Online Monitoring of Spatio-Temporal Properties for Imprecise Signals

Ennio Visconti  
TU Wien  
Vienna, Austria

Michele Loreti  
Università di Camerino  
Camerino, Italy

Ezio Bartocci  
TU Wien  
Vienna, Austria

Laura Nenzi  
TU Wien  
Vienna, Austria

ABSTRACT

From biological systems to cyber-physical systems, monitoring the behavior of such dynamical systems often requires to reason about complex spatio-temporal properties of physical and/or computational entities that are dynamically interconnected and arranged in a particular spatial configuration. Spatio-Temporal Reach and Escape Logic (STREL) is a recent logic-based formal language designed to specify and to reason about spatio-temporal properties. STREL considers each system’s entity as a node of a dynamic weighted graph representing their spatial arrangement. Each node generates a set of mixed-analog signals describing the evolution over time of computational and physical quantities characterising the node’s behavior. While there are offline algorithms available for monitoring STREL specifications over logged simulation traces, here we investigate for the first time an online algorithm enabling the runtime verification during the system’s execution or simulation. Our approach extends the original framework by considering imprecise signals and by enhancing the logics’ semantics with the possibility to express partial guarantees about the conformance of the system’s behavior with its specification. Finally, we demonstrate our approach in a real-world environmental monitoring case study.

CCS CONCEPTS

• Theory of computation → Logic and verification; Modal and temporal logics; • Software and its engineering → Abstraction, modeling and modularity.

KEYWORDS

Runtime verification, online monitoring, spatio-temporal logic, imprecise signal, signal temporal logic

ACM Reference Format:
Ennio Visconti, Ezio Bartocci, Michele Loreti, and Laura Nenzi. 2021. Online Monitoring of Spatio-Temporal Properties for Imprecise Signals. In MEMOCODE ’21: ACM-IEEE International Conference on Formal Methods and Models for System Design, November 20–22, 2021, Beijing, China. ACM, New York, NY, USA, 14 pages. https://doi.org/---------------

1 INTRODUCTION

Complex emergent spatio-temporal patterns such as traffic congestion or travelling waves are central in the understanding of networked dynamical systems where locally interacting entities are operating at different time and spatial scales. We can observe these patterns both in biological systems [3, 5, 11] as well as human engineered artefacts such as Collective Adaptive Systems [22] (CAS) and Cyber-Physical Systems [31] (CPS). CAS and CPS consist of a large number of heterogeneous (physical and computational in CPS) and spatially distributed entities featuring complex interactions among themselves, with humans and other systems. Example include biking sharing systems, the internet of things, contact tracing devices preventing the epidemic spread, vehicular networks and smart cities. Many of these systems are also safety-critical [31], meaning that a failure could result in loss of life or in catastrophic consequences for the environment.

The complex interaction with the physical environment in which these systems are embedded prevents them from being exhaustively verified at design time. A common alternative is testing [4]: traces generated during their execution/simulation are stored and monitored offline with respect to a formal specification used as an oracle. However, testing may provide a limited coverage and does not take into account physical failures that may happen during the execution. Online monitoring is instead a preferable solution when the monitoring verdict requires the immediate action of a policy maker during the system’s execution or when it is very computationally expensive [32] generating and storing the system’s execution traces to be monitored offline.

Motivating example. As a case study we consider a sensor network for environmental monitoring. Air pollution is the primary cause of the loss of biodiversity, of the reduction of agricultural productivity and of many diseases for humans’ lungs and cardiovascular system. Policy makers are constantly monitoring the amount of $NO_2$, an air pollutant that forms from the combustion of fossil fuels, to activate special policies that mitigate its release when levels grow too significantly for the public concern. For example, in the Italian region of Lombardy, there are over 80 stations distributed throughout the region monitoring the level of $NO_2$ in the air. Figure 1 shows
Figure 1: On the left the NO2 measuring stations of Lombardy; the lighter-colored dot represents the station 6859 - Rezzato. On the right the detailed view of the NO2 measured by the Rezzato station [20] in terms of $\mu g/m^3$. Red crosses and dotted line represent missing values. Data from the open data initiative of ARPA Lombardia [21].

In this paper we address the problem of online monitoring of spatio-temporal properties over systems from which we can observe noisy signals with possible missing data or out-of-order samples. Spatio-temporal monitoring. In the last decade there has been a great effort to develop logic-based specification languages and monitoring frameworks for spatio-temporal properties. Examples include SpaTel [12], SSTL [29], (SaSTL) [23, 24] and STREL [1]. For more details on the underlying spatial models and the language expressiveness we refer the reader to [28]. In this paper, we consider STREL [1] a spatio-temporal logic operating over a dynamic weighted graph representing the spatial arrangement of spatially distributed entities. Each node generates a set of mixed-analog signals describing the evolution over time of computational and physical quantities characterising the node’s behavior. STREL extends the Signal Temporal Logic (STL) [25] with the reach and escape operators that generalizes the somewhere, everywhere and surrounded spatial modalities, simplifying the monitoring that can be computed locally with respect to each node. However, the original work on STREL [1, 2] provides only an offline monitoring algorithm. In contrast we present here the first online monitoring algorithm for STREL and in general for spatio-temporal monitoring.

Online Monitoring. To the best of our knowledge, the only online monitoring techniques [7, 8, 14, 15, 26, 27, 30] that are available in the literature can handle temporal specification languages such as STL [25] and Metric Temporal Logic [17] (MTL). One of the main challenges for online monitoring is how and when to decide the satisfaction/violation of a formula with temporal operators reasoning about future and not yet observed events. In [8] the authors provides, for the first time, a dynamic programming algorithm for the online monitoring of the robustness metric of MTL formulas with bounded future and unbounded past. The past formula is used to reason about the robustness of the actual system observations, while for evaluating the future formula they use a predictor to estimate the likely robustness. However, the value forecasted by the predictor needs to be trusted because it is not the real value that the system will provide. Other approaches [14, 15, 30] address the problem of deciding about the future using a technique called pastification that rewrites the future operators as past ones and delays the verdict. Similarly the works of Mamouras et al [26, 27] delay the output verdict until some part of the future input is seen. In [7] the authors present an efficient online algorithm to compute the robust interval semantics for bounded horizon formulas. All these approaches assume that the data and the events to be observed come synchronously and in-order. Our contribution In contrast to these works, we present a novel approach to monitor online imprecise spatio-temporal signals (signals are defined on intervals, not just the robustness) where the samples can be processed also out-of-order. The notion of an interval is instrumental when representing partial knowledge about a value that is at least known to be within some boundaries. This might be because of errors in the measurement, or maybe because of some other sources of uncertainty throughout the process of acquiring and processing them. We define both a Boolean and a quantitative interval semantics for STREL and we prove the soundness and the correctness of the
robust interval semantics. We design and implement, as extensions to the Moonlight tool\(^1\), the first online monitoring algorithm for not-in-order sampled signals, and the first online spatio-temporal monitoring tool. Our experiments demonstrate also convincing performances comparing with the state-of-the-art tool BREACH \(^9\) for the online monitoring of temporal properties over in-order sampled signals.

**Paper organization** The rest of this paper is structured as follows. We provide the important aspects of interval algebra and our notion of imprecise signals in Section 2. In Section 3 we introduce the interval extension of the STREL logic, and its primary results, while in Section 4 we present our approach for the online monitoring of imprecise signals. Lastly, we present a realistic use case in Section 5 and we share our concluding remarks in Section 6.

## 2 INTERVAL ALGEBRA, SIGNALS AND SPATIAL MODEL

In this section, we define the key elements of interval algebra, signal and spatial model, which will be useful to characterize samples of the kind depicted in Figure 1.

**Definition 2.1** (Intervals). Let \(I(\mathbb{R}^m)\) be the set of intervals defined over the set \(\mathbb{R}^m = \mathbb{R} \cup \{+\infty, -\infty\}\). We call **closed interval** (or simply interval) any set \(I \subseteq \mathbb{R}^m\) such that \(I \equiv [a, b] := \{x \in \mathbb{R}^m : a < x < b, a, b \in \mathbb{R}^m\}\). For any \(I \equiv [a, b] \in I(\mathbb{R}^m)\), we will indicate as \(I, I^{-}\) the extremes of the interval.

In addition to the classical notion of interval, it is useful to recall some basic operations that can be performed.

**Definition 2.2** (Interval Basic Operations). Consider \(I, I_1, I_2 \in I(\mathbb{R}^m), c \in \mathbb{R}^m\), we define the following interval operators:

\[
\begin{align*}
&c + I := [I + c, I + c] &-I := [-I, -I] \\
&I_1 + I_2 := [I_1 + I_2, I_1 + I_2] &I_1 - I_2 := I_1 + (-I_2) \\
&[\max](I_1, I_2) := [\max(I_1), \max(I_2)] \quad [\max](I_1, I_2) := [\max(I_1), \min(I_2)] \\
&[\min](I_1, I_2) := [\min(I_1), \min(I_2)] \\
\end{align*}
\]

We also consider the extensions of \([\min]\) and \([\max]\) operators defined over an arbitrary subset \(A \subseteq I(\mathbb{R}^m)\), denoted by \((\cdot)\) instead of \((\cdot)\) for function arguments.

We call **interval radius** of \(I\), the operator \([I] := [\max([I], [I])], [\min([I], [I])])\)

Interval relations are described in this way:

**Definition 2.3** (Interval Inequalities). Let \(I_1, I_2 \in I(\mathbb{R}^m)\), we say that \(I_1 < I_2\) when \(I_1 \subseteq I_2\). Symmetrically, we say that \(I_1 > I_2\) when \(I_1 \supseteq I_2\), for \(c \in \mathbb{R}^m\).

To measure distances between intervals we consider the **Hausdorff Distance**.

**Definition 2.4** (Hausdorff Distance). Let \(X, Y\) be two non-empty subsets of a metric space \((M, d)\), we will call (Hausdorff) distance the function \(d_H : \mathcal{P}(X) \times \mathcal{P}(Y) \to \mathbb{R}_{\geq 0}\) defined as

\[
d_H := \max \left\{ \sup_{x \in X} \inf_{y \in Y} d(x, y), \sup_{y \in Y} \inf_{x \in X} d(x, y) \right\}
\]

**Source code available at:** github.com/MoonLightSuite/MoonLight

\(^1\)We will write \(I < c\) (respectively \(I > c\)) in place of \(I < [c, c]\) (resp. \(I > [c, c]\))

In practice, in our context, we can just consider the metric space defined by the euclidean distance over the real numbers, and thus \(d_H\) reduces to computing \(\max(\|I_1 - I_2\|, \|I_1 - I_2\|)\) for any two \(I_1, I_2 \in I(\mathbb{R}^m)\), although for doing that we say, by definition, that if both \(I_1, I_2\) (or both \(I_1, I_2\)) are infinite, then their Hausdorff distance is 0.

Now we have all the tools to introduce the concept of imprecise signals.

**Definition 2.5** (Imprecise Temporal Signal). Let \(\mathbb{T} \equiv [0, \infty)\) be a set representing the time domain, and let \(\mathcal{F}(\mathbb{T}, D^n)\), with \(D \subseteq \mathbb{I}(\mathbb{R})\) for a fixed \(n \in \mathbb{N}\), be the family of functions over Cartesian products of real intervals; we call **imprecise time signal**, any \(\sigma \in \mathcal{F}(\mathbb{T}, D^n)\), i.e. any function \(\sigma : \mathbb{T} \to D^n\).

It is convenient, in some cases, to slice the signals based on the domain \(D\) of interest, for that reason, we recall the concept of (signal) projection.

**Definition 2.6** (Signal Projection). Let \(\pi_i : D_1 \times \cdots \times D_i \times \cdots D_n \to D_i\) be the function that takes the \(i\)-th projection of the set-theoretic Cartesian product \(D^n\), we will indicate as \(\pi_i(\sigma)\) the projection of \(\sigma(t)\) to the \(i\)-th 1-dimensional signal \(s_i : \mathbb{T} \to D_i\).

To represent a set of signals distributed in the space, we introduce the following definition.

**Definition 2.7** (Imprecise Spatio-Temporal Signal). Let \(\mathcal{F}(\mathbb{L}, \mathbb{T}, D^n)\) be the family of functions of space and time over real intervals, with \(\mathbb{L}\) a set of locations, we call **imprecise spatio-temporal signal** or **just signal** when there is no risk of ambiguity – any \(s \in \mathcal{F}(\mathbb{L}, \mathbb{T}, D^n)\), i.e. any function: \(s : \mathbb{L} \times \mathbb{T} \to D^n\).

Considering the pollution example, where \(\mathbb{L}\) is the set of stations, \(\mathbb{T} = [0, 10]\) is the time domain corresponding to an interval of 10 days, and \(D\) is the possible range for nitrogen-dioxide values (NO\(_2\)) in the air; then the spatio-temporal signal \(s : \mathbb{L} \times \mathbb{T} \to D\) returns at each time, in each location the value of NO\(_2\), \(s(t, l)\).

We can naturally describe the distance between spatio-temporal signals by considering the Hausdorff distance from Definition 2.4 over all possible locations of the space, and time instants.

**Definition 2.8** (Spatio-Temporal Signal Distance). Let \(s_1, s_2 \in \mathcal{F}(\mathbb{L}, \mathbb{T}, D^n)\), we will call **signal distance** the largest Hausdorff distance over space and time, defined as:

\[
||s_1 - s_2||_{\infty} := \max_{i \leq n, t \in \mathbb{T}} \max_{l \in \mathbb{L}} |d_H(\pi_i(s_1(t, l), \pi_i(s_2(t, l)))|
\]

To describe the interplay of signals in different locations, we need to encompass the information related to the spatial distribution of the locations.

**Definition 2.9** (Spatial model). We call **spatial model** the tuple \(\mathcal{S} = (\mathbb{L}, W)^3\), where \(\mathbb{L}\) is a set of locations and \(W \subseteq \mathbb{L} \times \mathbb{R}_{\geq 0} \times \mathbb{L}\) is a **proximity function** associating at most one label \(w \in \mathbb{R}_{\geq 0}\) to each distinct pair \(l_1, l_2 \in \mathbb{L}\).

An obvious spatial model for the region of Figure 1 is a graph where every location is connected to all the others, and the proximity function is defined by labels corresponding to the minimal aerial

\(^3\)We focus on real-valued positive labels, to convey the intuitive meaning of distance between two locations. For alternative definitions of \(W\), the interested reader might refer to [1].
distance between each pair of locations. Finally to consider distance on paths of locations we introduce the notion of routes over the spatial model.

**Definition 2.10 (Routes).** A route \( r \) on \( S \) is a (potentially infinite) sequence \( \ell_0, \ell_1, \ldots, \ell_k, \ldots \) such that for any \( \ell_1, \ell_{i+1} \in r \), there is a label \((\ell_i, w, \ell_{i+1}) \in W \). We indicate by \( \Lambda(I) \) the set of routes on \( S \) starting at \( \ell \). Moreover, we will use \( r[i] \) to denote the \( i \)-th node of the route, \( \tau[i] \ldots \) to denote the subroute starting at the \( i \)-th node, and \( \tau(\ell_i) \) to denote the first occurrence of \( \ell_i \) in \( \tau \). Lastly, we will indicate by \( \ell_1 < \tau(\ell_2) \) the fact that \( \ell_1 \) precedes \( \ell_2 \) in the route \( \tau \).

Routes have the same intuitive meaning as they have in the physical world, and, similarly to the real world, we can define the concept of route (or travel) distance, as the aggregated sum of all the labels traversed by the route.

**Definition 2.11 (Route Distance).** For a given \( r \) on \( S \), the distance \( d_r[I] \) is:

\[
d_r[I] = \begin{cases}
0, & \text{if } I = 0 \\
W + d_r[I-1] & \text{if } I > 0 \text{ and } (r[0], w, r[1]) \in W
\end{cases}
\]

Lastly, routes allow us to conveniently define the distance between any two locations \( \ell_1, \ell_2 \in \mathcal{L} \), whichever the spatial model being considered. In fact, from the location \( \ell_1 \) to \( \ell_2 \), one can consider the minimal distance among all the routes starting at \( \ell_1 \) and ending in \( \ell_2 \):

\[
d_{\min}(\ell_1, \ell_2) = \min_{r \in \Lambda(I)} d_r[I]
\]

### 3 STREL WITH INTERVAL SEMANTICS

We present in this section an interval semantics that allows for specifying properties related to the spatial configuration, and in this context, the exact meaning is that at least a location in a range of less than 10 km must have a level of nitrogen dioxide lower than 400 \( \mu g/m^3 \). We will see other examples of the logic language in the next sections. For a more detailed description of the logic, we refer the reader to [1]. We present now the Boolean and quantitative interval semantics for STREL.

**Definition 3.2 (STREL Boolean Semantics).** Let \( \chi : F(\mathbb{L}, \mathbb{T}, D^p) \times \mathbb{L} \times \mathbb{T} \times \mathcal{L} \rightarrow \{-1, 0, 1\} \) be a function defined as follows:

\[
\chi(s, t, \top) = 1
\]

\[
\chi(s, t, \bot) = -1
\]

\[
\chi(s, t, p \circ c) = \begin{cases}
1, & \text{if } p_0(s(t, t)) \circ c \\
-1, & \text{if } p_0(s(t, t)) \circ -c \\
0, & \text{otherwise}^4
\end{cases}
\]

\[
\chi(s, t, \neg \varphi) = -\chi(s, t, \varphi)
\]

\[
\chi(s, t, \varphi_1 \lor \varphi_2) = \max(\chi(s, t, \varphi_1), \chi(s, t, \varphi_2))
\]

\[
\chi(s, t, [\varphi_1 ∧ \varphi_2]) = \min(\chi(s, t, \varphi_1'), \chi(s, t, \varphi_2'))
\]

\[
\chi(s, t, [\varphi_1 \circ \varphi_2]) = \max_{e \in \pi} \min_{\tau \in \tau[\ell]} \chi(s, t, \varphi_1')
\]

\[
\chi(s, t, [\varphi_1' \circ \varphi_2]) = \max_{e \in \pi} \min_{\tau \in \tau[\ell]} \chi(s, t, \varphi_1')
\]

This is a three-valued semantics, which is equal to 1 if the interval signal \( \chi(s, t, \varphi) \) satisfies \( \varphi \), -1 if the formula is not satisfied, and 0 if we cannot answer. The semantics is directly derived from the standard Boolean semantics and the interval algebra described in the previous section. For atomic proposition \( \chi(s, t, p \circ c) \equiv 1 \) iff the inequality \( p_0(s(t, t)) \circ c \) is true. This means, e.g., \( \chi(s, t, NO_2 > 400) \equiv 1 \) iff \( NO_2 \) the left extreme of the projected signal \( NO_2 \), is greater than 400. \( \chi(s, t, NO_2 < 400) \equiv 1 \) if the right extreme is less than 400, and \( \chi(s, t, NO_2 > 400) \equiv 0 \) otherwise, so if \( NO_2 \) is 400 \( NO_2 \) interval value. Similar calculation can be done for the other combination of \( o \) and \( c \).

The three-valued Boolean semantics can be sufficient in applications where the interest is only whether or not the specification is satisfied. However, in many complex cases, one might be interested in getting some insights about the degree by which a property is satisfied or violated. In the following, we introduce an extension of the quantitative semantics that provides numerical bounds to the robustness degree of a specification.

**Definition 3.3 (STREL Robust Interval Semantics).** Let \( \rho : F(\mathbb{L}, \mathbb{T}, D^p) \times \mathbb{L} \times \mathbb{T} \times \mathcal{L} \rightarrow \{\text{real intervals}\} \) be the function mapping signals, locations, time instants, and formulae defined as follows:

\[
\phi_{<10\text{km}} NO_2 < 400
\]

Spatial operators like \( \phi \) instead specify properties related to the spatial configuration, and in this context, the exact meaning is that at least a location in a range of less than 10 km must have a level of nitrogen dioxide lower than 400 \( \mu g/m^3 \). \( NO_2 \) will see other examples of the logic language in the next sections. For a more detailed description of the logic, we refer the reader to [1]. We present now the Boolean and quantitative interval semantics for STREL.
\[ \rho(s, t, \tau) = [s' \in [s, s'+1) \cap [t, t+1], c - p(s, t, \tau) \leq c' - p(s, t, \tau) \text{ if } c' > c \]
\[ \rho(s, t, \tau) = [s' \in [s, s'+1) \cap [t, t+1], c - p(s, t, \tau) \geq c' - p(s, t, \tau) \text{ if } c' < c \]
\[ \rho(s, t, p \circ c) = \{ p(s, t, \tau) - c, \text{ if } o \text{ is } ' > ' \} \]
\[ \rho(s, t, \neg \varphi) = -\rho(s, t, \varphi) \]
\[ \rho(s, t, \varphi_1 \lor \varphi_2) = \max \{ \rho(s, t, \varphi_1), \rho(s, t, \varphi_2) \} \]
\[ \rho(s, t, \varphi_1 \land \varphi_2) = \left\{ \begin{array}{ll}
\min \left( \rho(s, t', \varphi_2), \min_{t' \in [s, s')} \left( \rho(s, t', \varphi_1) \right) \right) & \text{if } \forall t' \leq t \\
\max \left( \rho(s, t', \varphi_2), \min_{t' \in [s, s')} \left( \rho(s, t', \varphi_1) \right) \right) & \text{if } \exists t' > t
\end{array} \right. \]

We will indicate with \( \rho^\varphi : \mathbb{L} \times \mathbb{T} \rightarrow I(\mathbb{R}^\infty) \) the robustness signal, i.e., the signal generated by the partial application of the \( \rho \) function to a given formula \( \varphi \) and a given signal \( s \), so that \( \rho^\varphi(s, t, \varphi) \).

Note that without the interval semantics we have defined, missing values should be substituted by some values that approximate the actual value (e.g., by linear interpolation), and therefore only approximate the actual value of satisfaction or robustness of a given property at that specific time point. Conversely, by exploiting the interval semantics, one could actually get upper/lower bounds at those points, which can actually be sufficient in real-world applications.

**Theorem 3.1 (Soundness of Robust Interval Semantics).** The Robust Interval Semantics of Definition 3.3 is sound w.r.t the Boolean Semantics of Definition 3.2, i.e., for any \( s \in \mathbb{F}(\mathbb{L}, \mathbb{T}, \mathbb{D}^\varphi) \), \( t \in \mathbb{L} \), \( \varphi \in \mathbb{L} \):

- if \( \rho(s, t, \varphi) > 0 \) then \( \chi(s, t, \varphi) = 1 \)
- if \( \rho(s, t, \varphi) < 0 \) then \( \chi(s, t, \varphi) = -1 \)
- if \( \rho(s, t, \varphi) = 0 \) then \( \chi(s, t, \varphi) = 0 \)

**Proof.** See the extended version of this article for the proof. \( \square \)

To provide the correctness of the interval semantics over imprecise signals, we introduce the following lemma:

**Lemma 3.1 (Metric Lemma).** Let \( s_1, s_2 \in \mathbb{F}(\mathbb{L}, \mathbb{T}, \mathbb{D}^\varphi) \). For any \( t \in \mathbb{T} \), for any \( \varphi \in \mathbb{L} \), for any \( \delta > 0 \), we have:

- if \( ||s_1 - s_2||_\infty < \delta \) then \( ||\rho^\varphi_{s_1} - \rho^\varphi_{s_2}||_\infty < \delta \)

**Proof.** See the extended version of this article for the proof. \( \square \)

**Theorem 3.2 (Correctness of Robust Interval Semantics).** The Robust Interval Semantics of Definition 3.3 is correct w.r.t the Boolean Semantics of Definition 3.2, i.e., for any \( s \in \mathbb{F}(\mathbb{L}, \mathbb{T}, \mathbb{D}^\varphi) \), \( t \in \mathbb{L} \), \( \varphi \in \mathbb{L} \):

- if \( ||s_1 - s_2||_\infty < ||\rho(s_1, t, \varphi) - \rho(s_2, t, \varphi)||_\infty < \delta \) then \( \chi(s_1, t, \varphi) = \chi(s_2, t, \varphi) \)

for all \( i \leq n \), where \( || \cdot || \) is the interval radius of Definition 2.2.

**Proof.** See the extended version of this article for the proof. \( \square \)

## 4 ONLINE MONITORING

In this section a novel online (out-of-order) monitoring algorithm for STREL is presented. Differently from the standard offline approach, where all the data is available at the beginning of the execution, online monitoring is performed incrementally, when a new piece of data is available. In this case, the uncertainty related to the absence of information must be taken into account. For that aim, the machinery of imprecise signals can be exploited to represent the uncertainty, where the result of the monitoring process, whether it is a satisfaction or a robustness signal, is refined as soon as new updates of the input arrive.

The semantics for STREL is defined for arbitrary signals, but algorithms, for computational reason, are provided for piecewise constant ones, along the lines of [1, 7]. This class of signals is convenient, and frequently chosen as the class of reference for a number of reasons: (i) it naturally describes digital signals, (ii) it can be stored in memory very efficiently, and processed fast enough to be considered for real-time applications, (iii) it allows to express the vast majority of real-valued signals of practical use with a limited loss of information. Since the presented signals are Lipschitz-continuous (we consider only inequalities on the variables of the system), we can always bound our error, considering the minimum time step and the maximum of their individual Lipschitz constants. An imprecise piecewise-constant signal \( \sigma : \mathbb{T} \rightarrow I(\mathbb{R}^\infty) \), can be characterized in the following way:

\[ \sigma(t) = \begin{cases} 
I_{t_{i}}, & \text{for } t_{i} \leq t < t_{i+1} \\
I_{t_{t}}, & \text{for } t_{t} \leq t < \infty
\end{cases} \]

and graphically represented as in Figure 2. Note that frequently when monitoring real-time application the last part of the signal will be characterized by the widest interval possible, as this denotes the fact that the knowledge collected so far is insufficient for providing any insight about the monitored specification for future values of the signal. Similar infinite interval can be considered for missed values.

We consider space-synchronized (s.s.) signals, i.e., signals defined on the same time intervals for any location of the space model. More precisely, a p.c. s.s. signal \( s : \mathbb{L} \times \mathbb{T} \rightarrow \mathbb{D}^\varphi \) is a signal that can be represented as a sequence of pairs \( \{(t_i, V_i)\}_{i \in \mathbb{N}} \), where each pair \( (t_i, V_i) \) of the sequence represents a piece of the signal, such that it maps any time-instant between \( t_i \) and \( t_{i+1} \) in \( \mathbb{T} \), to the \( |\mathbb{L}| \times n \) matrix \( V_i \) that represents the values of the \( n \) dimensions of the signal at each location \( t \) in \( \mathbb{L} \). The space-synchronization restriction might appear to be a severe limitation, but this shows one of the conceptual differences between online and offline monitoring: in an offline setting, the space-synchronization hypothesis would likely have detrimental effects on the performances, as it would force all the processing to happen at a temporal granularity that is the union of the temporal granularities of the signals at the different locations. In an online setting, on the other hand, the temporal granularity is determined by the time when new information is available, and the
space-synchronization hypothesis makes it possible to exploit in future work the Single-Instruction Multiple-Data (SIMD) capabilities of modern processors (see [13, 18]), resulting in execution times that are virtually independent from the number of locations, when appropriate hardware is available. In this context, we call signal update \( u \) the triplet \((t_a, t_b, V)\), representing a mapping to the value matrix \( V \) for any time instant between \( t_a \) (included) and \( t_b \) (excluded). Signal updates can be seen as some special kinds of signals that we use to represent upcoming partial information from the online behavior of the monitored system. In the context of our analysis, we always assume updates to provide truthful information (the case of hard faults, future work), and, for that reason, we can always assume updates to be well-formed, meaning that the interval \( V \) they provide is always included in the previous interval of the signal we stored for that time and location. To express the online nature of the computation we want to pursue, we need some way of describing the incremental evaluation of new information.

**Definition 4.1 (Signal refinement).** Let \( s_1, s_2 \in \mathcal{F}(\mathbb{L}, T, D^n) \), we say that \( s_1 \) is refined by \( s_2 \), and we write \( s_1 \succ s_2 \), iff for any \( \ell \in \mathbb{L}, i \in T, i \leq n, \pi_\ell(s_2(t, \ell)) \subseteq \pi_\ell(s_1(t, \ell)) \), and there is some \( t' \in \mathbb{L}, t' \in T, i' \leq n, \) such that \( \pi_{\ell'}(s_2(t', \ell')) \subseteq \pi_{\ell'}(s_1(t', \ell')) \), i.e. each interval of the co-domain of the signal \( s_2 \) is contained in the corresponding interval of the signal \( s_1 \), and some of them are strictly contained.

The refinement relation expresses the fact that \( s_1 \) and \( s_2 \) represent the same information, except that \( s_2 \) has a smaller degree of uncertainty. By the notions of signal update and signal refinement, we can easily represent the online evolution of a signal as a chain of signal refinements \( s_0 > s_1 > s_2 > \ldots \), where the signal \( s_{j+1} \) at the step \( j + 1 \) can be computed from \( s_j \) and update \( u_j \) like in Algorithm 1.

The \textsc{Refine}() procedure takes a signal \( s_j \) as a sequence of ordered pairs, and an update as the triplet \((t_a, t_b, V)\). In practice, it removes all the pieces of the signal that start within the interval \([t_a, t_b]\), and adds a piece with value \( V \) in the case \( t_a \) and/or \( t_b \) lay in between of \( t_i \) and \( t_{i+1} \). Clearly, for efficiency reasons, the algorithm can jump to the next pair each time \( t_{i+1} < t_a \), and can terminate as soon as \( t_i > t_b \). The updated signal at the end of the execution is the next element of the refinement chain, i.e. \( s_{j+1} \).

**The Monitoring Problem.** When monitoring online a given specification \( \varphi \), let \( s_0 \) be the signal representing the starting information on which the atoms of the formula \( \varphi \) are defined. Let also \((u_j)_{j \in \mathbb{L}}\) denote a (finite or infinite) sequence of signal updates. The online monitoring problem can be framed as computing the robustness signal \( R_{\varphi, s_0}^{u_1 \ldots u_n} \) (or, alternatively, the satisfaction signal \( \chi_{\varphi, s_0}^{u_1 \ldots u_n} \)), with \( s_j \succ s_{j+1} \), starting from \( s_0 \). A naïve implementation of an online monitor could just ignore the information coming from previous monitoring steps and restart the computation over the whole signal each time new information is available. As already noted in [7, 8], such an implementation would result in huge amounts of wasted resources when monitoring time signals, and it would therefore be even more costly when monitoring space-time signals. To properly scope the effect that an out-of-order update of the input signal generates for the evaluation of a formula, it is convenient to think of updates as starting from the atoms of the monitored formula, and then propagating their effects up through the syntactic tree, generating a ripple effect where the impacted time span widens based on the operators of the subformulae. Figure 3 shows the ripple effect resulting from the propagation of update information through the syntactic tree. From this intuition we can define the update ripple function, to scope the resulting update’s time span, based on the provided update, and on the operator being computed in the following way:

**Definition 4.2 (Update Ripple).** Let \( \text{OP}_t \) denote any temporal operator on the time interval \( I \), given a signal update \( u \equiv (t_a, t_b, V) \), we call update ripple the function

\[
\text{ur}(u, \varphi) = \begin{cases} 
\{t_a, t_b\} - I & \varphi \equiv \text{OP}_t \varphi_1 \text{ or } \varphi \equiv \varphi_1 \text{OP}_t \varphi_2 \\
\{t_a, t_b\} & \text{otherwise}
\end{cases}
\]

Being able to assess the time boundaries of the effect of an update, we can therefore define an online monitoring procedure that updates the robustness (or satisfaction) signal when needed and that keeps the valid parts otherwise. In general, we have that at the step \( j + 1 \),

![Figure 2: A graphical view of a typical piecewise-constant imprecise signal. From the 8th second onward, there is total absence of knowledge about what values the signal could have, but until then, we have some bounds on the actual values observed.](image)

**Algorithm 1** Signal Refinement

1: procedure \textsc{Refine}(s : \{(t_0, V_0), \ldots, (t_N, V_N)\}, u : (t_a, t_b, V))
2: for \((t_i, V_i)\) in \( s \) do
3: if \( t_i < t_a < t_{i+1} \) and \( V \subset V_i \) then
4: \( s := s \cup (t_a, V) \)
5: else if \( t_i = t_a \) and \( V \subset V_i \) then
6: \( s := s \setminus (t_i, V_i) \cup (t_i, V) \)
7: else if \( t_a < t_i < t_b \) then
8: \( s := s \setminus (t_i, V_i) \)
9: end if
10: if \( t_i < t_b < t_{i+1} \) then
11: \( s := s \cup (t_b, V_i) \)
12: end if
13: end for
14: end procedure
Algorithm 2 Online Monitoring Procedure

1: procedure MONITOR(M, ϕ, u)
2: \hspace{1em} switch ϕ do
3: \hspace{2em} case p \circ c \rightarrow end procedure
4: \hspace{2em} case \neg ϕ or \dot{E}_{\geq_{q_{t}u}} \rightarrow end procedure
5: \hspace{2em} case ϕ \land ϕ_2 or ϕ \Rightarrow \rightarrow \hspace{1em} 1
6: \hspace{2em} case \text{COMPUTE}_P(O,P,V) \rightarrow \hspace{1em} 1
7: \hspace{2em} case \text{COMPUTE}_U(O,P,V) \rightarrow \hspace{1em} 1
8: \hspace{2em} case \text{COMPUTE}_E(O,P,V) \rightarrow \hspace{1em} 1
9: \hspace{2em} case \text{COMPUTE}_R(O,P,V) \rightarrow \hspace{1em} 1
10: \hspace{2em} case \text{UNBOUND}_{\wedge}(M, ϕ, u) \rightarrow \hspace{1em} 1
11: \hspace{2em} case \text{MONITOR}(M, ϕ, u) \rightarrow \hspace{1em} 1
12: \hspace{2em} case \text{REFINE}(M, [u]) \rightarrow \hspace{1em} 1
13: return \langle M, [u] \rangle \rightarrow \hspace{1em} 1
14: end procedure

Online Monitoring Of Non-temporal Operators When monitoring formulae containing Boolean or Spatial operators, the online evaluation can be performed very efficiently by simply updating the robustness signal at the times corresponding to the received update. Algorithm 3 shows the algorithm for monitoring atomic formulae, Algorithm 4 presents the one for monitoring unary operators (i.e. \neg and \dot{E}_{j}) and Algorithm 5 shows the one for binary operators (i.e. \lor and \dot{E}_{j}). We represent by \text{COMPUTE}_P(O,P,V) (and \text{COMPUTE}_U(O,P,V_1,V_2)) the execution of the semantic operation corresponding to the operator OP, along the lines of Definitions 3.2, 3.3, i.e. \lfloor \max \rfloor/\lfloor \min \rfloor direct computation for Booleans, and the classical reach/escape routines [1] for spatial operators. A key difference from the offline version of the spatial algorithms, however, is that in our online version the \text{COMPUTE}_P implementation has been crafted to enable spatial-parallelization, i.e. monitors’ users with appropriate hardware and the need to speed-up for large spaces, can opt-in for the multi-threaded version of the algorithm, where \text{COMPUTE}_P is executed in parallel for any location of the spatial model.

Algorithm 3 Atomic Formula Monitoring

1: function ATOM(M, p \circ c, u)
2: \hspace{1em} (t_a, t_b, V) \leftarrow u
3: \hspace{1em} u_{\text{PROC}} := (t_a, t_b, \text{COMPUTE}_P(p \circ c, V))
4: \hspace{1em} \text{REFINE}(M_{[p_{\text{PROC}}]} \cup [u_{\text{PROC}}])
5: return \langle M, [u_{\text{PROC}}] \rangle
6: end function

The algorithm for monitoring binary operators is slightly more complex, as it requires to take into account the corresponding value
Algorithm 4 Unary Operator Monitoring

```plaintext
1: function UNARY(M, OPψ, u)
2: ⟨M, {uψ}⟩ := MONITOR(M, ψ, u)
3: {uOPψ} := Ø
4: for (tₐ, tₚ, V) ∈ {uψ} do
5: {uOPψ} := {uOPψ} ∪ (tₐ, tₚ, COMPUTE_OP(OP,V))
6: end for
7: REFINE(M[ρOPψ], {uOPψ})
8: return ⟨M, {uOPψ}⟩
9: end function
```

of the other subformula when an update is processed. In this context, we indicate by SELECT(s, t₁, t₂) the restriction of the signal s to the time interval that starts at t₁ and ends at t₂ (excluded).

Algorithm 5 Binary Operator Monitoring

```plaintext
1: function BINARY(M, ψ₁, OPψ₂, u)
2: {uψ₁OPψ₂} := Ø
3: ⟨M, {uψ₁}⟩ := MONITOR(M, ψ₁, u)
4: for (tₐ, tₚ, Vψ₁) ∈ {uψ₁} do
5: Vψ₂ := SELECT(M[ρψ₂], tₐ, tₚ)
6: {uψ₁OPψ₂} := {uψ₁OPψ₂} ∪ (tₐ, tₚ, COMPUTE_OP(OP, Vψ₁, Vψ₂))
7: end for
8: for ψ₂ in ψ₂...
9: Repeat lines 3-8 symmetrically for ψ₂...
10: REFINE(M[ρOPψ₂], {uψ₁OPψ₂})
11: return ⟨M, {uψ₁OPψ₂}⟩
12: end function
```

Online Monitoring Of Temporal Operators To execute temporal operators quickly enough for online needs, on the other hand, we need to store some extra information throughout the process. Firstly, it is useful to recollect that, in general, every temporal operator can be decomposed [10] in the conjunction of two (efficiently computable) operators:

- the bounded eventually F₁ (or equivalently the bounded globally G₁)
- the unbounded until U

We propose here an enhanced algorithm for monitoring bounded globally/eventually operators with out-of-order updates. For that aim, we slightly adapted the classical sliding window algorithm from Lemire [19] so that it is constrained on the ur function and that it can deal seamlessly with numerical and interval values. Algorithm 6 presents primary routine of the sliding window for computing updates of bounded unary temporal operators OP₁. The algorithm exploits an additional data structure W that is a deque, such that new elements of the window are added at the end, and such that when the window is saturated (i.e. the elements inside denote a time span bigger than the definition interval I of the operator), they are removed from left and propagated as updates. The logic of the algorithm is essentially the following: for each update received in input, the sliding window is initialized on the fragment of the robustness signal of the subformula defined by the update ripple function ur. For each piece of the fragment, the sliding window is updated (line 10), and each time the new piece makes the data in the window exceed the maximum size, the sliding window slides to the right, removing from the window some elements that can be safely propagated as updates (line 8); some edge cases are not covered to keep the algorithm concise (e.g. the case when the current piece is by itself wider than the window size). The precise behaviour of SLIDE and ADD, that control the mutation of W can be examined in the extended version of the paper or in the Moonlight implementation.

Algorithm 6 Sliding Window

```plaintext
1: procedure SLIDINGWINDOW(M, OP[ła,tₚ]ψ, u)
2: ⟨M, {uψ}⟩ := MONITOR(M, ψ, u)
3: {uOPψ} := Ø
4: for uψ ∈ {uψ} do
5: (tₐ, tₚ) := ur(uψ, OP[ła,tₚ]ψ)
6: if t > tₚ + W.first.start then
7: {uOPψ} := {uOPψ} ∪ SLIDE(t − tₚ)
8: end if
9: ADD(t − tₚ, V)
10: end for
11: end procedure
```

The second fundamental temporal algorithm is the one for computing the unbounded until. Unfortunately, being unbounded, any update might require to recompute, in the worst case, the whole robustness/satisfaction signal. In our implementation, we consider the algorithm in [7]. Note that it requires to keep the minimum value of preceding computations of ψ₁ and the maximum value of preceding computations of the whole formula as secondary data structures. A last remark about the implementation must be made: while the algorithms have been developed with the goal to enable out-of-order execution, all of them have been implemented also in an in-order variant, so that the execution time penalty from not assuming that updates are at the end, does not affect the users of the tool, when the use case of interest allows to.

5 EXPERIMENTAL EVALUATION

The interval semantics we presented in Section 3 and the online (in-order and out-of-order) monitoring strategies of Section 4 have been implemented as extensions to the Moonlight tool. To showcase the kind of applications where they can be exploited, and to compare the performances with other state-of-the-art approaches, we propose three different examples: (i) we present and discuss the results here three different examples: (i) we present and discuss the results... (ii) we compare the performances of our approach for the evaluation of a temporal property on the Abstract Fuel Control Simulink model from the Breach [9] tool; (iii) we compare the performances of the online approach versus the offline version of Moonlight on a simulated sensor network adopting the ZigBee protocol.
5.1 Use case: Air pollution monitoring

Recalling Properties 1, 2 from Section 3, we can see in Figure 4 the results of the monitoring. Note that when both the upper and lower bounds are below the 0 threshold, the property is certainly violated, while when only the lower bound is below 0, then the property is potentially violated. Property 1 gives some important insights on the faults observed in Figure 1. In fact, we can see that of the six observed failures for the ten-days span of interest, only three happen for a time that is long enough to potentially trigger public concern, which correspond to the spikes to minus infinity in the lower bound of Figure 4 (left). In essence, with the interval semantics we learn that the property could potentially be violated in those time-spans, while it is certainly not for the other missing values. However, Property 2 tells us something more about the neighbourhood: in fact, by combining the observations registered from close location, it is apparent that just one of the failures (the one happening during March 20th) likely corresponds to a violation of the property, since there is no close location exhibiting low levels of nitrogen dioxide in Figure 4 (right).

5.2 Online comparison: Abstract Fuel Control

Consider a Simulink<sup>®</sup> model that describes a black-box representation of an engine’s air-fuel ratio controller aimed at complying to emission targets of a vehicle, where the user has direct control over the engine speed and pedal angle. Each input and output is represented as a signal that is sampled regularly, the outputs being the actual air-to-fuel ratio (AF), and the mean air-to-fuel ratio value for the given input parameters (AF<sub>ref</sub>) at a sampling period T = 0.1s. For a full description of the model, the reader can see [16], while [7] provides the reference implementation for online monitoring in Breach.

In our experiments, we monitored the following STL property (note that STREL is an extension of STL, and therefore each STL provides the reference implementation for online monitoring in Breach).

\[
\varphi = G_{10,30}([\text{AF} - \text{AF}_{\text{ref}}] > 0.1 \rightarrow (F_{[0,1]}[\text{AF} - \text{AF}_{\text{ref}}] < 0.1))
\]

for different sample sizes, considering both updates as an order chain and by shuffling them at random to simulate out-of-order retrieval and processing. The result of \([\text{AF} - \text{AF}_{\text{ref}}]\) from the model has been stored in a file and loaded before starting the stopwatch for both monitors, to eliminate the simulation and loading time from the performance evaluation. Breach monitor has been measured via the reference implementation as a Simulink model, while Moonlight is implemented as a Java program. Table 1 reports a summary of the performances of the monitors for different sample sizes.

| N. samples | Breach Exec. Time In-order | Moonlight Exec. Time In-order | Out-of-order |
|------------|---------------------------|-------------------------------|--------------|
| 500        | 7.603 s                   | 0.004 s                       | 0.157 s      |
| 1000       | 8.143 s                   | 0.016 s                       | 0.489 s      |
| 5000       | 10.770 s                  | 0.096 s                       | 9.790 s      |
| 10000      | 13.730 s                  | 0.113 s                       | 44.894 s     |

Table 1: Performances for monitoring the property \(\varphi\) for different sample size. Times are averaged over 100 runs.

The interesting insight of the comparison is the fact that, while our in-order implementation provides reliably faster performances (note that the offline version of Moonlight had already shown better performance than Breach in [2]), the penalty that comes from not assuming ordered inputs grows substantially with the increase of the input size, as this requires longer searches in the output signal, to find the spot where the update should be applied. Nevertheless, the biggest sample size we considered is quite extreme (ten thousand randomly-shuffled samples), yet the execution time (4.489 ms/sample on average) is way smaller than the sampling time (0.1 s), which therefore makes it reasonable for most real-time scenarios.

5.3 Moonlight comparison: ZigBee Protocol

Consider a collection of moving devices communicating via the ZigBee (IEEE 802.15.4) protocol. From the protocol description we know that the devices can have three roles: they can either be coordinators, routers or sensor-node. Each device is equipped with an humidity sensor \(h(t)\) that reports at each time instant the observed value of the humidity at the current location. The humidity observed can be described as an MA(0) process, i.e.

\[
h(t) = c_0 + \epsilon(t), \quad (\epsilon \sim WN(0, \lambda^2))
\]

where the observed value comes from the real value \(c_0\), with some perturbation from the zero-mean white noise \(\epsilon\) of variance \(\lambda^2\). Each device can communicate with the ones that are close enough directly, but they can also communicate with furthest ones, as long as there is some router between them that can bridges the communication. Let \(X_H\) denote the true value of humidity for a given device at a given time, let \(X_S\) denote the role of a given sensor, and \(T\) some time threshold to warn the observers. We monitored the following properties on the system:

\[
\varphi_1 = (X_H > 60) \rightarrow F_{[0,T]}(X_H < 30)
\]

\[
\varphi_2 = \Box_{d<10}(X_S = \text{coordinator})
\]

Property \(\varphi_1\) denotes an alert condition: if the humidity measured by a device \(X_H\) goes beyond 60%, then it must fall down at 30% afterwards, within the time threshold \(T\). Property \(\varphi_2\), on the other hand, defines a reachability criterion between the sensors: it checks whether it is true that from any location, it is possible to reach a coordinator \((X_S = \text{coordinator})\) in less than 10 hops. Similarly to previous versions of this model [1, 6], we can consider the spatial model as a graph where all the devices are the nodes, and the edges between the nodes are all labeled by \(d = 1\) to denote the networking hop from one device to another. Table 2 shows the difference in monitoring \(\varphi_1\) and \(\varphi_2\) both online and offline. It is interesting to see how the different algorithms behave on same formula and data: in fact, the online temporal algorithms are penalized by the complexity added by the fact that some values must be recomputed. Conversely, the online spatial ones benefit from the hypothesis of spatial synchronization of the locations, resulting in a slightly more efficient computations in the case we explored. Lastly, it can be seen that the benefit of parallelization is particularly evident when the number of nodes is strictly smaller than the number of cores of the CPU (10 in our case), while the benefit practically vanishes (actually resulting in more overhead) as the number of parallel threads grows significantly more than the cores available.
We extended the traditional definition of signals to also consider imprecise signals defined by intervals of values. We presented an interval semantics for STREL, we proved its soundness and correctness, and we introduced an online monitoring algorithm for STREL that exploits imprecise signals that can be refined by updates arriving in any order, and that can monitor updates on different locations in parallel. We implemented the proposed methodology in the Moonlight monitoring tool. We motivated our framework from an air pollution control specification with real data from the region of Lombardy, Italy. Lastly, we compared the new methodology with other state-of-the-art tools, and discussed the differences. Many directions of future work can be followed, for example, the space-synchronization hypothesis helped us simplifying the implementation of the algorithms, but is not needed from a theoretical point of view. It will be interesting in the future to clearly assess the satisfiability/robustness have been reached, as well as intensive parallelization and hardware acceleration of the main algorithms.

### 6 CONCLUSIONS & FUTURE WORK

We extended the traditional definition of signals to also consider imprecise signals defined by intervals of values. We presented an interval semantics for STREL, we proved its soundness and correctness, and we introduced an online monitoring algorithm for STREL that exploits imprecise signals that can be refined by updates arriving in any order, and that can monitor updates on different locations in parallel. We implemented the proposed methodology in the Moonlight monitoring tool. We motivated our framework from an air pollution control specification with real data from the region of Lombardy, Italy. Lastly, we compared the new methodology with other state-of-the-art tools, and discussed the differences. Many directions of future work can be followed, for example, the space-synchronization hypothesis helped us simplifying the implementation of the algorithms, but is not needed from a theoretical point of view. It will be interesting in the future to clearly assess the satisfiability/robustness have been reached, as well as intensive parallelization and hardware acceleration of the main algorithms.

ACKNOWLEDGMENTS

The authors would like to acknowledge Davide Prandini for his thesis work (unpublished) where a preliminary work on imprecise signals for STL had been conducted, together with many ideas that have been used for developing the proofs of the theorems presented. This research has been partially supported by the Austrian FWF projects ZK-35 and LogiCS DK W1255-N23; and by Italian MIUR project PRIN 2017FTXR7S IT MATTERS and by Marche Region in implementation of the financial programme POR MARCHE FESR 2014-2020, project "Miracle".

REFERENCES

[1] Ezio Bartocci, Luca Bortolussi, Michele Loreti, and Laura Nenzi. 2017. Monitoring mobile and spatially distributed cyber-physical systems. In Proc. of MEMOCODE 2017: the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design. ACM, Vienna, Austria, 146–155. https://doi.org/10.1145/3127041.3127050

[2] Ezio Bartocci, Luca Bortolussi, Michele Loreti, Laura Nenzi, and Simone Silvetti. 2020. MoonLight: A Lightweight Tool for Monitoring Spatio-Temporal Properties. In Runtime Verification, Jyotirmoy Deshmukh and Dejan Niˇckovi´c (Eds.). Springer International Publishing, Cham, 417–428.

[3] Ezio Bartocci, Luca Bortolussi, Dimitrios Milios, Laura Nenzi, and Guido Sanguinetti. 2015. Studying Emergent Behaviours in Morphogenesis Using Signal Spatio-Temporal Logic. In Hybrid Systems Biology, Alessandro Abate and David Šárnánek (Eds.). Springer International Publishing, Cham, 156–172.

[4] Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donzé, Georgios Fainekos, Oded Maler, Dejan Niˇckovi´c, and Sriram Sankaranarayanan. 2018. Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications. Springer International Publishing, Cham, 135–175. https://doi.org/10.1007/
A PROOFS

A.1 Proof of Theorem 3.1

Proof. The theorem can be proved by induction on the subformulae of the formula $\varphi$.

- $\varphi \equiv T$: immediate from semantics, since $\rho(s, t, T) > 0$, and $\chi(s, t, T) = 1$ for any $s, t$.
- $\varphi \equiv \bot$: immediate from semantics, since $\rho(s, t, \bot) < 0$, and $\chi(s, t, \bot) = -1$ for any $s, t$.
- $\varphi \equiv p \circ c$: immediate from semantics, in fact, we can distinguish two cases based on $\circ$. (i) let $o \equiv ' > ', \rho(s, t, p \circ c) > 0$ iff $\pi_p(s, t, f(t)) > c$, which implies $\chi(s, t, p \circ c) = 1$; conversely $\rho(s, t, p \circ c) < 0$ iff $\pi_p(s, t, f(t)) < c$, but this means $\pi_p(s, t, f(t)) < 0$, which implies $\chi(s, t, p \circ c) = -1$. (ii) let $o \equiv ' < ', \rho(s, t, p \circ c) < 0$ iff $c - \pi_p(s, t, f(t)) > 0$, but this means $\pi_p(s, t, f(t)) > c$, which implies $\chi(s, t, p \circ c) = 1$; conversely $\rho(s, t, p \circ c) > 0$ iff $c - \pi_p(s, t, f(t)) < 0$, but this means $\pi_p(s, t, f(t)) > c$, which implies $\chi(s, t, p \circ c) = -1$. The case for $0 \in \rho(s, t, p \circ c)$ trivially holds by semantics since $\chi(s, t, p \circ c) = 0$ in all other cases.
- $\varphi \equiv \neg \varphi_2$: Let $\rho(s, t, \neg \varphi) > 0$, from semantics we have that $\rho(s, t, \neg \varphi) = -\rho(s, t, \varphi)$, and therefore $\rho(s, t, \neg \varphi) < 0$.
- By inductive hypothesis $\chi(s, t, \neg \varphi) = -1$, then $\chi(s, t, \varphi) = 1$. Let $\rho(s, t, \varphi) < 0$, from semantics we have that $\rho(s, t, \varphi) = -\rho(s, t, \neg \varphi)$, and therefore $\rho(s, t, \varphi) > 0$.
- By inductive hypothesis $\chi(s, t, \varphi) = 1$, then $\chi(s, t, \neg \varphi) = -1$. The case for $0 \in \rho(s, t, \varphi)$ is similar.
- $\varphi \equiv \varphi_1 \lor \varphi_2$: consider $\{\max(\rho(s, t, \varphi_1), \rho(s, t, \varphi_2))\}$.
- Let $\{\max(\rho(s, t, \varphi_1), \rho(s, t, \varphi_2))\}$.
- By inductive hypothesis, the theorem holds on both $\varphi_1$ and $\varphi_2$, and therefore either $\chi(s, t, \varphi_1) = 1$ or $\chi(s, t, \varphi_2) = 1$. But, by semantics we have that $\chi(s, t, \varphi_1, \varphi_2) = 1$, and therefore the theorem holds.
- Let $\{\min(\rho(s, t, \varphi_1), \rho(s, t, \varphi_2))\} < 0$.
- By inductive hypothesis, the theorem holds on both $\varphi_1$ and $\varphi_2$, and therefore $\chi(s, t, \varphi_1) = \chi(s, t, \varphi_2) = -1$. But, by semantics we have that $\chi(s, t, \varphi_1, \varphi_2) = -1$, and therefore the theorem holds.

The case for $0 \in \rho(s, t, \varphi_1, \varphi_2)$ is similar.

- $\varphi \equiv \varphi_1 \land \varphi_2$: consider $\{\max(\rho(s, t, \varphi_1), \rho(s, t, \varphi_2))\}$, with $\{\min(\rho(s, t, \varphi_1), \rho(s, t, \varphi_2))\}$.
- Let $\{\max(\rho(s, t, \varphi_1), \rho(s, t, \varphi_2))\} > 0$.
- By inductive hypothesis, this means that $\chi(s, t, \varphi_1, \varphi_2) = 1$, and all $t' \in [t', t]$. Therefore, by semantics, we have that $\chi(s, t, \varphi_1, \varphi_2) = -1$.
- $\varphi \equiv \neg \varphi_3$: consider $\{\max(\rho(s, t, \varphi_3), \rho(s, t, \varphi_3))\}$, with $\{\min(\rho(s, t, \varphi_3), \rho(s, t, \varphi_3))\}$.
- Let $\{\max(\rho(s, t, \varphi_3), \rho(s, t, \varphi_3))\} > 0$.
- By inductive hypothesis, this means that $\chi(s, t, \varphi_3, \varphi_3) = 1$, for all $t' \in [t', t]$. Therefore, by semantics, we have that $\chi(s, t, \varphi_3, \varphi_3) = 1$, and the theorem holds.
- $\{\min(\rho(s, t, \varphi_3), \rho(s, t, \varphi_3))\} < 0$.
- By inductive hypothesis, this means that $\chi(s, t, \varphi_3, \varphi_3) = -1$. Therefore, the theorem holds.

The case for $0 \in \rho(s, t, \varphi_3)$ is similar.

A.2 Proof of Lemma 3.1

Proof. The theorem can be proved by induction on the subformulae of the formula $\varphi$.

- $\varphi \equiv T|\bot$: $||p_1^\varphi - p_2^\varphi||_\infty = 0$ for any $s_1, s_2$.
- $\varphi \equiv \rho: c$: $||p_1^\varphi - p_2^\varphi||_\infty < \delta$ implies that $\max_{\rho \in T} \max_{\varphi_1, \varphi_2} \{d_H(\pi_\varphi(s_1(t), t)), d_H(\pi_\varphi(s_2(t), t))\} < \delta$ for any $\varphi \in AP$, but for hypothesis semantics we know that $\max_{\varphi_1, \varphi_2} \{d_H(\pi_\varphi(s_1(t), t)), d_H(\pi_\varphi(s_2(t), t))\} < \delta$ for any $s_1, s_2$.
- $\varphi \equiv ~\neg \psi$: $||p_1^\varphi - p_2^\varphi||_\infty = \max_{\varphi_1, \varphi_2} \max_{\rho \in T} \{d_H(\rho(s_1^\varphi(t), t), \rho(s_2^\varphi(t), t))\} = ||p_1^\varphi - p_2^\varphi||_\infty$. For inductive hypothesis we have that $||p_1^\varphi - p_2^\varphi||_\infty < \delta$ for some $t \in T$. Therefore, the theorem holds.
- $\varphi \equiv \psi \land \varphi_2$: $||p_1^\varphi - p_2^\varphi||_\infty = \max_{\varphi_1, \varphi_2} \max_{\rho \in T} \{d_H(\rho(s_1^\varphi(t), t), \rho(s_2^\varphi(t), t))\} = ||p_1^\varphi - p_2^\varphi||_\infty$. The proof can be split in two cases:
  - (1) $|\varphi_1| - |\varphi_2| < \delta$: at any time instant $t$, we will have four cases, depending on the values between $p_{s_1}^\varphi(t), p_{s_2}^\varphi(t), p_{s_1}^\varphi(t), p_{s_2}^\varphi(t)$, and $p_{s_1}^\varphi(t), p_{s_2}^\varphi(t)$. The proof can be split in two cases:
    - (2) $|\varphi_1| - |\varphi_2| > \delta$: at any time instant $t$, we will have four cases, depending on the values between $p_{s_1}^\varphi(t), p_{s_2}^\varphi(t), p_{s_1}^\varphi(t), p_{s_2}^\varphi(t)$, and $p_{s_1}^\varphi(t), p_{s_2}^\varphi(t)$.
(a) \( |\rho_{s_2}^{\psi}(t, t) - \rho_{s_2}^{\psi}(t, t')| < \delta \) with \( \rho_{s_2}^{\psi}(t, t) \geq \rho_{s_2}^{\psi}(t, t') \) and \( \rho_{s_2}^{\psi}(t, t') \geq \rho_{s_2}^{\psi}(t, t) \). Then for inductive hypothesis the theorem holds.

(b) \( |\rho_{s_2}^{\psi}(t, t) - \rho_{s_2}^{\psi}(t, t')| < \delta \) with \( \rho_{s_2}^{\psi}(t, t) \geq \rho_{s_2}^{\psi}(t, t') \) and \( \rho_{s_2}^{\psi}(t, t') \geq \rho_{s_2}^{\psi}(t, t) \). Then for inductive hypothesis the theorem holds.

(c) \( |\rho_{s_2}^{\psi}(t, t) - \rho_{s_2}^{\psi}(t, t')| < \delta \) with \( \rho_{s_2}^{\psi}(t, t) \geq \rho_{s_2}^{\psi}(t, t') \) and \( \rho_{s_2}^{\psi}(t, t') \geq \rho_{s_2}^{\psi}(t, t) \). Either (i) \( \rho_{s_2}^{\psi}(t, t) < \rho_{s_2}^{\psi}(t, t') + \delta \), which, combined with case’s hypothesis, implies that \( \rho_{s_2}^{\psi}(t, t) > \rho_{s_2}^{\psi}(t, t') - \delta \geq \rho_{s_2}^{\psi}(t, t') - \delta \), and therefore \( \rho_{s_2}^{\psi}(t, t) - \rho_{s_2}^{\psi}(t, t') > \delta \), which is true by inductive hypothesis. Or (ii) \( \rho_{s_2}^{\psi}(t, t) > \rho_{s_2}^{\psi}(t, t') - \delta \), which, combined with case’s hypothesis, implies that \( \rho_{s_2}^{\psi}(t, t) > \rho_{s_2}^{\psi}(t, t') - \delta \geq \rho_{s_2}^{\psi}(t, t') - \delta \), and therefore \( \rho_{s_2}^{\psi}(t, t) - \rho_{s_2}^{\psi}(t, t') > \delta \), which is true by inductive hypothesis.

(d) \( |\rho_{s_2}^{\psi}(t, t) - \rho_{s_2}^{\psi}(t, t')| < \delta \) with \( \rho_{s_2}^{\psi}(t, t) \geq \rho_{s_2}^{\psi}(t, t') \) and \( \rho_{s_2}^{\psi}(t, t') \geq \rho_{s_2}^{\psi}(t, t) \). This case is analogous to the previous one, with the robustness signals switched.

2. \( \|\varphi\|_{\mathcal{L}_{\infty}} < \delta \). This case is the dual of the previous one, where \( \varphi \) is substituted with \( -\varphi \).

• \( \varphi \equiv \varphi_{1(\mathbb{L})}^\forall : \rho_{s_1}^\varphi - \rho_{s_2}^\varphi \|_{\mathcal{L}_{\infty}} = \max_{t' \in \mathbb{T}} \max_{t \in \mathbb{T}} d_H(t, t') \)< \( \delta \), which requires that for any \( t \in \mathbb{L}, t' \in \mathbb{T} \), both \( \|\varphi\|_{\mathcal{L}_{\infty}} < \delta \), and

\[ \|\varphi\|_{\mathcal{L}_{\infty}} < \delta, \text{ with } \|\varphi_{1(\mathbb{L})}^\forall\|_{\mathcal{L}_{\infty}} = \max_{t \in \mathbb{L}} \max_{t' \in \mathbb{T}} \left( \max_{t' \in \mathbb{T}} d_H(t, t') \right) \]

1. \( \|\varphi\|_{\mathcal{L}_{\infty}} < \delta \): for any \( t \in \mathbb{L} \), for any \( t' \in \mathbb{T} \), we have that \( \|\varphi\|_{\mathcal{L}_{\infty}} < \delta \) iff for all \( t' \in I + t, A < \delta \), with \( \|\varphi\|_{\mathcal{L}_{\infty}} = 0 \) and \( \|\varphi\|_{\mathcal{L}_{\infty}} = 0 \).

A.3 Proof of Theorem 3.2

PROOF. The proof can be split in three possible cases for the hypothesis, depending on the value of \( \|\rho_{s_1}(t, t, \varphi)\|_{\mathcal{L}_{\infty}} \).

• Let \( \rho_{s_1}(t, t, \varphi) > 0 \): we must then have that, \( \|\varphi\|_{\mathcal{L}_{\infty}} < \rho_{s_1}(t, t, \varphi) \). Moreover, by Theorem 3.1 we have that \( \|\varphi\|_{\mathcal{L}_{\infty}} < \rho_{s_1}(t, t, \varphi) \). But this implies that \( \rho_{s_1}(t, t, \varphi) < \rho_{s_1}(t, t, \varphi) \), from the Definition 2.8. From definition of \( d_H \), this also implies that \( \rho_{s_1}(t, t, \varphi) < \rho_{s_1}(t, t, \varphi) \), and therefore that \( \rho_{s_1}(t, t, \varphi) < \rho_{s_1}(t, t, \varphi) \), which by Theorem 3.1 implies \( \rho_{s_1}(t, t, \varphi) \).

• Let \( \rho_{s_1}(t, t, \varphi) < 0 \): this case is symmetric to the first one, for \( \rho_{s_1}(t, t, \varphi) < 0 \).

Let \( 0 < \rho_{s_1}(t, t, \varphi) < 0 \): we have that \( \|\varphi\|_{\mathcal{L}_{\infty}} < \rho_{s_1}(t, t, \varphi) \). But this implies that \( \rho_{s_1}(t, t, \varphi) < \rho_{s_1}(t, t, \varphi) \), from the Definition 2.8. From definition of \( d_H \), this also implies that \( \rho_{s_1}(t, t, \varphi) < \rho_{s_1}(t, t, \varphi) \), and therefore that \( \rho_{s_1}(t, t, \varphi) < \rho_{s_1}(t, t, \varphi) \), which by Theorem 3.1 implies \( \rho_{s_1}(t, t, \varphi) \).

* Choosing \( \rho_{s_1}(t, t, \varphi) < 0 \), from the metric we have that \( \rho_{s_1}(t, t, \varphi) < \rho_{s_1}(t, t, \varphi) \), but this implies that \( d_H (\rho_{s_1}(t, t, \varphi), \rho_{s_1}(t, t, \varphi)) \) from the Definition 2.8. From definition of \( d_H \), this also implies
that $|\overline{\rho}(s_1, t, \varphi) - \overline{\rho}(s_2, t, \varphi)| < \overline{\rho}(s_1, t, \varphi)$. But this also implies that $\overline{\rho}(s_1, t, \varphi) < \overline{\rho}(s_1, t, \varphi) + \overline{\rho}(s_2, t, \varphi)$, and therefore that $\overline{\rho}(s_2, t, \varphi) > 0$.

* Choosing $\delta := -\rho(s_1, t, \varphi)$, from the metric lemma we have that $||\rho_s^p(\varphi) - \rho_s^p(\varphi)|| < -\rho(s_1, t, \varphi)$, but this implies that $d_{t, t}(\rho(s_1, t, \varphi), \rho(s_2, t, \varphi)) < -\rho(s_1, t, \varphi)$, from Definition 2.8. From definition of $d_{t, t}$, this also implies that $|\rho(s_1, t, \varphi) - \rho(s_2, t, \varphi)| < -\rho(s_1, t, \varphi)$. But this also implies that $\rho(s_1, t, \varphi) - \rho(s_2, t, \varphi) < -\rho(s_1, t, \varphi)$, and therefore that $\rho(s_2, t, \varphi) < 0$.

By Theorem 3.1, since we have $0 \in \rho(s_2, t, \varphi)$ then we must have $\gamma(s_2, t, \varphi) = 0$.

\[ \Box \]

## B ALGORITHMS

Sliding window mutation primitives are described here.

**Algorithm 7 Slide**

1. function SLIDE(t)
2.   \((t_i, V_i) := W\text{.removeFirst}()\)
3. while \(W\text{.first.start} < t\) do
4.   \((t_{i+1}, V_{i+1}) := W\text{.removeFirst}()\)
5.   \(\{u^{OP}\} := \{u^{OP}\} \cup (t_i, t_{i+1}, V_i)\)
6.   if \(t_{i+1} > t\) then
7.     \(W\text{.addFirst}(t_i, V_i)\)
8.     \(W\text{.addFirst}(t_{i+1}, V_{i+1})\)
9.   else
10.   \((t_i, V_i) := (t_{i+1}, V_{i+1})\)
11. end if
12. end while
13. \(\{u^{OP}\} := \{u^{OP}\} \cup (t_i, t_{i+1}, V_i)\)
14. \(W\text{.addFirst}(t_i, V_i)\)
15. return \(\{u^{OP}\}\)
16. end function

**Algorithm 8 Add**

1. procedure ADD(\(OP, t, V\))
2.   \((t_2, V_2) := W\text{.removeLast}()\)
3.   \(V_3 := \text{COMPUTE}_{OP}(OP, V, V_2)\)
4. if \(V_2 = V\) then
5.   \(W\text{.addLast}(t_2, V_2)\)
6. else if \(V_3 = V\) then
7.   \(\text{ADD}(OP, t_2, V_3)\)
8. else if \(V_3 \neq V_2\) then
9.   \(\text{ADD}(OP, t_2, V_3)\)
10. \(W\text{.addLast}(t, V)\)
11. else
12.   \(W\text{.addLast}(t_2, V_2)\)
13.   \(W\text{.addLast}(t, V)\)
14. end if
15. end procedure