$$

The considered time structure with alternating points and open intervals is similar to the one for timed automata, see [5].

2.2 Signal Temporal Logic (STL)

Values of traces are real numbers that vary over time. Hence, STL is a temporal logic to describe properties of continuous-timed and real-valued traces. We briefly introduce STL following Maler et al. [16]. An STL formula consists of atomic predicates (such as $x > 3$), which are composed using logical and temporal operators. The syntax of an STL formula over a finite set of atomic predicates $p \in AP$ is

$$\varphi := p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1\, \mathcal{U}_{[a,b]} \varphi_2.$$

The trace satisfaction semantics of an STL formula φ for a trace ξ is defined recursively on φ:

$$\begin{aligned}
\xi \models_T p &\iff \pi_p(\xi(0)) = true \\
\xi \models_T \neg\varphi &\iff \neg(\xi \models_T \varphi) \\
\xi \models_T \varphi_1 \vee \varphi_2 &\iff (\xi \models_T \varphi_1) \vee (\xi \models_T \varphi_2) \\
\xi \models_T \varphi_1 \, \mathcal{U}_{[a,b]} \varphi_2 &\iff \exists t \in [a,b] : \langle\xi\rangle_t \models_T \varphi_2 \text{ and } \forall t' \in [0,t) : \langle\xi\rangle_{t'} \models_T \varphi_1
\end{aligned}$$

using a predicate evaluation function π_p and the suffix notation $\langle\xi\rangle_a(t) = \xi(t+a)$, which shifts the trace in time. For instance, the *until*-operator $p \, \mathcal{U}_{[a,b]} q$ states that p has to hold for all times until q holds for one point in time. Other common temporal operators can be derived from these operators, such as the *finally*-operator $\mathcal{F}_{[a,b]}\varphi := true \, \mathcal{U}_{[a,b]}\varphi$ and the *globally*-operator $\mathcal{G}_{[a,b]}\varphi := \neg\mathcal{F}_{[a,b]}\neg\varphi$. For brevity of notation, we also introduce the continuous *next*-operator

$$\xi \models_T \mathcal{X}_a\varphi \quad \Leftrightarrow \quad \langle\xi\rangle_a \models_T \varphi \quad \Leftrightarrow \quad \xi \models_T true \, \mathcal{U}_{[a,a]}\varphi.$$

An STL formula in which no temporal operators are present is called a non-temporal formula in the following. Inspired by LTL, we define the statisfaction of an STL formula on a set of traces M as

$$M \models_T \varphi \quad \Leftrightarrow \quad \forall \xi \in M : \xi \models_T \varphi.$$

Formally, the STL verification task for a hybrid system S is to check whether $\mathit{Traces}(S) \models_T \varphi$ holds. Since a verification method has to reason about uncountably many traces, the problem is often replaced by falsification in practice, searching for a trace ξ with $\xi \not\models_T \varphi$. However, falsification cannot prove that φ holds. Note that $\mathit{Traces}(S) \not\models_T \varphi$ does not imply $\mathit{Traces}(S) \models_T \neg\varphi$, because of the $\forall$-quantifier over the traces.

3 Reachset Temporal Logic (RTL)

Evaluation of an STL formula cannot be directly done for an infinite set of traces. Therefore, we introduce a new temporal logic that is defined on reach sequences instead of traces (such as STL), which we refer to as *Reachset Temporal Logic* (RTL). By transforming an STL formula into an RTL formula, we can leverage RTL for model checking the STL formula on a hybrid system, as visualized in Fig. 1. The syntax and semantics of RTL are defined so that STL formulas can be transferred and expressed on reach sequences and have therefore some commonalities with STL, but also important differences.

Definition 1 (RTL syntax). *An RTL formula has the syntax*

$$\psi := \mathcal{A}\varrho \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \mathcal{X}_a\psi \mid \psi_1 \, \mathcal{U}_{[a,b]} \, \psi_2 \mid \psi_1 \, \mathcal{R}_{[a,b]} \, \psi_2$$

where ϱ is a propositional formula $\varrho := p \mid \varrho_1 \vee \varrho_2 \mid \neg\varrho$ over a finite set AP of predicates $p \in AP$.

Note that since we want to work with overapproximations of exact reachable sets, we have the negation operator only for non-temporal formulas, which is the reason for the syntactic split into ψ and ϱ.

Definition 2 (RTL semantics). *For a propositional formula ϱ and a state r the semantics is*

$$\begin{aligned} r \models_P p \quad &\Leftrightarrow \quad \pi_p(r) \\ r \models_P \neg\varrho \quad &\Leftrightarrow \quad r \not\models_P \varrho \\ r \models_P \varrho_1 \vee \varrho_2 \quad &\Leftrightarrow \quad (r \models_P \varrho_1) \vee (r \models_P \varrho_2)\,. \end{aligned}$$

For a reach sequence $\mathbf{R}$ and a formula ψ, the semantics is defined as

$$\mathbf{R} \models_R \mathcal{A}\varrho \quad \Leftrightarrow \forall r \in \mathbf{R}(0) : r \models_P \varrho \tag{4}$$
$$\mathbf{R} \models_R \psi_1 \wedge \psi_2 \quad \Leftrightarrow (\mathbf{R} \models_R \psi_1) \wedge (\mathbf{R} \models_R \psi_2) \tag{5}$$
$$\mathbf{R} \models_R \psi_1 \vee \psi_2 \quad \Leftrightarrow (\mathbf{R} \models_R \psi_1) \vee (\mathbf{R} \models_R \psi_2) \tag{6}$$
$$\mathbf{R} \models_R \mathcal{X}_a\psi \quad \Leftrightarrow \langle\mathbf{R}\rangle_a \models_R \psi \tag{7}$$
$$\mathbf{R} \models_R \psi_1\, \mathcal{U}_{[a,b]}\psi_2 \Leftrightarrow \exists t \in [a,b] : (\langle\mathbf{R}\rangle_t \models_R \psi_2) \wedge (\forall i \in [0,t) : \langle\mathbf{R}\rangle_i \models_R \psi_1) \tag{8}$$
$$\mathbf{R} \models_R \psi_1 \mathcal{R}_{[a,b]}\psi_2 \Leftrightarrow \forall t \in [a,b] : (\langle\mathbf{R}\rangle_t \models_R \psi_2) \vee (\exists i \in [0,t) : \langle\mathbf{R}\rangle_i \models_R \psi_1) \tag{9}$$

where $\langle\mathbf{R}\rangle_a(t) := \mathbf{R}(t+a)$ is the shift operator and $a \in \mathbb{R}_{\geq 0}, b \in \mathbb{R}_{\geq 0}$ with $a \leq b$.

Two RTL formulas ψ_1, ψ_2 are equivalent, denoted as $\psi_1 \equiv \psi_2$, iff the satisfaction is the same for all possible reach sequences. The operators $\mathcal{F}$ and $\mathcal{G}$ are defined similarly to STL:

$$\mathcal{F}_{[a,b]}\psi := \mathit{true}\, \mathcal{U}_{[a,b]}\psi \text{ (finally)} \quad \text{and} \quad \mathcal{G}_{[a,b]}\psi := \mathit{false}\mathcal{R}_{[a,b]}\psi \text{ (globally)}$$

To give an example, we consider the formula $\mathcal{F}_{[0,1]}\mathcal{A}\varrho$. A reach sequence $\mathbf{R}$ has to satisfy that ϱ holds for all states in one $\mathbf{R}(t)$ between time 0 and 1. Expressed on the set of traces $\mathcal{C}(\mathbf{R})$ corresponding to $\mathbf{R}$, this implies that all traces satisfy ϱ for one common point in time, compared to the requirement $\mathcal{F}_{[0,1]}\varrho$ for all traces:

$$\mathbf{R} \models_R \mathcal{F}_{[0,1]}\mathcal{A}\varrho \quad \Leftrightarrow \quad \exists t \in [0,1] : \mathcal{C}(\mathbf{R}) \models_T \mathcal{F}_{[t,t]}\varrho \quad \Rightarrow \quad \mathcal{C}(\mathbf{R}) \models_T \mathcal{F}_{[0,1]}\varrho.$$

Since a set of traces satisfies an STL formula if each trace satisfies the formula, the traces are "checked" independently of each other, i.e. it is not possible to reason about a variable point $t \in [a,b]$ in time at which something holds for all traces in a set. Therefore, this cannot be expressed by STL. In contrast, RTL is able to express common satisfaction of predicates.

4 Transformation from STL to RTL

Differences of STL and RTL described in the previous section have some important implications for the transformation between these temporal logics. In this

section we present a transformation Υ mapping an STL formula to an RTL formula. We first give some properties of a sound and complete transformation and then present a transformation for sampled time formulas and finitely represented reach sequences (Sec. 4.1). We further show that the results can be extended by transforming general STL formulas to sampled time formulas (Sec. 4.2). The methods will be used later to model check STL formulas, as shown in Fig. 1.

With a mapping Υ from STL to RTL we are able to transfer the verification task on the traces of a reach sequence $\mathcal{C}(\mathbf{R}) \models_T \varphi$ into a reach sequence verification task $\mathbf{R} \models_R \Upsilon(\varphi)$. Since we do not want to lose expressiveness, we demand from the transformation Υ that

$$\mathbf{R} \models_R \Upsilon(\varphi) \quad \Rightarrow \quad \mathcal{C}(\mathbf{R}) \models_T \varphi \quad \text{and} \quad \mathcal{C}(\mathbf{R}) \models_T \varphi \quad \Rightarrow \quad \mathbf{R} \models_R \Upsilon(\varphi)$$

holds, which we call soundness and completeness, respectively, for the reach sequence abstraction. If soundness and completeness is given for Υ, the semantical domain can be changed without changing the verification result. The following lemma gives some properties of a sound and complete Υ.

Lemma 1. *Let the STL formulas φ_i and the non-temporal formula ϱ be given. A sound and complete transformation Υ has the following properties:*

$$\Upsilon(\varrho) \equiv \mathcal{A}\varrho \qquad \textit{non-temporal transformation} \tag{10}$$

$$\Upsilon(\varphi_1 \wedge \varphi_2) \equiv \Upsilon(\varphi_1) \wedge \Upsilon(\varphi_2) \qquad \wedge\textit{-distributivity.} \tag{11}$$

Furthermore, the $\vee$-distributivity

$$\Upsilon(\varphi_1 \vee \varphi_2) \equiv \Upsilon(\varphi_1) \vee \Upsilon(\varphi_2) \qquad \vee\textit{-distributivity} \tag{12}$$

does also hold, if $tsupp(\varphi_1) \cap tsupp(\varphi_2) = \emptyset$ for

$$tsupp(\varphi) := \{t \mid \exists \xi, \xi' : \mathbb{R} \to \mathbb{R}^n, (\xi \models_T \varphi) \wedge (\xi' \not\models_T \varphi) \wedge (\forall t' \neq t : \xi(t') = \xi'(t'))\},$$

which are the points in time where a change in the trace can affect whether φ is true or not.

Proof. For non-temporal properties ϱ, (10) follows from

$$\mathcal{C}(\mathbf{R}) \models_T \varrho \quad \Leftrightarrow \ \forall \xi \in \mathcal{C}(\mathbf{R}) : \xi \models_T \varrho \quad \Leftrightarrow \ \forall r \in \mathbf{R}(0) : r \models_P \varrho \quad \Leftrightarrow \ \mathbf{R} \models_R \mathcal{A}\varrho.$$

From soundness and completeness of Υ and the RTL semantics follows

$$\mathbf{R} \models_R \Upsilon(\varphi_1 \wedge \varphi_2) \Leftrightarrow \mathcal{C}(\mathbf{R}) \models_T \varphi_1 \wedge \varphi_2 \tag{13}$$

$$\begin{aligned} \mathcal{C}(\mathbf{R}) \models_T \varphi_1 \wedge \varphi_2 &\Leftrightarrow \forall \xi \in \mathcal{C}(\mathbf{R}) : \xi \models_T \varphi_1 \wedge \varphi_2 \\ &\Leftrightarrow \forall \xi \in \mathcal{C}(\mathbf{R}) : \xi \models_T \varphi_1 \wedge \forall \xi \in \mathcal{C}(\mathbf{R}) : \xi \models_T \varphi_2 \end{aligned} \tag{14}$$

$$\begin{aligned} \mathcal{C}(\mathbf{R}) \models_T \varphi_1 \wedge \mathcal{C}(\mathbf{R}) \models_T \varphi_2 &\Leftrightarrow \mathbf{R} \models_R \Upsilon(\varphi_1) \wedge \mathbf{R} \models_R \Upsilon(\varphi_2) \\ &\Leftrightarrow \mathbf{R} \models_R \Upsilon(\varphi_1) \wedge \Upsilon(\varphi_2) \end{aligned} \tag{15}$$

which proves (11). The equivalences (13) and (15) hold also for $\vee$. Let us assume $\mathcal{C}(\mathbf{R}) : \xi \models_T \varphi_1 \vee \varphi_2$ holds, but not $\mathcal{C}(\mathbf{R}) : \xi \models_T \varphi_1 \vee \mathcal{C}(\mathbf{R}) : \xi \models_T \varphi_2$. Then,

there exist ξ_1, ξ_2 with $\xi_1 \not\models_T \varphi_1$ and $\xi_2 \not\models_T \varphi_2$. Because of the empty time support intersection and the special structure of $\mathcal{C}(\mathbf{R})$, we can construct ξ with $\xi(t) = \xi_1(t)$ for $t \in tsupp(\varphi_1)$ and $\xi(t) = \xi_2(t)$ otherwise. Since $\xi \in \mathcal{C}(\mathbf{R})$ and $\xi \not\models_T \varphi_1$, $\xi \not\models_T \varphi_2$, this is a contradiction and therefore (12) holds, because the other direction can also be easily shown. □

Based on the properties from Lemma 1, one can see the subtle differences between a well-defined complete and a non-complete transformation. Let us consider the STL formula $\varphi := (\varrho_0 \wedge \mathcal{X}_1 \varrho_1) \vee \mathcal{X}_1 \varrho_0$, which could be transformed to $\psi := (\mathcal{A}\varrho_0 \wedge \mathcal{X}_1 \mathcal{A}\varrho_1) \vee \mathcal{X}_1 \mathcal{A}\varrho_0$ by simply adding the $\mathcal{A}$-operator to the non-temporal subformulas of the STL formula φ. However, if we first rewrite φ to the equivalent formula $(\varrho_0 \vee \mathcal{X}_1 \varrho_0) \wedge \mathcal{X}_1(\varrho_0 \vee \varrho_1)$ and transform it, we get $\psi' := (\mathcal{A}\varrho_0 \vee \mathcal{X}_1 \mathcal{A}\varrho_0) \wedge \mathcal{X}_1 \mathcal{A}(\varrho_0 \vee \varrho_1) \equiv (\mathcal{A}\varrho_0 \wedge \mathcal{X}_1 \mathcal{A}(\varrho_0 \vee \varrho_1)) \vee \mathcal{X}_1 \mathcal{A}\varrho_0$. The formula ψ' does not force all the traces to satisfy ϱ_1 at time 1, if one trace does not satisfy ϱ_0 at time 1. Since ψ' also implies φ, it is a sound transformation of φ which is less restrictive than ψ. As one can see from this example, a sound and complete transformation cannot simply be constructed by structural induction over the parts of an STL formula, even if no nested temporal operators are used. Different parts of a formula are able to interact with each other if they are composed with the $\vee$-operator. In the following, we build upon Lemma 1 and give a sound and complete transformation function for sampled time formulas.

4.1 Sound and complete transformation for sampled time formulas

Operators can appear arbitrarily nested in STL formulas. Given a fixed $c > 0$, we call the subclass of STL which restricts formulas to

$$\varphi := \varrho \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathcal{X}_c \varphi \mid \mathcal{F}_{(0,c)} \varrho \mid \mathcal{G}_{(0,c)} \varrho, \quad \varrho := p \mid \neg\varrho \mid \varrho_1 \vee \varrho_2$$

sampled time STL with timestep c. For example $p \vee \mathcal{F}_{(0,c)} p \vee \mathcal{X}_c \left(p \vee \mathcal{F}_{(0,c)} p \vee \mathcal{X}_c p\right)$ can be seen as a sampled time version of the STL formula $\mathcal{F}_{[0,2c]} p$. Since standard equivalences hold on STL formulas, such as $\neg\mathcal{X}_c \varphi \equiv \mathcal{X}_c \neg\varphi$, $\neg\mathcal{F}_{(0,c)} \varphi \equiv \mathcal{G}_{(0,c)} \neg\varphi$, and $\mathcal{X}_c (\varphi_1 \vee \varphi_2) \equiv \mathcal{X}_c \varphi_1 \vee \mathcal{X}_c \varphi_2$, each sampled time formula has an equivalent sampled time formula in conjunctive normal form $\bigwedge_i \bigvee_j \mathcal{X}_c^j (\varphi_{ij} \vee \varrho_{ij})$ with φ_{ij} of the form $\bigvee_k \mathcal{F}_{(0,c)} \varrho_k \vee \bigvee_l \mathcal{G}_{(0,c)} \varrho_l$, non-temporal formulas ϱ_{ij}, and the $\mathcal{X}$-operator in series $\mathcal{X}_c^j := \mathcal{X}_{j \cdot c}$. Based on the conjunctive normal form, we are able to introduce a sound and complete transformation Υ considering finitely represented reach sequences and given that c divides all time intervals of the reach sequence. Since finitely represented reach sequences can be produced by Cora [2] and SpaceEx [12] for instance, this is of practical relevance.

Lemma 2. *Let a sampled time formula be given in conjunctive normal form. Then, the transformation Υ from STL to RTL defined via*

$$\Upsilon\left(\bigwedge_i \bigvee_j \mathcal{X}_c^j (\varphi_{ij} \vee \varrho_{ij})\right) := \bigwedge_i \bigvee_j \mathcal{X}_c^j (\Upsilon(\varphi_{ij}) \vee \mathcal{A}\varrho_{ij}) \tag{16}$$

$$\Upsilon\left(\bigvee_{k\in K}\mathcal{F}_{(0,c)}\varrho_k \vee \bigvee_{l\in L}\mathcal{G}_{(0,c)}\varrho_l\right) := \mathcal{X}_{\frac{c}{2}}\left(\mathcal{A}\varrho' \vee \bigvee_{l\in L}\mathcal{A}\left(\varrho_l \vee \varrho'\right)\right), \qquad (17)$$
$$\text{with } \varrho' := \bigvee_{k\in K}\varrho_k$$

is sound and complete for finitely represented reach sequences $\mathbf{R} = (t_0, R_0)$ $((t_0, t_1), \overline{R}_0)$ (t_1, R_1) $\ldots$ $((t_m, t_{m+1}), \overline{R}_m)$*, which are c-divisible, where c-divisibility holds if and only if* $t_i \in \mathbb{N}c := \{0, c, 2c, \ldots\}$ *holds for all i.*

Proof. Soundness and completeness can be proven by structural induction. Since we define the transformation such that $\Upsilon(\varphi_1 \wedge \varphi_2) \equiv \Upsilon(\varphi_1) \wedge \Upsilon(\varphi)$ holds, it can be shown similarly as in Lemma 1 that it is sufficient to show soundness and completeness for $\bigvee_j \mathcal{X}_c^j(\varphi_{ij} \vee \varrho_{ij})$, which works similarly, because different time branches have different time supports $tsupp(\mathcal{X}_c^j(\varphi_{ij} \vee \varrho_{ij})) \subseteq [j, j+1)$. Therefore it is sufficient to show soundness and completeness for (17). For brevity reasons, we do not give the proof for general formulas, but prove that the two terms

$$\mathcal{C}(\mathbf{R}) \models_T \mathcal{G}_{(0,c)}\varrho_1 \vee \mathcal{G}_{(0,c)}\varrho_2 \vee \mathcal{F}_{(0,c)}\varrho_3 \qquad (18)$$
$$\mathbf{R} \models_R \mathcal{X}_{\frac{c}{2}}\mathcal{A}(\varrho_1 \vee \varrho_3) \vee \mathcal{X}_{\frac{c}{2}}\mathcal{A}(\varrho_2 \vee \varrho_3) \qquad (19)$$

are equivalent. Let us assume that (19) holds and therefore without loss of generality $\mathbf{R} \models_R \mathcal{X}_{\frac{c}{2}}\mathcal{A}(\varrho_1 \vee \varrho_3)$ holds. Since $\mathbf{R}$ is a finitely represented reach sequence which changes values only at points in time divisible by c, also $R \models_R \mathcal{G}_{(0,c)}\mathcal{A}(\varrho_1 \vee \varrho_3)$ and therefore $\mathcal{C}(\mathbf{R}) \models_T \mathcal{G}_{(0,c)}(\varrho_1 \vee \varrho_3)$ holds, which implies (18). On the other hand, let us assume (19) does not hold. Therefore

$$\begin{aligned}&\mathbf{R} \not\models_R \mathcal{X}_{\frac{c}{2}}\mathcal{A}(\varrho_1 \vee \varrho_3) \wedge \mathbf{R} \not\models_R \mathcal{X}_{\frac{c}{2}}\mathcal{A}(\varrho_2 \vee \varrho_3)\\ \Rightarrow &\exists r_1 \in \mathbf{R}\left(\frac{c}{2}\right) : r_1 \models_p \neg\varrho_1 \wedge \neg\varrho_3 \quad \wedge \quad \exists r_2 \in \mathbf{R}\left(\frac{c}{2}\right) : r_2 \models_p \neg\varrho_2 \wedge \neg\varrho_3\end{aligned}$$

holds. Hence, Eq. (18) does not hold, because the trace

$$\xi(t) := \begin{cases} r_1, & t \in \left(0, \frac{c}{2}\right) \\ r_2, & t \in \left[\frac{c}{2}, c\right) \\ \text{any } r \in R(t), & \text{otherwise} \end{cases}$$

is contained in $\mathcal{C}(\mathbf{R})$ but does not satisfy the formula in (18). □

Lemma 2 proves that the RTL formula $\psi := (\mathcal{A}\varrho_0 \wedge \mathcal{X}_1\mathcal{A}(\varrho_0 \vee \varrho_1)) \vee \mathcal{X}_1\mathcal{A}\varrho_0$ is a sound and complete transformation of the formula $(\varrho_0 \wedge \mathcal{X}_1\varrho_1) \vee \mathcal{X}_1\varrho_0$ considered in the previous section. As we have seen above, the formula $\mathcal{F}_{[0,2c]}p$ has an equivalent sampled time notation. Therefore, it can be transformed to $\psi' := \mathcal{A}p \vee \mathcal{X}_{\frac{c}{2}}\mathcal{A}p \vee \mathcal{X}^2_{\frac{c}{2}}\mathcal{A}p \vee \mathcal{X}^3_{\frac{c}{2}}\mathcal{A}p \vee \mathcal{X}^4_{\frac{c}{2}}\mathcal{A}p$ using Lemma 2. Since we do not have any temporal operators but the shift operator in ψ and ψ', the formulas can easily be checked on a reach sequence. This is the basis for our model checking approach in Sec. 5. Note that $\frac{c}{2}$ can be seen as a compatible sample time that jumps from one point in time kc to the next open interval $(kc, (k+1)c)$ or from an open interval $(kc, (k+1)c)$ to the next point in time $(k+1)c$ respectively, as shown in Fig. 4.

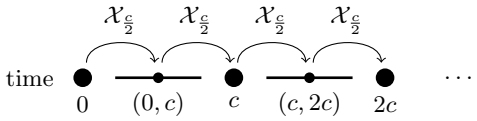


Fig. 4. The next operator $\mathcal{X}_{\frac{c}{2}}$ points to the next interval or point.

4.2 Transformation of general STL to sampled time STL

Rewriting a general STL formula as an sampled time formula enables us to use the results of the previous section for general STL formulas. The rewriting is sound and therefore, we are able to reason about the satisfaction of an STL formula on reach sequences. The main idea is to leverage the finite representation of a given STL formula φ for rewriting and use rules of the form $\xi \models_T \varphi \Leftarrow \xi \models_T \varphi'$ to rewrite φ to a sampled time version φ' in a sound manner. If we have such rules, they can also be applied to $\mathcal{C}(\mathbf{R})$.

Lemma 3. *Let φ be an STL formula which can be written as $f(\varphi_1, \ldots, \varphi_n)$, where f is a function composing φ_i by $\wedge$, $\vee$, and $\mathcal{X}_c$. Let $\xi \models_T \varphi_i \Leftarrow \xi \models_T \varphi_i'$ for all i and ξ. Then*

$$\mathcal{C}(\mathbf{R}) \models_T \varphi \Leftarrow \mathcal{C}(\mathbf{R}) \models_T \varphi'$$

holds with $\varphi' = f(\varphi_1', \ldots, \varphi_n')$.

Proof. Let us assume $\xi \models_T \varphi_1' \wedge \varphi_2'$, which is equivalent to $\xi \models_T \varphi_1' \wedge \xi \models_T \varphi_2'$, holds for all ξ. From the rewriting rules it follows that $\xi \models_T \varphi_1 \wedge \xi \models_T \varphi_2$ and therefore $\xi \models_T \varphi_1 \wedge \varphi_2$ holds also. The proof follows from

$$\begin{aligned} \mathcal{C}(\mathbf{R}) \models_T \varphi_1 \wedge \varphi_2 \quad &\Leftrightarrow \quad \forall \xi \in \mathcal{C}(\mathbf{R}) : \xi \models_T \varphi_1 \wedge \varphi_2 \\ \Leftarrow \quad \forall \xi \in \mathcal{C}(\mathbf{R}) : \xi \models_T \varphi_1' \wedge \varphi_2' \quad &\Leftrightarrow \quad \mathcal{C}(\mathbf{R}) \models_T \varphi_1' \wedge \varphi_2' \end{aligned}$$

by structural induction over $\wedge$, $\vee$, and $\mathcal{X}_c$. □

Finally, we need a set of rewriting rules that are sufficient to rewrite general STL formulas as sampled time ones.

Lemma 4. *Let an STL formula φ be given, which is c-divisible, where c-divisibility holds if c divides all bounds of temporal operators of φ. Without loss of generality, we assume that φ is in negation normal form. Hence, φ can be written as $f(\varphi_1, \ldots, \varphi_n)$, where f is a function composing φ_i by $\wedge$, $\vee$, and $\mathcal{X}_c$ and the outmost operator of each φ_i is a temporal operator or φ_i is non-temporal. Then, for any temporal φ_i there is a rewriting in Table 1 or one of the following equivalences using subformulas $\widehat{\varphi}_i$*

$$\widehat{\varphi}_1 \, \mathcal{U}_{[0,0]} \widehat{\varphi}_2 \equiv \widehat{\varphi}_2, \quad \widehat{\varphi}_1 \, \mathcal{R}_{[0,0]} \widehat{\varphi}_2 \equiv \widehat{\varphi}_2, \quad \mathcal{F}_I \mathcal{X}_1 \widehat{\varphi}_1 \equiv \mathcal{X}_1 \mathcal{F}_I \widehat{\varphi}_1, \quad \mathcal{G}_I \mathcal{X}_1 \widehat{\varphi}_1 \equiv \mathcal{X}_1 \mathcal{G}_I \widehat{\varphi}_1$$

such that φ can be rewritten to $rw(\varphi) = f(\varphi_1', \ldots, \varphi_n')$ in a sound manner. The formula φ can be rewritten to a sampled time version with timestep c by iteratively using the rewriting $\varphi \mapsto rw(\varphi) \mapsto rw^2(\varphi) \mapsto \ldots$ until no rewriting rule matches anymore.

Table 1. For all ξ the formula $\xi \models_T \varphi_i \Leftarrow \xi \models_T \varphi_i'$ holds for each pair φ_i, φ_i' in the table. For readability reasons, we use $I = (0, c)$ and assume $c = 1$.

φ_i	φ_i'
$\varphi_1 \,\mathcal{U}_{[i,j]} \varphi_2$	$\varphi_1 \wedge \mathcal{G}_I \varphi_1 \wedge \mathcal{X}_1 \left(\varphi_1 \,\mathcal{U}_{[i-1,j-1]} \varphi_2\right)$
$\varphi_1 \,\mathcal{U}_{[0,j]} \varphi_2$	$\varphi_2 \vee \left(\varphi_1 \wedge \mathcal{G}_I \varphi_1 \wedge \left(\mathcal{F}_I \varphi_2 \vee \mathcal{X}_1 \left(\varphi_1 \,\mathcal{U}_{[0,j-1]} \varphi_2\right)\right)\right)$
$\varphi_1 \,\mathcal{R}_{[i,j]} \varphi_2$	$\varphi_1 \vee \mathcal{F}_I \varphi_1 \vee \mathcal{X}_1 \left(\varphi_1 \,\mathcal{R}_{[i-1,j-1]} \varphi_2\right)$
$\varphi_1 \,\mathcal{R}_{[0,j]} \varphi_2$	$\varphi_2 \wedge \left(\varphi_1 \vee \left(\mathcal{G}_I \varphi_2 \wedge \left(\mathcal{F}_I \varphi_1 \vee \mathcal{X}_1 \left(\varphi_1 \,\mathcal{R}_{[0,j-1]} \varphi_2\right)\right)\right)\right)$
$\mathcal{G}_I(\varphi_1 \wedge \varphi_2)$	$\mathcal{G}_I \varphi_1 \wedge \mathcal{G}_I \varphi_2$
$\mathcal{G}_I(\varphi_1 \vee \varphi_2)$	$\mathcal{G}_I \varphi_1 \vee \mathcal{G}_I \varphi_2$
$\mathcal{F}_I(\varphi_1 \wedge \varphi_2)$	$(\mathcal{G}_I \varphi_1 \wedge \mathcal{F}_I \varphi_2) \vee (\mathcal{F}_I \varphi_1 \wedge \mathcal{G}_I \varphi_2)$
$\mathcal{F}_I(\varphi_1 \vee \varphi_2)$	$\mathcal{F}_I \varphi_1 \vee \mathcal{F}_I \varphi_2$
$\mathcal{F}_I(\varphi_1 \,\mathcal{U}_{[i,j]} \varphi_2)$	$\mathcal{G}_I \varphi_1 \wedge \mathcal{X}_1 \left(\varphi_1 \wedge \mathcal{G}_I \varphi_1 \wedge \mathcal{F}_I(\varphi_1 \,\mathcal{U}_{[i-1,j-1]} \varphi_2)\right)$
$\mathcal{F}_I(\varphi_1 \,\mathcal{U}_{[0,j]} \varphi_2)$	$\mathcal{F}_I \varphi_2 \vee \left(\mathcal{G}_I \varphi_1 \wedge \mathcal{X}_1 \left(\varphi_2 \vee \left(\varphi_1 \wedge \mathcal{G}_I \varphi_1 \wedge \mathcal{F}_I(\varphi_1 \,\mathcal{U}_{[0,j-1]} \varphi_2)\right)\right)\right)$
$\mathcal{G}_I(\varphi_1 \,\mathcal{U}_{[i,j]} \varphi_2)$	$\mathcal{G}_I \varphi_1 \wedge \mathcal{X}_1 \left(\varphi_1 \wedge \mathcal{G}_I \varphi_1 \wedge \mathcal{G}_I(\varphi_1 \,\mathcal{U}_{[i-1,j-1]} \varphi_2)\right)$
$\mathcal{G}_I(\varphi_1 \,\mathcal{U}_{[0,j]} \varphi_2)$	$\mathcal{G}_I \varphi_2 \vee \left(\mathcal{G}_I \varphi_1 \wedge \mathcal{X}_1 \left(\varphi_2 \vee \left(\varphi_1 \wedge \mathcal{G}_I \varphi_1 \wedge \mathcal{G}_I(\varphi_1 \,\mathcal{U}_{[0,j-1]} \varphi_2)\right)\right)\right)$
$\mathcal{F}_I(\varphi_1 \,\mathcal{R}_{[i,j]} \varphi_2)$	$\mathcal{F}_I \varphi_1 \vee \mathcal{X}_1 \left(\varphi_1 \vee \mathcal{F}_I \varphi_1 \vee \mathcal{F}_I(\varphi_1 \,\mathcal{R}_{[i-1,j-1]} \varphi_2)\right)$
$\mathcal{F}_I(\varphi_1 \,\mathcal{R}_{[0,j]} \varphi_2)$	$\mathcal{F}_I(\varphi_1 \wedge \varphi_2) \vee \left(\mathcal{G}_I \varphi_2 \wedge \mathcal{X}_1 \left(\varphi_2 \wedge \left(\varphi_1 \vee \left(\mathcal{G}_I \varphi_2 \wedge \mathcal{F}_I \,\varphi_1 \,\mathcal{R}_{[0,j-1]} \varphi_2\right)\right)\right)\right)$
$\mathcal{G}_I(\varphi_1 \,\mathcal{R}_{[i,j]} \varphi_2)$	$\mathcal{G}_I \varphi_1 \vee \mathcal{X}_1 \left(\varphi_1 \vee \mathcal{G}_I(\varphi_1 \,\mathcal{R}_{[i-1,j-1]} \varphi_2)\right)$
$\mathcal{G}_I(\varphi_1 \,\mathcal{R}_{[0,j]} \varphi_2)$	$\mathcal{G}_I(\varphi_2) \wedge \left(\mathcal{G}_I \varphi_1 \vee \mathcal{X}_1 \left(\varphi_2 \wedge \left(\varphi_1 \vee \left(\mathcal{G}_I \varphi_2 \wedge \mathcal{G}_I(\varphi_1 \,\mathcal{R}_{[0,j-1]} \varphi_2)\right)\right)\right)\right)$

Proof. Since we assume c-divisibility and negation normal form, each temporal operator of the subformula is a $\mathcal{U}$-operator or an $\mathcal{R}$-operator and one of the first 4 rewriting rules of Table 1 can be applied. After the first rewriting step, there are potentially formulas nested in $\mathcal{G}_I$ or $\mathcal{F}_I$. For every possible operator there is exactly one rewriting rule. With Lemma 3, it is sufficient to prove the soundness of the rewriting rules in Table 1. Let us consider $c = 1$ and the formula $\varphi_1 \,\mathcal{U}_{[0,j]} \varphi_2$, which is true if φ_2 holds, $\exists t \in (0,1) : \mathcal{G}_{[0,t)} \varphi_1 \wedge \mathcal{X}_t \varphi_2$ holds, or $\mathcal{G}_{[0,1)} \varphi_1 \wedge \mathcal{X}_1(\varphi_1 \,\mathcal{U}_{[0,j-1]} \varphi_2)$ holds. By overapproximating $\exists t \in (0,1) : \mathcal{G}_{[0,t)} \varphi_1 \wedge \mathcal{X}_t \varphi_2$ with $\mathcal{G}_{[0,1)} \varphi_1 \wedge \mathcal{F}_{(0,1)} \varphi_2$ we obtain the rewritten formula. The other formulas can be proven similarly. □

If needed, temporal formula such as $p \,\mathcal{U}_{[0,0.9]} q$ can als be rewritten to $p \,\mathcal{U}_{[0,1]} q$ in a sound manner, if $c = 1$ should be enforced. However, this is typically not needed since one can choose alternatives such as $c = 0.9$ or $c = 0.1$ which also depends on the reach sequence. As an example, the formula $\varphi := p \,\mathcal{U}_{[0,2]} q$ with atomic propositions p and q can be rewritten as follows:

$$\begin{aligned} \varphi &\to \varphi_2 \vee \left(\varphi_1 \wedge \mathcal{G}_I \varphi_1 \wedge \left(\mathcal{F}_I \varphi_2 \vee \mathcal{X}_1 \left(\varphi_1 \,\mathcal{U}_{[0,1]} \varphi_2\right)\right)\right) \\ &\to \varphi_2 \vee (\varphi_1 \wedge \mathcal{G}_I \varphi_1 \wedge (\mathcal{F}_I \varphi_2 \vee \mathcal{X}_1 (\varphi_2 \vee (\varphi_1 \wedge \mathcal{G}_I \varphi_1 \wedge (\mathcal{F}_I \varphi_2 \vee \mathcal{X}_1 \varphi_2))))) \,. \end{aligned}$$

Now that we have solved the problem of transforming an STL formula to an RTL formula defined on the reach sequence, we present a model checking algorithm in the next section.

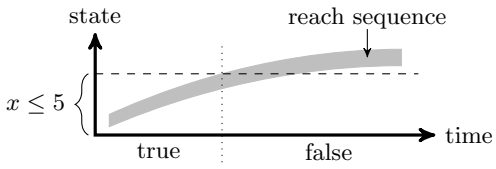


Fig. 5. Predicate evaluation for several points in time t: $\mathbf{R} \models_R \mathcal{X}_t \mathcal{A}(x \leq 5)$.

5 STL Model Checking

Our model checking approach for STL formulas is presented in the following. The foundation of the approach follows from Lemma 2 to 4 and is summarized in the following theorem.

Theorem 1. *Let φ be an STL formula, $\mathbf{R}$ be a reach sequence of a hybrid automaton S, and $\mathbf{R}$ and φ be c-divisible. The formula φ can be transformed to an RTL formula $\psi = \bigwedge_i \bigvee_j \mathcal{X}^j_{\frac{c}{2}} \bigvee_k \mathcal{A}\varrho_{ijk}$ with non-temporal properties ϱ_{ijk}, where*

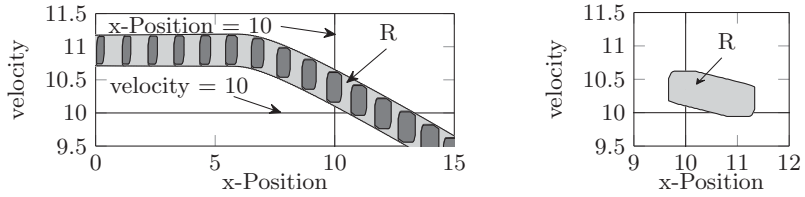
$$\mathcal{C}(\mathbf{R}) \models_T \varphi \quad \Leftarrow \quad \mathbf{R} \models_R \bigwedge_i \bigvee_j \mathcal{X}^j_{\frac{c}{2}} \bigvee_k \mathcal{A}\varrho_{ijk}$$

holds and therefore, the transformation is sound. If φ is equivalent to a sampled time STL formula, the transformation is complete. Hence, $\mathbf{R} \models_R \psi$ implies $Traces(S) \models_T \varphi$, which proves φ for the hybrid automaton S.

It remains to show how $\bigwedge_i \bigvee_j \mathcal{X}^j_{\frac{c}{2}} \bigvee_k \mathcal{A}\varrho_{ijk}$ can be evaluated on a reach sequence $\mathbf{R}$. This can be reduced to the problem $\mathbf{R} \models_R \mathcal{X}^j_{\frac{c}{2}} \mathcal{A}\varrho_{ijk}$. The satisfaction result is obtained by evaluating all such subformulas and then computing the Boolean value of the remaining logical formula.

Our RTL syntax and semantics, as well as the transformation from STL to RTL, are independent of the representation of the reachable sets $\mathbf{R}(t)$ and the predicates used. However, to implement a model checking algorithm, we have to define a representation and a set of predicates we rely on. Therefore, we assume that the reachable sets are represented by (sets of) polytopes as in SpaceEx [12] and Cora [2]. Given a set of vectors $c_1, \ldots, c_k$ and values $d_1, \ldots, d_k$, a *polytope* is defined as the set $poly(c_1, \ldots, c_k, d_1, \ldots, d_k) = \bigcap_{i=1}^k \{x \in \mathbb{R}^n \mid c_i^T x \leq d_i\}$, which is the intersection of halfspaces. We consider the set AP of atomic predicates of the form $a^T x \sim b$, where $a \in \mathbb{R}^n$, $b \in \mathbb{R}$, and $\sim \in \{<, \leq, >, \geq\}$, which are also halfspace restrictions. For instance, the evaluation of $\mathcal{A}(x \leq 5)$ for a reach sequence is visualized in Fig. 5. Note that the formula is only satisfied if all states x satisfy $x \leq 5$.

Given a formula of the type $\mathcal{A}\varrho$, the logical part ϱ can be transformed into disjunctive normal form $\varrho = \bigvee_i \bigwedge_j (a_{ij}^T x \sim b_{ij})$ with $\sim \in \{<, \leq\}$. Because $\bigwedge_j (a_{ij}^T x \sim b_{ij})$ corresponds to the polytope region $poly_i = poly(a_{i1}, \ldots, b_{i1}, \ldots)$, the check $\mathbf{R} \models_R \mathcal{X}_t \mathcal{A}\varrho$ can be performed by the polytope inclusion check $\mathbf{R}(t) \subseteq \bigcup poly_i$, which can be implemented using standard polytope libraries.



(a) Reachable sets for t_i (dark) and (t_i, t_{i+1}) (light) (b) Reachable set R

Fig. 6. Automated driving example for a reach sequence.

6 Example

In the following, we provide an example for our model checking method from the domain of automated driving. For automated driving, it is important to verify safety properties such as the absence of collisions. While driving, this can be done by periodically checking that a collision is not possible for a bounded time of the planned trajectory using the reach sequence [3]. However, there are also other safety relevant *temporal* properties which should be verified. Based on the results in this paper, the verification of these properties can be easily integrated in the existing verification scheme.

For example, when a vehicle is traversing a crossing, it should not block the crossing and should maintain a certain velocity until it reaches the other side. This can be expressed on the traces as an STL property similar to $\varphi := v \geq 10\ \mathcal{U}_{[0,2]} x \geq 10$, where v is the velocity and x is the distance covered. We use Cora [2] and the vehicle model of Althoff and Dolan [3] to compute the reachable sequence of the vehicle as visualized in Fig. 6. To verify φ with the reach sequence, we transform φ to a sampled time RTL formula. An exemplary transformation result for φ is

$$\mathcal{A}q \vee (\mathcal{A}(p \vee q) \wedge \mathcal{X}_{\frac{c}{2}} \mathcal{A}p \wedge (\mathcal{X}_{\frac{c}{2}} \mathcal{A}q \vee \mathcal{X}^2_{\frac{c}{2}} \mathcal{A}q \vee (\mathcal{X}^2_{\frac{c}{2}} \mathcal{A}(p \vee q) \wedge \mathcal{X}^3_{\frac{c}{2}} \mathcal{A}p \wedge (\mathcal{X}^3_{\frac{c}{2}} \mathcal{A}q \vee \mathcal{X}^4_{\frac{c}{2}} \mathcal{A}q))))$$

for $c = 1$, $p = v \geq 10$, and $q = x \geq 10$. In this example, reachability analysis, which is the basis for verification of both safety and temporal properties, takes 3.8 seconds. Checking that the resulting reach sequence satisfies the RTL formula takes only 0.15 additional seconds. With Thm. 1 we can conclude that the STL formula φ holds for all possible evolutions of the system.

7 Conclusion

We introduce a model checking technique for STL formulas, which leverages reachable sets computed by reachability analysis tools. This is done by: (i) Defining the Reachset Temporal Logic (RTL), whose semantics is defined on reachable sets instead of traces, on which previous temporal logics are defined (e.g. STL); (ii) introducing a sound and complete transformation from sampled time STL to

RTL for finitely represented reach sequences; (iii) introducing a rewriting scheme for general STL formula to sampled time STL formula; and (iv) introducing a model checking method for RTL formulas obtained by the transformation. The approach is especially useful for non-deterministic models that naturally exhibit uncountably many traces due to necessary abstractions from original dynamics. Our model checking technique is independent of the way reach sequences are obtained and represented. Therefore, all reachability analysis tools can benefit from our approach by extending their reasoning from non-temporal (safety) properties to temporal properties. This is demonstrated by an example from automated driving, where the online verification of the absence of collisions is extended to online verification of temporal properties.

Future work could intensify the interconnection of the reachability analysis and the verification part to develop the method further. Additionally, the semantics of RTL can be extended in the sense of robust semantics as used by Metric Temporal Logic [11].

ACKNOWLEDGMENT The authors gratefully acknowledge financial support by the European Commission project UnCoVerCPS under grant number 643921.

References

1. S. N. Ahmadyan, J. A. Kumar, and S. Vasudevan. Runtime verification of nonlinear analog circuits using incremental time-augmented RRT algorithm. In *Proc. of Design, Test & Automation in Europe*, 2013.
2. M. Althoff. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151, 2015.
3. M. Althoff and J. M. Dolan. Reachability computation of low-order models for the safety verification of high-order road vehicle models. In *American Control Conference*, pages 3559–3566. IEEE, 2012.
4. R. Alur, C. Courcoubetis, and D. L. Dill. Model-checking for real-time systems. In *Proc. of 5th Symposium on Logic in Computer Science*, pages 414–425, 1990.
5. R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.
6. E. Asarin et al. Recent progress in continuous and hybrid reachability analysis. In *Conference on Computer Aided Control Systems Design*, pages 1582–1587, 2006.
7. C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
8. D. Bresolin. HyLTL: a temporal logic for model checking hybrid systems. In *Proceedings Third International Workshop on Hybrid Autonomous Systems, HAS*, pages 73–84, 2013.
9. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 2000.
10. P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok. C2E2: A verification tool for stateflow models. In *TACAS*, pages 68–82. 2015.
11. G. E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262 – 4291, 2009.
12. G. Frehse et al. SpaceEx: scalable verification of hybrid systems. In *Computer Aided Verification*, pages 379–395, 2011.

13. H. Guéguen, M. Lefebvre, J. Zaytoon, and O. Nasri. Safety verification and reachability analysis for hybrid systems. *Annual Reviews in Control*, 33(1):25–36, 2009.
14. E. A. Lee. CPS foundations. In *Design Automation Conf.*, pages 737–742, 2010.
15. I. Lee, S. Kannan, M. Kim, O. Sokolsky, and M. Viswanathan. Runtime assurance based on formal specifications. In *Proc. of the IntemationaI Conference on Parallel and Distributed Processing Techniques and Applications*, 1999.
16. O. Maler, D. Nickovic, and A. Pnueli. Checking temporal properties of discrete, timed and continuous behaviors. volume 4800 of *Lecture Notes in Computer Science*, pages 475–505. Springer, 2008.
17. O. Maler and D. Ničković. Monitoring properties of analog and mixed-signal circuits. *Journal on Software Tools for Technology Transfer*, 15:247–268, 2013.
18. S. Mitra, T. Wongpiromsarn, and R. M. Murray. Verifying cyber-physical interactions in safety-critical systems. *IEEE Security and Privacy*, 11(4):28–37, 2013.
19. A. Pinto, A. Sangiovanni-Vincentelli, L. P. Carloni, and R. Passerone. Interchange formats for hybrid systems: Review and proposal. In *HSCC*, LNCS 3414, pages 526–541. Springer, 2005.
20. A. Platzer and E. M. Clarke. The image computation problem in hybrid systems model checking. In *HSCC*, LNCS 4416, pages 473–486. Springer, 2007.
21. R. Poovendran. Cyberphysical systems: Close encounters between two parallel worlds. *Proceedings of the IEEE*, 98(8):1363–1366, 2010.
22. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: The next computing revolution. In *Design Automation Conference*, pages 731–736, 2010.
23. M. U. Sanwal and O. Hasan. Formal verification of cyber-physical systems: Coping with continuous elements. In *Proc. of the 16th International Conference on Computational Science and Its Applications*, pages 358–371, 2013.
24. G. Sauter, H. Dierks, M. Fränzle, and M. R. Hansen. Lightweight hybrid model checking facilitating online prediction of temporal properties. In *Proceedings of the 21st Nordic Workshop on Programming Theory, NWPT09*, pages 20–22, 2009.
25. G. V. Smirnov. *Introduction to the Theory of Differential Inclusions*. American Mathematical Society, 2002.
26. L. Tan, J. Kim, O. Sokolsky, and I. Lee. Model-based testing and monitoring for hybrid embedded systems. In *Model-based testing and monitoring for hybrid embedded systems*, pages 487–492, 2004.
27. Z. Wang, M. H. Zaki, and S. Tahar. Statistical runtime verification of analog and mixed signal designs. In *Conference on Signals, Circuits and Systems*, 2009.