The First-Order Logic of Signals

Alexey Bakhirkin*, Thomas Ferrère†, Thomas A. Henzinger†, and Dejan Ničković‡
*Univ. Grenoble Alpes, CNRS, Grenoble INP, VERIMAG, 38000 Grenoble, France
†IST Austria
‡AIT Austrian Institute of Technology

***Abstract*—Formalizing properties of systems with continuous dynamics is a challenging task. In this paper, we propose a formal framework for specifying and monitoring rich temporal properties of real-valued signals. We introduce *signal first-order logic* (SFO) as a specification language that combines first-order logic with linear-real arithmetic and unary function symbols interpreted as piecewise-linear signals. We first show that while the satisfiability problem for SFO is undecidable, its membership and monitoring problems are decidable. We develop an offline monitoring procedure for SFO that has polynomial complexity in the size of the input trace and the specification, for a fixed number of quantifiers and function symbols. We show that the algorithm has computation time linear in the size of the input trace for the important fragment of *bounded-response* specifications interpreted over input traces with finite variability. We can use our results to extend *signal temporal logic* with first-order quantifiers over *time* and *value* parameters, while preserving its efficient monitoring. We finally demonstrate the practical appeal of our logic through a case study in the micro-electronics domain.**

I. Introduction

Cyber-physical systems (CPS) are networks of computational and physical elements that interact with their environment via sensors and actuators. CPS applications are today everywhere around us, deeply impacting our society. Medical devices that enable adaptive therapy based on the continuous measurement of patients' parameters, smart buildings that optimize heating strategy to the users' habits, and industrial plants that collaborate to minimize production costs are just a few examples of CPS applications. The inherent complexity of CPS and the intricate interactions with their physical environment makes their verification a challenging task.

Property-based monitoring is a light-weight verification technique that combines formal specifications with the analysis of individual system behaviors. It is a pragmatic, yet rigorous approach to reason about complex systems. Central to the property-based monitoring approach is the specification language that allows to precisely describe the intended system behaviors. *Signal temporal logic* (STL) [1] is a specification formalism for expressing real-time temporal properties of real-valued signals. It extends *linear-time temporal logic* (LTL) [2] with linear inequalities over signal values and with real-time bounds associated to temporal operators [3]. The monitoring of STL enjoys an efficient procedure [1], with time complexity linear in the size of the signals. Several research directions around STL have been explored in the recent years: the extension of the language with quantitative semantics and robust monitoring [4]–[6], parameter estimation [7]–[9], specification mining [10]–[12] and diagnosis [13]. STL specifications were used to analyze complex systems in the automotive, biomedical, robotics, analog system design and education application domains (see the survey [14] for more details).

This research was supported in part by the Austrian Science Fund (FWF) under grants S11402-N23 (RiSE/SHiNE) and Z211-N23 (Wittgenstein Award), by the EU ICT COST Action IC1402 (ARVI), and by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement nr. 306595 "STATOR".

The success of STL in recent years is mainly due to the simple monitoring procedures and the ability to capture many asynchronous sequential behaviors of real-valued signals observed in control applications. The *bounded stabilization* requirement is a typical example of a temporal specification than can be naturally expressed in STL. Given a Boolean signal b and a real-valued signal f, the bounded stabilization requirement is formulated as follows: "Whenever the control signal b is on its rising edge, the absolute value of f must go inside the interval $[4.5, 5.5]$ within 10 time units and continuously remain within that same interval for at least 8 time units". This informal requirement, illustrated in Figure 1, is expressed as the following STL specification.

$$\Box(\uparrow b \rightarrow \Diamond_{[0,10]} \Box_{[0,8]}(|f - 5| \leq 0.5)$$

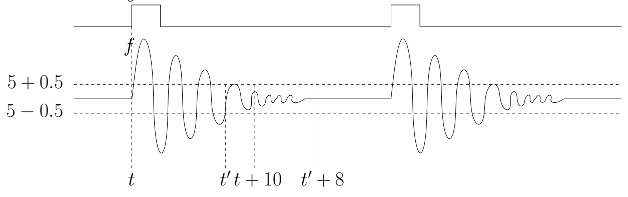

Fig. 1. Bounded stabilization with fixed threshold $r = 5$.

Nevertheless, the expressiveness of STL has some limitations. For instance, the bounded stabilization property requires apriori knowledge of thresholds and timing bounds. In real-life applications, these bounds may not be known in advance and may even change dynamically during system execution. A more general formulation of the bounded stabilization property requires the signal f to stabilize around *some* value of r, which can vary during the execution of the system as shown in Figure 2. This more general specification cannot be expressed in STL. Intuitively, the general bounded stabilization requirement could be formulated in STL extended with quantification over the threshold value as follows.

$$\Box(\uparrow b \rightarrow \exists r : \Diamond_{[0,10]} \Box_{[0,8]}(|f - r| \leq 0.5)$$

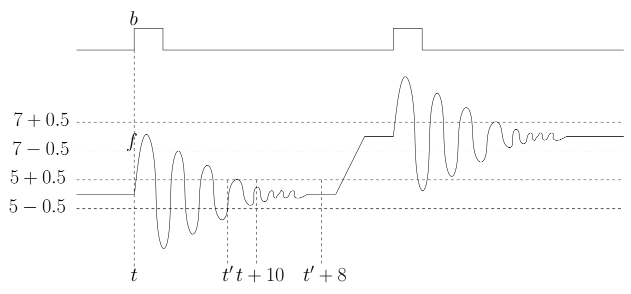

Fig. 2. Bounded stabilization without fixed threshold r.

When introducing variable *time* parameters, temporal logic operators become unnecessary because the quantification over time implicit in temporal operators is then made explicit in syntax. In particular, by using first-order quantification, the modality $\Diamond_{[s,s]}$, where s is a free variable, can express all other forms of temporal operators.

Motivated by the lack of a clean specification language that is sufficiently expressive to capture rich temporal properties, we propose *signal first-order logic* (SFO) as a powerful declarative formalism for expressing real-valued signal requirements. SFO combines first-order logic with linear real arithmetic and uninterpreted unary function symbols, which represent real-valued signals over time. The syntax of SFO allows quantification over time and value variables. This constitutes a general semantic framework with a syntax that can be used to directly specify rich temporal requirements and that provides a clean theoretical basis from which other specification formalisms can be derived. *Quantified signal temporal logic* (QSTL) is one such formalism, which, as we will demonstrate, is equally expressive as SFO.

SFO is an expressive formalism with undecidable satisfiability. In contrast, we show that the membership problem for SFO relative to piecewise-linear signals is decidable in time that is doubly exponential in the number of quantifiers and function symbols appearing in the specification, which is typically small in practice. A problem related to membership is known as *monitoring*. The monitoring problem takes as input a signal trace w and a formula φ with one free (time) variable and asks for the Boolean signal u such that $u(t) = 1$ iff φ is satisfied by w at time t. We propose an efficient monitoring algorithm based on a polyhedral representation of the set of free variables interpretations that satisfy a formula for a given input signal. We then define a class of *bounded-response* SFO specifications, over which we show that our algorithm performs in time linear in the length of the input signal. Informally, a bounded-response property φ can be associated with a horizon h such that the satisfaction of $\varphi[t]$ by a trace w does not depend on values of w at any t' with $|t' - t| > h$. Finally, we illustrate the expressiveness of SFO specifications on a case study from the micro-electronics application domain.

II. Examples

In this section, we introduce SFO with several examples. We first formulate the bounded stabilization property from Section I by using the SFO syntax, and then introduce *control* and *rise time* specifications. Function symbols f, g range over real-valued signals and b ranges over Boolean signals. Value variable r and time variables c, c', d, d', e, e' and t range over the reals.

Example 1 (Bounded stabilization). *Before formulating the main stabilization formula, we define what is a rising edge* $\uparrow b$ *of a Boolean signal* b *at time* t.

$$\uparrow b[t] \equiv b(t) = 1 \wedge \exists c \in (0, \infty) : \forall c' \in (0, c) : b(t - c') = 0$$

We now express the bounded stabilization property in SFO as follows.

$$\varphi_1 \equiv \uparrow b[t] \rightarrow \exists r : \exists c \in [0, 10] : \forall d \in [0, 8] : \\ |f(t + c + d) - r| \leq 0.5$$

Example 2 (Control). *Consider the following property.*

$$\varphi_2 \equiv \forall r : (\forall c \in [0, 10] : |f(t + c) - r| \leq 1) \\ \rightarrow (\forall d \in [5, 10] : |g(t + d) - r| \leq 2)$$

Formula φ_2 *requires that if* f *is stable (stays within* 1.0 *of some value* r *for* 10 *time units), then* g *stabilizes around the same value (stays within* 2.0 *of* r *within* 5 *time units).*

Example 3 (Rise time). *Consider the following property.*

$$\varphi_3 \equiv \forall c \in (0, 10) : \begin{pmatrix} f(t) = 1 \wedge f(t + c) = 2 \\ \wedge\ \forall c' \in (0, c) : 1 < f(t + c') < 2 \end{pmatrix} \\ \rightarrow \exists d \in (0, 10) : 9c \leq 10d \leq 11c \wedge \exists e \in (0, 20) : \\ \begin{pmatrix} \forall e' \in (0, e) : g(t + e') < 1 \\ \wedge\ g(t + e') = 1 \wedge g(t + e + d) = 2 \\ \wedge\ \forall d' \in (0, d) : 1 < g(t + e + d') < 2 \end{pmatrix}$$

Formula φ_3 *expresses the fact that if signal* f *has a positive edge (from* 1.0 *to* 2.0 *in less than* 10 *time units) then signal* g *subsequently (in less than* 20 *time units) has a positive edge whose rise time is within* 10% *of that of* f.

III. Signal First-Order Logic

In this section we define *signal first-order logic* (SFO), and study the decidability of its satisfiability and membership problems.

A. Definitions

A *time domain* $\mathbb{T}$ is a non-singular interval of $\mathbb{R}$, and a (*real-valued*) *signal* is a function $\mathbb{T} \rightarrow \mathbb{R}$. Let $F = \{f_1, f_2, \ldots\}$ be a set of function symbols. We call *Boolean* a signal with value in $\{0, 1\}$. A *trace* w is an interpretation of functions symbols $f \in F$ as signals, denoted $[\![f]\!]_w$. Let $X = \{x_1, x_2, \ldots\}$ be a set of variables. A *valuation* v is an interpretation of variables $x \in X$ as real numbers, denoted $[\![x]\!]_v$.

Signal first-order logic is obtained from variables in X and the signature $\langle f_1, f_2, \ldots, \mathbb{Z}, -, +, < \rangle$ where $f_1, f_2, \ldots \in F$

are uninterpreted unary function symbols, $\mathbb{Z}$ are integer constants, and $-, +, <$ are the usual arithmetic binary functions and order relation. For practical purposes, we partition the set of variables X into $T \cup R$ where $t \in T$ are called *time* variables and $r \in R$ are called *value* variables.

Definition 4 (SFO syntax)**.** *Terms τ and ρ of SFO are given by the grammar*

$$\tau ::= t \mid n \mid \tau_1 - \tau_2 \mid \tau_1 + \tau_2$$
$$\rho ::= r \mid f(\tau) \mid n \mid \rho_1 - \rho_2 \mid \rho_1 + \rho_2$$

where $n \in \mathbb{Z}$, $r \in R$, and $t \in T$. A term θ is either a time *term τ or a* value *term ρ as above.*

Formulas φ of SFO are given by the grammar

$$\varphi ::= \theta_1 < \theta_2 \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \exists r : \varphi \mid \exists t \in I : \varphi$$

where $n \in \mathbb{Z}$, θ_1, θ_2 are terms, and I are real intervals with bounds in $\mathbb{Z} \cup \{\pm\infty\}$.

We use standard abbreviations such as $\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$, $\varphi_1 \wedge \varphi_2 \equiv \neg(\neg\varphi_1 \vee \neg\varphi_2)$, and $\forall r : \varphi \equiv \neg\exists r : \neg\varphi$. We also write $\forall t \in I : \varphi$ as a shorthand for $\neg\exists t \in I : \neg\varphi$.

Definition 5 (SFO semantics)**.** *The value of a term θ over a trace w and valuation v, denoted $[\![\theta]\!]_{w,v}$, extends the interpretation of function symbols and variables given by w inductively as follows: $[\![n]\!]_{w,v} = n$ for all $n \in \mathbb{Z}$, $[\![\theta_1 - \theta_2]\!]_{w,v} = [\![\theta_1]\!]_{w,v} - [\![\theta_2]\!]_{w,v}$ and $[\![\theta_1 + \theta_2]\!]_{w,v} = [\![\theta_1]\!]_{w,v} + [\![\theta_2]\!]_{w,v}$. The satisfaction of a formula φ over a trace w and valuation v, denoted $(w, v) \models \varphi$, is inductively defined by letting $(w, v) \models \theta_1 < \theta_2$ iff $[\![\theta_1]\!]_{w,v} < [\![\theta_2]\!]_{w,v}$, $(w, v) \models \exists r : \varphi$ iff $(w, v[r \leftarrow a]) \models \varphi$ for some $a \in \mathbb{R}$, similarly for quantification over* time *variables, and as expected for Boolean connectives.*

We can also consider variants of SFO extented to:

- non-linear arithmetic, with formulas over the signature $\langle f_1, f_2, \ldots, \mathbb{Z}, -, +, \times, < \rangle$;

or restricted to:

- difference logic, with formulas over $\langle f_1, f_2, \ldots, +\mathbb{Z}, < \rangle$;
- linear order, with formulas over $\langle f_1, f_2, \ldots, < \rangle$.

Here by $+\mathbb{Z}$ we denote the set of functions $t \mapsto t + n$ for all $n \in \mathbb{Z}$.

The proposed syntax of SFO may appear overly restrictive as compared to arbitrary formulas over the signature $\langle f, g, \ldots, \mathbb{Z}, -, +, < \rangle$ for $f, g, \ldots \in F$. Two natural restrictions are enforced; we show that neither are essential.

1) The nesting of uninterpreted function symbols is disallowed. We did not encounter examples of properties of real-valued signals requiring such a nesting. Remark that nesting of function symbols can be eliminated by change of variable: any atomic formula of the form $\gamma[f(\tau)]$ can be rewritten as $\exists r : r = f(\tau) \wedge \gamma[r]$ and applying this rewrite repeatedly will yield terms without nested function symbols.
2) Quantification over time variables is according to the form $\exists t \in I : \varphi$. This is a natural way of expressing many temporal properties. Enforcing this particular form is obviously not a restriction since taking $I = (-\infty, +\infty)$ is allowed.

We sometimes use *variable bounds* in quantification over time, as in formula $\forall c' \in (0, c) : \varphi$. Such bounds can also be eliminated and replaced by integers when the corresponding free variables are themselves bounded. In our example, assuming c is bounded by $(0, 1)$, the formula can be rewritten as $\forall c' \in (0, 1) : 0 < c' < c \wedge \varphi$. The general motivation for restricting the quantification over time to a fixed or variable interval is to obtain an efficient monitoring algorithm. The bounds on time variables can be used to limit the scope of the resulting operations over signals to a neighborhood of each linear segment, as we shown in Section IV-C.

Notice that *time* and *value* variables are not fully separated since they can be compared in formulas $\theta_1 < \theta_2$ where θ_1 and θ_2 feature both types of variables. If this was undesirable one could enforce that θ_1 and θ_2 are of the same type. Comparing times with signal values is necessary in order to constrain derivatives of signals.

B. Satisfiability

We show that the satisfiability of SFO is undecidable.

Theorem 6. *The satisfiability of SFO is undecidable in general, and remains undecidable*

1) *over piecewise-linear or piecewise-constant signals;*
2) *over a fixed, bounded or unbounded time domain;*
3) *restricted to linear order;*
4) *restricted to difference logic over Boolean signals.*

Proof. In general, we can proceed by reduction to the halting of two-counter machines, a well-known undecidable problem [15]. The run of a two-counter machine can be encoded using functions f_1, f_2 whose value at integer times is directly that of counters, and a function f_3 holding the instruction number. Then every instruction of a given two-counter machine can be encoded as an SFO formula. For instance, an instruction i incrementing counter 1 is written $\forall t \in [0, +\infty) : f_3(t) = i \rightarrow f_1(t + 1) = f_1(t) + 1 \wedge f_3(t + 1) = i + 1$. The state of the machine can be maintained at integer times, definable via some signal f_0 satisfying $f_0(0) = 0 \wedge \forall t \in [0, +\infty) : f_0(t) = 0 \rightarrow f_0(t + 1) = 0 \wedge \forall c \in (0, 1) : f_0(t + c) \neq 0$. This does not pose any difficulty. We now show that the undecidability still holds under the various restrictions.

1) We can use the same reduction as in the general case. Since nothing is required of f_1, f_2, and f_3 between integer times the formula is satisfiable over restricted models iff it is satisfiable over unrestricted models.
2) We can modify the previous reduction by maintaining the state of the two-counter machine not at integer times but at times of the form $c + id$ for $i \in \mathbb{N}$, given an initial time $c \in (-\infty, +\infty)$ and a period $d \in (0, 1)$. For instance, incrementing the counter 1 in instruction i is

now written $\forall t \in [c, +\infty) : f_3(t) = i \to f_1(t+d) = f_1(t) + 1 \wedge f_3(t+d) = i + 1$. The main formula is prefixed by existential quantification over c and d.

3) We introduce function symbols g, h that we use to mark the beginning of time slots where the state of the counter machine is encoded. We require $\exists t \in (-\infty, +\infty) : g(t) = h(t) \wedge \forall t \in (-\infty, +\infty) : g(t) = h(t) \to \exists t' \in (t, +\infty) : g(t') = h(t') \wedge \forall t'' \in (t, t') : g(t'') \neq h(t'')$. This ensures the existence of a monotone, infinitely countable sequence of times $t_0, t_1, \ldots$ where g and h are equal at t_i and differ in between t_i and t_{i+1} for all i. We then encode machine configurations between t_i and t_{i+1} as follows. Signals f_1, f_2 do not hold the value of counters or instruction numbers, but assume a sequence of unique values. As an example, to increment a counter, we require the existence of a unique value that is in the next time slot but not in the previous one, and symmetrically for decrementing.
4) We proceed by reduction to the satisfiability of metric temporal logic [3], which is undecidable in dense time [16]. Propositions p of metric temporal logic translate as atomic formulas $f_p(t) = 1$ for some time variable t. The temporal operator *until*, denoted $\varphi \mathcal{U}_I \psi$ with timing interval I, can be defined as $\exists c \in I : \varphi[t + c] \wedge \forall c' \in (0, c) : \psi[t + c']$ where φ and ψ are formulas with one free time variable. $\square$

C. Membership

We show that by contrast to the satisfiability problem, the simpler membership problem for SFO is decidable.

Theorem 7. *The membership problem of SFO relative to piecewise-linear traces is decidable.*

Proof. Let w be a piecewise-linear trace and φ a closed formula. We reduce the membership of w in the language of φ to the satisfiability of first-order linear arithmetic. Formulas W_i translating signals $[\![f_i]\!]_w$ can be constructed with size $|W_i| = O(|w|)$. They guarantee $W_i[a, b]$ iff $[\![f_i]\!]_w(a) = b$ for all $a \in \mathbb{T}$, $b \in \mathbb{R}$. Then we replace every term $\rho[f_i(\tau)]$ by $\exists x : \rho[x] \wedge W_i[\tau, x]$ recursively until all function symbols are eliminated. We obtain a formula of linear real arithmetic, i.e. without function symbols. The satisfiability of linear real arithmetic formulas is decidable by quantifier elimination. $\square$

Fischer and Rabin showed that the complexity of satisfiability of linear real arithmetic is at least exponential time for nondeterministic computations [17]. The input to the membership problem is given as a formula and a trace, and while the complexity relative to the formula size is subject to this lower bound, the complexity relative to the size of the trace is not. In particular, we show that the reduction of Theorem 7 yields an algorithm that for a fixed formula decides the membership problem relative to piecewise-linear traces in polynomial time. The quantifier elimination problem in linear real arithmetic produced by this reduction can be solved by the method of Ferrante and Rackoff [18], or its improvement by Loos and Weispfenning [19], both proceeding by *virtual substitution*. Such a method eliminates each existential quantifier $\exists x$ in some subformula $\exists x : F[x]$ in two steps that we briefly recall.

In the first step, terms featuring variable x are put in the form $x < \theta$ or $\theta < x$ where θ is a term not featuring x. Note that this may create rational coefficients, we temporarily allow them. Such terms θ form a set Θ.

In the second step, the subformula $\exists x : F[x]$ is replaced by a finite disjunction $\bigvee_{\gamma \in \Gamma} F[x /\!\!/ \gamma]$ over a set of terms Γ. Ferrante and Rackoff take $\Gamma = \{\frac{\theta_1 + \theta_2}{2} \mid \theta_1, \theta_2 \in \Theta\} \cup \{\pm\infty\}$, while Loos and Weispfenning take $\Gamma = \{\theta, \theta \pm \epsilon \mid \theta \in \Theta\}$ for some infinitesimal ϵ. The substitution is virtual because such terms are not proper, but one can define atomic formulas equivalent to their substitution, for example $\frac{y}{3} < z + \epsilon$ is equivalent to $y < 3z$. The process is repeated for each quantifier.

By applying the method of [18] or [19], eliminating k quantifiers from a linear real arithmetic formula of size m can be done in time $m^{2^{O(k)}}$. We thus obtain the following:

Theorem 8. *The membership problem of an SFO formula φ relative to a piecewise-linear trace w is computable in time $(m+n)^{2^{O(k+l)}}$ where k is the number of quantifiers in φ, l is the number of occurrences of function symbols in φ, m is the length of φ, and n is the length of w.*

Proof. We translate the SFO formula φ into a quantified formula F of linear real arithmetic, using the procedure of Theorem 7. Formula F has size $O(m + nl)$ and $k + l$ quantifiers. Eliminating all quantifiers from F can be done in time $(m+n)^{2^{O(k+l)}}$, and deciding the validity of the resulting ground formula can be done in linear time. $\square$

The cylindric algebraic decomposition method of Collins [20], although much more involved, has time complexity with the same bounds so that Theorem 8 also extends to SFO with non-linear arithmetic and piecewise-polynomial signals. In the case of difference logic, the quantifier elimination problem can be decided in exponential time by a similar method [21], so that the membership problem of SFO restricted to difference logic over piecewise-constant signals is decidable in exponential time.

We first remark that the (doubly) exponential complexity comes solely from the number of function symbols and quantifiers, which is expected to be small in practice. The complexity relative to trace size still appears prohibitive of practical applications that involve long traces. We will show in Section IV that for a fairly large class of formulas, the complexity relative to the trace size can be reduced to linear time.

D. Signal Second-Order Logic

Let us call signal second-order logic (SSO) the extension of SFO with quantifiers over uninterpreted function symbols. We briefly survey the theoretical properties of this extension. Since SSO contains SFO, its satisfiability problem is evidently not decidable. We show that indeed even its membership problem is not decidable.

Theorem 9. *The membership problem of SSO relative to piecewise-linear traces is undecidable.*

Proof. For any closed SFO formula φ whose uninterpreted function symbols are $f_1, \ldots, f_n$, the membership in $\exists f_1 \exists f_2 \ldots \exists f_n : \varphi$ is equivalent to the satisfiability of φ. By Theorem 6, the satisfiability of an arbitrary SFO formula φ is undecidable. $\square$

IV. Monitoring

In this section, we study the monitoring problem for SFO, defined as follows. We call *temporal* a formula φ with exactly one free variable t, of type *time*, and such that t appears with coefficient 0 or 1 in every term τ argument of a function symbol.

Definition 10. *The* satisfaction signal *of formula φ relative to trace w is the Boolean signal denoted w_φ such that $w_\varphi(t) = 1$ iff $(w, t) \models \varphi$ for all $t \in \mathbb{T}$.*

We restrict our attention to piecewise-linear traces over a bounded time domain. The satisfaction signal of some formula φ relative to such a trace w is a piecewise-constant Boolean signal, and can be represented as a finite union of intervals over which the formula is true.

Definition 11. *The* monitoring *problem is the task of computing, given a temporal formula φ and a signal w, the satisfaction signal of φ relative to w.*

The monitoring problem provides a slight generalization of the membership problem. It is particularly relevant when seen as an *online* computation problem, where the satisfaction signal should be produced along with the trace. It is also relevant as an *offline* computation problem, when used for testing. The time at which the formula is true and false provides valuable information for the debugging of simulation traces. Such information is explicitly computed for *monitoring*, but not strictly required for *membership*.

A. Polyhedral Representation

The interpretation of every term and every formula can be represented as a set of convex polyhedra. Our monitoring procedure of piecewise-linear signals for SFO over linear real arithmetic is based on this principle. Below, we give a high-level polyhedral characterization of operations over SFO terms and formula and later we give the corresponding concrete algorithms. We denote convex polyhedra as a set of linear equalities and inequalities, and use operations $\sqcup$, $\sqcap$, *complement*, and *eliminate* denoting the union, intersection, complement, and projection of polyhedra or sets of polyhedra. Their implementation is presented in full details in Section IV-B.

Functions We assume that the interpretation of every function f is given as a set of (convex) polyhedra $\mathcal{P}_f$ with two free variables: t_f denoting time and v_f denoting the value of f at the given time point.

Terms In general, the interpretation of a term can be seen as a function from the values of its free variables to a real value. Thus a term θ (as in Definition 4) can be represented as a set of convex polyhedra $\mathcal{P}_\theta$, with variables corresponding to free variables of the term and one extra variable v_θ that corresponds to the value of the term. Polyhedron $\mathcal{P}_\theta$ and variable v_θ can be defined by induction over the term structure as follows.

- For $\theta \equiv \tau$, $\mathcal{P}_\theta = \{v' = \tau\}$, $v_\theta = v'$, where τ is a time term as in Definition 4, i.e., a linear constraint over time variables, and v' is a fresh variable.
- For $\theta \equiv r$, $\mathcal{P}_\theta = \{v' = r\}$, $v_\theta = v'$, where r is a value variable, and v' is a fresh variable.
- For $\theta \equiv n$, $\mathcal{P}_\theta = \{v' = n\}$, $v_\theta = v'$, where $n \in \mathbb{Z}$ and v' is a fresh variable.
- For $\theta \equiv f(\tau)$, $\mathcal{P}_\theta = \mathcal{P}_f[t_f \mapsto \tau, v_f \mapsto v']$, $v_\theta = v'$, where τ is a linear expression as in Definition 4, v' is a fresh variable, and $\sqcap$ denotes intersection of sets of polyhedra.
- For $\theta \equiv \theta_1 \pm \theta_2$, $\mathcal{P}_\theta = \mathit{eliminate}(v_{\theta_1}, v_{\theta_2}, \{v' = v_{\theta_1} \pm v_{\theta_2}\} \sqcap \mathcal{P}_{\theta_1} \sqcap \mathcal{P}_{\theta_2})$, where v' is a fresh variable, and *eliminate* denotes the elimination of the given variables from the set of polyhedra.

Formulas Similarly, the interpretation of a formula φ can be seen as a function from the values of its free variables to a Boolean value and thus can represented as a set of polyhedra $\mathcal{P}_\varphi$, with variables corresponding to free variables of the formula. $\mathcal{P}_\varphi$ can be defined by induction over the formula structure as follows.

- For $\varphi \equiv \theta_1 < \theta_2$, $\mathcal{P}_\varphi = \mathit{eliminate}(v_{\theta_1}, v_{\theta_2}, \{v_{\theta_1} < v_{\theta_2}\} \sqcap \mathcal{P}_{\theta_1} \sqcap \mathcal{P}_{\theta_2})$, where $\mathit{eliminate}(V, \mathcal{P})$ denotes the elimination of variables V in the polyhedral set $\mathcal{P}$.
- For $\varphi \equiv \neg\varphi'$, $\mathcal{P}_\varphi = \mathit{complement}(\mathcal{P}_{\varphi'})$, where *complement* takes the complement of a polyhedral set.
- For $\varphi \equiv \varphi_1 \vee \varphi_2$, $\mathcal{P}_\varphi = \mathcal{P}_{\varphi_1} \sqcup \mathcal{P}_{\varphi_2}$, where $\sqcup$ stands for the union of two sets of polyhedra.
- For $\varphi \equiv \exists r : \varphi'$, $\mathcal{P}_\varphi = \mathit{eliminate}(r, \mathcal{P}_{\varphi'})$.
- For $\varphi \equiv \exists t \in I : \varphi'$, $\mathcal{P}_\varphi = \mathit{eliminate}(t, \{t \in I\} \sqcap \mathcal{P}_{\varphi'})$.

B. Monitoring Algorithm

In this section we present a monitoring algorithm for SFO. The key idea behind this algorithm is that for a temporal formula φ, its free time variable t is often the most significant dimension in the following sense: for a fixed value of t the interpretation of φ as a non-convex polyhedron has few convex parts. We internally represent every convex polyhedron P in the form $\{t \in I\} \sqcap Q$, where I is an interval with bounds in $\mathbb{Z} \cup \{\pm\infty\}$. We use interval I to order polyhedra along the time axis according to $\sup(I)$, which enables to rule out many empty intersections of polyhedra.

Representing the interpretations An interpreted function f that corresponds to an input signal is assumed to be piecewise-linear, have bounded domain and finite variability and thus can be represented as a list of polyhedra that are disjoint and ordered by time t_f; each polyhedron represents the value of the function on a separate bounded interval of time.

For every term and every formula, we represent its interpretation as a list ordered by the upper bound on variable t, the free variable of the top-level formula. We make sure that every operation on lists of polyhedra (union, intersection, complement) preserves this ordering. Operations on polyhedra used as subroutines in our algorithm all admit efficient implementations based the double description method, see [22].

Terms Algorithm 1 implements the function *term* that builds a list of polyhedra representing the term θ (as in Definition 4). The function returns a pair of values: list of polyhedra and the variable denoting the value. The extra argument P_{dom} is a set of constraints on bound time variables that was collected when traversing the formula.

Algorithm 1 Creating a list of polyhedra for a term.

```
function $term(\theta, P_{\mathrm{dom}})$
  if $\theta \equiv \tau$ then
    $v' \leftarrow fresh\ \ variable$
    return $\{v' = \tau\}, v'$
  else if $\theta \equiv r$ then
    $v' \leftarrow fresh\ \ variable$
    return $\{v' = r\}, v'$
  else if $\theta \equiv n$ then
    $v' \leftarrow fresh\ \ variable$
    return $\{v' = n\} \sqcap P_{\mathrm{dom}}, v'$
  else if $\theta \equiv f(\tau)$ then
    $v' \leftarrow fresh\ \ variable$
    return $\mathcal{P}_f[t_f \mapsto \tau, v_f \mapsto v'] \sqcap P_{\mathrm{dom}}, v'$
  else if $\theta \equiv \theta_1 \pm \theta_2$ then
    $\mathcal{P}_1, v_1 \leftarrow term(\theta_1, P_{\mathrm{dom}})$
    $\mathcal{P}_2, v_2 \leftarrow term(\theta_2, P_{\mathrm{dom}})$
    $v' \leftarrow fresh\ \ variable$
    return
      $eliminate(v_1, v_2, \{v' = v_1 \pm v_2\} \sqcap \mathcal{P}_1 \sqcap \mathcal{P}_2), v'$
  end
end
```

We note that when constructing the list of polyhedra for $f(\tau)$, the term τ is a linear expression over time variables, and thus substitution of t_f with τ is a valid polyhedral operation. Since t can only appear positively in τ, the resulting list will naturally be sorted by the upper bound on variable t.

Formulas Fig. 2 shows the function *formula* that builds a list of polyhedra representing the formula φ. Again, θ ranges over the terms as in Definition 4. The extra argument P_{dom} is again a set of constraints on bound variables that was collected when traversing the formula. For the top-level formula, we pass $\{\top\}$ as the P_{dom} argument.

Variable elimination The operation *eliminate* traverses the given list of polyhedra and eliminates from every polyhedron the given set of variables, which is a standard polyhedral operation. The projection operation leaves the time-ordering of polyhedra unaffected.

Algorithm 2 Creating a list of polyhedra for a formula.

```
function $formula(\varphi, P_{\mathrm{dom}})$
  if $\varphi \equiv \theta_1 < \theta_2$ then
    $\mathcal{P}, v \leftarrow term(\theta_1 - \theta_2, P_{\mathrm{dom}})$
    return $eliminate(v, \mathcal{P} \sqcap \{v < 0\})$
  else if $\varphi \equiv \neg\varphi'$ then
    $\mathcal{P} \leftarrow formula(\varphi', P_{\mathrm{dom}})$
    return $P_{\mathrm{dom}} \sqcap complement(\mathcal{P})$
  else if $\varphi \equiv \varphi_1 \vee \varphi_2$ then
    $\mathcal{P}_1 \leftarrow formula(\varphi_1, P_{\mathrm{dom}})$
    $\mathcal{P}_2 \leftarrow formula(\varphi_2, P_{\mathrm{dom}})$
    return $\mathcal{P}_1 \sqcup \mathcal{P}_2$
  else if $\varphi \equiv \exists r : \varphi'$ then
    $\mathcal{P} \leftarrow formula(\varphi', P_{\mathrm{dom}})$
    return $eliminate(r, \mathcal{P})$
  else if $\varphi \equiv \exists s \in I : \varphi'$ then
    $\mathcal{P} \leftarrow formula(\varphi', P_{\mathrm{dom}} \sqcap \{s \in I\})$
    return $eliminate(s, \mathcal{P})$
  end
end
```

Union The operation $\sqcup$ denotes the union of two lists of polyhedra. When the input lists are ordered, we can merge them into a new list that is also ordered. This is a standard algorithm on ordered lists, which preforms in linear-time.

Intersection The operation $\sqcap$ denotes the intersection, and we use the same symbol to denote one of the three operations:

- The intersection of two convex polyhedra.
- The intersection of a polyhedron and a list, which can be implemented with pairwise intersection. This intersection operation preserves the time-ordering of polyhedra.
- The intersection of two lists of polyhedra, which can be implemented efficiently by exploiting the time ordering and the fact that two polyhedra, whose time projections do not intersect, themselves do not intersect.

Complement In general, a list of polyhedra can be complemented by following De Morgan's laws. For sets of polyhedra representing formulas, we can make the procedure more efficient by following the intuition that the projections of polyhedra on the main time variable t are in most cases bounded and every projection intersects only a small number of projections of "nearby" polyhedra. The function *complement* in Algorithm 3 implements this intuition. It starts with $\mathcal{P}$ being the list of polyhedra to complement and $\mathcal{C}$ being a singleton containing the "universal" polyhedron that only restricts t to the time domain. In every step, the function takes the first polyhedron P in the list $\mathcal{P}$ which is presented in the form $P = \{t \in I\} \sqcap Q$ and partitions $\mathcal{C}$ according to I. Polyhedron P is subtracted from $\mathcal{C}$, by subtracting $\{t \in I\}$ on one hand, subtracting Q from the intersection of $\mathcal{C}$ with $\{t \in I\}$ on the other hand, and then taking the union of the two. Subtraction (denoted by *subtract*) can be done by a straightforward intersection with complement, following De

Algorithm 3 Complementation over a list of polyhedra. The operation *subtract* between a list of polyhedra and a single polyhedron denotes straightforward intersection with complement following De Morgan's laws.

```
function $complement(\mathcal{P})$
  $\mathcal{C} \leftarrow \{t \in \mathbb{T}\}$
  while $\mathcal{P} \neq \varnothing$ do
    pick $P \in \mathcal{P}$
    $\{t \in I\} \sqcap Q \leftarrow P$
    $\mathcal{S} \leftarrow \mathcal{C} \sqcap \{t \in I\}$
    $\mathcal{C} \leftarrow subtract(\mathcal{C}, \{t \in I\}) \sqcup subtract(\mathcal{S}, Q)$
    $\mathcal{P} \leftarrow \mathcal{P} \setminus P$
  end
  return $\mathcal{C}$
end
```

Morgan's laws. Since $\mathcal{C}$ is ordered by t and polyhedra are often bounded in t, subtracting $\{t \in I\}$ from $\mathcal{C}$ can usually be done by scanning only a few elements of $\mathcal{C}$. Intersecting $\mathcal{C}$ with $\{t \in I\}$ is done similarly, and by the same intuition we usually only need to scan a few elements in the neighborhood of interval I. The result of this intersection, denoted $\mathcal{S}$, is expected to contain few elements. Then Q is subtracted from $\mathcal{S}$. Taking the union of the two parts is done using the merge subroutine previously discussed. We are done processing P and we remove it from the list $\mathcal{P}$.

C. Bounded-Response Formulas

We now define a fragment of SFO that we call *bounded-response*. In the following section, we show that for the bounded-response fragment of SFO and bounded-variability traces, Algorithm 2 has time complexity linear in the trace length.

Definition 12 (Bounded-Response SFO). *We call* bounded-response *the fragment of SFO in which all intervals I in subformulas of the form $\exists t : t \in I$ are such that* $\inf I \neq -\infty$ *and* $\sup I \neq +\infty$.

One may argue that the *bounded-response* restriction is too strong. We observe that properties involving unbounded time can still be expressed. In the context of monitoring, this still allows for one unbounded time variable. This fragment is thus sufficient to monitor safety properties in which bad prefixes can be identified by a segment of the input of bounded duration, a very common situation in practice.

D. Complexity

In this section, we characterize the complexity of our algorithm, first in the general case, and then we identify a fragment of SFO for which the complexity is reduced to linear-time relative to the trace length.

Theorem 13. *Algorithm 2 operates in time $2^{(m+n)^{2^{O(k+l)}}}$ where k is the number of quantifiers in φ, l is the number of occurrences of function symbols in φ, m is the length of φ, and n is the length of w.*

Proof. For any union of polyhedra $\mathcal{P}$, we denote by $\#(\mathcal{P})$ the number of unique linear constraints that appear (possibly repeated) in $\mathcal{P}$. We show that for any formula φ and trace w, if $\mathcal{P}$ is a polyhedral representation of $[\![\varphi]\!]_w$, then $\#(P) \leq (m+n)^{2^{O(k+l)}}$, where k, l, m, n are as in the statement of Theorem 7. One can see that $\#(\mathcal{P}) = nl$ when φ is atomic, $\#(complement(\mathcal{P})) = \#(\mathcal{P})$, and $\#(\mathcal{P}_1 \sqcup \mathcal{P}_2) \leq \#(\mathcal{P}_1) + \#(\mathcal{P}_2)$. The projection operation is such that $\#(eliminate(x, \mathcal{P})) \leq \#(\mathcal{P})^2$. Thus the bound holds by induction on the formula structure.

Now any set of convex polyhedra $\mathcal{P}$ has at most $2^{\#(\mathcal{P})}$ elements, each with at most $\#(\mathcal{P})$ constraints. Algorithm 2 ensures that convex polyhedra are not duplicated, since it removes every convex polyhedron that is already covered by another convex polyhedron. Observe that every subroutine of Algorithm 2 has running time polynomial in the size of their combined input and output. This gives us the bound as desired. □

In the general case the complexity of monitoring is related to that of the membership problem. One notable difference is that the membership problem is a decision procedure, and its output has constant size. When using a quantifier elimination procedure, it is best not to enforce disjunctive normal form [18]. Since the formula is closed, the result will be a ground formula that can easily be decided without using such a special form. By contrast the output of the monitoring can grow fast with the size of the signal. When using a quantifier elimination procedure, we must at least in the last step use a disjunctive normal form, since the required format of the output is a union of intervals. Algorithm 2 indeed enforces a disjunctive normal form at every step of the quantifier elimination, as with Fourier-Motzkin elimination. We note that the complexity of such a procedure is not worse than virtual substitution method when the output is required to be in disjunction normal form, see also [23].

Let us now consider the bounded-response fragment of SFO. We call *horizon* of a bounded-response formula the sum of the absolute value of all its time constants. We call *variability* of a piecewise-linear signal w relative to some bounded-response formula φ the maximum number of linear segments in w during any time period as long as the horizon of φ.

Theorem 14. *Algorithm 2 operates in time $n2^{(m+j)^{2^{O(k+l)}}}$ over the class of bounded-response formulas, where j is the variability of w relative to φ, k is the number of quantifiers in φ, l is the number of occurrences of function symbols in φ, m is the length of φ, and n is the length of w.*

Proof. Let h be the horizon of formula φ, and (t, t') be an interval over which signal w is linear. It follows from the definition *variability* that the size of the restriction of w to the time interval $[t - 2h, t' + 2h]$ is at most $5h$. Similarly as in the proof of Theorem 13, we have that if $\mathcal{P}$ is a polyhedral representation of the set

$$S_{t,t'} = \{(t'', v) \mid t - h \leq t'' \leq t' + h \text{ and } (t'', v) \in [\![\varphi]\!]_w\},$$

then the number of unique constraints in $\mathcal{P}$ is $(m+h)^{2^{O(k+l)}}$. This is because the part of the signal that can influence the set S_t is contained in $[t - 2h, t' + 2h]$. The set $[\![\varphi]\!]_w$ is covered by the $S_{t,t'}$ obtained from all t, t' chosen as endpoints of linear pieces of w. Thus the polyhedral representation of $[\![\varphi]\!]_w$ has size $n2^{(m+h)^{2^{O(k+l)}}}$. Subroutines of Algorithm 2 have running time at most proportional to the size of their input multiplied by maximum the number of polytopes that overlap in time. Indeed the only superlinear operation is that of *complement*. For bounded-response formulas, the set of polyhedra that overlap in time is $2^{(m+h)^{2^{O(k+l)}}}$. This gives us the desired computation time upper bound. □

Remark that following a similar argument, one could obtain an *online* monitoring algorithm with matching complexity for bounded-response SFO.

V. Quantified Signal Temporal Logic

In this section we define quantified signal temporal logic (QSTL), which extends STL [1] with first order quantification, and study its relation with SFO.

Terms of type *time* and *value* of QSTL are given by the grammar:

$$\tau ::= t \mid n \mid \tau_1 - \tau_2 \mid \tau_1 + \tau_2$$
$$\rho ::= r \mid f \mid x \mid n \mid \rho_1 - \rho_2 \mid \rho_1 + \rho_2$$

where $n \in \mathbb{Z}$, $t \in T$, $r \in R$, and $f \in F$. Formulas of QSTL are given by the grammar:

$$\varphi ::= \theta_1 < \theta_2 \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \exists x : \varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2 \mid \Diamond_{[\tau_1,\tau_2]}\, \varphi \mid \varphi_1 \,\mathcal{S}\, \varphi_2 \mid \diamondminus_{[\tau_1,\tau_2]}\, \varphi$$

where $x \in X$ and τ_1, τ_2, θ_1, and θ_2 are QSTL terms, such that τ_1 and τ_2 are of type *time*.

In our SFO framework, the temporal modalities of QSTL are seen as syntactic sugar that may be applied to SFO temporal formulas. Let us assume a distinguished time variable t, standing for the absolute time. Atomic formulas $\theta_1 < \theta_2$ are defined as $\theta_1(t) < \theta_2(t)$, where for some term θ, writing $\theta(t)$ means applying the argument t to all function symbols in θ. Operator *until*, denoted $\mathcal{U}$, is defined by letting

$$\varphi \,\mathcal{U}\, \psi \equiv \exists c \in (0, +\infty) : \varphi[t + c] \wedge \forall d \in (0, c) : \psi[t + d]$$

for all temporal formulas φ and ψ. The definition of operator *since* $\mathcal{S}$ is symmetrical. Operator *eventually*, denoted $\Diamond_{[\tau_1,\tau_2]}$ for time terms τ_1, τ_2, is defined by letting

$$\Diamond_{[\tau_1,\tau_2]}\, \varphi \equiv \exists c : 0 \leq \tau_1 \leq c \leq \tau_2 \wedge \varphi[t + c]$$

for all temporal formulas φ. The definition of operator *once*, denoted $\diamondminus_{[\tau_1,\tau_2]}$, is symmetrical. One can also define operator *always*, denoted $\Box_{[\tau_1,\tau_2]}$, by letting $\Box_{[\tau_1,\tau_2]}\, \varphi \equiv \neg \Diamond_{[\tau_1,\tau_2]} \neg\varphi$, and operator *historically*, denoted $\boxminus_{[\tau_1,\tau_2]}$, by letting $\boxminus_{[\tau_1,\tau_2]}\, \varphi \equiv \neg \diamondminus_{[\tau_1,\tau_2]} \neg\varphi$.

It is easy to see that every SFO formula can be written in QSTL form, so that QSTL is complete for SFO. This is because the modality $\exists r : \Diamond_{[t,t]}\, f(t) = r$ allows to access the value r of function f at any time t explicitly. In more details, putting an SFO formula in QSTL form is a two-stage process. The first stage recursively moves signals outside of inequalities by rewriting a formula φ into formula $\underline{\varphi}$ defined as follows:

$$\underline{\gamma[f(\tau)]} \equiv \exists t \in (-\infty, +\infty) : \exists r : r = f(t) \wedge t = \tau \wedge \underline{\gamma[r]}$$
$$\underline{\neg\varphi} \equiv \neg\underline{\varphi}$$
$$\underline{\varphi_1 \vee \varphi_2} \equiv \underline{\varphi_1} \vee \underline{\varphi_2}$$
$$\underline{\exists x : \varphi} \equiv \exists x : \underline{\varphi}$$

The second stage consists in replacing every subformulas $r = f(t)$ by $\Diamond_{[t,t]}\, f = r$. We have removed all arguments of function symbols and thus obtain a QSTL formula. Since this formula has no nested temporal operators, it is equivalent at time zero to the original SFO formula.

VI. Case Study

We present in this section the DDR2 memory interface case study. DDR2 memory interface acts as a bus between the memory and other components and exhibits communication of digital data implemented at the analog level of abstraction. In DDR2, the data access is controlled by a single-ended or differential strobe signal, which acts as an asynchronous clock. In the remainder of this section, we consider the single-ended data strobe DDR2-400 memory interface.

We focus more specifically on the alignment property between data signal *DS* and data signal strobe *DQS*. The alignment between the two signals is defined with respect to their respective crossings of thresholds defined in Table I. In the remainder of the section, we refer to different thresholds by their identifier.

TABLE I
DDR2-400 specification of threshold values.

Threshold name	Threshold id	Value (V)
V_{DDQ}	V_1	1.800
$V_{IH(AC)_{min}}$	V_2	1.250
$V_{IH(DC)_{min}}$	V_3	1.025
$V_{REF(DC)}$	V_4	0.900
$V_{IL(DC)_{max}}$	V_5	0.775
$V_{IL(AC)_{max}}$	V_6	0.650

More specifically, the proper alignment between *DS* and *DQS* is determined by two values, the *setup* time *tDS* and *hold* time *tDH*. The setup and hold times of *DS* and *DQS* are checked both on their *falling* and *rising* edges of the strobe signal. We focus on the setup time *tDS* at the falling edge of *DQS*, the other three cases are symmetric. We illustrate this property with the timing diagram shown in Figure 3.

Informally, we can express the setup time alignment property at the falling edge of *DQS* as follows: *Whenever* DQS *is*

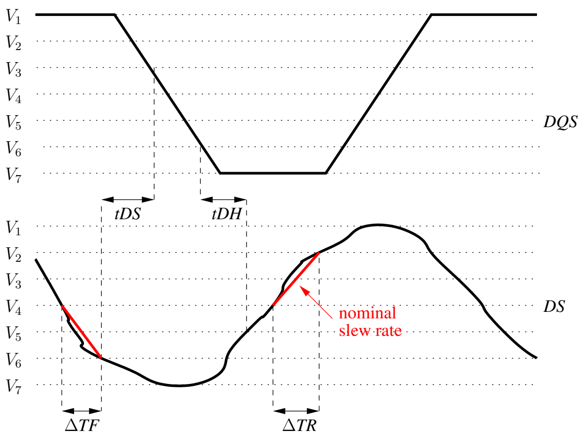

Fig. 3. Data *DS* and data strobe *DQS* alignment with *tDS* setup and *tDH* hold time on the falling edge.

on its falling edge, the distance from the previous falling edge in DS *is at least* tDS *time*. The falling edge of *DQS* (*DS*) is characterized by *DQS* (*DS*) crossing V_3 (V_6) from above.

In order to express the above requirement in SFO, we first define the *falling edge* operator $\downarrow(s, c)[t]$, which holds at t iff signal f crosses the threshold c from above. We formalize this operators as follows:

$$\downarrow(f, c)[t] \equiv f(t) = c \wedge \exists d \in (0, +\infty) : \\ \forall d' \in (0, d) : f(t + d') < c$$

It is now straightforward to define the alignment requirement as the following SFO $\psi[t]$ formula:

$$\psi[t] \equiv \downarrow(DQS, V_3)[t] \rightarrow \boxminus_{[0,tDS]} \neg\downarrow(DS, V_6)[t]$$

We note that falling edges of a signal f below threshold c can also be captured in pure STL by letting $\downarrow(f, c) \equiv (f = c) \wedge (f < c)\mathcal{U}\top$. As a consequence, the alignment requirement does not seem to need the power of SFO. In fact, the above STL specification relies on the assumption that *tDS* is a constant value. In reality, this is not the case. The setup time *tDS* varies dynamically according to the *slew rates* (slopes) of *DS* and *DQS*. According to the DDR2 standard, the setup time *tDS* is the sum of a constant *base term tDS(base)* and a variable *correction term*

$$\Delta tDS = tDS - tDS(base)$$

where *tDS(base)* equals to $150ps$ for single-ended DDR2-400 and the correction term ΔtDS is a value that depends directly on slew rates ssr_{DS} and ssr_{DQS} measured in *DS* and *DQS*. The setup slew rate ssr_s of a falling signal s being defined as

$$ssr_s = \frac{V_2 - V_4}{\Delta TF_s}$$

where ΔTF_s is the time that the signal s spends between V_4 and V_2 (see Figure 3). The exact computation of $\Delta tDS = \delta(\Delta TF_{DQS}, \Delta TF_{DS})$ is done by interpolating the values from the look-up table depicted in Table II.

TABLE II
FRAGMENT OF THE LOOK-UP TABLE USED TO COMPUTE THE CORRECTION TERM ΔtDS AS A FUNCTION OF ssr_{DQS} AND ssr_{DS}.

		DQS slew rate (V/ns)				
		2	1.5	1	0.9	0.8
DS slew rate (V/ns)	2	188	146	63	-	-
	1.5	167	125	42	43	-
	1	125	83	0	-2	-13
	0.9	-	69	-14	-13	-27
	0.8	-	-	-31	-30	-44

To summarize, checking the setup time for *DQS* and *DS* requires dynamically measuring their nominal slew rates via ΔTF_{DQS} and ΔTF_{DS} and adapting accordingly the setup time bound *tDS*. This specification is clearly beyond the expressiveness of STL.

In the remainder of this section, we will demonstrate the formalization of the full alignment property in SFO. We We first define two auxiliary operators $\ominus_d^{c,c'}(f)[t]$ and $\bigcirc_d^{c,c'}(f)[t]$ as the shorthands

$$\ominus_d^{c,c'}(f)[t] \equiv \exists e \in (0, \infty) : \\ \left(\begin{array}{l} \forall e' \in (0, e) : f(t - e') < c \\ \wedge\ f(t - e) = c \\ \wedge\ \forall e' \in (e, e + d) : c < f(t - e') < c' \\ \wedge\ f(t - e - d) = c' \end{array}\right)$$

and

$$\bigcirc_d^{c,c'}(f)[t] \equiv \exists e \in (0, \infty) : \\ \left(\begin{array}{l} \forall e' \in (0, e) : f(t + e') < c \\ \wedge\ f(t + e) = c \\ \wedge\ \forall e' \in (e, e + d) : c < f(t + e') < c' \\ \wedge\ f(t + e + d) = c' \end{array}\right)$$

Intuitively, $\ominus_d^{c,c'}(f)[t]$ ($\bigcirc_d^{c,c'}(f)$) holds at time t iff the previous (next) duration that f spends in the region defined by the interval $[c, c']$ has the duration d.

We are now ready to express the alignment requirement $\psi'[t]$ is SFO as follows:

$$\psi'[t] \equiv \downarrow(DQS, V_3)[t] \rightarrow \exists \Delta TF_{DQS}, \Delta TF_{DS} : \\ \ominus_{\Delta TF_{DS}}^{V_6,V_4}(DS)[t] \wedge \bigcirc_{\Delta TF_{DQS}}^{V_6,V_4}(DQS)[t] \wedge \\ tDS = \delta(\Delta TF_{DQS}, \Delta TF_{DS}) \wedge \boxminus_{[0,tDS]} \neg\downarrow(DS, V_6)[t]$$

The specification states that whenever *DS* is on its falling edge, there exist timing parameters ΔTF_{DQS} and ΔTF_{DS} such that (1) ΔTF_{DQS} is the duration that *DQS* spends in the region $[V_6, V_4]$ used to measure its slew rate, (2) ΔTF_{DS} is the duration that *DS* spends in the region $[V_6, V_4]$ used to measure its slew rate, (3) *tDS* is the setup time as a function of ΔTF_{DQS} and ΔTF_{DS}, and (4) the previous falling edge of *DS* is at least *tDS* time away.

VII. Conclusion

We introduced *signal first-order logic* (SFO) for the purpose of specifying and monitoring properties of real-valued signal. On the specification side, we show that this logic is equivalent in expressive power as *signal temporal logic* extended with quantifiers (QSTL). We believe that SFO provides a clear and general framework for studying the specification of real-valued temporal behaviors. On the monitoring side, the algorithm that we propose works recursively on the SFO formula structure, and computes the satisfaction set (or validity domain) of the formula at every time point. The monitoring of SFO is hard, and part of it is due to the lack of temporal structure of the formula. The hardness of this problem is indeed already present for monadic first-order logic. The simpler membership problem of SFO is at least nondeterministic exponential time relative to the formula size. We identify the fragment of *bounded-response* formulas, for which our monitoring algorithm terminates in linear-time relative to the trace length.

The problem of monitoring SFO or QSTL can be related to the parametric identification of STL. In the latter problem, the objective is to find the set of (tightest) parameter values such that the STL formula is satisfied. Variables in *parametric signal temporal logic* (PSTL) [7] can be seen as free variables in QSTL. We also remark that QSTL semantically contains STL* [24], which supplements STL by *freeze* quantifiers [25]. The monitoring algorithm of [24] proceeds by representing the satisfaction set using several time dimensions: the reference time, and the time at which values are *frozen*. On the other hand, our algorithm represents the satisfaction set relative to value dimensions.

In the future, we plan to implement our monitoring algorithm and assess its performance. We will also investigate the adaptation of our SFO monitoring algorithm to the problem of pattern matching. The problem of pattern matching over Boolean and real-valued signals is studied in [26] and in [27], respectively. Both work consider *timed regular expressions* of [28] as a specification language. A star-free variant of such expressions could readily be encoded in SFO. The problem of finding an adequate counterpart in logic to the Kleene closure remains open.

Acknowledgments

We thank Eugene Asarin and Oded Maler for their helpful comments.

References

[1] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems (FORMATS/FTRTFT)*, 2004, pp. 152–166.

[2] A. Pnueli, "The temporal logic of programs," in *Foundations of Computer Science, 1977., 18th Annual Symposium on*. IEEE, 1977, pp. 46–57.

[3] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-time systems*, vol. 2, no. 4, pp. 255–299, 1990.

[4] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theor. Comput. Sci.*, vol. 410, no. 42, pp. 4262–4291, 2009.

[5] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2010, pp. 92–106.

[6] A. Donzé, T. Ferrere, and O. Maler, "Efficient robust monitoring for STL," in *International Conference on Computer Aided Verification*. Springer, 2013, pp. 264–279.

[7] E. Asarin, A. Donzé, O. Maler, and D. Nickovic, "Parametric identification of temporal properties," in *Runtime Verification*, 2011, pp. 147–160.

[8] H. Yang, B. Hoxha, and G. Fainekos, "Querying parametric temporal logic properties on embedded systems," in *IFIP International Conference on Testing Software and Systems*. Springer, 2012, pp. 136–151.

[9] A. Bakhirkin, T. Ferrère, and O. Maler, "Efficient parametric identification for STL," in *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*. ACM, 2018, pp. 177–186.

[10] E. Bartocci, L. Bortolussi, and G. Sanguinetti, "Data-driven statistical learning of temporal logic properties," in *Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2014, pp. 23–37.

[11] X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia, "Mining requirements from closed-loop control models," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 34, no. 11, pp. 1704–1717, 2015.

[12] Z. Kong, A. Jones, and C. Belta, "Temporal logics for learning and detection of anomalous behavior," *IEEE Trans. Automat. Contr.*, vol. 62, no. 3, pp. 1210–1222, 2017.

[13] T. Ferrère, O. Maler, and D. Nickovic, "Trace diagnostics using temporal implicants," in *Automated Technology for Verification and Analysis*, 2015, pp. 241–258.

[14] E. Bartocci, J. V. Deshmukh, A. Donzé, G. E. Fainekos, O. Maler, D. Nickovic, and S. Sankaranarayanan, "Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications," in *Lectures on Runtime Verification - Introductory and Advanced Topics*, 2018, pp. 135–175.

[15] M. L. Minsky, *Computation: finite and infinite machines*. Prentice-Hall, 1967.

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

[17] M. J. Fischer and M. O. Rabin, "Super-exponential complexity of presburger arithmetic," in *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Springer, 1998, pp. 122–135.

[18] J. Ferrante and C. Rackoff, "A decision procedure for the first order theory of real addition with order," *SIAM Journal on Computing*, vol. 4, no. 1, pp. 69–76, 1975.

[19] R. Loos and V. Weispfenning, "Applying linear quantifier elimination," *The computer journal*, vol. 36, no. 5, pp. 450–462, 1993.

[20] G. E. Collins, "Quantifier elimination for real closed fields by cylindrical algebraic decompostion," in *Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern, May 20–23, 1975*. Springer, 1975, pp. 134–183.

[21] M. Koubarakis, "Complexity results for first-order theories of temporal constraints," in *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 1994, pp. 379–390.

[22] R. Bagnara, P. M. Hill, and E. Zaffanella, "Not necessarily closed convex polyhedra and the double description method," *Formal Asp. Comput.*, vol. 17, no. 2, pp. 222–257, 2005.

[23] D. Monniaux, "A quantifier elimination algorithm for linear real arithmetic," in *International Conference on Logic for Programming Artificial Intelligence and Reasoning*. Springer, 2008, pp. 243–257.

[24] L. Brim, P. Dluhoš, D. Šafránek, and T. Vejpustek, "STL*: Extending signal temporal logic with signal-value freezing operator," *Information and Computation*, vol. 236, pp. 52–67, 2014.

[25] R. Alur and T. A. Henzinger, "A really temporal logic," *Journal of the ACM (JACM)*, vol. 41, no. 1, pp. 181–203, 1994.

[26] D. Ulus, T. Ferrère, E. Asarin, and O. Maler, "Timed pattern matching," in *Formal Modeling and Analysis of Timed Systems (FORMATS)*. Springer, 2014, pp. 222–236.

[27] A. Bakhirkin, T. Ferrère, O. Maler, and D. Ulus, "On the quantitative semantics of regular expressions over real-valued signals," in *Formal Modeling and Analysis of Timed Systems (FORMATS)*. Springer, 2017, pp. 189–206.

[28] E. Asarin, P. Caspi, and O. Maler, "Timed regular expressions," *Journal of the ACM*, vol. 49, no. 2, pp. 172–206, 2002.