Online Monitoring of Spatio-Temporal Properties for Imprecise Signals

Ennio Visconti
TU Wien
Vienna, Austria

Ezio Bartocci
TU Wien
Vienna, Austria

Michele Loreti
University of Camerino
Camerino, Italy

Laura Nenzi
TU Wien
Vienna, Austria
University of Trieste
Trieste, Italy

ABSTRACT

From biological systems to cyber-physical systems, monitoring the behavior of such dynamical systems often requires reasoning about complex spatio-temporal properties of physical and 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 reason about spatio-temporal properties. STREL considers each system's entity as a node of a dynamic weighted graph representing its spatial arrangement. Each node generates a set of mixed-analog signals describing the evolution over time of computational and physical quantities characterizing 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, 11 pages. https://doi.org/10.1145/3487212.3487344

MEMOCODE '21, November 20–22, 2021, Beijing, China
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-9127-6/21/11…$15.00
https://doi.org/10.1145/3487212.3487344

1 INTRODUCTION

Complex emergent spatio-temporal patterns such as traffic congestion or traveling waves are central in understanding networked dynamical systems where locally interacting entities operate at different temporal and spatial scales. We can observe these patterns both in biological systems [3, 5, 11] as well as human-engineered artifacts 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. Examples 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 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 policymaker 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. Policymakers 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

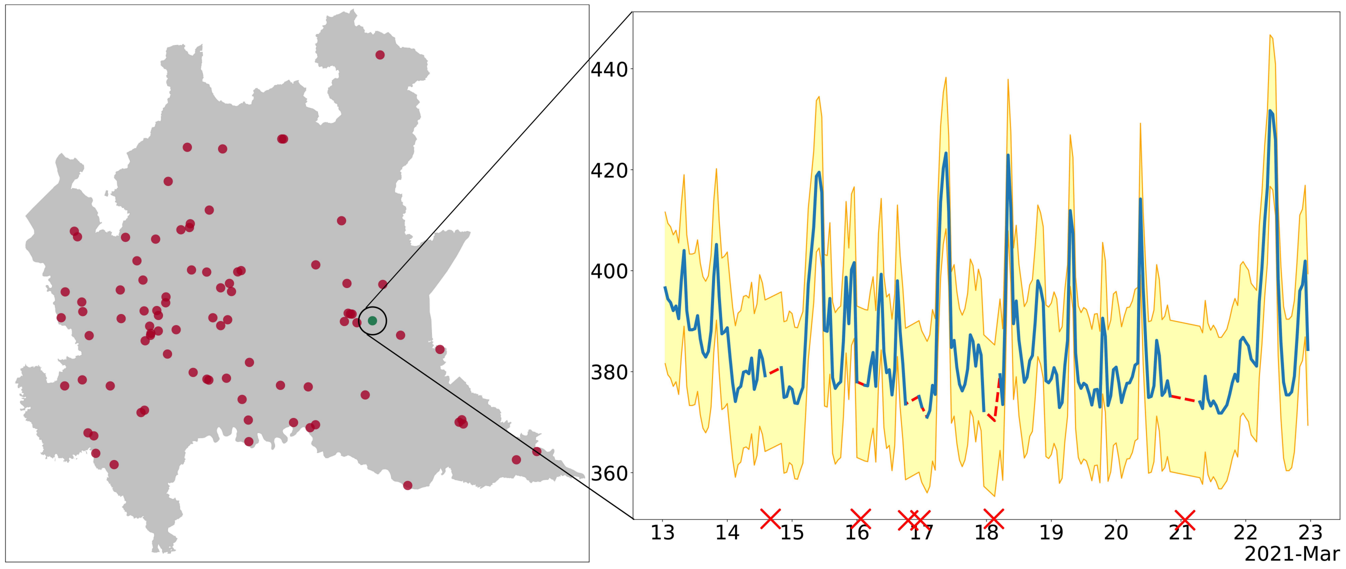

Figure 1: On the left the NO_2 measuring stations of Lombardy; the lighter-colored dot represents the station 6859 - Rezzato. On the right the detailed view of the NO_2 measured by the Rezzato station [20] in terms of $\mu g/m^3$. Red crosses and dotted line represent missing values. Data from the open data initiative of ARPA Lombardia [21].

the location of these stations, and the value reported in the town of Rezzato for the first quarter of 2021. The measurements happen regularly every hour, but sometimes the sensors fail to communicate the measurements because of meteorological issues or temporary faults, and the actual values are provided later on. Furthermore, the values measured by the sensors are noisy, and they have a certain degree of uncertainty. Another way to deal with the missing values could be to check that nearby locations (e.g., within 10 Km) do not register alarming values of particles in the air, which is possible only if we consider spatio-temporal properties.

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 characterizing the node's behavior. STREL extends the Signal Temporal Logic (STL) [25] with the *reach* and *escape* operators that generalize 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 provide, 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 actual 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 monitoring online imprecise spatio-temporal signals (signals are defined on intervals, not just the robustness) where the samples can also be processed 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 robust interval semantics' soundness and the correctness. 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 also demonstrate 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 essential 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 $\mathcal{I}(\mathbb{R}^\infty)$ be the set of intervals defined over the set $\mathbb{R}^\infty \equiv \mathbb{R} \cup \{+\infty, -\infty\}$. We call *closed interval* (or simply *interval*) any set $I \subseteq \mathbb{R}^\infty$ such that $I \equiv [a, b] := \{x \in \mathbb{R}^\infty : a \leqslant x \leqslant b; a, b \in \mathbb{R}^\infty\}$. For any $I \equiv [a, b] \in \mathcal{I}(\mathbb{R}^\infty)$, we will indicate as $\underline{I} \equiv a$ and $\overline{I} \equiv b$ the extremes of the interval.

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

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

$$c + I := [\underline{I} + c, \overline{I} + c] \qquad -I := [-\overline{I}, -\underline{I}]$$
$$I_1 + I_2 := [\underline{I_1} + \underline{I_2}, \overline{I_1} + \overline{I_2}] \qquad I_1 - I_2 := I_1 + (-I_2)$$
$$[\max](I_1, I_2) := [\max(\underline{I_1}, \underline{I_2}), \max(\overline{I_1}, \overline{I_2})]$$
$$[\min](I_1, I_2) := [\min(\underline{I_1}, \underline{I_2}), \min(\overline{I_1}, \overline{I_2})]$$

We also consider the extensions of [min] and [max] operators defined over an arbitrary subset $A \subseteq \mathcal{I}(\mathbb{R}^\infty)$, denoted by {} instead of () for function arguments.
We call *interval radius* of I, the operator:

$$|I| := [\min(|\underline{I}|, |\overline{I}|), \max(|\underline{I}|, |\overline{I}|)]$$

Interval relations are described in this way:

Definition 2.3 (Interval Inequalities)**.** Let $I_1, I_2 \in \mathcal{I}(\mathbb{R}^\infty)$, we say that $I_1 < I_2$ when $\overline{I_1} < \underline{I_2}$. Symmetrically, we say that $I_1 > I_2$ when $\underline{I_1} > \overline{I_2}$[2].

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 $\langle M, d\rangle$, we will call (Hausdorff) distance the function $d_H : \mathcal{P}(X) \times \mathcal{P}(X) \to \mathbb{R}_{\geq 0}$ defined as

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

In practice, in our context, we can limit at considering the metric space defined by the euclidean distance over the real numbers, and thus d_H reduces to computing $\max(|\underline{I_1} - \underline{I_2}|, |\overline{I_1} - \overline{I_2}|)$ for any two $I_1, I_2 \in \mathcal{I}(\mathbb{R}^\infty)$, although for doing that we say, by definition, that if both $\underline{I_1}, \underline{I_2}$ (or both $\overline{I_1}, \overline{I_2}$) 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 $D \subseteq \mathcal{I}(\mathbb{R})$ 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 will be convenient in the upcoming sections 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 $\pi_i : D_1 \times \cdots \times D_i \times \cdots \times D_n \to D_i$ be the function that takes the i-th projection of the set-theoretic Cartesian product D^n, we will indicate as $\pi_i(\sigma(t))$ 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: $\mathbf{s} : \mathbb{L} \times \mathbb{T} \to D^n$

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 $\mathbf{s} : \mathbb{L} \times \mathbb{T} \to D$ returns at each time, in each location the value of NO_2, $\mathbf{s}(\ell, t)$.

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

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

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

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 $\mathcal{S} = \langle \mathbb{L}, W\rangle$ [3], where $\mathbb{L}$ is a set of locations and $W \subseteq \mathbb{L} \times \mathbb{R}^\infty_{\geq 0} \times \mathbb{L}$ is

[1] Source code available at: `github.com/MoonLightSuite/MoonLight`
[2] We will write $I < c$ (respectively $I > c$) in place of $I < [c, c]$ (resp. $I > [c, c]$)

[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].

a *proximity function* associating at most one label $w \in \mathbb{R}^{\infty}_{\geq 0}$ to each distinct pair $\ell_1, \ell_2 \in \mathbb{L}$

A prominent 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 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 τ on $\mathcal{S}$ is a (potentially infinite) sequence $\ell_0, \ell_1, \dots \ell_k \dots$, such that for any $\ell_i, \ell_{i+1} \in \tau$, there is a label $(\ell_i, w, \ell_{i+1}) \in W$. We indicate by $\Lambda(\ell)$ the set of routes on $\mathcal{S}$ starting at ℓ. Moreover, we will use $\tau[i]$ to denote the i-th node of the route, $\tau[i\dots]$ to denote the subroute starting at the i-th node, and $\tau(\ell_i)$ to denote the first occurrence of ℓ_i in τ. Lastly, we will indicate by $\ell_1 < \tau(\ell_2)$ the fact that ℓ_1 precedes ℓ_2 in the route τ.

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 τ on $\mathcal{S}$, the distance $d_\tau[i]$ is:

$$d_\tau[i] = \begin{cases} 0, & \text{if } i = 0 \\ w + d_{\tau[1\dots]}[i-1], & \text{if } i > 0 \text{ and } (\tau[0], w, \tau[1]) \in W \end{cases}$$

Lastly, routes allow us to conveniently define the distance between any two locations $\ell_1, \ell_2 \in \mathbb{L}$. From the location ℓ_1 to ℓ_2, one can consider the minimal distance among all the routes starting at ℓ_1 and ending in ℓ_2:

$$d_{\mathcal{S}}[\ell_1, \ell_2] = \min_{\tau \in \Lambda(\ell_1)} \{d_\tau[\ell_2]\}$$

3 STREL WITH INTERVAL SEMANTICS

We present in this section an interval semantics that allows for a conservative analysis that considers both the minimum and the maximum values of intervals. This way, a plethora of use scenarios can fit into this specification language, spanning from traditional offline monitoring of a given specification over imprecise signals to online monitoring with out-of-order updates. All the proofs of theorems and lemmas can be checked in the extended version of this article at [33].

Definition 3.1 (STREL Syntax)**.** We consider logical formulae belonging to the language $\mathcal{L}$ generated by the following BNF grammar:

$$\varphi := \top \mid \bot \mid p \ \circ \ c \mid \neg \varphi \mid \varphi \ \vee \ \varphi \mid \varphi \, \mathrm{U}_I \, \varphi \mid \varphi \, \mathcal{R}_{\leq d} \varphi \mid \mathcal{E}_{\geq d} \varphi$$

where $\circ \in \{>, <\}$, $c \in \mathbb{R}$, p is associated to a projection function π of Definition 2.6, i.e. $p \ \circ \ c$ are inequalities on the variables of the systems. U_I is the *until* temporal operator, with I real interval, while $\mathcal{R}_{\leq d}$ and $\mathcal{E}_{\geq d}$ are the spatial operators *reach* and *escape* , with $d \in \mathbb{R}_{>0}$. In addition, we have the derived Boolean operators as *and* ($\wedge$) and *implies* ($\rightarrow$), temporal operators *eventually* (F_I) and *globally* (G_I), and spatial operators *somewhere* ($\diamonddiamond_{\leq d}$) and *everywhere* ($\boxdot_{\leq d}$).

Considering the air pollution case study again, the current Lombardy regulation requires taking action when the level of nitrogen dioxide (NO_2) exceeds the threshold of $400\mu g/m^3$ for more than three hours. Let $NO_2 < 400$ denote the atomic proposition that states that the level of nitrogen dioxide is lower than $400\mu g/m^3$. A requirement as the previous one could be expressed as:

$$\mathrm{F}_{[0,3hours]} NO_2 < 400 \qquad (1)$$

Temporal operators like F specify properties on the dynamic evolution of the system. In fact, when (1) is violated, the alerting procedure could be triggered to inform the citizens about the danger. However, since it is known that noise and local faults frequently happen, one could also consider alerting the population when the close neighborhood (e.g., within 10 Km) exhibits a similar phenomenon. For this aim, a property like (2) can be monitored.

$$\diamonddiamond_{<10km} NO_2 < 400 \qquad (2)$$

Spatial operators like $\diamonddiamond$ 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 $400\mu g/m^3$. We will see other examples of the logic language in the following sections. For a more detailed description of the logic, we refer the reader to [1]. We now present the Boolean and quantitative interval semantics for STREL.

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

$$\chi(\mathbf{s}, \ell, t, \top) = 1$$

$$\chi(\mathbf{s}, \ell, t, \bot) = -1$$

$$\chi(\mathbf{s}, \ell, t, p \circ c) = \begin{cases} 1, & \text{if } \pi_p(\mathbf{s}(\ell, t)) \circ c \\ -1, & \text{if } \pi_p(\mathbf{s}(\ell, t)) \circ^{-1} c \\ 0, & \text{otherwise}^4 \end{cases}$$

$$\chi(\mathbf{s}, \ell, t, \neg\varphi) = -\chi(\mathbf{s}, \ell, t, \varphi)$$

$$\chi(\mathbf{s}, \ell, t, \varphi_1 \vee \varphi_2) = \max(\chi(\mathbf{s}, \ell, t, \varphi_1), \chi(\mathbf{s}, \ell, t, \varphi_2))$$

$$\chi(\mathbf{s}, \ell, t, \varphi_1 \mathrm{U}_I \varphi_2) = \max_{t' \in t+I} \{\min(\chi(\mathbf{s}, \ell, t', \varphi_2), \min_{t'' \in [t', t]} \{\chi(\mathbf{s}, \ell, t'', \varphi_1)\})\}$$

$$\chi(\mathbf{s}, \ell, t, \varphi_1 \mathcal{R}_{\leq d} \varphi_2) = \max_{\tau \in \Lambda(\ell)} \max_{\ell' \in \tau : d_\tau[\ell'] \leq d} \{\min(\chi(\mathbf{s}, \ell', t, \varphi_2), \min_{\ell'' < \tau(\ell')} \{\chi(\mathbf{s}, \ell'', t, \varphi_1)\}\}$$

$$\chi(\mathbf{s}, \ell, t, \mathcal{E}_{\geq d} \varphi) = \max_{\tau \in \Lambda(\ell)} \max_{\ell' \in \tau : d_{\mathcal{S}}[\ell, \ell'] \geq d} \min_{\ell'' < \tau(\ell')} \{\chi(\mathbf{s}, \ell'', t, \varphi)\}$$

This is a three-valued semantics, which is equal to 1 if the interval signal $\chi(\mathbf{s}, \ell, t, \varphi)$ satisfies φ, -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 $\chi(\mathbf{s}, \ell, t, p \circ c) = 1$ iff the inequality $\pi_p(\mathbf{s}(\ell, t)) \circ c$ is true. This means, e.g., $\chi(\mathbf{s}, \ell, t, NO_2 > 400) = 1$ iff $\underline{NO_2}$, the left extreme of the projected signal NO_2, is greater than $\overline{400}$. $\chi(\mathbf{s}, \ell, t, NO_2 > 400) = -1$ if the right extreme is less than 400, and $\chi(\mathbf{s}, \ell, t, NO_2 > 400) = 0$ otherwise, so if $400 \in NO_2$ interval value. A similar calculation can be done for the other combination of $\circ$ and c.

The three-valued Boolean semantics can be sufficient in applications where the interest is only whether or not the specification

[4] Note that '>' and '<'are used in this context to represent interval inequalities, which do not define a total ordering.

is satisfied. However, in many complex cases, one might be interested in getting some insights into the degree to 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 $\rho : \mathcal{F}(\mathbb{L}, \mathbb{T}, D^n) \times \mathbb{L} \times \mathbb{T} \times \mathcal{L} \to \mathcal{I}(\mathbb{R})$ be the function mapping signals, locations, time instants, and formulae defined as follows:

$$\rho(\mathbf{s}, \ell, t, \top) = [+\infty, +\infty]$$

$$\rho(\mathbf{s}, \ell, t, \bot) = [-\infty, -\infty]$$

$$\rho(\mathbf{s}, \ell, t, p \circ c) = \begin{cases} \pi_p(\mathbf{s}(\ell, t)) - c, & \text{if } \circ \text{ is ` > '} \\ c - \pi_p(\mathbf{s}(\ell, t)), & \text{if } \circ \text{ is ` < '} \end{cases}$$

$$\rho(\mathbf{s}, \ell, t, \neg\varphi) = -\rho(\mathbf{s}, \ell, t, \varphi)$$

$$\rho(\mathbf{s}, \ell, t, \varphi_1 \vee \varphi_2) = [\max](\rho(\mathbf{s}, \ell, t, \varphi_1), \rho(\mathbf{s}, \ell, t, \varphi_2))$$

$$\rho(\mathbf{s}, \ell, t, \varphi_1 \mathrm{U}_I \varphi_2) = \left[\max_{t' \in t+I}\right] \left\{ [\min] \left(\rho(\mathbf{s}, \ell, t', \varphi_2), \left[\min_{t'' \in [t', t]}\right] \{\rho(\mathbf{s}, \ell, t'', \varphi_1)\} \right) \right\}$$

$$\rho(\mathbf{s}, \ell, t, \varphi_1 \mathcal{R}_{\leq d} \varphi_2) = \left[\max_{\tau \in \Lambda(\ell)}\right] \left[\max_{\ell' \in \tau : d_\tau[\ell'] \leq d}\right] \left\{ [\min] (\rho(\mathbf{s}, \ell', t, \varphi_2), \left[\min_{\ell'' < \tau(\ell')}\right] \{\rho(\mathbf{s}, \ell'', t, \varphi_1)\} \right\}$$

$$\rho(\mathbf{s}, \ell, t, \mathcal{E}_{\geq d} \varphi) = \left[\max_{\tau \in \Lambda(\ell)}\right] \left[\max_{\ell' \in \tau : d_S[\ell, \ell'] \geq d}\right] \left[\min_{\ell'' < \tau(\ell')}\right] \{\rho(\mathbf{s}, \ell'', t, \varphi)\}$$

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

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 get upper/lower bounds at those points, which can 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 $\mathbf{s} \in \mathcal{F}(\mathbb{L}, \mathbb{T}, D^n)$, $\ell \in \mathbb{L}$, $t \in \mathbb{T}$, $\varphi \in \mathcal{L}$:

$$\begin{array}{llll} \text{if} & \rho(\mathbf{s}, \ell, t, \varphi) > 0 & \text{then} & \chi(\mathbf{s}, \ell, t, \varphi) = 1 \\ \text{if} & \rho(\mathbf{s}, \ell, t, \varphi) < 0 & \text{then} & \chi(\mathbf{s}, \ell, t, \varphi) = -1 \\ \text{if} & 0 \in \rho(\mathbf{s}, \ell, t, \varphi) & \text{then} & \chi(\mathbf{s}, \ell, t, \varphi) = 0 \end{array}$$

Proof Sketch. The theorem can be proved by induction on the subformulae of the formula φ, starting from $\varphi \equiv \top$, which is immediate from semantics, since $\rho(\mathbf{s}, \ell, t, \top) > 0$.

See the extended version of this article [33] for the complete proof. □

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

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

$$\text{if} \quad ||\mathbf{s_1} - \mathbf{s_2}||_\infty < \delta \quad \text{then} \quad ||\rho^\varphi_{s_1} - \rho^\varphi_{s_2}||_\infty < \delta$$

Proof. See the extended version of this article [33] for the proof. □

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 $\mathbf{s} \in \mathcal{F}(\mathbb{L}, \mathbb{T}, D^n)$, $\ell \in \mathbb{L}$, $t \in \mathbb{T}$, $\varphi \in \mathcal{L}$:

$$\text{if } ||\mathbf{s_1} - \mathbf{s_2}||_\infty < |\rho(\mathbf{s_1}, \ell, t, \varphi)| \text{ then } \chi(\mathbf{s_1}, \ell, t, \varphi) = \chi(\mathbf{s_2}, \ell, t, \varphi)$$

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

Proof Sketch. The proof essentially consists in two cases depending on the value of $|\rho(\mathbf{s_1}, \ell, t, \varphi)|$. If $\rho(\mathbf{s_1}, \ell, t, \varphi) > 0$ (the case for < 0 is symmetric), then by Theorem 3.1 $\chi(\mathbf{s_1}, \ell, t, \varphi) = 0$. From Lemma 3.1, we have that $d_H(\rho(\mathbf{s_1}, \ell, t, \varphi),$ $\rho(\mathbf{s_2}, \ell, t, \varphi)) < \overline{\rho}(\mathbf{s_1}, \ell, t, \varphi)$. But this also implies that $\underline{\rho}(\mathbf{s_1}, t, \varphi) - \underline{\rho}(\mathbf{s_2}, \ell, t, \varphi) < \underline{\rho}(\mathbf{s_1}, \ell, t, \varphi)$, and therefore that $\underline{\rho}(\mathbf{s_2}, \ell, t, \varphi) > 0$. If, instead, $0 \in \rho(\mathbf{s_1}, \ell, t, \varphi)$: we have that $||\mathbf{s_1} - \mathbf{s_2}||_\infty < \min(\overline{\rho}(\mathbf{s_1}, \ell, t, \varphi), -\underline{\rho}(\mathbf{s_1}, \ell, t, \varphi))$ and that $\rho(\mathbf{s_1}, \ell, t, \varphi)) = 0$. Whichever of the two is the minimum, it can be shown that $0 \in \rho(\mathbf{s_2}, \ell, t, \varphi)$, therefore we must have $\chi(\mathbf{s_2}, \ell, t, \varphi) = 0$.

See the extended version of this article [33] for the complete proof. □

4 ONLINE MONITORING

In this section, a novel *online (out-of-order) monitoring algorithm* for STREL is presented. Unlike 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 reasons, 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 several 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* $\sigma : \mathbb{T} \to \mathcal{I}(\mathbb{R}^\infty)$, can be characterized in the following way:

$$\sigma(t) = \begin{cases} I_i, & \text{for } t_i \le t < t_{i+1} \\ \vdots \\ I_n, & \text{for } t_n \le t < \infty \end{cases}$$

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 intervals can be considered for missed values.

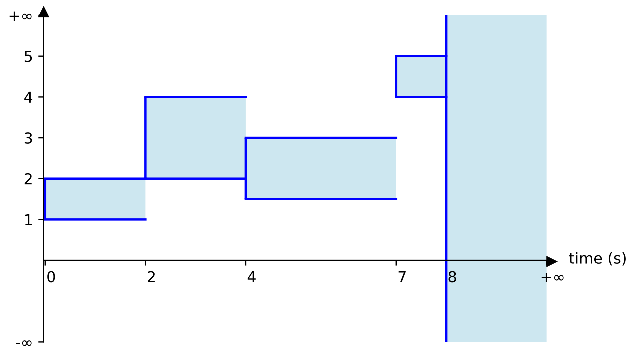

Figure 2: A graphical view of a typical piecewise-constant imprecise signal. From the 8-th second onward, there is total absence of knowledge about what values the signal could have, but until then, we have some bounds on the actual values observed.

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 $s : \mathbb{L} \times \mathbb{T} \to D^n$ is a signal that can be represented as a sequence of pairs $\{(t_i, \mathbf{V}_i)\}_{i \in \mathbb{N}}$, where each pair $(t_i, \mathbf{V}_i)$ 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 $\mathbf{V}_i$ that represents the values of the n dimensions of the signal at each location ℓ 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* $\mathbf{u}$ the triplet $(t_a, t_b, \mathbf{V})$, representing a mapping to the value matrix $\mathbf{V}$ 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, i.e., where updates provide wrong information, will be explored in future work), and, for that reason, we can always assume updates to be well-formed, meaning that the interval $\mathbf{V}$ 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 $\mathbf{s}_1, \mathbf{s}_2 \in \mathcal{F}(\mathbb{L}, \mathbb{T}, D^n)$, we say that $\mathbf{s}_1$ *is refined by* $\mathbf{s}_2$, and we write $\mathbf{s}_1 \succ \mathbf{s}_2$, iff for any $\ell \in \mathbb{L}, t \in \mathbb{T}, i \le n$, $\pi_i(\mathbf{s}_2(\ell, t)) \subseteq \pi_i(\mathbf{s}_1(\ell, t))$, and there is some $\ell' \in \mathbb{L}, t' \in \mathbb{T}, i' \le n$, such that $\pi_{i'}(\mathbf{s}_2(\ell', t')) \subset \pi_{i'}(\mathbf{s}_1(\ell', t'))$, i.e. each interval of the co-domain of the signal $\mathbf{s}_2$ is contained in the corresponding interval of the signal $\mathbf{s}_1$, and some of them are strictly contained.

The *refinement* relation expresses the fact that $\mathbf{s}_1$ and $\mathbf{s}_2$ represent the same information, except that $\mathbf{s}_2$ has a lower 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 $\mathbf{s}_0 \succ \cdots \succ \mathbf{s}_j \succ \ldots$, where the signal $\mathbf{s}_{j+1}$ at the step $j+1$ can be computed from $\mathbf{s}_j$ and update $\mathbf{u}_j$ like in Algorithm 1.

Algorithm 1 Signal Refinement

```
1:  procedure REFINE($\mathbf{s} : \{(t_0, \mathbf{V}_0), \ldots, (t_N, \mathbf{V}_N)\}$, $\mathbf{u} : (t_a, t_b, \mathbf{V})$)
2:      for $(t_i, \mathbf{V}_i)$ in $\mathbf{s}$ do
3:          if $t_i < t_a < t_{i+1}$ and $\mathbf{V} \subset \mathbf{V}_i$ then
4:              $\mathbf{s} := \mathbf{s} \cup (t_a, \mathbf{V})$
5:          else if $t_i = t_a$ and $\mathbf{V} \subset \mathbf{V}_i$ then
6:              $\mathbf{s} := \mathbf{s} \setminus (t_i, \mathbf{V}_i) \cup (t_i, \mathbf{V})$
7:          else if $t_a < t_i < t_b$ then
8:              $\mathbf{s} := \mathbf{s} \setminus (t_i, \mathbf{V}_i)$
9:          end if
10:         if $t_i < t_b < t_{i+1}$ then
11:             $\mathbf{s} := \mathbf{s} \cup (t_b, \mathbf{V}_i)$
12:         end if
13:     end for
14: end procedure
```

The REFINE( ) procedure takes a signal $\mathbf{s}_j$ as a sequence of ordered pairs and an update as the triplet $(t_a, t_b, \mathbf{V})$. 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 $\mathbf{V}$ 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 $t_i > t_b$. At the end of the execution, the updated signal is the next element of the refinement chain, i.e., $\mathbf{s}_{j+1}$.

The Monitoring Problem. When monitoring online a given specification φ, let $\mathbf{s}_0$ be the signal representing the starting information on which the atoms of the formula φ are defined. Let also $(\mathbf{u}_j)_{j \in \mathbb{N}}$ denote a (finite or infinite) sequence of signal updates. The online monitoring problem can be framed as computing the robustness signal $\rho^{\varphi}_{\mathbf{s}_{j+1}}$, given $\rho^{\varphi}_{\mathbf{s}_j}$ (or, alternatively, the satisfaction signal $\chi^{\varphi}_{\mathbf{s}_{j+1}}$ given $\chi^{\varphi}_{\mathbf{s}_j}$), with $\mathbf{s}_j \succ \mathbf{s}_{j+1}$, starting from $\mathbf{s}_0$. A naïve implementation of an online monitor could simply ignore the information coming from previous monitoring steps and restart the computation over

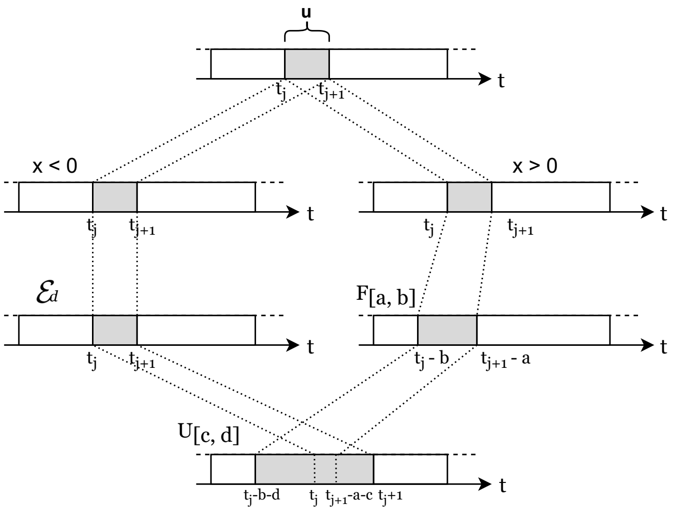

Figure 3: The ripple effect generated from the propagation of an update of the signal x, over the formula $\varphi := (\mathcal{E}_d x < 0)\ \mathrm{U}_{[c,d]}(\mathrm{F}_{[a,b]} x > 0)$.

the whole signal each time new information is available. As already noted in [7, 8], such an implementation would result in vast amounts of wasted resources when monitoring time signals, and it would 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 propagating the update's 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 $\mathbf{OP}_I$ denote any temporal operator on the time interval I, given a signal update $\mathbf{u} \equiv (t_a, t_b, \mathbf{V})$, we call *update ripple* the function

$$\mathrm{ur}(\mathbf{u}, \varphi) = \begin{cases} [t_a, t_b) - I & \varphi \equiv \mathbf{OP}_I\psi \text{ or } \varphi \equiv \psi_1 \mathbf{OP}_I \psi_2 \\ [t_a, t_b) & \text{otherwise} \end{cases}$$

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 $j+1$, the monitoring function can be evaluated as:

$$\rho_{j+1}(\mathbf{s}_{j+1}, \ell, t, \varphi) = \begin{cases} \textsc{monitor}(\mathbf{u}_j, \varphi)[\ell, t] & t \in \mathrm{ur}(\mathbf{u}_j, \varphi) \\ \rho_j(\mathbf{s}_j, \ell, t, \varphi) & otherwise \end{cases}$$

Note that in all the cases where the updates overlap, they must be processed sequentially in order to generate correct results.

Monitoring Procedure. To compute the monitoring result signal online, it is crucial to be able to exploit the knowledge from the past each time new information is available. The most natural way for doing so is to develop a stateful algorithm that stores the relevant information from previous computations. We will represent by $\mathcal{M}$ the persistent memory (i.e., the state) that we keep throughout the various iterations of the monitoring process and by $\mathcal{M}[x]$ the access to the item x from memory. The memory $\mathcal{M}$ is organized around a data structure representing the set of robustness signals of all the subformulae of the monitored formula φ, as computed in the last iterations. This set can be encoded as an array indexed on some ordering of the subformulae. We represent as $\mathcal{M}[\rho^{\psi}]$ the access to the respective robustness signal for some formula ψ. This data structure is essential to maximize the time performance of the monitoring process, as the subsequent iterations will re-compute only the differing fragments based on the update ripple. Before starting the monitoring process, the memory is initialized by storing an *undefined signal* for any subformula ψ in the set of the subformulae of the formula φ being monitored. We call *undefined signal* the special signal $s : \mathbb{L} \times \mathbb{T} \to [-\infty, +\infty]$, which represents the total absence of knowledge about the value at any possible time instant.

Once the memory is initialized, the monitoring can start. We assume that the signal is always received as a sequence of signal updates $\mathbf{u}_j$, starting from $j = 0$, where the input signal is considered to be undefined. Algorithm 2 represents the base routine triggered when receiving an update $\mathbf{u}$ of the input signal. The recursive procedure $\textsc{monitor}(\mathcal{M}, \varphi, \mathbf{u})$ is responsible for propagating the input signal update to the subformulae and then fetching the corresponding updates of the robustness signal. We indicate by $\langle \mathcal{M}, \{u^{\varphi}\} \rangle$ the return value of the algorithm, to mean that it returns an updated version of the memory, and a list of robustness updates of the formula φ that might either be used by the caller or discarded. The general procedure of Algorithm 2 calls the specific procedures of Algorithms 3-6 depending on the operators encountered while traversing the tree of the formula. Note that all of the above exploit the REFINE( ) primitive operation from Algorithm 1.

Algorithm 2 Online Monitoring Procedure

```
1:  procedure MONITOR($\mathcal{M}, \varphi, \mathbf{u}$)
2:      switch $\varphi$ do
3:          case $p \circ c$
4:              $\langle \mathcal{M}, \{u^{\varphi}\} \rangle := \textsc{atom}(\mathcal{M}, \varphi, \mathbf{u})$
5:          case $\neg\psi$ or $\mathcal{E}_{\geq d}\psi$
6:              $\langle \mathcal{M}, \{u^{\varphi}\} \rangle := \textsc{unary}(\mathcal{M}, \varphi, \mathbf{u})$
7:          case $\psi_1 \vee \psi_2$ or $\psi_1 \mathcal{R}_{\leq d} \psi_2$
8:              $\langle \mathcal{M}, \{u^{\varphi}\} \rangle := \textsc{binary}(\mathcal{M}, \varphi, \mathbf{u})$
9:          case $\mathrm{F}_I \psi$
10:             $\langle \mathcal{M}, \{u^{\varphi}\} \rangle := \textsc{slidingWindow}(\mathcal{M}, \varphi, \mathbf{u})$
11:         case $\psi_1 \mathrm{U} \psi_2$
12:             $\langle \mathcal{M}, \{u^{\varphi}\} \rangle := \textsc{unboundedUntil}(\mathcal{M}, \varphi, \mathbf{u})$
13:     return $\langle \mathcal{M}, \{u^{\varphi}\} \rangle$
14: end procedure
```

Online Monitoring Of Non-temporal Operators When monitoring formulae contain Boolean or Spatial operators, the online evaluation can be performed very efficiently by simply updating the robustness signal at 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 $\mathcal{E}_d$), and Algorithm 5 shows the one for binary operators (i.e., $\vee$ and $\mathcal{R}_d$). We represent by COMPUTE_OP($\mathbf{OP}, \mathbf{V}$) (and COMPUTE_OP($\mathbf{OP}, \mathbf{V}_1, \mathbf{V}_2$)) the execution of the semantic operation corresponding to the operator **OP**, along the lines of Definitions 3.2,3.3, i.e., [max]/[min] 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 COMPUTE_OP 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 COMPUTE_OP is executed in parallel for any location of the spatial model.

Algorithm 3 Atomic Formula Monitoring

```
function ATOM($\mathcal{M}, p \circ c, \mathbf{u}$)
    $(t_a, t_b, \mathbf{V}) := \mathbf{u}$
    $\mathbf{u}^{p \circ c} := (t_a, t_b, \text{COMPUTE\_OP}(p \circ c, \mathbf{V}))$
    REFINE($\mathcal{M}[\rho^{p \circ c}], \{\mathbf{u}^{p \circ c}\}$)
    return $\langle \mathcal{M}, \{\mathbf{u}^{p \circ c}\} \rangle$
end function
```

Algorithm 4 Unary Operator Monitoring

```
function UNARY($\mathcal{M}, \mathbf{OP}\psi, \mathbf{u}$)
    $\langle \mathcal{M}, \{\mathbf{u}^{\psi}\} \rangle$ := MONITOR($\mathcal{M}, \psi, \mathbf{u}$)
    $\{\mathbf{u}^{\mathbf{OP}\psi}\} := \emptyset$
    for $(t_a, t_b, \mathbf{V}) \in \{\mathbf{u}^{\psi}\}$ do
        $\{\mathbf{u}^{\mathbf{OP}\psi}\} := \{\mathbf{u}^{\mathbf{OP}\psi}\} \cup (t_a, t_b, \text{COMPUTE\_OP}(\mathbf{OP}, \mathbf{V}))$
    end for
    REFINE($\mathcal{M}[\rho^{\mathbf{OP}\psi}], \{\mathbf{u}^{\mathbf{OP}\psi}\}$)
    return $\langle \mathcal{M}, \{\mathbf{u}^{\mathbf{OP}\psi}\} \rangle$
end function
```

The algorithm for monitoring binary operators is slightly more complex, as it requires taking into account the corresponding value of the other subformula when an update is processed. In this context, we indicate by SELECT($\mathbf{s}, t_1, t_2$) the restriction of the signal $\mathbf{s}$ to the time interval that starts at t_1 and ends at t_2 (excluded).

Algorithm 5 Binary Operator Monitoring

```
function BINARY($\mathcal{M}, \psi_1 \mathbf{OP} \psi_2, \mathbf{u}$)
    $\{\mathbf{u}^{\psi_1 \mathbf{OP} \psi_2}\} := \emptyset$
    $\langle \mathcal{M}, \{\mathbf{u}^{\psi_1}\} \rangle$ := MONITOR($\mathcal{M}, \psi_1, \mathbf{u}$)
    for $(t_a, t_b, \mathbf{V}^{\psi_1}) \in \{\mathbf{u}^{\psi_1}\}$ do
        $\mathbf{V}^{\psi_2}$ := SELECT($\mathcal{M}[\rho^{\psi_2}], t_a, t_b$)
        $\{\mathbf{u}^{\psi_1 \mathbf{OP} \psi_2}\} := \{\mathbf{u}^{\psi_1 \mathbf{OP} \psi_2}\} \cup$
                    $(t_a, t_b, \text{COMPUTE\_OP}(\mathbf{OP}, \mathbf{V}^{\psi_1}, \mathbf{V}^{\psi_2}))$
    end for
    Repeat lines 3-8 symmetrically for $\psi_2$...
    REFINE($\mathcal{M}[\rho^{\psi_1 \mathbf{OP} \psi_2}], \{\mathbf{u}^{\psi_1 \mathbf{OP} \psi_2}\}$)
    return $\langle \mathcal{M}, \{\mathbf{u}^{\psi_1 \mathbf{OP} \psi_2}\} \rangle$
end function
```

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

- the bounded eventually F_I (or equivalently the bounded globally G_I)
- 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 the primary routine of the sliding window for computing updates of bounded unary temporal operators $\mathbf{OP}_I$. 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 behavior of SLIDE and ADD that control the mutation of `W` can be examined in the extended version of the paper [33] or in the Moonlight implementation.

Algorithm 6 Sliding Window

```
1:  procedure SLIDINGWINDOW($\mathcal{M}, \mathbf{OP}_{[t_a,t_b]}\psi, \mathbf{u}$)
2:      $\langle \mathcal{M}, \{\mathbf{u}^{\psi}\} \rangle$ := MONITOR($\mathcal{M}, \psi, \mathbf{u}$)
3:      $\{\mathbf{u}^{\mathbf{OP}\psi}\} := \emptyset$
4:      for $\mathbf{u}^{\psi} \in \{\mathbf{u}^{\psi}\}$ do
5:          $(t_s, t_e)$ := ur($\mathbf{u}^{\psi}, \mathbf{OP}_{[t_a,t_b]}\psi$)
6:          for $(t, \mathbf{V}) \in$ SELECT($\mathcal{M}[\rho^{\psi}], t_s, t_e$) do
7:              if $t > t_b +$ W.first.start then
8:                  $\{\mathbf{u}^{\mathbf{OP}\psi}\} := \{\mathbf{u}^{\mathbf{OP}\psi}\} \cup$ SLIDE($t - t_b$)
9:              end if
10:             ADD($t - t_a, \mathbf{V}$)
11:         end for
12:     end for
13:     REFINE($\mathcal{M}[\rho^{\mathbf{OP}\psi}], \{\mathbf{u}^{\mathbf{OP}\psi}\}$)
14:     return $\langle \mathcal{M}, \{\mathbf{u}^{\mathbf{OP}\psi}\} \rangle$
15: 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 keeping the minimum value of preceding computations of φ_1 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 also been implemented 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 here three different examples: (i) we present and discuss the results of the properties previously introduced in the context of air quality monitoring; (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. All the computations have been executed on an Intel® Core™ i7-5820K CPU @ 3.30GHz, 15M cache, 6 cores (12 threads), with 32GB RAM, running Ubuntu® 20.04.2 LTS, and Matlab™ R2021a.

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 into 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 neighborhood: in fact, by combining the observations registered from close locations, 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® model that describes a black-box representation of an engine's air-fuel ratio controller aimed at complying with 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_{ref}) at a sampling period $T = 0.1s$. For a complete 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 formula is also a STREL formula)

$$\varphi = \text{G}_{[10,30]}(|AF - AF_{ref}| > 0.1 \rightarrow (\text{F}_{[0,1]}|AF - AF_{ref}| < 0.1))$$

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 $|AF - AF_{ref}|$ 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 (removing the execution time of the 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	Moonlight Exec. Time	
	In-order	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 φ 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 a 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.

$$h(t) = c_0 + \varepsilon(t), \qquad (\varepsilon \sim WN(0, \lambda^2))$$

where the observed value comes from the real value c_0, with some perturbation from the zero-mean white noise ε of variance λ^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 bridge the communication. Let X_H denote the actual value of humidity for a given device at a given time, let X_S denote the role of a given sensor, and T some

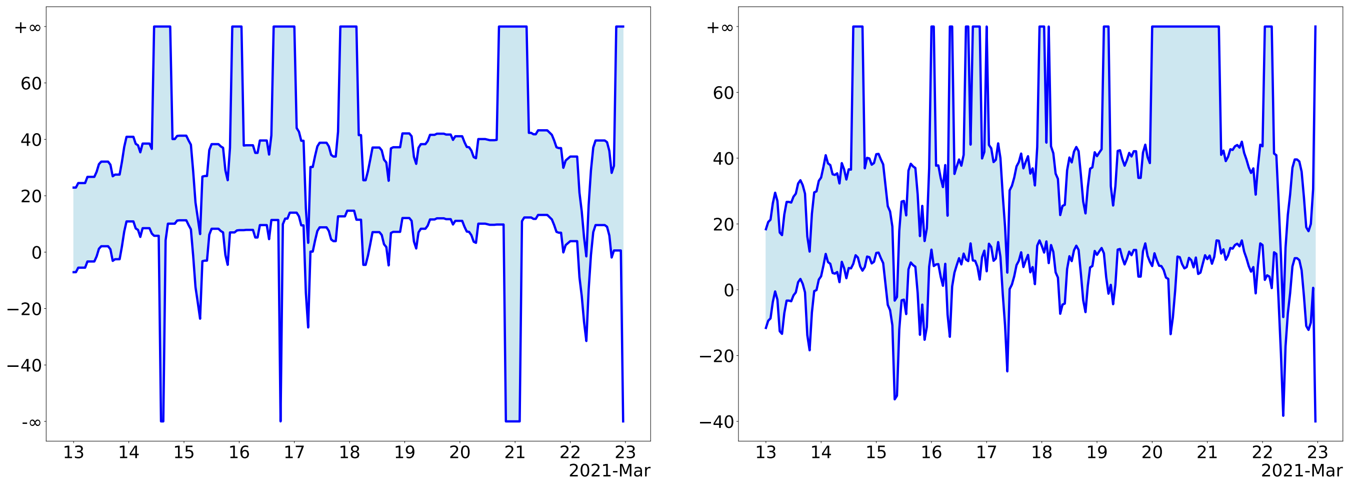

Figure 4: The results of monitoring robustness for Property 1 (on the left), and Property 2 (on the right) at the Rezzato station. Despite the missing values, reliable values of the robustness of the property can be provided.

time threshold to warn the observers. We monitored the following properties on the system:

$$\varphi_1 = (X_H > 60) \rightarrow \mathrm{F}_{[0,T]}(X_H < 30)$$

$$\varphi_2 = \boxdot \Diamond\!\!\!\Diamond_{d<10}(X_S = \texttt{coordinator})$$

Property φ_1 denotes an alert condition: if the humidity measured by a device X_H goes beyond 60%, then it must fall down at 30% afterward, within the time threshold T. Property φ_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 ($X_S = \texttt{coordinator}$) in less than 10 hops. Similar 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 φ_1 and φ_2 both online and offline. It is interesting to see how the different algorithms behave on the 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 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). In contrast, the benefit practically vanishes (actually resulting in more overhead) as the number of parallel threads grows significantly more than the cores available.

Time samples	N. nodes	Offline		Online		Online(Parallel)	
		φ_1	φ_2	φ_1	φ_2	φ_1	φ_2
100	10	9	77	116	29	49	58
	50	8	1028	151	430	84	583
	100	15	6919	197	2993	137	3017
500	10	8	200	621	45	461	760
	50	17	4058	1901	1783	1549	2009
	100	25	32561	3333	15641	2889	15486

Table 2: Execution times registered for monitoring φ_1 and φ_2 with different versions of Moonlight. Times in ms, averaged over 100 runs.

6 CONCLUSIONS & FUTURE WORK

We extended the traditional definition of signals also to 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 for 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 computational advantages and disadvantages of that hypothesis and to which extent it can be relaxed. Another intriguing topic for future development concerns spatial models representing (and interacting as) distributed systems. In that context, multiple directions could be pursued, like considering an ownership model for the atomic formulae, or by reasoning on an actor-based communication model among locations. Another interesting idea could be to expand the kind of failures we can monitor. For example, we could consider some form of error correction in case some received updates later prove to have provided wrong information (maybe because of some broken sensors). Lastly, different forms of computational optimization could be explored, like stopping when some bounds on the satisfiability/robustness have been reached, and 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 Science Fund (FWF): 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 Ničković (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 Šafránek (Eds.). Springer International Publishing, Cham, 156–172.

[4] Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donzé, Georgios Fainekos, Oded Maler, Dejan Ničković, 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/10.1007/978-3-319-75632-5_5

[5] Ezio Bartocci, Ebru Aydin Gol, Iman Haghighi, and Calin Belta. 2018. A Formal Methods Approach to Pattern Recognition and Synthesis in Reaction Diffusion Networks. *IEEE Trans. Control. Netw. Syst.* 5, 1 (2018), 308–320. https://doi.org/10.1109/TCNS.2016.2609138

[6] L. Bortolussi, J. Hillston, D. Latella, and M. Massink. 2013. Continuous Approximation of Collective Systems Behaviour: a Tutorial. *Performance Evaluation* 70, 5 (May 2013), 317–349. https://doi.org/10.1016/j.peva.2013.01.001

[7] Jyotirmoy V. Deshmukh, Alexandre Donzé, Shromona Ghosh, Xiaoqing Jin, Garvit Juniwal, and Sanjit A. Seshia. 2017. Robust online monitoring of signal temporal logic. *Formal Methods in System Design* 51, 1 (01 Aug 2017), 5–30. https://doi.org/10.1007/s10703-017-0286-7

[8] Adel Dokhanchi, Bardh Hoxha, and Georgios Fainekos. 2014. On-Line Monitoring for Temporal Logic Robustness. In *Runtime Verification*, Borzoo Bonakdarpour and Scott A. Smolka (Eds.). Springer International Publishing, Cham, 231–246.

[9] Alexandre Donzé. 2010. Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems. In *Computer Aided Verification*, Tayssir Touili, Byron Cook, and Paul Jackson (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 167–170.

[10] Alexandre Donzé, Thomas Ferrère, and Oded Maler. 2013. Efficient Robust Monitoring for STL. In *Computer Aided Verification*, Natasha Sharygina and Helmut Veith (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 264–279. https://doi.org/10.1007/978-3-642-39799-8_19

[11] Radu Grosu, Scott A. Smolka, Flavio Corradini, Anita Wasilewska, Emilia Entcheva, and Ezio Bartocci. 2009. Learning and detecting emergent behavior in networks of cardiac myocytes. *Commun. ACM* 52, 3 (2009), 97–105. https://doi.org/10.1145/1467247.1467271

[12] Iman Haghighi, Austin Jones, Zhaodan Kong, Ezio Bartocci, Radu Grosu, and Calin Belta. 2015. SpaTeL: a novel spatial-temporal logic and its applications to networked systems. In *Proc. of HSCC'15: the 18th International Conference on Hybrid Systems: Computation and Control*. IEEE, Seattle, WA, USA, 189–198. https://doi.org/10.1145/2728606.2728633

[13] Timothy Hayes, Oscar Palomar, Osman Unsal, Adrian Cristal, and Mateo Valero. 2016. Future Vector Microprocessor Extensions for Data Aggregations. *SIGARCH Comput. Archit. News* 44, 3 (June 2016), 418–430. https://doi.org/10.1145/3007787.3001182

[14] Stefan Jaksic, Ezio Bartocci, Radu Grosu, Reinhard Kloibhofer, Thang Nguyen, and Dejan Nickovic. 2015. From signal temporal logic to FPGA monitors. In *Proc. of MEMOCODE 2015: the 13th ACM/IEEE International Conference on Formal Methods and Models for Codesign*. IEEE, New York City at 3 Park Ave, 218–227. https://doi.org/10.1109/MEMCOD.2015.7340489

[15] Stefan Jaksic, Ezio Bartocci, Radu Grosu, and Dejan Nickovic. 2018. An Algebraic Framework for Runtime Verification. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 37, 11 (2018), 2233–2243. https://doi.org/10.1109/TCAD.2018.2858460

[16] Xiaoqing Jin, Jyotirmoy V. Deshmukh, James Kapinski, Koichi Ueda, and Ken Butts. 2014. Powertrain Control Verification Benchmark. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control* (Berlin, Germany) *(HSCC '14)*. Association for Computing Machinery, New York, NY, USA, 253–262. https://doi.org/10.1145/2562059.2562140

[17] Ron Koymans. 1990. Specifying Real-Time Properties with Metric Temporal Logic. *Real-Time Systems* 2, 4 (1990), 255–299. https://doi.org/10.1007/BF01995674

[18] Daniel Kusswurm. 2020. *Armv8-32 SIMD Architecture*. Apress, Berkeley, CA, 131–140. https://doi.org/10.1007/978-1-4842-6267-2_7

[19] D. Lemire. 2006. Streaming Maximum-Minimum Filter Using No More than Three Comparisons per Element. *Nord. J. Comput.* 13 (2006), 328–339.

[20] ARPA Lombardia. 2021. Dati sensori aria. https://www.dati.lombardia.it/Ambiente/Dati-sensori-aria/nicp-bhqi.

[21] ARPA Lombardia. 2021. Stazioni qualità dell'aria. https://www.dati.lombardia.it/Ambiente/Stazioni-qualit-dell-aria/ib47-atvt.

[22] Michele Loreti and Jane Hillston. 2016. *Modelling and Analysis of Collective Adaptive Systems with CARMA and its Tools*. Springer International Publishing, Cham, 83–119. https://doi.org/10.1007/978-3-319-34096-8_4

[23] Meiyi Ma, Ezio Bartocci, Eli Lifland, John A. Stankovic, and Lu Feng. 2020. SaSTL: Spatial Aggregation Signal Temporal Logic for Runtime Monitoring in Smart Cities. In *Proc. of ICCPS 2020: the 11th ACM/IEEE International Conference on Cyber-Physical Systems*. IEEE, Sydney, Australia, 51–62. https://doi.org/10.1109/ICCPS48487.2020.00013

[24] Meiyi Ma, Ezio Bartocci, Eli Lifland, John A. Stankovic, and Lu Feng. 2021. A Novel Spatial-Temporal Specification-Based Monitoring System for Smart Cities. *IEEE Internet of Things Journal* 8, 15 (2021), 11793–11806. https://doi.org/10.1109/JIOT.2021.3069943

[25] Oded Maler and Dejan Nickovic. 2013. Monitoring properties of analog and mixed-signal circuits. *STTT* 15, 3 (2013), 247–268. https://doi.org/10.1007/s10009-012-0247-9

[26] Konstantinos Mamouras, Agnishom Chattopadhyay, and Zhifu Wang. 2021. Algebraic Quantitative Semantics for Efficient Online Temporal Monitoring. In *Tools and Algorithms for the Construction and Analysis of Systems*, Jan Friso Groote and Kim Guldstrand Larsen (Eds.). Springer International Publishing, Cham, 330–348.

[27] Konstantinos Mamouras and Zhifu Wang. 2020. Online Signal Monitoring With Bounded Lag. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 11 (2020), 3868–3880. https://doi.org/10.1109/TCAD.2020.3013053

[28] Laura Nenzi, Ezio Bartocci, Luca Bortolussi, Michele Loreti, and Ennio Visconti. 2020. Monitoring Spatio-Temporal Properties (Invited Tutorial). In *Runtime Verification*, Jyotirmoy Deshmukh and Dejan Ničković (Eds.). Springer International Publishing, Cham, 21–46.

[29] Laura Nenzi, Luca Bortolussi, Vincenzo Ciancia, Michele Loreti, and Mieke Massink. 2015. Qualitative and Quantitative Monitoring of Spatio-Temporal Properties. In *Proc. of RV 2015: the 6th International Conference on Runtime Verification*, Vol. 9333. Springer, Vienna, Austria, 21–37. https://doi.org/10.1007/978-3-319-23820-3_2

[30] Dejan Ničković and Tomoya Yamaguchi. 2020. RTAMT: Online Robustness Monitors from STL. In *Automated Technology for Verification and Analysis*, Dang Van Hung and Oleg Sokolsky (Eds.). Springer International Publishing, Cham, 564–571.

[31] Denise Ratasich, Faiq Khalid, Florian Geissler, Radu Grosu, Muhammad Shafique, and Ezio Bartocci. 2019. A Roadmap Toward the Resilient Internet of Things for Cyber-Physical Systems. *IEEE Access* 7 (2019), 13260–13283. https://doi.org/10.1109/ACCESS.2019.2891969

[32] Konstantin Selyunin, Stefan Jaksic, Thang Nguyen, Christian Reidl, Udo Hafner, Ezio Bartocci, Dejan Nickovic, and Radu Grosu. 2017. Runtime Monitoring with Recovery of the SENT Communication Protocol. In *Computer Aided Verification*, Rupak Majumdar and Viktor Kunčak (Eds.). Springer International Publishing, Cham, 336–355.

[33] Ennio Visconti, Ezio Bartocci, Michele Loreti, and Laura Nenzi. 2021. Online Monitoring of Spatio-Temporal Properties for Imprecise Signals. arXiv:arXiv:2109.08081