Quantitative Monitoring of Signal First-Order Logic

Marek Chalupa[1], Thomas A. Henzinger[2], N. Ege Saraç[3], Emily Yu[4]

[1] Zeroth Research, United Kingdom
marek.chalupa@zeroth.org
[2] Institute of Science and Technology Austria, Austria
tah@ista.ac.at
[3] CISPA Helmholtz Center for Information Security, Germany
ege.sarac@cispa.de
[4] Leiden University, Netherlands
z.yu@liacs.leidenuniv.nl

Abstract. Runtime monitoring checks, during execution, whether a partial signal produced by a hybrid system satisfies its specification. Signal First-Order Logic (SFO) offers expressive real-time specifications over such signals, but currently comes only with Boolean semantics and has no tool support. We provide the first robustness-based quantitative semantics for SFO, enabling the expression and evaluation of rich real-time properties beyond the scope of existing formalisms such as Signal Temporal Logic. To enable online monitoring, we identify a past-time fragment of SFO and give a pastification procedure that transforms bounded-response SFO formulas into equisatisfiable formulas in this fragment. We then develop an efficient runtime monitoring algorithm for this past-time fragment and evaluate its performance on a set of benchmarks, demonstrating the practicality and effectiveness of our approach. To the best of our knowledge, this is the first publicly available prototype for online quantitative monitoring of full SFO.

Keywords: Signal first-order logic · Robustness-based quantitative semantics · Online runtime monitoring.

1 Introduction

Runtime verification checks executions of a system against formal specifications by monitoring the system's behavior during runtime. Online monitors are lightweight pieces of software that operate in parallel with the system, checking partial traces on the fly as the execution unfolds. In contrast, offline monitoring analyzes the complete trace only after the system has finished executing.

Previous work [5] introduced Signal First-Order Logic (SFO) as a rich specification language for expressing real-time properties of real-valued signals. SFO offers greater expressiveness than the widely used Signal Temporal Logic (STL) [27], enabling more precise modeling of complex system behaviors. For example, SFO

allows the formalization of bounded stabilization properties: following a significant disturbance, the signal must return to its pre-disturbance value within 10 timestamps. This level of expressiveness goes beyond traditional temporal logics and makes SFO well suited for reasoning about the dynamic behavior of hybrid systems. Technically, SFO combines first-order logic with linear arithmetic and uninterpreted unary function symbols to represent real-valued signals over time. Its syntax supports quantification over both time and value variables, allowing for rich and flexible property specification. However, this expressiveness comes at a cost: formulas whose satisfaction depends on unbounded future behavior are in general not efficiently monitorable online, which makes the full logic ill suited for practical runtime monitoring.

So far, the semantics of SFO have been defined only in Boolean terms, i.e., a system trajectory either satisfies or violates a specification [5]. In many real-world applications, especially those involving continuous dynamics and numerical quantities, such binary outcomes are often insufficient. Quantitative semantics, which measure how well or to what extent a specification is satisfied or violated, offer more informative feedback and can significantly improve decision making in practice.

At a high level, we develop the semantic foundations and algorithmic machinery needed for online quantitative monitoring of SFO. In particular, we make the following contributions:

1. *Quantitative semantics for SFO.* We introduce a robust quantitative semantics for SFO, generalizing the standard Boolean semantics and enabling quantitative evaluation of real-valued signals.
2. *Past-time fragment and pastification.* We define a syntactic fragment of SFO, called past-time SFO (pSFO), which is well suited for online monitoring. To enable its use in practice, we provide a pastification procedure that transforms bounded-response SFO formulas into equisatisfiable pSFO formulas.
3. *Online monitoring algorithm.* We propose an efficient algorithm for online monitoring of pSFO formulas over piecewise-linear signals. Our approach computes robustness values symbolically using polyhedral representations.
4. *Implementation and experiments.* We implement our monitoring algorithm in a prototype tool and evaluate its performance on several benchmark scenarios. The results demonstrate the effectiveness of our approach.

Related work. Signal Temporal Logic (STL) was introduced as a specification language for continuous signals [27], and there is now a large body of work on its quantitative semantics and monitoring algorithms. Robustness-based semantics for STL [15,14], based on signed distance to violation, has since been supported by efficient offline and online monitoring algorithms [13,11,12,28,25,19]. Our work is complementary to this STL-based line: we lift robustness-based monitoring from STL to full SFO and identify a past-time, bounded-response fragment that still admits efficient online algorithms.

Beyond STL, the work most closely related to ours is Signal First-Order Logic (SFO) [5]: Bakhirkin et al. define a Boolean semantics for SFO and an offline

monitoring algorithm, but do not release an implementation or report experimental results. Menghi et al. [30] introduce Restricted Signal First-Order Logic (RFOL), a syntactic fragment of SFO with quantitative semantics. RFOL forbids quantification over value variables, enforces that each subformula contains at most one free time variable, and requires all formulas to be closed. In contrast, we define a generic robustness-based quantitative semantics for *full* SFO and identify a fragment tailored for efficient online monitoring.

The extension of STL with freeze quantifiers (STL*) [10] can record the current signal value for later comparison. While STL* can express relative changes such as local extrema and oscillations, SFO is strictly more expressive: it can capture additional classes of signal-based properties, including punctual derivative constraints that STL* cannot express; see [9] for a detailed taxonomy and an expressiveness comparison of STL, STL*, and SFO.

In a different direction, Bakhirkin and Basset [4] propose a quantitative extension of STL with generalized sliding-window and until operators, tailored to express and monitor bounded-stabilization properties. However, this extension remains quantifier-free: it cannot bind time or value parameters and reuse them later in the formula, so it cannot capture SFO specifications that quantify over dynamically chosen event durations and relate these durations across different signals, such as rise-time properties (e.g., if signal f has a positive edge then signal g subsequently has a positive edge whose rise time is within 10% of that of f; see [5, Ex. 3]). Silvetti et al. [34] extend this framework with integral-based and filtered sliding-window operators, which strengthens its ability to express cumulative and conditional quantitative requirements (e.g., average power consumption restricted to periods where a mode is active). These arithmetic extensions are largely orthogonal: SFO extended with suitable arithmetic operators can encode such sliding-window constructs and, in addition, express duration-parameterized properties that these formalisms cannot capture.

Orthogonal to logic-specific robustness monitoring, recent work develops logic-independent notions of quantitative monitorability, characterizing when (and how well) quantitative trace properties can be monitored (or approximated from finite prefixes) [24,22,20]. Relatedly, quantitative analogs of the safety-liveness dichotomy and corresponding decompositions have been established for quantitative properties and automata [23,6,7]. These general frameworks provide semantic criteria for what is monitorable (possibly approximately), whereas we give a robustness semantics for full SFO and exact online monitoring algorithms.

To our knowledge, this paper is the first to provide a generic robustness semantics for full SFO, together with an online monitoring algorithm for a practically relevant past-time bounded-response fragment.

2 Signal First-Order Logic (SFO)

In this section, we recall Signal First-Order Logic (SFO) and its standard Boolean semantics [5]. A signal is a function $w : \mathbb{T} \to \mathbb{R}$ mapping a temporal domain $\mathbb{T} \subseteq \mathbb{R}$ to real values. Let $\mathcal{F}$ be a set of function symbols and $\mathcal{X} = \mathcal{T} \cup \mathcal{R}$ be a

set of variables, partitioned into time variables $\mathcal{T}$ and value variables $\mathcal{R}$. Both time terms τ and value terms ρ evaluate to real numbers. A valuation ν assigns values to variables $x \in \mathcal{X}$, denoted $[\![x]\!]_\nu$. Function symbols remain uninterpreted at the logical level; a signal trace w assigns a real-valued signal $[\![f]\!]_w$ to each function symbol $f \in \mathcal{F}$.

Definition 1 (Syntax). *The syntax of an SFO formula ϕ is defined by the following grammar:*

$$\phi ::= \theta < \theta \mid \neg\phi \mid \phi \vee \phi \mid \exists r \in R.\, \phi \mid \ \exists s \in I.\, \phi \qquad \theta ::= \rho \mid \tau$$
$$\rho ::= r \mid f(\tau) \mid n \mid \rho - \rho \mid \rho + \rho \qquad \tau ::= s \mid n \mid \tau - \tau \mid \tau + \tau$$

where $R \subseteq \mathbb{R}$ and $I \subseteq \mathbb{R}$ are intervals with rational end points, $n \in \mathbb{Q}$ is a rational number, $r \in \mathcal{R}$ is a value variable, $s \in \mathcal{T}$ is a time variable, and ρ and τ denote value and time terms, respectively.

Definition 2 (Boolean Semantics). *Consider an SFO formula ϕ, a trace w, a valuation ν, and a term θ. The satisfaction relation $w, \nu \models \phi$ and the value $[\![\theta]\!]_{w,\nu}$ are defined inductively as follows:*

$$w,\nu \models \theta_1 < \theta_2 \Leftrightarrow [\![\theta_1]\!]_{w,\nu} < [\![\theta_2]\!]_{w,\nu} \qquad [\![n]\!]_{w,\nu} = n \quad (n \in \mathbb{Q})$$
$$w,\nu \models \neg\phi \Leftrightarrow w,\nu \not\models \phi \qquad [\![x]\!]_{w,\nu} = \nu(x) \quad (x \in \mathcal{X})$$
$$w,\nu \models \phi_1 \vee \phi_2 \Leftrightarrow w,\nu \models \phi_1 \text{ or } w,\nu \models \phi_2 \qquad [\![\theta_1 \pm \theta_2]\!]_{w,\nu} = [\![\theta_1]\!]_{w,\nu} \pm [\![\theta_2]\!]_{w,\nu}$$
$$w,\nu \models \exists r \in R.\, \phi \Leftrightarrow \exists a \in R : w,\nu[r \leftarrow a] \models \phi \qquad [\![f(\tau)]\!]_{w,\nu} = [\![f]\!]_w\big([\![\tau]\!]_{w,\nu}\big)$$
$$w,\nu \models \exists s \in I.\, \phi \Leftrightarrow \exists a \in I : w,\nu[s \leftarrow a] \models \phi$$

We use standard syntactic abbreviations: $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$, $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$, and $\forall x \in S.\, \phi \equiv \neg\exists x \in S.\, \neg\phi$. Abbreviations for derived comparison operators ($\leq$, $\geq$, $=$) and absolute values ($|\rho|$) are defined as expected.

Remark 1 (Well-definedness). Formulas are interpreted over traces w with a temporal domain $\mathbb{T} \subseteq \mathbb{R}$. We only define the satisfaction relation $w, \nu \models \phi$ for pairs (w, ν) for which all signal accesses in ϕ are well-defined: whenever a term $f(\tau)$ occurs in ϕ, we require that the interpreted time $[\![\tau]\!]_{w,\nu}$ lies in $\mathbb{T}$. If this side condition fails, we leave $w, \nu \models \phi$ undefined. In particular, quantifiers range over those instantiations that yield well-defined signal accesses in the quantified subformula; if no such instantiation exists, the formula is undefined. Whenever we write $w, \nu \models \phi$, we implicitly assume that this side condition holds.

At the semantic level, a trace w interprets each function symbol f as a concrete real-valued function $[\![f]\!]_w$ on its temporal domain $\mathbb{T}$. We do not impose any additional assumptions on how these functions are represented or how values between samples are obtained; the semantics depends only on the resulting pointwise function values at the times where the formula accesses the signals.

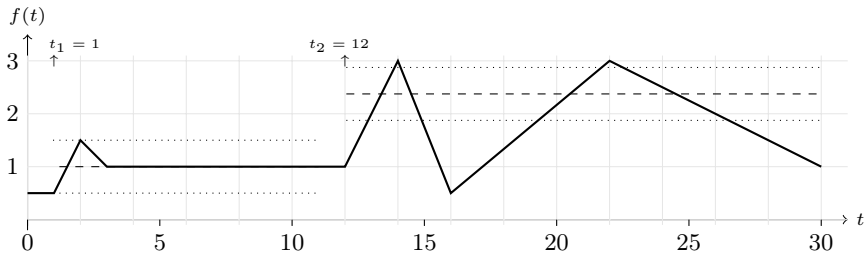

Figure 1: Illustration of the quantitative semantics for the bounded stabilization property given in Example 1. For convenience, we use piecewise-linear interpolation. Upward arrows indicate rising edges at $t_1 = 1$ and $t_2 = 12$. Following t_1, the signal f stabilizes rapidly within the ± 0.5 tolerance band around $r_1 = 1$. Conversely, following t_2, the signal exhibits wide oscillations that violate the band. The quantitative evaluation considers all 8-unit windows starting within 10 units of t_2, identifying the window with the minimum range to determine the maximal robustness value.

3 Quantitative Semantics of SFO

We introduce a quantitative semantics for SFO, assigning a robustness value to each formula relative to a specific signal and valuation. This score quantifies the degree to which a signal satisfies or violates a formula. We further establish the soundness of this semantics with respect to the Boolean semantics.

Definition 3 (Quantitative Semantics). *Given an SFO formula ϕ, a trace w, and a valuation ν, the* robustness value $\mu(w, \nu, \phi) \in \mathbb{R} \cup \{-\infty, +\infty\}$ *is defined inductively as follows.*

$$
\begin{aligned}
\mu(w, \nu, \theta_1 < \theta_2) &= [\![\theta_2]\!]_{w,\nu} - [\![\theta_1]\!]_{w,\nu} \\
\mu(w, \nu, \neg\phi) &= -\mu(w, \nu, \phi) \\
\mu(w, \nu, \phi_1 \vee \phi_2) &= \max\{\mu(w, \nu, \phi_1), \mu(w, \nu, \phi_2)\} \\
\mu(w, \nu, \exists r \in R.\phi) &= \sup_{a \in R} \mu(w, \nu[r \leftarrow a], \phi) \\
\mu(w, \nu, \exists s \in I.\phi) &= \sup_{a \in I} \mu(w, \nu[s \leftarrow a], \phi)
\end{aligned}
$$

The quantitative semantics follows the well-definedness convention from Remark 1: $\mu(w, \nu, \phi)$ is defined only when all signal accesses needed to evaluate ϕ are defined. In particular, in the quantified clauses, the supremum ranges only over those instantiations for which the quantified subformula has a defined robustness value; if no such instantiation exists, then $\mu(w, \nu, \phi)$ is undefined.

The definition above is independent of any particular interpolation scheme. When a trace is given as sampled data, an interpolation choice fixes the concrete functions $[\![f]\!]_w$ on $\mathbb{T}$. We illustrate the quantitative semantics of SFO using the bounded stabilization example from the introduction, using piecewise-linear interpolation purely for exposition.

Example 1 (Bounded Stabilization). Let $b : \mathbb{T} \to \{0, 1\}$ be a Boolean mode signal and $f : \mathbb{T} \to \mathbb{R}$ be a real-valued signal. A *rising edge* of b at time t is

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

Note that since Boolean signals are piecewise constant and right-continuous, isolated single-point spikes do not arise. The *bounded stabilization* requirement states that whenever b rises at t, the signal f must enter and remain within a tolerance band ± 0.5 around some value r. This stabilization must occur no later than $c \leq 10$ time units after t and persist for at least 8 time units:

$$\phi \equiv \uparrow b[t] \to \exists r \in \mathbb{R}. \exists c \in [0, 10]. \forall d \in [0, 8]. |f(t + c + d) - r| \leq 0.5.$$

Quantitative semantics allow for computing the degree to which a signal and valuation satisfy this property. For this example, we consider only time points where a rising edge occurs and focus on the consequent of the implication, denoted by ψ. Given a signal w and a valuation ν, the quantitative semantics $\mu(w, \nu, \psi)$ is given by:

$$\sup_{r \in \mathbb{R}} \sup_{c \in [0,10]} \inf_{d \in [0,8]} (0.5 - |f(t + c + d) - r|).$$

This expression captures the optimal robustness over all candidate stabilization levels $r \in \mathbb{R}$ and settling delays $c \in [0, 10]$. Specifically, the inner term represents the worst-case deviation from the ± 0.5 band around r over a window of size 8 starting at $t + c$. Intuitively, for a given t, we seek a target value r and a delay $c \in [0, 10]$ that minimize the signal's deviation from r in the subsequent 8-unit window. The robustness of ψ is upper-bounded by 0.5 (indicating perfect stabilization) and is unbounded from below.

Consider the signal in Fig. 1. After the first rising edge at $t_1 = 1$, the signal stabilizes perfectly around $r_1 = 1$ within 2 time units. Thus, the parameters $r = 1$ and $c = 2$ (for any $d \in [0, 8]$) maximize the robustness value to 0.5. Conversely, after the second rising edge at $t_2 = 12$, the signal oscillates widely and slowly. To evaluate quantitative satisfaction here, we examine windows of size 8 starting within 10 time units of t_2. For each candidate delay $c \in [0, 10]$, we consider the segment $[t_2 + c, t_2 + c + 8]$ and compute the range of f. The robustness value for each segment is $0.5 - \frac{1}{2}(\max f - \min f)$, and the overall robustness is the supremum over all feasible c and r. For the signal in Fig. 1, the smallest range of f over any such 8-unit window occurs in the interval $[19, 27]$, which places the peak at $t = 22$ in the window while the minima occur at the endpoints. Over this interval, the minimum and maximum values of f are 1.75 and 3.0, respectively, yielding a range of 1.25. Consequently, the optimal stabilization target is $r_2 = (1.75 + 3.0)/2 = 2.375$, and the corresponding robustness is $\mu(w, \nu[t \leftarrow 12], \psi) = 0.5 - \frac{1.25}{2} = -0.125$.

This negative value indicates that no matter how parameters r and c are chosen, the signal fails to remain within the ± 0.5 band around any constant level for a full 8-unit interval starting within 10 units of the rising edge. Intuitively,

the robustness value quantifies the signal's overshoot in the "best" window: even in the window with minimal range, the signal deviates by 0.125 outside the allowed band at its worst point. This represents the minimal uniform perturbation required to repair the signal to satisfy the property. Shifting the maximum down and the minimum up by 0.125 compresses the range sufficiently to meet the ± 0.5 requirement. Equivalently, the existential quantifiers synthesize parameters: for each fixed c, the maximizing r is the midrange of f on $[t+c, t+c+8]$, and the maximizing pair $(r^\star, c^\star)$ witnesses the best robustness value [2]. □

Remark 2 (Units and Compatibility). Because atomic predicates are interpreted as differences (i.e., $\theta_2 - \theta_1$), each atomic robustness value carries the units of the compared quantity, and we assume that atomic predicates are dimensionally well-formed. When a formula combines predicates over signals with different physical units, the max aggregator in disjunctions compares incompatible quantities, so the resulting robustness value lacks a direct physical interpretation. This limitation is inherited from the standard robustness semantics for STL [15,14] and is not specific to SFO. A practical mitigation is to evaluate all predicates on dimensionless quantities, e.g., by normalizing signals and thresholds using known operating bounds. Alternatively, one can adopt an explicitly normalized predicate-level score [30].

Finally, we relate the signed robustness in our quantitative semantics to the standard Boolean semantics.

Theorem 1 (Soundness of Quantitative Semantics). *Let ϕ be an SFO formula, w a signal trace, and ν a valuation. If $\mu(w, \nu, \phi) > 0$, then $w, \nu \models \phi$. If $\mu(w, \nu, \phi) < 0$, then $w, \nu \not\models \phi$.*

4 Past-Time SFO (pSFO)

Online monitoring is simplest for specifications whose evaluation at time t depends only on the prefix observed up to t. Past-time specifications have this property, so a monitor can emit verdicts (or robustness values) immediately, without buffering future samples. In this section, we isolate a past-time fragment of SFO and provide a procedure to translate SFO formulas with bounded future lookahead to pSFO formulas.

We call an SFO formula ϕ *temporal* if it has exactly one free time variable t, and every signal access is anchored at t: whenever $f(\tau)$ occurs in ϕ, the time term τ contains t with coefficient 1 (equivalently, τ can be rewritten as $t + \delta$ where t does not occur free in δ). This excludes references to fixed absolute times and ensures that each signal access is at a well-defined offset from the reference time t.

Definition 4 (Access time terms and pSFO). *Let ϕ be a temporal SFO formula with unique free time variable t. Let M_ϕ be the set of time terms τ such that some occurrence $f(\tau)$ appears in ϕ. For $\tau \in M_\phi$ and valuation ν,*

define the offset *of τ relative to t by $\mathit{off}_\nu(\tau) = [\![\tau]\!]_{\nu[t\leftarrow 0]}$. We call ϕ a* past-time SFO *(pSFO) formula if for every $\tau \in M_\phi$ and every valuation ν of bound time variables consistent with their quantification intervals (evaluated relative to $t = 0$), we have $\mathit{off}_\nu(\tau) \leq 0$.*

Intuitively, $\mathit{off}_\nu(\tau)$ is the time shift encoded by τ relative to the reference time t. The pSFO condition says that every signal read happens at time $t + \mathit{off}_\nu(\tau) \leq t$, i.e., never strictly in the future.

The *bounded-response* fragment of SFO consists of formulas where every time-quantifier interval I in $\exists s \in I.\, \psi$ is bounded, i.e., $\inf I \neq -\infty$ and $\sup I \neq \infty$. In this fragment, the offsets induced by bound time variables range over bounded sets, so the formula's time dependence can be summarized by finite lookahead and history bounds.

Below, we provide a pastification procedure for temporal bounded-response SFO formulas. Note that this does not preclude unbounded time entirely: the free time variable t ranges over the entire temporal domain, whereas quantified times act as *relative offsets* and are constrained to bounded ranges. This allows the expression of safety (and co-safety) properties whose violations (and satisfactions) are witnessed by input segments of bounded length, as is common in practice (e.g., when enforcing deadlines).

The key quantity for pastification is the *forward horizon*, an upper bound on how far into the future the formula may read relative to t. Dually, the *backward horizon* bounds how far into the past it may read. For bounded-response formulas, both quantities are finite.

Definition 5 (Horizons). *Let ϕ be a temporal bounded-response formula. For each $\tau \in M_\phi$, let $\ell_\tau = \inf_\nu \mathit{off}_\nu(\tau)$ and $u_\tau = \sup_\nu \mathit{off}_\nu(\tau)$, where ν ranges over valuations of the bound time variables consistent with their quantification intervals. We define the* forward *and* backward *horizons, respectively, as $H^+(\phi) = \max_{\tau \in M_\phi} \max(0, u_\tau)$ and $H^-(\phi) = \max_{\tau \in M_\phi} \max(0, -\ell_\tau)$, with the convention that $H^+(\phi) = H^-(\phi) = 0$ when $M_\phi = \emptyset$.*

Thus, $H^+(\phi)$ is the maximal future offset of any signal access in ϕ, and $H^-(\phi)$ is the maximal history length needed to evaluate ϕ at a given reference time. We now define the pastification operator. Without loss of generality, we assume that bound variables are named apart from free variables.

Definition 6 (Pastification). *For $d \geq 0$, the pastification of ϕ is $\Pi_d(\phi) = \phi[t \mapsto t - d]$, obtained by substituting every free occurrence of t by $(t - d)$.*

Pastification turns future signal reads into past reads by shifting the reference time backwards inside the formula. Concretely, if a signal access is at $t + \delta$ in ϕ, then it becomes $(t - d) + \delta = t + (\delta - d)$ in $\Pi_d(\phi)$; choosing d large enough makes all these new offsets non-positive.

Example 2 (Pastification for bounded stabilization). Recall the bounded stabilization property from Example 1: The rising-edge predicate $\uparrow b[t]$ contains an

unbounded time quantifier, so ϕ is not bounded-response. However, the consequent $\psi \equiv \exists r \in \mathbb{R}.\ \exists c \in [0,10].\ \forall d \in [0,8].\ |f(t+c+d) - r| \leq 0.5$ is bounded-response and can be pastified. Here $M_\psi = \{\, t+c+d \,\}$, and for any valuation ν consistent with $c \in [0,10]$ and $d \in [0,8]$ we have $\mathit{off}_\nu(t+c+d) = [\![t+c+d]\!]_{\nu[t\leftarrow 0]} = \nu(c)+\nu(d)$. Hence $\ell_{t+c+d} = 0$ and $u_{t+c+d} = 18$, so $H^+(\psi) = 18$ (and $H^-(\psi) = 0$). Applying Π_{18} yields

$$\Pi_{18}(\psi) \equiv \exists r \in \mathbb{R}.\ \exists c \in [0,10].\ \forall d \in [0,8].\ |f(t-18+c+d) - r| \leq 0.5.$$

Now every access time has offset $\mathit{off}_\nu(t-18+c+d) = \nu(c)+\nu(d)-18 \leq 0$, so $\Pi_{18}(\psi)$ is a pSFO formula. Evaluating $\Pi_{18}(\psi)$ at time t only reads f at times up to t, whereas evaluating ψ at time t would need f up to $t+18$. Moreover, ψ and $\Pi_{18}(\psi)$ are equisatisfiable via the reference-time shift $t \mapsto t+18$. □

Choosing d to be at least the forward horizon of the input ensures that all time references lie in the past, i.e., a pSFO formula. Finally, we establish that pastification preserves satisfiability up to a constant shift of the reference time.

Theorem 2 (Equisatisfiability under Pastification). *Let ϕ be a temporal bounded-response SFO formula with free time variable t, and let $h \geq H^+(\phi)$. For every trace w and valuation ν, we have $w, \nu \models \phi$ iff $w, \nu' \models \Pi_h(\phi)$, where $\nu'(t) = \nu(t) + h$ and $\nu'(x) = \nu(x)$ for all $x \neq t$.*

5 Online Quantitative Monitoring of pSFO

We present an online quantitative monitoring algorithm for bounded-response pSFO formulas, which capture a broad class of safety and co-safety properties. For any such formula, the monitor operates over a sliding window corresponding to the formula's backward horizon. As execution progresses, robustness values are updated incrementally using exclusively the data within this window. The construction is compositional, enabling the aggregation of monitors for pSFO subformulas while enforcing a bounded memory footprint. Furthermore, the outputs of these monitors can be combined via Boolean operations, extending the monitoring capability beyond safety and co-safety.

Following the well-definedness convention of Remark 1, the algorithm evaluates robustness only over valuations for which all signal accesses fall within the available trace domain; quantifier ranges that partly exceed the domain are effectively trimmed to the well-defined region.

When ϕ is a temporal formula with a unique free time variable t, we write $\mu(w, t_0, \phi)$ for $\mu\big(w, \nu[t \leftarrow t_0], \phi\big)$ to denote the *pointwise robustness* of ϕ at time $t_0 \in \mathbb{T}$, where ν is any valuation of the remaining (non-temporal) free variables of ϕ; if t is the only free variable, the choice of ν is irrelevant. Analogously, we write $w, t_0 \models \phi$ for $w, \nu[t \leftarrow t_0] \models \phi$. Since monitoring evaluates a formula at each time point along a trace, this pointwise view is the one we adopt throughout.

Definition 7 (Online Monitoring Problem). *The* online monitoring problem *for a temporal pSFO formula ϕ over a stream $w_1, w_2, \ldots$ of input segments*

requires producing an output segment v_i for each w_i. In Boolean monitoring*, v_i is a Boolean segment indicating whether ϕ holds pointwise. In* quantitative monitoring*, v_i is an extended-real-valued segment providing the pointwise robustness of ϕ, that is, a partial function $t \mapsto \mu(w, t, \phi) \in \mathbb{R} \cup \{-\infty, +\infty\}$ where defined.*

The core idea of the algorithm is to represent signals, terms, and formulas geometrically as collections of convex polyhedra defined by linear constraints. Each signal is encoded as a list of polyhedra over time and value variables; arithmetic operations translate into intersections and projections on these polyhedra; and Boolean connectives become polyhedral case-splits. This geometric perspective, which enables Boolean monitoring [5], forms the foundation for our quantitative extension: by maintaining an auxiliary variable that tracks the robustness of each subformula, we extract quantitative information directly from the polyhedral representation.

Throughout this section, we assume the input signal is sampled at regular time intervals and linearly interpolated between samples. The assumption of regular sampling is a presentational convenience: the polyhedral operations do not require uniform sampling, and extending to irregular time steps would only affect bookkeeping, not the algorithms themselves. Although uniform sampling simplifies certain specifications (for instance, property P2 in the obstacle avoidance experiments in Section 6 approximates a derivative using the fixed step size), SFO and our algorithm could be adapted to monitor such properties without this assumption.

We denote by $\sqcup$, $\sqcap$, Negate, and Eliminate the standard operations of union, intersection, complement, and projection, applied to individual polyhedra or lists thereof. Lists of polyhedra are maintained in increasing order of the upper bound of the free time variable t. The intersection operation denotes intersections between (i) two polyhedra, (ii) a polyhedron and a list (pairwise), and (iii) two ordered lists of polyhedra. These operations are efficiently implemented via the double description method [32,18].

Each function f is interpreted as a signal, represented by a list of convex polyhedra $\mathcal{P}_f$ parameterized by two free variables: t_f (time) and v_f (the value of f). These polyhedra are updated upon the arrival of new signal segments; consequently, we treat them as read-only global variables within helper functions.

Finally, we assume that input pSFO formulas are given in a normalized form: all first-order quantifiers are pushed inward as far as possible, and derived propositional connectives are eliminated; negations are pushed inward through propositional structure as far as possible, without pushing them across quantifiers. This preprocessing preserves the semantics and simplifies the monitoring construction, since quantified variables then appear in local scopes and each quantifier block can be handled independently. The rewriting is linear in formula size, and pushing quantifiers inward only narrows their scope to the subformulas that mention the bound variable, without duplicating any subformula, so normalization does not increase formula size.

Monitor (Algorithm 1) outlines the high-level procedure. The monitor maintains a history of signal polyhedra $\mathcal{P}_f$. Upon receiving segment w_i, it

Algorithm 1: Monitor

Require: pSFO formula ϕ with backward horizon h; sampling period Δ; initial signal values w_0; stream of signal segments $w_1, w_2, \ldots$
Ensure: stream of segments $v_1, v_2, \ldots$ encoding the robustness of w for ϕ

$\mathcal{P}_f \leftarrow \{\text{constraints from } w_0\}$ for all $f \in \mathcal{F}$
for $i \geq 1$ **do**
  Receive w_i
  $\mathcal{P}_f \leftarrow \mathcal{P}_f \sqcup \{\text{constraints from } w_i\}$ for all $f \in \mathcal{F}$
  $P_d \leftarrow \{\, t \in [(i-1)\Delta, i\Delta) \,\}$
  $(\mathcal{P}, v) \leftarrow$ FormulaRobust(ϕ, P_d)
  Output $v_i \leftarrow (\mathcal{P}, v)$
  Drop from $\mathcal{P}_f$ all signal pieces ending before $i\Delta - h$ for all $f \in \mathcal{F}$

Algorithm 2: FormulaRobust

Require: pSFO formula ϕ; domain-constraints P_d for the current segment
Ensure: list of polyhedra $\mathcal{P}$ and robustness variable v for a subformula on the current segment

if $\phi \equiv (\theta_1 < \theta_2)$ **then**
  $(\mathcal{P}', v') \leftarrow$ Term$(\theta_2 - \theta_1, P_d)$
  return $(\mathcal{P}', v')$
else if $\phi \equiv \neg\phi'$ **then**
  $(\mathcal{P}', v') \leftarrow$ FormulaRobust(ϕ', P_d)
  $v'' \leftarrow$ FreshVar()
  $\mathcal{P}'' \leftarrow \mathcal{P}' \sqcap \{\, v'' = -v' \,\}$
  return (Eliminate$(\{v'\}, \mathcal{P}''), v'')$
else if $\phi \equiv \exists r \in R : \phi'$ **then**
  $(\mathcal{P}', v') \leftarrow$ FormulaRobust$(\phi', P_d \sqcap \{r \in R\})$
  $(\mathcal{P}'', v'') \leftarrow$ EliminateBySup$(\mathcal{P}', v', r, R)$
  return $(\mathcal{P}'', v'')$
else if $\phi \equiv \exists s \in I : \phi'$ **then**
  $(\mathcal{P}', v') \leftarrow$ FormulaRobust$(\phi', P_d \sqcap \{s \in I\})$
  $(\mathcal{P}'', v'') \leftarrow$ EliminateBySup$(\mathcal{P}', v', s, I)$
  return $(\mathcal{P}'', v'')$
else if $\phi \equiv \phi_1 \vee \phi_2$ **then**
  $(\mathcal{P}_1, v_1) \leftarrow$ FormulaRobust(ϕ_1, P_d)
  $(\mathcal{P}_2, v_2) \leftarrow$ FormulaRobust(ϕ_2, P_d)
  $v' \leftarrow$ FreshVar()
  $\mathcal{P}^1_{\max} \leftarrow \mathcal{P}_1 \sqcap \mathcal{P}_2 \sqcap \{\, v_1 \geq v_2,\ v' = v_1 \,\}$
  $\mathcal{P}^2_{\max} \leftarrow \mathcal{P}_1 \sqcap \mathcal{P}_2 \sqcap \{\, v_1 < v_2,\ v' = v_2 \,\}$
  $\mathcal{P}' \leftarrow$ Eliminate$\big(\{v_1, v_2\},\ \mathcal{P}^1_{\max} \sqcup \mathcal{P}^2_{\max}\big)$
  return $(\mathcal{P}', v')$

Algorithm 3: Term

Require: a term θ, a polyhedron P_d collecting domain constraints
Ensure: list of polyhedra $\mathcal{P}$ and robustness variable v for the term on the current segment

if $\theta \equiv n$ **then**
  $v \leftarrow$ FreshVar()
  return $(\{\, v = n \,\} \sqcap P_d, v)$
else if $\theta \equiv r$ **then**
  $v \leftarrow$ FreshVar()
  return $(\{\, v = r \,\} \sqcap P_d, v)$
else if $\theta \equiv s$ **then**
  $v \leftarrow$ FreshVar()
  return $(\{\, v = s \,\} \sqcap P_d, v)$
else if $\theta \equiv \theta_1 \pm \theta_2$ **then**
  $(\mathcal{P}_1, v_1) \leftarrow$ Term(θ_1, P_d)
  $(\mathcal{P}_2, v_2) \leftarrow$ Term(θ_2, P_d)
  $v \leftarrow$ FreshVar()
  $\mathcal{P} \leftarrow (\mathcal{P}_1 \sqcap \mathcal{P}_2) \sqcap \{\, v = v_1 \pm v_2 \,\} \sqcap P_d$
  return (Eliminate$(\{v_1, v_2\}, \mathcal{P}), v)$
else if $\theta \equiv f(\tau)$ **then**
  $v \leftarrow$ FreshVar()
  return $(\mathcal{P}_f[t_f \mapsto \tau,\ v_f \mapsto v] \sqcap P_d, v)$

Algorithm 4: EliminateBySup

Require: list of polyhedra $\mathcal{P}$; current robustness variable v; quantified variable x; interval $I \subseteq \mathbb{R}$
Ensure: list of polyhedra $\mathcal{P}'$ (without x) and robustness variable v' for the quantified formula on the current segment

$\mathcal{P}' \leftarrow \emptyset$
$v' \leftarrow$ FreshVar()
for all $P \in \mathcal{P}$ **do**
  $P_I \leftarrow P \sqcap \{x \in I\}$
  $(P'_I, o) \leftarrow$ NormalizeObj(P_I, v, x)
  $(\mathcal{Q}, v') \leftarrow$ PlpMaximize(P'_I, o, x, v')
  $\mathcal{P}' \leftarrow \mathcal{P}' \sqcup \mathcal{Q}$
$\mathcal{P}' \leftarrow \{(Y, v') \in \mathcal{P}' \mid \neg\exists \hat{v} : (Y, \hat{v}) \in \mathcal{P}' \wedge \hat{v} > v'\}$
return $(\mathcal{P}', v')$

Algorithm 5: PlpMaximize

Require: polyhedron P over Y and x; affine objective o over (Y, x); quantified variable x; robustness v'
Ensure: list of polyhedra $\mathcal{Q}$ over Y with $v'(y) = \sup\{o(y, x) \mid (y, x) \in P\}$

$(\alpha, \beta) \leftarrow$ SplitCoeff(o, x)
$(\mathcal{L}, \mathcal{U}, P_0) \leftarrow$ IsolateBounds(P, x)
$P_Y \leftarrow$ Eliminate$(\{x\}, P)$; $\mathcal{Q} \leftarrow \emptyset$
$G^+ \leftarrow P_0 \sqcap \{\alpha > 0\}$
$G^- \leftarrow P_0 \sqcap \{\alpha < 0\}$
$G^0 \leftarrow P_0 \sqcap \{\alpha = 0\}$
if $G^+ \neq \emptyset$ **then**
  if $\mathcal{U} = \emptyset$ **then**
    $\mathcal{Q} \leftarrow \mathcal{Q} \sqcup (G^+ \sqcap P_Y \sqcap \{v' = +\infty\})$
  else
    for all $u \in \mathcal{U}$ **do**
      $A_u \leftarrow \bigsqcap_{u' \in \mathcal{U}} \{u \leq u'\}$
      $F_u \leftarrow \bigsqcap_{\ell \in \mathcal{L}} \{\ell \leq u\}$
      $\mathcal{Q} \leftarrow \mathcal{Q} \sqcup (G^+ \sqcap A_u \sqcap F_u \sqcap P_Y \sqcap \{v' = \alpha u + \beta\})$
if $G^- \neq \emptyset$ **then**
  if $\mathcal{L} = \emptyset$ **then**
    $\mathcal{Q} \leftarrow \mathcal{Q} \sqcup (G^- \sqcap P_Y \sqcap \{v' = +\infty\})$
  else
    for all $\ell \in \mathcal{L}$ **do**
      $A_\ell \leftarrow \bigsqcap_{\ell' \in \mathcal{L}} \{\ell \geq \ell'\}$
      $F_\ell \leftarrow \bigsqcap_{u \in \mathcal{U}} \{\ell \leq u\}$
      $\mathcal{Q} \leftarrow \mathcal{Q} \sqcup (G^- \sqcap A_\ell \sqcap F_\ell \sqcap P_Y \sqcap \{v' = \alpha\ell + \beta\})$
if $G^0 \neq \emptyset$ **then**
  $\mathcal{Q} \leftarrow \mathcal{Q} \sqcup (G^0 \sqcap P_Y \sqcap \{v' = \beta\})$
return $(\mathcal{Q}, v')$

appends constraints to $\mathcal{P}_f$, restricts the reference time to the current segment by constructing P_d, and calls FormulaRobust. The result is a list of polyhedra $\mathcal{P}$ together with a distinguished robustness variable v; this pair encodes the piecewise-affine robustness function on the current segment. Finally, using the finite backward horizon $h = H^-(\phi)$, the monitor discards signal pieces whose (right) endpoint is strictly smaller than $i\Delta - h$, which are never needed to evaluate ϕ for any $t \in [(i-1)\Delta, i\Delta)$.

FormulaRobust (Algorithm 2) recursively constructs polyhedra for subformulas, mapping logical operators to linear constraints. *Negation* introduces a fresh output variable and enforces $v'' = -v'$, then immediately projects out the old robustness variable; this keeps the representation functional (one designated robustness variable per returned subformula piece). *Disjunction* encodes the max operator by case-splitting on $v_1 \geq v_2$ versus $v_1 < v_2$ over the intersection of the two operand regions. *Existential quantification* computes the supremum of the robustness over the quantified variable's interval via EliminateBySup.

Term (Algorithm 3) transforms arithmetic terms into polyhedra annotated with a value variable. In all base cases (constants and variables), the returned polyhedron is conjoined with P_d so that any currently active domain constraints (e.g., quantifier bounds) are preserved. Sums and differences intersect subterm regions and add the defining equality for the output variable, then project away intermediate variables. For function terms $f(\tau)$, we substitute τ for the signal time variable t_f and constrain the signal value variable v_f to equal the fresh output variable.

EliminateBySup (Algorithm 4) eliminates a quantified variable x by taking the supremum of the current robustness variable v over an interval I. It first restricts to the closure $[\inf I, \sup I]$, which does not affect the supremum. Then, it applies a normalization step: NormalizeObj(P_I, v, x) extracts from P_I the affine expression $o(Y, x)$ that equals v on that piece and substitutes it to eliminate the robustness variable, yielding a reduced polyhedron P_I' over (Y, x) together with the explicit objective o. The reduced polyhedron and explicit objective are then passed to PlpMaximize. Finally, the last line removes dominated points so that, for each parameter valuation Y, only maximal v' values remain; this implements the supremum over a union of pieces as the pointwise maximum of the per-piece suprema.

PlpMaximize (Algorithm 5) performs parametric maximization of an affine objective $o = \alpha x + \beta$ over the feasible x-interval induced by a polyhedron $P(Y, x)$. The optimum is achieved at an extremal bound of x determined by the sign of α; if the relevant side is unbounded, the supremum is $+\infty$. If for some y there is no feasible x (i.e., the section P_y is empty), then $y \notin P_Y$ and the procedure produces no output polyhedron at y, leaving the robustness undefined there (cf. Remark 1).

Lines 1–6. SplitCoeff(o, x) decomposes the affine objective into $o = \alpha x + \beta$, where α and β are affine in the parameters Y. IsolateBounds(P, x) extracts the affine lower bounds $\mathcal{L}$ $(x \geq \ell(Y))$, upper bounds $\mathcal{U}$ $(x \leq u(Y))$, and the x-independent guard P_0. Projecting out x yields $P_Y = \{y \mid \exists x.(y, x) \in P\}$, defining

parameter regions where the feasible x-interval is nonempty. The algorithm partitions the parameter space by the sign of α: G^+ (positive slope, optimum at an upper bound), G^- (negative slope, optimum at a lower bound), and G^0 (zero slope).

Lines 7–14 ($\alpha > 0$). For positive slopes, the supremum occurs at the maximal feasible x. If $\mathcal{U} = \emptyset$, the feasible region is unbounded above and the objective diverges to $+\infty$, yielding $v' = +\infty$ on $G^+ \sqcap P_Y$. Otherwise, for each candidate $u \in \mathcal{U}$, we isolate the region where u is the tightest upper bound (A_u) and feasible w.r.t. all lower bounds (F_u). On this region the supremum is $v' = \alpha u + \beta$.

Lines 15–24 ($\alpha < 0$ and $\alpha = 0$). Symmetrically, for negative slopes the supremum occurs at the minimal feasible x. If $\mathcal{L} = \emptyset$, the feasible region is unbounded below and the objective diverges to $+\infty$ as $x \to -\infty$, yielding $v' = +\infty$ on $G^- \sqcap P_Y$. Otherwise, for each $\ell \in \mathcal{L}$, the region $A_\ell \sqcap F_\ell$ isolates the tightest feasible lower bound, and the supremum is $v' = \alpha\ell + \beta$. Finally, for $\alpha = 0$, the objective is independent of x, so $v' = \beta$ on $G^0 \sqcap P_Y$.

Example 3 (Supremum Elimination). Let $\psi \equiv \exists c \in [0,2]\,.\, 0 < f(t-c)$ and consider the samples $f(0) = 0$, $f(1) = -3$, $f(2) = -1$, $f(3) = 1$ with piecewise-linear interpolation: $f(s)$ is given by $-3s$ for $s \in [0,1)$, and by $2s-5$ for $s \in [1,2)$ and $s \in [2,3)$. We encode f as the list $\mathcal{P}_f$ of convex polyhedra over (t_f, v_f): $P_f^{(1)} = \{0 \le t_f < 1, v_f = -3t_f\}$, $P_f^{(2)} = \{1 \le t_f < 2, v_f = 2t_f - 5\}$, and finally $P_f^{(3)} = \{2 \le t_f < 3, v_f = 2t_f - 5\}$. We monitor on the current segment with bounds $\mathcal{P}_d = \{2 \le t < 3, 0 \le c \le 2\}$. Applying Term to the atom $0 < f(t-c)$ substitutes $\tau := t - c$ for t_f and introduces a fresh value variable v. For each piece of $\mathcal{P}_f$, this produces constraints in (t, c, v), which we intersect with $\mathcal{P}_d$ and simplify for $t \in [2,3)$:

$$\begin{aligned}
&\text{(A)}\ \ 2 \le t < 3,\ t-1 \le c \le 2,\ v = 3c - 3t\\
&\text{(B)}\ \ 2 \le t < 3,\ t-2 \le c \le t-1,\ v = -2c + 2t - 5,\\
&\text{(C)}\ \ 2 \le t < 3,\ 0 \le c \le t-2,\ v = -2c + 2t - 5.
\end{aligned}$$

Hence, FormulaRobust returns their union with v as the atomic robustness.

The quantifier $\exists c \in [0,2]$ is handled by EliminateBySup, which first conjoins $c \in [0,2]$ and then normalizes each piece: NormalizeObj extracts the affine objective $o(t,c)$ (here, simply $o = v$ as given by the displayed equalities) and projects out v. Then PlpMaximize writes $o = \alpha c + \beta$ (via SplitCoeff) and isolates the affine lower and upper bounds on c.

For (A) we have $t - 1 \le c \le 2$, so $\mathcal{L} = \{t-1\}$, $\mathcal{U} = \{2\}$ and $\alpha = +3 > 0$ (the G^+ branch). Among the candidate upper bounds, the guard $A_u \wedge F_u$ with $u = 2$ is the only nonempty one (feasible for all $t \in [2,3)$, since $t - 1 \le 2$), so the supremum is realized at $u = 2$, yielding

$$v'_A(t) = \alpha u + \beta = 3 \cdot 2 + (-3t) = -3t + 6 \quad (t \in [2,3)).$$

For (B) we have $t-2 \le c \le t-1$, so $\mathcal{L} = \{t-2\}, \mathcal{U} = \{t-1\}$ and $\alpha = -2 < 0$ (the G^- branch). Among the candidate lower bounds, the guard $A_\ell \wedge F_\ell$ with

$\ell = t-2$ is the only nonempty one (feasible for all $t \in [2,3)$, since $t-2 \geq 0$ and $t-2 \leq t-1$), so the supremum is realized at $\ell = t-2$, yielding

$$v'_B(t) = \alpha\ell + \beta = -2(t-2) + (2t-5) = -1 \quad (t \in [2,3)).$$

For (C) we have $0 \leq c \leq t-2$, so $\mathcal{L} = \{0\}, \mathcal{U} = \{t-2\}$ and again $\alpha = -2 < 0$ (the G^- branch). Among the candidate lower bounds, the guard $A_\ell \wedge F_\ell$ with $\ell = 0$ is the only nonempty one (feasible for all $t \in [2,3)$, since $0 \leq t-2$), so the supremum is realized at $\ell = 0$, yielding

$$v'_C(t) = \alpha\ell + \beta = -2 \cdot 0 + (2t-5) = 2t-5 \quad (t \in [2,3)).$$

Because $t \in [2,3)$ and $c \in [0,2]$, every evaluation time $\tau = t - c$ lies in $[0,3)$, so all three pieces $P_f^{(1)}, P_f^{(2)}, P_f^{(3)}$ are feasible on this segment.

Eliminating c over the union corresponds to taking the pointwise maximum of the three affine outcomes $v'_A(t) = -3t+6$, $v'_B(t) = -1$, and $v'_C(t) = 2t-5$. For all $t \in [2,3)$ we have $v'_B(t) = -1 \leq \max\{v'_A(t), v'_C(t)\}$, so v'_B never attains the supremum. Hence the robustness is given by

$$g(t) = \mu(w,t,\psi) = \max\{-3t+6,\ 2t-5\}.$$

The intersection $-3t+6 = 2t-5$ occurs at $t = 11/5 = 2.2$, so we obtain

$$g(t) = \begin{cases} -3t+6, & t \in [2, 11/5), \\ 2t-5, & t \in [11/5, 3). \end{cases}$$

Equivalently, over (t, v'), the monitor outputs

$$\{2 \leq t < 11/5,\ v' = -3t+6\} \sqcup \{11/5 \leq t < 3,\ v' = 2t-5\}.$$

Note that if we instead quantify $\exists c \in [1,2]$, then for $t \in [0,1)$ we have $\tau = t - c \in [-2,0)$, which lies outside the available trace domain. In this case, the polyhedral construction yields an empty projection onto t for that region, and therefore v' is undefined on $[0,1)$ (cf. Remark 1). □

Finally, we show that our monitoring algorithm is correct by establishing the correctness of each subroutine. We first prove PlpMaximize correct by characterizing the supremum of an affine function over a polyhedral interval, which is always realized at a boundary determined by affine guards. This result extends directly to EliminateBySup, which applies the same mechanism to each piece of a polyhedral union and then takes pointwise maxima via dominated-point removal. We prove afterward that Term is correct by structural induction on terms, since each case preserves the intended affine relation between the output variable and the term's value. Similarly, we handle FormulaRobust by induction on formulas, showing that each logical operator is correctly mapped to its quantitative counterpart; in particular, disjunction intersects the domains of both operands, enforcing the uniform well-definedness convention (Remark 1),

while existential quantification ranges only over instantiations for which the robustness of the quantified subformula is defined. The monitor's correctness follows by composing these results: (i) at each iteration the stored signal pieces cover the backward horizon window $[t-h,t]$ for every t in the current segment, so all subroutine preconditions are met, and (ii) garbage collection preserves this invariant for all future iterations while ensuring bounded memory.

Theorem 3 (Correctness of MONITOR (Algorithm 1)). *Let ϕ be a pSFO formula with a backward horizon of h, and let w be a piecewise-linear signal sampled with period Δ. For every $i \geq 1$ and every $t \in [(i-1)\Delta, i\Delta)$ such that $\mu(w,t,\phi)$ is defined, MONITOR outputs v_i satisfying $v_i(t) = \mu(w,t,\phi)$. For every $t \in [(i-1)\Delta, i\Delta)$ such that $\mu(w,t,\phi)$ is undefined, $v_i(t)$ is undefined. Moreover, whenever $v_i(t)$ is defined, it depends only on the restriction of w to $[t-h,t]$.*

6 Experiments

We implemented our algorithm in Python using *SymPy* [31] for symbolic mathematics and *pplpy* [1], a Python wrapper for the *Parma Polyhedra Library* [3]. The implementation[1] provides an SFO parser, polyhedra representations and operations, and the monitoring algorithm.

Our experiments address a basic feasibility question: can the proposed online monitor compute robustness values fast enough to keep up with the sampling rate of representative cyber-physical benchmarks? Concretely, we measure the per-segment runtime of robustness computation and examine how it scales with specification complexity, in particular when moving from horizon-0 (history-free) constraints to longer-horizon formulas, and to formulas with nested quantifiers or Boolean combinations of multiple properties.

All experiments were executed on a MacBook Pro M4 (macOS Sequoia 15.7.1) with 48 GB of memory and 4.5 GHz CPU. Although our algorithm is online, experiments were run offline on pre-recorded CSV files for reproducibility.

6.1 Benchmarks and Properties

(1) *Dynamic Obstacle Avoidance.* The first benchmark [17,33] involves a parcel delivery drone flying through an urban environment populated by 8 other drones that act as obstacles. The ego drone is controlled by a neural-network policy, while the remaining drones follow pre-defined trajectories. The environment's state space is eight-dimensional, defined by the vector $x = [x, y, z, v_x, v_y, v_z, \theta_x, \theta_y]$, which includes the drone's three spatial coordinates, three velocity components, and two angular variables representing roll and pitch. The control inputs are angular accelerations of θ_x and θ_y, as well as the vertical thrust T. The control interval is $0.1s$. We check three different properties in this benchmark:

[1] `https://github.com/ista-vamos/artifact-fm-qsfo`

Table 1: The average time in seconds of computing the robustness value for each new segment.

Avoidance				*Altitude Control*			
P1	*P2*	*P3*	*P1 + P2 + P3*	*P1*	*P2*	*P3*	*P1 + P2 + P3*
0.002	0.005	0.014	0.021	0.002	0.002	0.210	0.532

P1: *Safe velocity.* From time t_s onward, the absolute value of velocity in every direction must be less than $v_{max} m/s$. This property corresponds to the formula $t > t_s \implies (|v_x(t)| < v_{max} \wedge |v_y(t)| < v_{max} \wedge |v_z(t)| < v_{max})$.

P2: *Motion smoothness.* The time-derivative of velocity is bounded: $|\dot{v_x}| \leq a_{max}$, $|\dot{v_y}| \leq a_{max}$, $|\dot{v_z}| \leq a_{max}$. Because we have fixed sampling of $0.1s$, we can approximate $\dot{f}$ as $\frac{(f(t)-f(t-0.1)}{0.1}$ and then rewrite the whole inequality to the expression $|f(t) - f(t-0.1)| \leq c$ where $c = 0.1 \cdot a_{max}$. We monitor the conjunction of these inequalities for each derivative above.

P3: *Obstacle separation.* For all obstacles, the distance in every direction must be greater than d_{min} at all times.

(2) *Altitude Control.* This benchmark [21] models a simplified F-16 fighter jet that must maintain a safe altitude above the ground. The state space is represented by the vector $x = [v_a, \alpha, \beta, \phi, pitch, yaw, \dot{\phi}, \dot{pitch}, \dot{yaw}, n, e, alt, p_{lag}]$ which includes the airspeed (v_a), angle of attack (α), sideslip angle (β), roll (ϕ), pitch, yaw, roll rate, pitch rate, yaw rate, northward and eastward displacements, altitude (alt), engine power lag. The sampling time of this environment is fixed at 0.033 s. We monitor the following properties:

P1: *Safe altitude.* The altitude shall satisfy $300m \leq \text{alt}(t) \leq 13000m$ at all times.

P2: *Safe airspeed.* The airspeed shall remain within a safe range $v_{\min} \leq v_a(t) \leq v_{\max}$ for all t.

P3: *Altitude recovery.* Whenever $\text{alt}(t) < 500\,\text{m}$, then within τ_{rec} the altitude shall be at least 700 m and remain so for at least τ_{hold}. We monitor the formula $alt(t) < 500 \implies \exists t_r \in [0, \tau_{rec}].\forall t_h \in [0, \tau_{hold}].\, alt(t+t_r+t_h) \geq 700$.

6.2 Results and Discussion

We generated 100 simulation traces from both scenarios with randomized initial states. For *Dynamic Obstacle Avoidance*, each simulation either stops when the drone collides into nearby drones or reaches a maximum of 1000 simulation steps, whereas the *Altitude Control* simulation has a uniform length of 1000 steps. The results of our experiments are summarized in Table 1. For the *Dynamic Obstacle Avoidance* scenario, the time of analyzing each observed segment is smaller than the control time (0.1 s) by a margin. This means that the monitor is timely enough to be used for making control decisions. One of the reasons

for such low computation times is that formulas of properties *P1* and *P3* have horizon 0, which means we do not have to remember and analyze any history. For property *P2* (and combination of all properties *P1 + P2 + P3*), the horizon is 0.1 which means remembering one previous segment is sufficient for evaluating the formulas.

For the *Altitude Control* benchmark, first two properties are very similar to the first property of the avoidance benchmark, and therefore the times are similar. Property *P3*, however, is more interesting as the formula contains quantifiers. We used constants $\tau_{rec} = 10$ and $\tau_{hold} = 10$, which means that the formula has horizon 20. Since sampling rate is $0.033\,s$, the monitor has to remember more than 600 last segments. Still, it was able to analyze each new segment in around $0.2\,s$. Monitoring all three properties together (*P1 + P2 + P3*) increases the runtime further as the monitor has to compute symbolically minima and maxima over conjunctions and disjunctions. These computation times are not low enough to be sufficient for control, but are sufficient for raising warnings.

7 Conclusion

We introduced the first robustness-based quantitative semantics for Signal First-Order Logic (SFO) and an online monitoring algorithm for a past-time, bounded-response fragment. Our approach combines pastification with symbolic polyhedral computations to obtain robustness values for expressive SFO specifications over piecewise-linear signals. A prototype implementation and experiments on representative benchmarks demonstrate that quantitative SFO monitoring is feasible in practice and scales to non-trivial specifications.

Promising directions for future work include using robustness as an objective for controller synthesis, for example as a reward signal in reinforcement learning for cyber-physical systems [29]. We also aim to develop compositional and decentralized monitoring for multi-agent and distributed settings, where approximate monitoring can improve scalability [8]. Another natural extension is integrating SFO monitoring with runtime enforcement [16] (a.k.a. shielding [26]), where the monitor's low latency relative to the control period could enable a feedback loop that steers the controller away from violations.

Acknowledgments We thank the anonymous reviewers for their helpful comments. This work was supported by the European Research Council (ERC) Grants VAMOS (No. 101020093) and HYPER (No. 101055412), and by the Advanced Research and Invention Agency under the Safeguarded AI programme.

References

1. pplpy: Python interface to the parma polyhedra library. `https://pypi.org/project/pplpy/`. Accessed: 2025-12-05.

2. Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. Parametric identification of temporal properties. In Sarfraz Khurshid and Koushik Sen, editors, *Runtime Verification - Second International Conference, RV 2011, San Francisco, CA, USA, September 27-30, 2011, Revised Selected Papers*, volume 7186 of *Lecture Notes in Computer Science*, pages 147–160. Springer, 2011.
3. Roberto Bagnara, Patricia M Hill, Enea Zaffanella, and Abramo Bagnara. The parma polyhedra library. *Sci. Comput. Program*, 72, 2006.
4. Alexey Bakhirkin and Nicolas Basset. Specification and efficient monitoring beyond STL. In Tomás Vojnar and Lijun Zhang, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part II*, volume 11428 of *Lecture Notes in Computer Science*, pages 79–97. Springer, 2019.
5. Alexey Bakhirkin, Thomas Ferrère, Thomas A. Henzinger, and Dejan Nickovic. The first-order logic of signals: keynote. In Björn B. Brandenburg and Sriram Sankaranarayanan, editors, *Proceedings of the International Conference on Embedded Software, EMSOFT 2018, Torino, Italy, September 30 - October 5, 2018*, page 1. IEEE, 2018.
6. Udi Boker, Thomas A. Henzinger, Nicolas Mazzocchi, and N. Ege Saraç. Safety and liveness of quantitative automata. In Guillermo A. Pérez and Jean-François Raskin, editors, *34th International Conference on Concurrency Theory, CONCUR 2023, Antwerp, Belgium, September 18-23, 2023*, volume 279 of *LIPIcs*, pages 17:1–17:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.
7. Udi Boker, Thomas A. Henzinger, Nicolas Mazzocchi, and N. Ege Saraç. Safety and liveness of quantitative properties and automata. *Log. Methods Comput. Sci.*, 21(2), 2025.
8. Borzoo Bonakdarpour, Anik Momtaz, Dejan Nickovic, and N. Ege Saraç. Approximate distributed monitoring under partial synchrony: Balancing speed & accuracy. In Erika Ábrahám and Houssam Abbas, editors, *Runtime Verification - 24th International Conference, RV 2024, Istanbul, Turkey, October 15-17, 2024, Proceedings*, volume 15191 of *Lecture Notes in Computer Science*, pages 282–301. Springer, 2024.
9. Chaima Boufaied, Maris Jukss, Domenico Bianculli, Lionel Claude Briand, and Yago Isasi Parache. Signal-based properties of cyber-physical systems: Taxonomy and logic-based characterization. *J. Syst. Softw.*, 174:110881, 2021.
10. Lubos Brim, Petr Dluhos, David Safránek, and Tomas Vejpustek. Stl*: Extending signal temporal logic with signal-value freezing operator. *Inf. Comput.*, 236:52–67, 2014.
11. Jyotirmoy V. Deshmukh, Alexandre Donzé, Shromona Ghosh, Xiaoqing Jin, Garvit Juniwal, and Sanjit A. Seshia. Robust online monitoring of signal temporal logic. *Formal Methods Syst. Des.*, 51(1):5–30, 2017.
12. Adel Dokhanchi, Bardh Hoxha, and Georgios Fainekos. On-line monitoring for temporal logic robustness. In *RV*, volume 8734 of *Lecture Notes in Computer Science*, pages 231–246. Springer, 2014.
13. Alexandre Donzé, Thomas Ferrère, and Oded Maler. Efficient robust monitoring for STL. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 264–279. Springer, 2013.

14. Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In Krishnendu Chatterjee and Thomas A. Henzinger, editors, *Formal Modeling and Analysis of Timed Systems - 8th International Conference, FORMATS 2010, Klosterneuburg, Austria, September 8-10, 2010. Proceedings*, volume 6246 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2010.
15. Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theor. Comput. Sci.*, 410(42):4262–4291, 2009.
16. Yliès Falcone, Laurent Mounier, Jean-Claude Fernandez, and Jean-Luc Richier. Runtime enforcement monitors: composition, synthesis, and enforcement abilities. *Formal Methods Syst. Des.*, 38(3):223–262, 2011.
17. Thor I Fossen. A survey on nonlinear ship control: From theory to practice. *IFAC Proceedings Volumes*, 33(21):1–16, 2000.
18. Komei Fukuda and Alain Prodon. Double description method revisited. In Michel Deza, Reinhardt Euler, and Yannis Manoussakis, editors, *Combinatorics and Computer Science, 8th Franco-Japanese and 4th Franco-Chinese Conference, Brest, France, July 3-5, 1995, Selected Papers*, volume 1120 of *Lecture Notes in Computer Science*, pages 91–111. Springer, 1995.
19. Ebru Aydin Gol. Efficient online monitoring and formula synthesis with past STL. In *5th International Conference on Control, Decision and Information Technologies, CoDIT 2018, Thessaloniki, Greece, April 10-13, 2018*, pages 916–921. IEEE, 2018.
20. Felipe Gorostiaga and César Sánchez. General monitorability of totally ordered verdict domains. *Innov. Syst. Softw. Eng.*, 21(2):673–686, 2025.
21. Peter Heidlauf, Alexander Collins, Michael Bolender, and Stanley Bak. Verification challenges in f-16 ground collision avoidance and other automated maneuvers. *ARCH@ ADHS*, 2018, 2018.
22. Thomas A. Henzinger, Nicolas Mazzocchi, and N. Ege Saraç. Abstract monitors for quantitative specifications. In Thao Dang and Volker Stolz, editors, *Runtime Verification - 22nd International Conference, RV 2022, Tbilisi, Georgia, September 28-30, 2022, Proceedings*, volume 13498 of *Lecture Notes in Computer Science*, pages 200–220. Springer, 2022.
23. Thomas A. Henzinger, Nicolas Mazzocchi, and N. Ege Saraç. Quantitative safety and liveness. In Orna Kupferman and Pawel Sobocinski, editors, *Foundations of Software Science and Computation Structures - 26th International Conference, FoSSaCS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22-27, 2023, Proceedings*, volume 13992 of *Lecture Notes in Computer Science*, pages 349–370. Springer, 2023.
24. Thomas A. Henzinger and N. Ege Saraç. Quantitative and approximate monitoring. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–14. IEEE, 2021.
25. Stefan Jaksic, Ezio Bartocci, Radu Grosu, Thang Nguyen, and Dejan Nickovic. Quantitative monitoring of STL with edit distance. *Formal Methods Syst. Des.*, 53(1):83–112, 2018.
26. Bettina Könighofer, Mohammed Alshiekh, Roderick Bloem, Laura R. Humphrey, Robert Könighofer, Ufuk Topcu, and Chao Wang. Shield synthesis. *Formal Methods Syst. Des.*, 51(2):332–361, 2017.
27. Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In Yassine Lakhnech and Sergio Yovine, editors, *Formal Techniques, Mod-*

elling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems, FORMATS 2004 and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24, 2004, Proceedings, volume 3253 of *Lecture Notes in Computer Science*, pages 152–166. Springer, 2004.

28. Oded Maler and Dejan Nickovic. Monitoring properties of analog and mixed-signal circuits. *Int. J. Softw. Tools Technol. Transf.*, 15(3):247–268, 2013.
29. Yue Meng and Chuchu Fan. Signal temporal logic neural predictive control. *IEEE Robotics Autom. Lett.*, 8(11):7719–7726, 2023.
30. Claudio Menghi, Shiva Nejati, Khouloud Gaaloul, and Lionel C. Briand. Generating automated and online test oracles for simulink models with continuous and uncertain behaviors. In *ESEC/SIGSOFT FSE*, pages 27–38. ACM, 2019.
31. Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.
32. T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall. *3. The Double Description Method*, pages 51–74. Princeton University Press, Princeton, 1953.
33. Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. Learning safe multi-agent control with decentralized neural barrier certificates. *arXiv preprint arXiv:2101.05436*, 2021.
34. Simone Silvetti, Michele Loreti, and Laura Nenzi. Modular and online monitoring of temporal logic specification with integral and filter. In Bettina Könighofer and Hazem Torfah, editors, *Runtime Verification - 25th International Conference, RV 2025, Graz, Austria, September 15-19, 2025, Proceedings*, volume 16087 of *Lecture Notes in Computer Science*, pages 120–139. Springer, 2025.

A Omitted Proofs

A.1 Proof of Theorem 1 (Soundness of Quantitative Semantics)

Statement. Let ϕ be an SFO formula, w a signal trace, and v a valuation. If $\mu(w, v, \phi) > 0$, then $w, v \models \phi$. If $\mu(w, v, \phi) < 0$, then $w, v \not\models \phi$.

Proof. We prove both implications simultaneously by structural induction on the structure of ϕ.

Atomic case. Let $\phi \equiv (\theta_1 < \theta_2)$. By definition, $\mu(w, v, \theta_1 < \theta_2) = [\![\theta_2]\!]_{w,v} - [\![\theta_1]\!]_{w,v}$. If $\mu(w, v, \phi) > 0$, then $[\![\theta_1]\!]_{w,v} < [\![\theta_2]\!]_{w,v}$, hence $w, v \models \theta_1 < \theta_2$. If $\mu(w, v, \phi) < 0$, then $[\![\theta_1]\!]_{w,v} > [\![\theta_2]\!]_{w,v}$, hence $w, v \not\models \theta_1 < \theta_2$.

Negation. Let $\phi \equiv \neg\psi$. Then $\mu(w, v, \neg\psi) = -\mu(w, v, \psi)$. If $\mu(w, v, \neg\psi) > 0$, then $\mu(w, v, \psi) < 0$ and by the inductive hypothesis we have $w, v \not\models \psi$, hence $w, v \models \neg\psi$. If $\mu(w, v, \neg\psi) < 0$, then $\mu(w, v, \psi) > 0$ and by the inductive hypothesis we have $w, v \models \psi$, hence $w, v \not\models \neg\psi$.

Disjunction. Let $\phi \equiv \phi_1 \vee \phi_2$. Then $\mu(w, v, \phi_1 \vee \phi_2) = \max\{\mu(w, v, \phi_1), \mu(w, v, \phi_2)\}$. If $\mu(w, v, \phi_1 \vee \phi_2) > 0$, then at least one of $\mu(w, v, \phi_1)$ or $\mu(w, v, \phi_2)$ is strictly greater than 0. By the inductive hypothesis, the corresponding disjunct is satisfied, and therefore $w, v \models \phi_1 \vee \phi_2$. If $\mu(w, v, \phi_1 \vee \phi_2) < 0$, then both $\mu(w, v, \phi_1) < 0$ and $\mu(w, v, \phi_2) < 0$. By the inductive hypothesis, $w, v \not\models \phi_1$ and $w, v \not\models \phi_2$, hence $w, v \not\models \phi_1 \vee \phi_2$.

Existential quantification. We treat $\exists r \in R.\, \psi$ and $\exists s \in I.\, \psi$ uniformly. Let $\phi \equiv \exists x \in J.\, \psi$, where x is either a value variable r (with $J = R$) or a time variable s (with $J = I$). Let $D := \{\, a \in J \mid \mu(w, v[x \leftarrow a], \psi) \text{ is defined} \,\}$. By the well-definedness convention, we have $D \neq \emptyset$, and $\mu(w, v, \exists x \in J.\, \psi) = \sup_{a \in D} \mu(w, v[x \leftarrow a], \psi)$. If $\mu(w, v, \exists x \in J.\, \psi) > 0$, then there exists some $a \in D$ with $\mu(w, v[x \leftarrow a], \psi) > 0$. By the inductive hypothesis, $w, v[x \leftarrow a] \models \psi$, hence $w, v \models \exists x \in J.\, \psi$. If $\mu(w, v, \exists x \in J.\, \psi) < 0$, then for all $a \in D$ we have $\mu(w, v[x \leftarrow a], \psi) < 0$. By the inductive hypothesis, for all $a \in D$ we have $w, v[x \leftarrow a] \not\models \psi$, hence $w, v \not\models \exists x \in J.\, \psi$. □

A.2 Proof of Theorem 2 (Equisatisfiability Under Pastification)

Statement. Let ϕ be a temporal bounded-response SFO formula with free time variable t, and let $h \geq H^+(\phi)$. For every trace w and valuation v, we have $w, v \models \phi$ iff $w, v' \models \Pi_h(\phi)$, where $v'(t) = v(t) + h$ and $v'(x) = v(x)$ for all $x \neq t$.

Proof. Since $\Pi_h(\cdot)$ is defined by substituting the free variable t with $(t - h)$, we apply the same substitution to terms and write $\Pi_h(\theta)$ for the term obtained from θ by replacing each occurrence of t by $(t - h)$.

First, we claim that for every (time or value) term θ,

$$[\![\Pi_h(\theta)]\!]_{w,v'} \;=\; [\![\theta]\!]_{w,v}. \tag{1}$$

We prove (1) by structural induction on θ. If θ is a rational constant n, then $\Pi_h(n) = n$ and both sides equal n. If θ is a variable $x \neq t$, then $\Pi_h(x) = x$

and $[\![\Pi_h(x)]\!]_{w,v'} = v'(x) = v(x) = [\![x]\!]_{w,v}$. If $\theta = t$, then $\Pi_h(t) = t - h$ and $[\![t-h]\!]_{w,v'} = v'(t)-h = (v(t)+h)-h = v(t) = [\![t]\!]_{w,v}$. If $\theta \equiv \theta_1 \pm \theta_2$, then (1) follows directly from the induction hypothesis and the compositional semantics of $\pm$. Finally, if $\theta \equiv f(\tau)$, then by the induction hypothesis applied to the time term τ, we have $[\![\Pi_h(f(\tau))]\!]_{w,v'} = [\![f(\Pi_h(\tau))]\!]_{w,v'} = [\![f]\!]_w\big([\![\Pi_h(\tau)]\!]_{w,v'}\big) = [\![f]\!]_w\big([\![\tau]\!]_{w,v}\big) = [\![f(\tau)]\!]_{w,v}$. This completes the proof of (1). As an immediate consequence, for every occurrence of a signal access $f(\tau)$ in ϕ, the access time $[\![\tau]\!]_{w,v}$ equals the corresponding access time $[\![\Pi_h(\tau)]\!]_{w,v'}$ in $\Pi_h(\phi)$; hence the well-definedness condition holds for (w, v, ϕ) iff it holds for $(w, v', \Pi_h(\phi))$.

We now prove by structural induction on ϕ that $w, v \models \phi$ iff $w, v' \models \Pi_h(\phi)$.

Atomic case. Let $\phi \equiv (\theta_1 < \theta_2)$. Using (1) for θ_1 and θ_2, we have $w, v \models \theta_1 < \theta_2$ iff $[\![\theta_1]\!]_{w,v} < [\![\theta_2]\!]_{w,v}$ iff $[\![\Pi_h(\theta_1)]\!]_{w,v'} < [\![\Pi_h(\theta_2)]\!]_{w,v'}$ iff $w, v' \models \Pi_h(\theta_1 < \theta_2)$.

Negation. If $\phi \equiv \neg\psi$, then we have $w, v \models \neg\psi$ iff $w, v \not\models \psi$ iff $w, v' \not\models \Pi_h(\psi)$ iff $w, v' \models \neg\Pi_h(\psi)$ iff $w, v' \models \Pi_h(\neg\psi)$, where the middle equivalence uses the induction hypothesis for ψ.

Disjunction. If $\phi \equiv \psi_1 \vee \psi_2$, then by the induction hypothesis, $w, v \models \psi_1 \vee \psi_2$ iff $(w, v \models \psi_1)$ or $(w, v \models \psi_2)$ iff $(w, v' \models \Pi_h(\psi_1))$ or $(w, v' \models \Pi_h(\psi_2))$ iff $w, v' \models \Pi_h(\psi_1) \vee \Pi_h(\psi_2)$ iff $w, v' \models \Pi_h(\psi_1 \vee \psi_2)$.

Existential quantification over values. If $\phi \equiv \exists r \in R.\, \psi$, then $w, v \models \exists r \in R.\, \psi$ iff $\exists a \in R :\ w, v[r \leftarrow a] \models \psi$. For any $a \in R$, shifting t commutes with updating r (since $r \neq t$), i.e., $(v[r \leftarrow a])' = v'[r \leftarrow a]$. Applying the induction hypothesis to ψ under the valuation $v[r \leftarrow a]$ yields $w, v[r \leftarrow a] \models \psi$ iff $w, v'[r \leftarrow a] \models \Pi_h(\psi)$. Therefore, $w, v \models \exists r \in R.\, \psi$ iff $\exists a \in R :\ w, v'[r \leftarrow a] \models \Pi_h(\psi)$ iff $w, v' \models \exists r \in R.\, \Pi_h(\psi)$ iff $w, v' \models \Pi_h(\exists r \in R.\, \psi)$.

Existential quantification over time. If $\phi \equiv \exists s \in I.\, \psi$, the same argument applies: since s is bound, it is not the free variable t, and thus $(v[s \leftarrow a])' = v'[s \leftarrow a]$ for all $a \in I$. The induction hypothesis for ψ then yields the desired equivalence. □

A.3 Proof of Theorem 3 (Correctness of Monitor (Algorithm 1))

Statement. Let ϕ be a pSFO formula with a backward horizon of h, and let w be a piecewise-linear signal sampled with period Δ. For every $i \geq 1$ and every $t \in [(i-1)\Delta, i\Delta)$ such that $\mu(w, t, \phi)$ is defined, Monitor outputs v_i satisfying $v_i(t) = \mu(w, t, \phi)$. For every $t \in [(i-1)\Delta, i\Delta)$ such that $\mu(w, t, \phi)$ is undefined, $v_i(t)$ is undefined. Moreover, whenever $v_i(t)$ is defined, it depends only on the restriction of w to $[t - h, t]$.

Conventions. To avoid confusion between valuations and the robustness variables used in the algorithms, we write ν for valuations of SFO variables. For a temporal formula ϕ with unique free time variable t, we use the standard abbreviation $\mu(w, t_0, \phi) \coloneqq \mu\big(w, \nu[t \leftarrow t_0], \phi\big)$. We also identify a returned pair $(\mathcal{P}, v)$ (a finite union of polyhedra and a distinguished output variable) with the partial function it encodes on the current segment: for $t_0 \in \mathbb{R}$, define

$$(\mathcal{P}, v)(t_0) \text{ is defined} \iff \exists a\, \exists P \in \mathcal{P} :\ \nu[t \leftarrow t_0,\, v \leftarrow a] \models P,$$

$$(\mathcal{P}, v)(t_0) \coloneqq \sup\{a \mid \exists P \in \mathcal{P} :\ \nu[t \leftarrow t_0,\, v \leftarrow a] \models P\}.$$

By construction (in particular, the case-splits and the dominated-point removal in Algorithm 4), the set under the supremum is either empty or a singleton, so this agrees with the intended pointwise robustness value whenever defined.

We prove correctness of each subroutine used by MONITOR and then compose the results.

Lemma 1 (Boundary Supremum). *Let $P \subseteq \mathbb{R}^m \times \mathbb{R}$ be a convex polyhedron over parameters $Y \in \mathbb{R}^m$ and a scalar variable $x \in \mathbb{R}$. For $y \in \mathbb{R}^m$, write*

$$P_y \coloneqq \{x \mid (y, x) \in P\} \quad \textit{and} \quad P_Y \coloneqq \{y \mid P_y \neq \emptyset\}.$$

Let $o(y, x) = \alpha(y)\, x + \beta(y)$. For every $y \in P_Y$, the set P_y is a (possibly open, possibly unbounded) interval, and we define its endpoints by $L(y) \coloneqq \inf P_y \in \mathbb{R} \cup \{-\infty\}$ and $U(y) \coloneqq \sup P_y \in \mathbb{R} \cup \{+\infty\}$. Then:

$$\sup_{x \in P_y} o(y, x) = \begin{cases} \alpha(y)\, U(y) + \beta(y) & \text{if } \alpha(y) > 0, \\ \alpha(y)\, L(y) + \beta(y) & \text{if } \alpha(y) < 0, \\ \beta(y) & \text{if } \alpha(y) = 0, \end{cases}$$

with the conventions that $\alpha(y) \cdot (+\infty) = +\infty$ for $\alpha(y) > 0$ and $\alpha(y) \cdot (-\infty) = +\infty$ for $\alpha(y) < 0$.

Proof. Fix $y \in P_Y$. Since P is convex and defined by finitely many linear inequalities, P_y is the intersection of finitely many halfspaces in $\mathbb{R}$ and is therefore a (possibly open, possibly unbounded) interval. If $\alpha(y) = 0$, then $o(y, x) = \beta(y)$ is constant in x and the claim follows.

Assume $\alpha(y) > 0$. Then $o(y, \cdot)$ is strictly increasing. If $U(y) = +\infty$, then for every M there exists $x \in P_y$ with $x > M$, hence $o(y, x)$ is arbitrarily large and the supremum is $+\infty$. If $U(y) < +\infty$, then for all $x \in P_y$ we have $x \leq U(y)$, hence $o(y, x) \leq \alpha(y)U(y) + \beta(y)$. Conversely, by definition of $U(y) = \sup P_y$, for every $\varepsilon > 0$ there exists $x_\varepsilon \in P_y$ with $x_\varepsilon > U(y) - \varepsilon$, so $o(y, x_\varepsilon) > \alpha(y)(U(y) - \varepsilon) + \beta(y)$. Letting $\varepsilon \to 0$ yields $\sup_{x \in P_y} o(y, x) = \alpha(y)U(y) + \beta(y)$.

The case $\alpha(y) < 0$ is symmetric: $o(y, \cdot)$ is strictly decreasing, so the supremum is achieved (as a supremum, not necessarily attained) at the lower endpoint $L(y)$, and if $L(y) = -\infty$ the objective diverges to $+\infty$ as $x \to -\infty$. □

Lemma 2 (Correctness of PLPMAXIMIZE (Algorithm 5)). *Let P be a convex polyhedron over variables (Y, x) and let $o(Y, x)$ be an affine objective. Let $(\mathcal{Q}, v')$ be the output of PLPMAXIMIZE(P, o, x, v'). Let $\pi_Y(\cdot)$ denote projection onto Y-variables. We interpret any output piece containing the special constraint $\{v' = +\infty\}$ as assigning the value $+\infty$ (i.e., v' ranges over $\mathbb{R} \cup \{+\infty\}$ on that piece). Then, for every valuation ν of the parameter variables Y,*

$$\nu \in \pi_Y\Big(\bigcup \mathcal{Q}\Big) \quad \Longleftrightarrow \quad \exists x \in \mathbb{R} :\ (\nu, x) \in P,$$

and whenever $\nu \in \pi_Y(\bigcup \mathcal{Q})$, the value $v'(\nu)$ induced by $\mathcal{Q}$ satisfies

$$v'(\nu) = \sup\{\, o(\nu, x) \mid (\nu, x) \in P \,\}.$$

Equivalently, under this convention, $\bigcup \mathcal{Q}$ is the graph of the (extended-real-valued) function $\nu \mapsto \sup\{o(\nu, x) \mid (\nu, x) \in P\}$ on its feasibility domain.

Proof. Let $o = \alpha x + \beta$ be the decomposition produced by SplitCoeff, where α and β are affine in Y. The routine IsolateBounds rewrites the constraints of P into the form

$$P \equiv P_0(Y) \;\sqcap\; \Big(\prod_{\ell \in \mathcal{L}} \{x \;\bowtie_\ell\; \ell(Y)\}\Big) \;\sqcap\; \Big(\prod_{u \in \mathcal{U}} \{x \;\bowtie_u\; u(Y)\}\Big),$$

where P_0 contains exactly those constraints of P not involving x, and each $\ell \in \mathcal{L}$ and $u \in \mathcal{U}$ is affine in Y. Moreover, each $\bowtie_\ell$ is either $\geq$ or $>$ and each $\bowtie_u$ is either $\leq$ or $<$, depending on whether the corresponding original constraint was non-strict or strict. Let $P_Y = \textsc{Eliminate}(\{x\}, P)$ be the projection onto Y; then $P_Y = \{y \mid P_y \neq \emptyset\}$.

Fix $\nu \in P_Y$. Instantiating $Y = \nu$ turns the bounds above into inequalities over x only, hence P_ν is an interval with endpoints $L(\nu) = \sup\{\ell(\nu) \mid \ell \in \mathcal{L}\}$ (or $-\infty$ if $\mathcal{L} = \emptyset$) and $U(\nu) = \inf\{u(\nu) \mid u \in \mathcal{U}\}$ (or $+\infty$ if $\mathcal{U} = \emptyset$). The algorithm splits the parameter space by the sign of α into $G^+ = P_0 \sqcap \{\alpha > 0\}$, $G^- = P_0 \sqcap \{\alpha < 0\}$, and $G^0 = P_0 \sqcap \{\alpha = 0\}$, and always intersects the produced pieces with P_Y, so all output points lie in the feasibility domain.

Case $\alpha(\nu) > 0$. If $\mathcal{U} = \emptyset$ then $U(\nu) = +\infty$ and by Lemma 1 the supremum is $+\infty$; the algorithm returns $v' = +\infty$ on $G^+ \sqcap P_Y$, hence in particular at ν. If $\mathcal{U} \neq \emptyset$, let $u^\star \in \mathcal{U}$ be an upper bound attaining the minimum at ν, i.e. $u^\star(\nu) = U(\nu)$ (such $u^\star$ exists because $\mathcal{U}$ is finite). Then $\nu \models A_{u^\star}$, since $u^\star(\nu) \leq u'(\nu)$ for all $u' \in \mathcal{U}$, and $\nu \models F_{u^\star}$, since feasibility implies $\ell(\nu) \leq U(\nu) = u^\star(\nu)$ for all $\ell \in \mathcal{L}$. Thus ν lies in the region where the algorithm sets $v' = \alpha u^\star + \beta = \alpha U + \beta$, which equals $\sup_{x \in P_\nu} o(\nu, x)$ by Lemma 1. (Ties between several minimizing upper bounds only create overlapping output pieces with the same v' value, which does not affect correctness.)

Case $\alpha(\nu) < 0$. The argument is symmetric. If $\mathcal{L} = \emptyset$, then $L(\nu) = -\infty$ and Lemma 1 yields supremum $+\infty$; the algorithm outputs $v' = +\infty$ on $G^- \sqcap P_Y$. If $\mathcal{L} \neq \emptyset$, pick $\ell^\star \in \mathcal{L}$ attaining the maximum at ν, so $\ell^\star(\nu) = L(\nu)$; then $\nu \models A_{\ell^\star}$ and $\nu \models F_{\ell^\star}$, and the algorithm sets $v' = \alpha \ell^\star + \beta = \alpha L + \beta = \sup_{x \in P_\nu} o(\nu, x)$.

Case $\alpha(\nu) = 0$. Then $o(\nu, x) = \beta(\nu)$ is independent of x, so $\sup_{x \in P_\nu} o(\nu, x) = \beta(\nu)$ and the algorithm outputs $v' = \beta$ on $G^0 \sqcap P_Y$.

Finally, for $\nu \notin P_Y$ we have $P_\nu = \emptyset$, and every output region is intersected with P_Y, so $\nu \notin \pi_Y(\bigcup \mathcal{Q})$. This proves both the domain equivalence and the supremum identity. □

Lemma 3 (Correctness of EliminateBySup (Algorithm 4)). *Let $\mathcal{P}$ be a finite union of convex polyhedra over variables (Y, x, v), and assume that on each polyhedron $P \in \mathcal{P}$ the variable v is uniquely determined as an affine*

function of (Y, x)*. Let* $I \subseteq \mathbb{R}$ *be the quantification interval and* $(\mathcal{P}', v') = \textsc{EliminateBySup}(\mathcal{P}, v, x, I)$*. Let* $\pi_Y(\cdot)$ *denote projection onto* Y*-variables. Then for every valuation* ν *of* Y*,*

$$\nu \in \pi_Y\Big(\bigcup \mathcal{P}'\Big) \quad \Longleftrightarrow \quad \exists b \in I :\ \exists P \in \mathcal{P} :\ \exists a :\ \nu[x \leftarrow b,\, v \leftarrow a] \models P,$$

and whenever $\nu \in \pi_Y(\bigcup \mathcal{P}')$*, the induced output value satisfies*

$$v'(\nu) = \sup\big\{\, a \,\big|\, \exists b \in I,\ \exists P \in \mathcal{P} :\ \nu[x \leftarrow b,\, v \leftarrow a] \models P \,\big\}.$$

In other words, $\bigcup \mathcal{P}'$ *is the graph of the (partial) map* $\nu \mapsto \sup\{\, a \mid \exists b \in I,\ \exists P \in \mathcal{P} :\ \nu[x \leftarrow b,\, v \leftarrow a] \models P \,\}$*.*

Proof. For each input polyhedron $P \in \mathcal{P}$, EliminateBySup first forms $P_I \coloneqq P \sqcap \{x \in I\}$. Thus, for every valuation ν of Y, the set of feasible instantiations for (x, v) in P_I is exactly $\big\{(b, a) \;\big|\; b \in I \ \wedge\ \nu[x \leftarrow b,\, v \leftarrow a] \models P\big\}$, so both the feasibility domain and the intended supremum over $x \in I$ are preserved.

Next, NormalizeObj eliminates the old robustness variable v by extracting the affine objective $o_P(Y, x)$ such that $P_I \models (v = o_P(Y, x))$, producing an equivalent polyhedron P_I' over (Y, x) together with objective o_P. Therefore, for every ν, we have $\sup\big\{\, a \,\big|\, \exists b \in I :\ \nu[x \leftarrow b,\, v \leftarrow a] \models P \,\big\} = \sup\{\, o_P(\nu, x) \mid (\nu, x) \in P_I' \,\}$, with the left-hand set is empty iff the right-hand feasible set is empty.

Applying PlpMaximize to (P_I', o_P) yields a union $\mathcal{Q}_P$ over (Y, v') such that, by Lemma 2, for all ν, if $\nu \in \pi_Y\Big(\bigcup \mathcal{Q}_P\Big)$ then $v'(\nu) = \sup\{\, o_P(\nu, x) \mid (\nu, x) \in P_I' \,\}$. Moreover, PlpMaximize intersects every generated guard with $P_Y \coloneqq \textsc{Eliminate}(\{x\}, P_I')$, hence $\pi_Y(\bigcup \mathcal{Q}_P) \subseteq P_Y$. Conversely, if $\nu \in P_Y$ then there exists x with $(\nu, x) \in P_I'$. Writing $o_P = \alpha x + \beta$, the case split $\alpha > 0$, $\alpha < 0$, $\alpha = 0$ places ν in one of the corresponding regions G^+, G^-, G^0 of Algorithm 5. In each case, either the relevant bound set is empty (and the algorithm adds the whole region intersected with P_Y), or a tightest bound is attained (the bound sets are finite), so ν satisfies the corresponding guard $A_u \sqcap F_u$ (or $A_\ell \sqcap F_\ell$), and hence $\nu \in \pi_Y(\bigcup \mathcal{Q}_P)$. Therefore $\pi_Y(\bigcup \mathcal{Q}_P) = P_Y$, i.e., $\nu \in \pi_Y\Big(\bigcup \mathcal{Q}_P\Big)$ iff $\exists x :\ (\nu, x) \in P_I'$ iff $\exists b \in I :\ \exists a :\ \nu[x \leftarrow b,\, v \leftarrow a] \models P$. The algorithm then takes the union $\bigcup_{P \in \mathcal{P}} \mathcal{Q}_P$ over all pieces. Fix ν such that $\exists b \in I,\ \exists P \in \mathcal{P},\ \exists a :\ \nu[x \leftarrow b,\, v \leftarrow a] \models P$. Since a supremum over a finite union of feasible sets equals the maximum of the per-piece suprema, we have

$$\begin{aligned} &\sup\big\{\, a \,\big|\, \exists b \in I,\ \exists P \in \mathcal{P} :\ \nu[x \leftarrow b,\, v \leftarrow a] \models P \,\big\} \\ &= \max_{P \in \mathcal{P}:\ \exists b \in I\ \exists a:\ \nu[x \leftarrow b,\, v \leftarrow a] \models P} \sup\big\{\, a \,\big|\, \exists b \in I :\ \nu[x \leftarrow b,\, v \leftarrow a] \models P \,\big\}. \end{aligned}$$

The final dominated-point removal step in Algorithm 4 keeps, for each fixed ν, exactly the points with maximal v' among all pieces, thereby implementing this pointwise maximum. In particular, it preserves the domain $\pi_Y(\bigcup \mathcal{P}')$ (it only removes non-maximal values at already-present ν). Thus $\bigcup \mathcal{P}'$ is precisely the graph of the supremum function on its feasibility domain, and it is empty over those ν for which no feasible instantiation $b \in I$ exists. □

Lemma 4 (Correctness of Term (Algorithm 3)). *Fix a trace w. Let θ be a term and let P_d be the current conjunction of domain constraints passed to* Term. *Let $(\mathcal{P}, v) =$ Term(θ, P_d). Then for every valuation ν of the variables occurring in θ or in P_d (excluding the fresh output variable v),*

$$\Big(\nu \models P_d \ \wedge\ [\![\theta]\!]_{w,\nu} \text{ is defined}\Big) \quad\Longleftrightarrow\quad \exists a:\ \nu[v \leftarrow a] \models \bigcup \mathcal{P},$$

and whenever this holds, the (necessarily unique) witnessing a equals $[\![\theta]\!]_{w,\nu}$.

Proof. Throughout the proof, we use that for each function symbol f the current list $\mathcal{P}_f$ encodes the graph of $[\![f]\!]_w$ on the time domain currently stored by the monitor (with piecewise-linear interpolation). We proceed by structural induction on θ.

If θ is a constant n or a variable (r or s), Term returns a single polyhedron enforcing $v = n$ (resp. $v = r$, $v = s$) conjoined with P_d, so the claim is immediate.

If $\theta \equiv \theta_1 \pm \theta_2$, let $(\mathcal{P}_1, v_1)$ and $(\mathcal{P}_2, v_2)$ be the recursive results. By the induction hypothesis, for any $\nu \models P_d$, satisfying $\bigcup \mathcal{P}_1$ (resp. $\bigcup \mathcal{P}_2$) is equivalent to $[\![\theta_1]\!]_{w,\nu}$ (resp. $[\![\theta_2]\!]_{w,\nu}$) being defined, and in that case the unique values of v_1 and v_2 equal these semantics. The algorithm constructs $\mathcal{P}$ by intersecting $\mathcal{P}_1$ and $\mathcal{P}_2$, adding the constraint $v = v_1 \pm v_2$, and projecting away v_1 and v_2. Therefore $\nu[v \leftarrow a] \models \bigcup \mathcal{P}$ holds exactly when both subterms are defined and $a = [\![\theta_1]\!]_{w,\nu} \pm [\![\theta_2]\!]_{w,\nu} = [\![\theta]\!]_{w,\nu}$.

If $\theta \equiv f(\tau)$, then by assumption $\mathcal{P}_f$ is a polyhedral graph representation of f: for any time point s in the currently stored signal domain there is a polyhedron in $\mathcal{P}_f$ enforcing $v_f = [\![f]\!]_w(s)$ when $t_f = s$. Term substitutes $t_f \mapsto \tau$ and $v_f \mapsto v$ and conjoins P_d. Hence, for any valuation $\nu \models P_d$, the constraints are satisfiable exactly when the access time $[\![\tau]\!]_{w,\nu}$ lies in the stored domain of f (i.e., the access is well-defined with respect to the available trace), and then they enforce $v = [\![f]\!]_w([\![\tau]\!]_{w,\nu}) = [\![f(\tau)]\!]_{w,\nu}$. □

Lemma 5 (Correctness of FormulaRobust (Algorithm 2)). *Let ϕ be a pSFO formula and let P_d be the current conjunction of domain constraints for the free variables of ϕ on the current segment. Let $(\mathcal{P}, v) =$ FormulaRobust(ϕ, P_d). Then for every valuation ν of the free variables of ϕ (excluding the fresh output variable v),*

$$\Big(\nu \models P_d \wedge \mu(w, \nu, \phi) \text{ is defined}\Big) \quad\Longleftrightarrow\quad \exists a \in \mathbb{R} \cup \{-\infty, +\infty\}:\ \nu[v \leftarrow a] \models \bigcup \mathcal{P},$$

and whenever this holds, the witnessing a is unique and equals $\mu(w, \nu, \phi)$.

Proof. We prove the claim by structural induction on ϕ.

Let $\phi \equiv (\theta_1 < \theta_2)$. Then FormulaRobust calls Term on the term $\theta_2 - \theta_1$ and returns the resulting pair $(\mathcal{P}, v)$. By Lemma 4, for every valuation ν we have that $\nu \models P_d$ and $[\![\theta_2 - \theta_1]\!]_{w,\nu}$ is defined iff there exists a (necessarily unique) $a \in \mathbb{R}$ with $\nu[v \leftarrow a] \models \bigcup \mathcal{P}$, and then $a = [\![\theta_2 - \theta_1]\!]_{w,\nu}$. Since

$$\mu(w, \nu, \theta_1 < \theta_2) = [\![\theta_2]\!]_{w,\nu} - [\![\theta_1]\!]_{w,\nu} = [\![\theta_2 - \theta_1]\!]_{w,\nu},$$

the claim follows.

Let $\phi \equiv \neg\phi'$ and let $(\mathcal{P}', v') = \textsc{FormulaRobust}(\phi', P_d)$. By the induction hypothesis, for every $\nu \models P_d$, $\mu(w, \nu, \phi')$ is defined iff there exists a unique $b \in \mathbb{R} \cup \{-\infty, +\infty\}$ such that $\nu[v' \leftarrow b] \models \bigcup \mathcal{P}'$, and then $b = \mu(w, \nu, \phi')$. The algorithm introduces a fresh output variable v, conjoins the constraint $v = -v'$, and projects out v'. Hence, for every $\nu \models P_d$, there exists a with $\nu[v \leftarrow a] \models \bigcup \mathcal{P}$ iff there exists b with $\nu[v' \leftarrow b] \models \bigcup \mathcal{P}'$ and $a = -b$. Therefore the output is defined exactly when $\mu(w, \nu, \phi')$ is defined, and whenever defined it satisfies

$$a = -\mu(w, \nu, \phi') = \mu(w, \nu, \neg\phi').$$

Let $\phi \equiv \phi_1 \vee \phi_2$ and let $(\mathcal{P}_i, v_i) = \textsc{FormulaRobust}(\phi_i, P_d)$ for $i \in \{1, 2\}$. By the induction hypothesis, for every $\nu \models P_d$ and each $i \in \{1, 2\}$, $\mu(w, \nu, \phi_i)$ is defined iff there exists a unique $a_i \in \mathbb{R} \cup \{-\infty, +\infty\}$ such that $\nu[v_i \leftarrow a_i] \models \bigcup \mathcal{P}_i$, and then $a_i = \mu(w, \nu, \phi_i)$. Under our well-definedness convention (Remark 1), $\mu(w, \nu, \phi_1 \vee \phi_2)$ is defined iff both $\mu(w, \nu, \phi_1)$ and $\mu(w, \nu, \phi_2)$ are defined. The algorithm enforces this by intersecting $\mathcal{P}_1$ and $\mathcal{P}_2$, splitting into the cases $v_1 \geq v_2$ and $v_1 < v_2$, setting $v = v_1$ and $v = v_2$ respectively, and projecting out v_1, v_2. Therefore, for every $\nu \models P_d$, the output is defined iff both a_1 and a_2 exist, and whenever defined the unique output value satisfies

$$a = \max\{a_1, a_2\} = \max\{\mu(w, \nu, \phi_1), \mu(w, \nu, \phi_2)\} = \mu(w, \nu, \phi_1 \vee \phi_2).$$

Let $\phi \equiv \exists x \in J.\, \phi'$, where x is either a value variable (so $J = R$) or a time variable (so $J = I$). The algorithm first computes $(\mathcal{P}', v') = \textsc{FormulaRobust}\big(\phi',\ P_d \sqcap \{x \in J\}\big)$, and then returns $(\mathcal{P}, v) = \textsc{EliminateBySup}(\mathcal{P}', v', x, J)$. Fix a valuation ν of the free variables of ϕ with $\nu \models P_d$, and define the set of admissible instantiations $D_\nu := \{\, b \in J \mid \mu(w, \nu[x \leftarrow b], \phi') \text{ is defined} \,\}$. By the quantitative semantics (together with the well-definedness convention for quantifiers), $\mu(w, \nu, \exists x \in J.\phi')$ is defined iff $D_\nu \neq \emptyset$, and then $\mu(w, \nu, \exists x \in J.\phi') = \sup_{b \in D_\nu} \mu(w, \nu[x \leftarrow b], \phi')$. By the induction hypothesis applied to the recursive call on ϕ', for every $b \in J$ the following holds: if $b \in D_\nu$, then there exists a unique value $a_b = \mu(w, \nu[x \leftarrow b], \phi') \in \mathbb{R} \cup \{-\infty, +\infty\}$ such that $\nu[x \leftarrow b,\, v' \leftarrow a_b] \models \bigcup \mathcal{P}'$. Equivalently, $\bigcup \mathcal{P}'$ is the graph of the partial map $(\nu, b) \mapsto \mu(w, \nu[x \leftarrow b], \phi')$ over those pairs with $b \in D_\nu$. Applying Lemma 3 to $\textsc{EliminateBySup}(\mathcal{P}', v', x, J)$ therefore yields that the resulting output is defined at ν iff $D_\nu \neq \emptyset$, and whenever defined its unique output value satisfies

$$a = \sup_{b \in D_\nu} a_b = \sup_{b \in D_\nu} \mu(w, \nu[x \leftarrow b], \phi') = \mu(w, \nu, \exists x \in J.\phi').$$

□

Proof (of Theorem 3). Fix $i \geq 1$ and $t_0 \in [(i-1)\Delta, i\Delta)$. Let $P_d = \{t \in [(i-1)\Delta, i\Delta)\}$, and let $v_i = (\mathcal{P}_i, v)$ be the pair output by Monitor at iteration i, i.e. $(\mathcal{P}_i, v) = \textsc{FormulaRobust}(\phi, P_d)$.

Signal availability on the required window. By construction of Monitor, after the garbage-collection step at the end of iteration $i-1$ (vacuously for

$i = 1$), each list $\mathcal{P}_f$ contains all signal pieces whose right endpoint is at least $(i-1)\Delta - h$. After receiving w_i, MONITOR appends the fresh piecewise-linear constraints for w_i. Consequently, immediately before calling FORMULAROBUST at iteration i, each $\mathcal{P}_f$ encodes the graph of $[\![f]\!]_w$ on (at least) the time interval $[(i-1)\Delta - h,\ i\Delta)$.

Correctness on the current segment. Because ϕ is pSFO and h is a backward horizon for ϕ, every signal access time term τ occurring in ϕ satisfies $t-h \leq \tau \leq t$ for every valuation of bound variables consistent with their quantifier bounds. Instantiating $t = t_0 \in [(i-1)\Delta, i\Delta)$ yields $\tau \in [t_0 - h, t_0] \subseteq [(i-1)\Delta - h, i\Delta)$. Thus, whenever $\mu(w, t_0, \phi)$ is defined, all signal values needed to evaluate ϕ at t_0 lie within the stored signal window represented by the current $\mathcal{P}_f$ lists. Since also $t_0 \models P_d$, we can apply Lemma 5 to the call FORMULAROBUST(ϕ, P_d) at time t_0. Therefore, $\mu(w, t_0, \phi)$ is defined iff $v_i(t_0)$ is defined, and in the defined case we have $v_i(t_0) = \mu(w, t_0, \phi)$. This proves the first two claims of the theorem.

Dependence only on $[t_0 - h, t_0]$. By the same horizon argument, every signal read performed when evaluating ϕ at time t_0 occurs at some time in $[t_0 - h, t_0]$. Hence, whenever $v_i(t_0)$ is defined, modifying w outside $[t_0 - h, t_0]$ does not change any accessed signal value and therefore cannot change $\mu(w, t_0, \phi) = v_i(t_0)$.

Soundness of the garbage-collection step. After producing v_i, MONITOR drops from each $\mathcal{P}_f$ all pieces ending strictly before $i\Delta - h$. For any later evaluation time $t \geq i\Delta$, the required history window satisfies $[t-h, t] \subseteq [i\Delta - h, t]$, so no dropped piece can be accessed in any subsequent iteration. Thus garbage collection preserves correctness of all future outputs while ensuring bounded memory. □