Towards a Set-based Signal Temporal Logic

Julien Alexandre dit Sandretto and Alexandre Chapoutot
U2IS, ENSTA Paris, Institut Polytechnique de Paris, Palaiseau, France
{julien.alexandre-dit-sandretto,alexandre.chapoutot}@ensta-paris.fr

Pierre-Loïc Garoche
ENAC, Université de Toulouse, Toulouse, France
pierre-loic.garoche@enac.fr

December 21, 2020

Abstract

Abstract model-checking methods are efficient to prove properties on systems with infinite state-space dimension. When considering continuous-time dynamical systems, adequate temporal logic has to be used to express specifications of systems. In this short paper, we present our current work on designing a subset of signal temporal logic (STL) amenable to bounded abstract model-checking methods. The proposed approach uses set-based operations as atomic proposition of STL. Set-based approach is more suitable to applied abstract model-checking methods as it allows to consider set of system trajectories instead of only one trajectory of the system. Interval analysis methods are used to give a computable counter-part version of this set-based STL and to compute sets of trajectories of dynamical systems. The correctness of the presented framework is asserted using an abstract interpretation framework.

Keywords: Abstract Interpretation theory, Interval Analysis, Validated Numerical Integration

Consider the study of cyber-physical systems modeled as dynamical systems. At early stage of the design, system-under-study is modeled either by differential equations or by a discrete-time dynamical system. Current means of studying these kinds of systems include the following approaches: (1) classical control system analyses, such as frequency domain analysis, unfortunately these approaches are restricted to limited class of systems such as linear ones. (2) reachability analysis which ensures that all reachable states belong to a safe set; either as a unique set denoting all states, or as a flow-pipe of time-dependent reachable states, as performed by hybrid system analysis tool such as Flow* or SpaceEx or DynIbex; (3) numerical simulation which evaluates of the system's behaviors for a set of given input vectors. These input vectors can be randomly

generated, as in the Monte Carlo approach, providing a reasonable yet non exhaustive, idea of the system's behaviors. Reachability analysis is considered in this article.

A key difficulty when it comes to formal verification is to be able to formally specify the requirements. Historically, in theoretical Computer Science, the set of functional properties of a system could be partitioned between safety properties, properties expressed over states, and liveness properties, properties expressed over traces. Safety properties are easier to express since they can usually be characterized by a safe or unsafe set of states. The verification amounts to show that the set of reachable states belong to the first set or has no shared states with the latter. Liveness properties are richer since they can express a notion of sequence of events, even in the case of infinite behaviors. These algorithms are complex and not yet well developed when it comes to such numerical systems. However checking a liveness property on a bounded horizon can be proved to expressible as a safety property. This enables the evaluation a limited temporal properties to finite horizon systems.

Last, recent tools, such as FRET [1] – the Formal Requirement Elicitaton Toolbox –, help the user to design its specification, providing an underlying formal logic based on future-time or past-time metric LTL. While existing tools, as S-TaLiRo [2], allow to observe the validity of such requirements for a given trace or trajectory, very few methods exist [3, 4] for bounded-horizon exhaustive analysis of such requirements for dynamical systems, either ODE-based or discrete time.

This paper describes our on-going work. We present a way to formally verify bounded horizon properties of an STL formula for dynamical system. Our goal is to lift the approach of STL observers proposed in [5] to sets of trajectories, using the abstract interpretation framework. Our proposal intends to be more generic than existing works [4, 3] while providing computable abstractions.

1 Setting 1: System Requirements with STL

Temporal logic [6] is an important formalism when it comes to specify the behavior of systems and to prove their functional properties. It was first introduced in the context of discrete-time systems – where traces are denoted as sequences of values – and has since been extended to deal with continuous-time or hybrid systems with a dense-time representation [5, 7] such as Signal temporal logic (STL). While STL is strictly less expressive than LTL since it considers finite time properties, it relies on similar temporal operators with bounded time horizon. As an example the *until* $\mathtt{U}$ operator has to be associated to a time interval within which it should hold $\mathtt{U}_{[t_1,t_2]}$. STL has become a well accepted formalism to design continuous-time system requirements [8].

We recall the syntax of STL formula [9]: $\varphi ::= \text{true} \mid \sigma \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \mathtt{U}_{[a,b]} \varphi_2 \mid \varphi_1 \mathtt{R}_{[a,b]} \varphi_2$. The time interval $[a, b]$ is such that $a < b$ and a, b in $\mathbb{Q}$. σ is an atomic proposition of the form $g(x) \leqslant 0$. As usual, constant false can be defined as false $\equiv \neg$true and $\varphi_1 \wedge \varphi_2 \equiv \neg(\neg\varphi_1 \vee \neg\varphi_2)$. Other temporal operators

$$
\begin{array}{lcl}
y(t), t_i \models \text{true} & & \\
y(t), t_i \models \sigma & \equiv & \sigma\left(y(t_i)\right) = \text{true} \\
y(t), t_i \models \neg\varphi & \equiv & y(t) \not\models \varphi \\
y(t), t_i \models \varphi_1 \vee \varphi_2 & \equiv & y(t), t_i \models \varphi_1 \vee \ y(t), t_i \models \varphi_2 \\
y(t), t_i \models \varphi_1 \mathtt{U}_{[a,b]} \varphi_2 & \equiv & \exists t' \in t_i + [a,b], y(t), t' \models \varphi_2 \wedge \forall t'' \in [t_i, t'], y(t), t'' \models \varphi_1 \\
y(t), t_i \models \varphi_1 \mathtt{R}_{[a,b]} \varphi_2 & \equiv & \forall t' \in t_i + [a,b], y(t), t' \models \varphi_2 \vee \exists t'' \in [t_i, t'], y(t), t'' \models \varphi_1
\end{array}
$$

Figure 1: Overview of the semantics of a Signal Temporal Logic (STL) formula.

as *always* $\Box$, *eventually* $\Diamond$ can be defined such that $\Diamond_{[a,b]}\varphi \equiv \text{true}\ \mathtt{U}_{[a,b]}\varphi$ and $\Box_{[a,b]}\varphi \equiv \neg\Diamond_{[a,b]}\neg\varphi$. Assuming that a system trajectory is denoted by $y(t)$, the semantics of a STL formula φ is defined inductively in Figure 1. Current tools for requirement elicitation or property checking based on STL formula mainly work by considering one trajectory at a time, acting as a monitor. In order to cover a wide range of system behaviors, one need to generate a large sample of trajectories, typically using simulation tools such as Simulink [2]. For example, the algorithm for monitoring STL formulas [5] provides an efficient checking algorithm to assert that a trajectory is a model of an STL formula.

2 Setting 2: Reachability Analysis of Dynamical Systems

We consider continuous-time systems defined as the solution of *initial value problem* of *ordinary differential equations* (IVP-ODE) defined by $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$ with $\mathbf{y}(0) = \mathbf{y}_0$. The non-linear function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is the dynamic of the system, with n the dimension of the state vector $\mathbf{y}$. Function $\mathbf{f}$ is assumed to be smooth enough to ensure existence and uniqueness of the solution. The exact solution $\mathbf{y}(t; \mathbf{y}_0)$ is usually not computable except for special cases, *e.g.*, linear systems. In consequence, numerical integration methods are used to produce an approximated solution. In abstract model-checking context, *reachability methods* [10, 11] or *guaranteed numerical integration methods* [12, 13] are used to produce an over-approximation of the set of trajectories $\mathcal{Y}(t; \mathcal{Y}_0) = \{\mathbf{y}(t; \mathbf{y}_0) : \forall \mathbf{y}_0 \in \mathcal{Y}_0\}$ starting from a set of initial conditions $\mathcal{Y}_0$. Note that in some cases, bounded parameters are also considered in the dynamic f which also induce a set of possible trajectories of the system.

Guaranteed numerical integration methods are interval counterpart of numerical integration methods. A validated numerical integration of a differential equation consists in a discretization of time, such that $t_0 \leqslant \cdots \leqslant t_{\text{end}}$, and a computation of enclosures of the set of states of the system $\mathbf{y}_0$, $\ldots$, $\mathbf{y}_{\text{end}}$, by the help of a guaranteed integration scheme. In details, it is made of (i) an integration method $\Phi(f, \mathbf{y}_j, t_j, h)$, starting from an initial value $\mathbf{y}_j$ at time t_j and a finite time horizon h (the step-size), producing an approximation $\mathbf{y}_{j+1}$ at time $t_{j+1} = t_j + h$, of the exact solution $\mathbf{y}(t_{j+1}; \mathbf{y}_j)$, *i.e.*, $\mathbf{y}(t_{j+1}; \mathbf{y}_j) \approx \Phi(f, \mathbf{y}_j, t_j, h)$; and (ii) a truncation error function $\text{lte}_\Phi(f, \mathbf{y}_j, t_j, h)$,

such that $\mathbf{y}(t_{j+1}; \mathbf{y}_j) = \Phi(f, \mathbf{y}_j, t_j, h) + \text{lte}_\Phi(f, \mathbf{y}_j, t_j, h)$. A validated numerical integration method is a two step method starting at time t_j and for which i) it computes an enclosure $[\tilde{\mathbf{y}}_j]$ of the solution of IVP-ODE over the time interval $[t_j, t_{j+1}]$ to bound $\text{lte}_\Phi(f, \mathbf{y}_j, t_j, h)$; ii) it computes a tight enclosure of the solution of IVP-ODE for the particular time instant t_{j+1}. There are many methods for these two steps among Taylor series and Runge-Kutta methods see [12, 13] and the references therein for more details. Usually guaranteed numerical integration methods are based on intervals or zonotopes abstractions.

Remark 1. *In context of abstract model checking where reachability methods can be used, checking the validity of atomic proposition of the form* $\mathbf{y}(t_i; \mathbf{y}_0) = 0$ *for a particular time instant* t_i *is not relevant as* $\mathcal{Y}(t, \mathcal{Y}_0)$ *is over-approximated by a thick function even if* $\mathcal{Y}_0$ *is a singleton. That is one of the main motivations of defining a set-based temporal logic formula, i.e., temporal logic formula based on set-based atomic propositions.*

3 Setting 3: Abstract Interpretation of Temporal Logic

While Abstract Interpretation theory has been mainly known for the static analysis of safety properties for imperative programs, mainly using numerical abstract domains such as interval or convex polyhedra, the theory is more general and could be applied to a wide range of semantics analyses.

An interesting line of work by Cousot and Cousot [14] and Ranzato *et al.* [15] characterizes temporal logics (μ-calculus and LTL, respectively) within this framework. The semantics of a temporal logic formula is described by its model, *i.e.*, the set of traces that verify the formula. This approach is developed by induction on the operators of the logic and, as an example, the negation operator computes the complement of the considered set of traces with respect to the full set of traces. We recall the basic definitions of those papers. $[\![\phi]\!]$ denotes the semantics of formula ϕ. For an atomic proposition p, $[\![p]\!] = \{\tau \in \text{Trace}(\Sigma) \mid \ell \in \ell(\tau(0))\}$ where Σ is the set of states and $\ell(\tau(0))$ the set of atomic propositions associated to the first state of trace π. $[\![\neg p]\!] = \text{Trace}(\Sigma) \backslash [\![p]\!]$, $[\![\phi_1 \vee \phi_2]\!] = [\![\phi_1]\!] \cup [\![\phi_2]\!]$, $[\![\phi_1 \mathcal{U}_{[a,b]} \phi_2]\!] = \{\tau \in \text{Trace}(\Sigma) \mid \exists k \in \mathbb{N}, \tau^k \in [\![\phi_2]\!]$ and $\forall j \in [0, k), \tau^j \in [\![\phi_1]\!]\}$ where σ^k is the suffix of trace σ that begins at index k.

These framework provides interesting connections between logics, model-checking and static analysis, but they lack concrete implementations. In other words, they provide a sound and strong theoretical framework able to compare the respective expressiveness of different formalisms but are not well suited for effective computations. An obvious limitation is the need to rely on a Boolean lattice with a $\neg$ completion operator which is rarely available in most numerical domains. Despite this lack of practical applications, we believe that is the proper framework to both express and apply set-based verification of STL formulas.

4 Motivation for Set-based STL Reasoning

We propose to restrict STL formulas to a logic where atomic propositions are based on set operations. Set-based predicates are appropriate when considering dynamical systems and their specifications. First, most specification are subject to uncertainties and typically express as distances or bounds over norms, which naturally translate to sets. Second, the model dynamics as an ODE is an approximation of the actual physical dynamics of the system. A sound way to address this is to provided sets for parameters, denoting a family of systems at once.

In terms of verification, we recall that the basic principle being model-checking is to check that the models of the systems belong to the models of the specification, which is also a set inclusion relation. Regarding precision of the analysis, most equality and inequality constraints are hardly representable in abstract domains. For example when computing with intervals, it is difficult to have a precise $\neg$ operator in the abstract. This lack of precision to represent equalities and their negation is a good motivation to restrict their uses.

5 Set-based atomic propositions

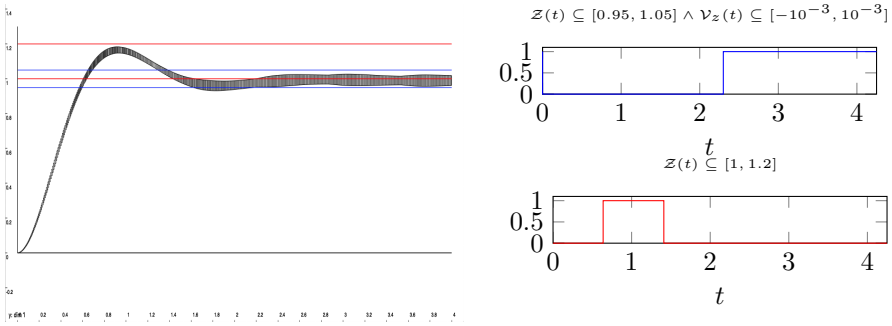
We propose to restrict STL to formulas with two kinds of atomic predicates. In the original definition, atomic predicates over system states are arbitrary real-valued functions $g(x) \leq 0$, which could be used to denote inequality and therefore equality. We rather propose to restrict atomic predicates to the following ones: $\sigma ::= \mathbf{g}(\mathcal{Y}) \subseteq \mathcal{A} \mid \mathbf{g}(\mathcal{Y}) \cap_{=\emptyset} \mathcal{A}$, with $A \subseteq \mathbb{R}^n$ a constant set and $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^n$ a non-linear function. The lifting of functions $\mathbf{g}$ over set $\mathcal{X}$ is such that $\mathbf{g}(\mathcal{X}) = \{\mathbf{g}(\mathbf{x}) : \forall \mathbf{x} \in \mathcal{X}\}$.

While, with these two predicates, we cannot exactly characterize the negation of each of them, one can remark that $\mathbf{g}(\mathcal{Y}) \cap_{=\emptyset} \mathcal{A} \implies \neg(\mathbf{g}(\mathcal{Y}) \subseteq \mathcal{A})$ and, respectively, $\mathbf{g}(\mathcal{Y}) \subseteq \mathcal{A} \implies \neg(\mathbf{g}(\mathcal{Y}) \cap_{=\emptyset} \mathcal{A})$. We can then use them soundly when over-approximating sets with an abstract negation or complement operator $\neg$, as for example in [16, 17].

Remark 2. *Set-based atomic proposition subsumes inequalities such as constraints of the form $g(\mathbf{x}) \geqslant 0$ can be encoded as $g(\mathbf{x}) \subset [0, +\infty[$.*

6 Set-based STL monitors

Previous work [4, 3] have also considered STL formula for the verification of continuous or hybrid dynamical systems. Indeed, [4] introduced a new Reachset Temporal Logic (RTL) in order to express temporal properties on set of trajectories. This RTL relies on the computation of reachable tube, discretized in time, and provides a convenient way to express RTL into conjunctive normal form of propositional formula. In [3], a direct interval-based extension of the STL monitoring algorithm proposed by [18] is defined. This work follows



(a) Drone's altitude with uncertainty

(b) Boolean abstraction

Figure 2: Altitude control example

the same philosophy, proving that a set of trajectories fulfills a given temporal specification. Nonetheless, the embedding in abstract interpretation theory can provide a new look on these previous works as a way to merge them into a common generic framework and provide insights to define new abstract domains (relational or not) for reachable tubes or STL formula.

In particular, following [18], monitoring STL formula is defined inductively on the syntax of the formula as most common static analyses by abstract interpretation. Moreover, this monitoring formula is based on a Boolean abstraction of continuous trajectories in order to use a temporal logic monitoring algorithm.

As an illustration of the benefit of set-based STL, a simple altitude control algorithm of a drone is defined by: $\dot{z} = v_z \quad \text{and} \quad \dot{v_z} = \frac{K_p(1-z) - K_d v_z + 3.2373}{[0.33, 0.34]} - g$ where g is the g-force equals to $9.81 m.s^{-2}$. Note that we consider an unknown but bounded mass of a drone equals to $[0.33, 0.34]$kg. The gains of the controller are $K_p = 5$ and $K_d = 1.25$. The goal of the mission is to put the control to pose $(0.0, 0.0)$ to pose $(1.0, 0.0)$ and hovers at this last pose. A simulation result is given in Figure 2a.

A property of interest is given by the set-based STL formula

$$\Box_{[0,3]} \left(\mathcal{Z}(t) \subseteq [1, 1.2] \implies \left(\Diamond_{[0.5,1]} \mathcal{Z}(t) \subseteq [0.95, 1.05] \wedge \mathcal{V}_z(t) \subseteq [-10^{-3}, 10^{-3}] \right) \right).$$

A first idea to define monitoring algorithm of set-based STL formula is to adapt the monitoring algorithm of [3] to use set-based atomic propositions. It should be straightforward to adapt it to our formalism since the Boolean abstraction of set-based predicate (cf Figure 2b) can be embedded in algorithm defined in [3].

7 Perspectives

Manipulating abstractions and reasoning about them could easily leads to false statements. We believe that this abstract interpretation-based reasoning over our restriction of STL formulas could lead to both sound and precise analyses. In

addition, we identified a need for a tool able to reason exhaustively on computed reachable tubes.

We are currently designing a modular instantiation of the temporal logic abstract interpretation frameworks mentioned above, able to effectively compute an over-approximation of both the models of a formula and the models of its negation. Our approach is to first design a simple abstraction that propagates abstract Boolean in $\{\text{True}, \text{False}, \top\}$ along Boolean operators (as in [3]), then to propagate models, and, finally, design time-dependent relational abstractions.

References

[1] Dimitra Giannakopoulou et al. "Formal Requirements Elicitation with FRET". In: *Joint Proceedings of REFSQ-2020 Workshops co-located with the 26th International Conference on Requirements Engineering: Foundation for Software Quality*. Vol. 2584. CEUR Workshop Proceedings. CEUR-WS.org, 2020.

[2] Georgios Fainekos, Bardh Hoxha, and Sriram Sankaranarayanan. "Robustness of Specifications and Its Applications to Falsification, Parameter Mining, and Runtime Monitoring with S-TaLiRo". In: *Proc. of Int. Conf. on Runtime Verification*. Vol. 11757. LNCS. Springer, 2019, pp. 27–47.

[3] Daisuke Ishii, Naoki Yonezaki, and Alexandre Goldsztejn. "Monitoring Temporal Properties Using Interval Analysis". In: *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* 99-A.2 (2016), pp. 442–453.

[4] Hendrik Roehm et al. "STL Model Checking of Continuous and Hybrid Systems". In: *Automated Technology for Verification and Analysis*. Vol. 9938. LNCS. 2016, pp. 412–427.

[5] Oded Maler, Dejan Nickovic, and Amir Pnueli. "Checking Temporal Properties of Discrete, Timed and Continuous Behaviors". In: *Pillars of Computer Science, Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*. Vol. 4800. LNCS. Springer, 2008, pp. 475–505.

[6] Zohar Manna and Amir Pnueli. "A Hierarchy of Temporal Properties". In: *Proc. of the ACM Symposium on Principles of Distributed Computing*. 1990, pp. 377–410.

[7] Alexandre Donzé and Oded Maler. "Robust Satisfaction of Temporal Logic over Real-Valued Signals". In: *Formal Modeling and Analysis of Timed Systems*. Vol. 6246. LNCS. Springer, 2010, pp. 92–106.

[8] Ezio Bartocci et al. "Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications". In: *Lectures on Runtime Verification - Introductory and Advanced Topics*. Vol. 10457. LNCS. Springer, 2018, pp. 135–175.

[9] Ron Koymans. "Specifying real-time properties with metric temporal logic". In: *Real-time systems* 2.4 (1990), pp. 255–299.

[10] Goran Frehse et al. “SpaceEx: Scalable Verification of Hybrid Systems”. In: *Proc. of Computer Aided Verification*. Vol. 6806. LNCS. Springer. 2011, pp. 379–395.

[11] Xin Chen, Erika Abraham, and Sriram Sankaranarayanan. “Taylor Model Flowpipe Construction for Non-linear Hybrid Systems”. In: *Proc. of IEEE Real-Time Systems Symposium*. 2012, pp. 183–192.

[12] Nedialko S. Nedialkov, Kenneth Jackson, and Georges Corliss. “Validated solutions of initial value problems for ordinary differential equations”. In: *Applied Mathematics and Computation* 105.1 (1999), pp. 21–68.

[13] Julien Alexandre dit Sandretto and Alexandre Chapoutot. “Validated Explicit and Implicit Runge-Kutta Methods”. In: *Reliable Computing* 22 (2016).

[14] P. Cousot and R. Cousot. “Temporal Abstract Interpretation”. In: *Conference Record of the Twentyseventh Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. Jan. 2000, pp. 12–25.

[15] Francesco Ranzato and Francesco Tapparo. “An Abstract Interpretation Perspective on Linear vs. Branching Time”. In: *Programming Languages and Systems*. Springer Berlin Heidelberg, 2005, pp. 69–85. ISBN: 978-3-540-32247-4.

[16] Julien Alexandre Dit Sandretto, Alexandre Chapoutot, and Olivier Mullier. “Constraint-Based Framework for Reasoning with Differential Equations”. In: *Cyber-Physical Systems Security*. Ed. by Çetin Kaya Koç. Springer, 2018, pp. 23–41.

[17] Julien Alexandre Dit Sandretto and Alexandre Chapoutot. “Logical Differential Constraints Based on Interval Boolean Tests”. In: *Fuzzy Techniques: Theory and Applications - Proceedings of the 2019 Joint World Congress of the International Fuzzy Systems Association and the Annual Conference of the North American Fuzzy Information*. Vol. 1000. Advances in Intelligent Systems and Computing. Springer, 2019, pp. 788–792.

[18] Oded Maler and Dejan Nickovic. “Monitoring Temporal Properties of Continuous Signals”. In: *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Vol. 3253. LNCS. Springer, 2004, pp. 152–166.