Acta Informatica (2018) 55:309–338
https://doi.org/10.1007/s00236-017-0295-4

ORIGINAL ARTICLE

Algorithms for monitoring real-time properties

David Basin[1] · **Felix Klaedtke**[2] · **Eugen Zălinescu**[3]

Received: 25 June 2015 / Accepted: 17 February 2017 / Published online: 3 March 2017

Abstract Real-time logics are popular specification languages for reasoning about systems intended to meet timing constraints. Numerous formalisms have been proposed with different underlying time models that can be characterized along two dimensions: dense versus discrete time and point-based versus interval-based. We present monitoring algorithms for the past-only fragment of metric temporal logics that differ along these two dimensions, analyze their complexity, and compare them on a class of formulas for which the point-based and the interval-based settings coincide. Our comparison reveals similarities and differences between the monitoring algorithms and highlights key concepts underlying our and prior monitoring algorithms. For example, point-based algorithms are conceptually simpler and more efficient than interval-based ones as they are invoked only at time points occurring in the monitored trace and their reasoning is limited to just those time points.

1 Introduction

Real-time logics [2,18] allow us to specify system properties involving timing constraints. Such specifications are useful when designing, developing, and verifying systems with hard

A preliminary version of this article appeared in the proceedings of the 2nd International Conference on Runtime Verification [7].

✉ Eugen Zălinescu
eugen.zalinescu@in.tum.de

David Basin
david.basin@inf.ethz.ch

Felix Klaedtke
felix.klaedtke@neclab.eu

1 Department of Computer Science, Institute of Information Security, ETH Zurich, Zurich, Switzerland

2 NEC Europe Ltd., Heidelberg, Germany

3 Institut für Informatik, Technische Universität München, Munich, Germany

real-time requirements, for example requirements on response time or the frequency of event occurrences, such as every request must be followed within 10 s by a grant. These logics also have applications in runtime verification, where monitors generated from specifications are used to check the correctness of system behaviors at runtime [15]. A monitor could, for example, raise an alarm when it detects incorrect system behavior.

Various monitoring algorithms for real-time logics have been developed [3,5,8,12,17,19,25,26,33] based on different time models. These time models can be characterized by two independent aspects. First, whether the time model is point-based or interval-based. In point-based time models, system traces are sequences of system states, where each state is timestamped. In interval-based time models, system traces consist of continuous (Boolean) signals of state variables. A second distinction is whether the time model is either dense or discrete, which depends on the underlying ordering on time points (i.e. elements of the time domain), that is, whether there are infinitely many or finitely many time points between any two distinct time points.

Real-time logics based on a dense, interval-based time model are more natural and general than their counterparts based on a discrete or point-based model. In fact, both discrete and point-based time models can be seen as abstractions of their dense, interval-based counterparts [2,29]. However, the satisfiability and the model-checking problems for many real-time logics with the more natural time model are computationally harder than the corresponding decision problems when the time model is discrete or point-based. See the survey [27] for further discussion and examples.

In this article, we analyze the impact of different time models on monitoring. We do this by presenting, analyzing, and comparing monitoring algorithms for real-time logics based on different time models. More concretely, we present monitoring algorithms for the past-only fragment of propositional metric temporal logics with a point-based and an interval-based semantics, also considering both dense and discrete time domains. We compare our algorithms on a class of formulas for which the point-based and the interval-based settings coincide. To define this class, we distinguish between event propositions and state propositions. The truth value of a state proposition always has a duration, whereas an event proposition cannot be continuously true between two distinct time points.

Our analysis explains the impact of different time models on monitoring. First, the impact of a dense versus a discrete time domain is minor. The algorithms are essentially the same and have similar computational complexities. Second, monitoring in a point-based setting is conceptually simpler than in an interval-based setting. In the point-based setting, reasoning is limited to just those time points occurring in the monitored trace. In contrast, the interval-based setting is more complex as it must also account for time points that do not occur in the monitored trace. Moreover, we show that our point-based monitoring algorithms perform better than our interval-based algorithms on the given class of formulas on which the two settings coincide. Our findings are, e.g., helpful when choosing or justifying the underlying time model for monitoring a system.

Overall, we see the contributions as follows. First, our monitoring algorithms simplify and clarify key concepts of previously presented algorithms [5,22,25,26]. In particular, we present the algorithms in full detail and we analyze their complexity, giving upper bounds on their time and space usage. Second, our monitoring algorithm for the dense, point-based time model has better complexity bounds than existing algorithms for the same time model [33]. Third, our comparison of the monitoring algorithms illustrates the similarities, differences, and trade-offs between the time models with respect to monitoring. Moreover, formulas in our fragment benefit from both settings: although they describe properties based on a more

natural time model, they can be monitored with respect to a point-based time model, which is more efficient.

The remainder of this article is structured as follows. In Sect. 2 we fix notation and terminology. In Sect. 3 we compare the point-based and the interval-based time model and define a class of formulas on which the two time coincide. In Sects. 4 and 5 we present and analyze our monitoring algorithms for the point-based setting and the interval-based setting, respectively, and we compare them in Sect. 6. In Sect. 7 we discuss related work and we draw conclusions in Sect. 8.

2 Preliminaries

2.1 Time domain and intervals

We denote the *time domain* by $\mathbb{T}$ and assume that it is ordered by $\leq$. If not stated differently, we assume a dense time domain, where $\mathbb{T}$ is $\mathbb{Q}_{\geq 0}$.[1] For the discrete time domain, we assume $\mathbb{T} = \mathbb{N}$.

A (*time*) *interval* is a non-empty set $I \subseteq \mathbb{T}$ such that if $\tau < \kappa < \tau'$ then $\kappa \in I$, for all $\tau, \tau' \in I$ and $\kappa \in \mathbb{T}$. We denote the set of all time intervals by $\mathbb{I}$. An interval is either left-open or left-closed and similarly either right-open or right-closed. We denote the left margin and the right margin of an interval $I \in \mathbb{I}$ by $\ell(I)$ and $r(I)$, respectively. For instance, the interval $I = \{\tau \in \mathbb{T} \mid 3 \leq \tau\}$, which we also write as $[3, \infty)$, is left-closed and right-open with margins $\ell(I) = 3$ and $r(I) = \infty$.

For an interval $I \in \mathbb{I}$, we define the extension $I^{\geq} := I \cup (\ell(I), \infty)$ to the right and its strict counterpart $I^{>} := I^{\geq} \backslash I$, which excludes I. We define ${}^{\leq}I := [0, r(I)) \cup I$ and ${}^{<}I := ({}^{\leq}I) \backslash I$ similarly. Note that ${}^{<}I = \mathbb{T} \backslash (I^{\geq})$ and $I^{>} = \mathbb{T} \backslash {}^{\leq}I$. An interval $I \in \mathbb{I}$ is *singular* if $|I| = 1$, *bounded* if $r(I) < \infty$, and *unbounded* if $r(I) = \infty$. The intervals $I, J \in \mathbb{I}$ are *adjacent* if $I \cap J = \emptyset$ and $I \cup J \in \mathbb{I}$. For $I, J \in \mathbb{I}$, $I \oplus J$ is the set $\{\tau + \tau' \mid \tau \in I \text{ and } \tau' \in J\}$.

An *interval partition* of $\mathbb{T}$ is a sequence $\langle I_i \rangle_{i \in N}$ of time intervals with $N = \mathbb{N}$ or $N = \{0, \ldots, n\}$ for some $n \in \mathbb{N}$ that fulfills the properties:

(i) I_{i-1} and I_i are adjacent and $\ell(I_{i-1}) \leq \ell(I_i)$, for all $i \in N \backslash \{0\}$, and
(ii) for each $\tau \in \mathbb{T}$, there is an $i \in N$ such that $\tau \in I_i$.

The interval partition $\langle J_j \rangle_{j \in M}$ *refines* the interval partition $\langle I_i \rangle_{i \in N}$ if for every $j \in M$, there is some $i \in N$ such that $J_j \subseteq I_i$. We often write $\bar{I}$ for a sequence of intervals instead of $\langle I_i \rangle_{i \in N}$. Moreover, we abuse notation by writing $I \in \langle I_i \rangle_{i \in N}$ if $I = I_i$, for some $i \in N$.

A *time sequence* $\langle \tau_i \rangle_{i \in \mathbb{N}}$ is a sequence of elements $\tau_i \in \mathbb{T}$ that is strictly increasing (i.e., $\tau_i < \tau_j$, for all $i, j \in \mathbb{N}$ with $i < j$) and progressing (i.e., for all $\tau \in \mathbb{T}$, there is $i \in \mathbb{N}$ with $\tau_i > \tau$). Similar to interval sequences, $\bar{\tau}$ abbreviates $\langle \tau_i \rangle_{i \in \mathbb{N}}$.

2.2 Boolean signals

A (*Boolean*) *signal* γ is a subset of $\mathbb{T}$ that fulfills the following *finite-variability* condition: for every bounded interval $I \in \mathbb{I}$, there are intervals $I_0, \ldots, I_{n-1} \in \mathbb{I}$ such that $\gamma \cap I = I_0 \cup \cdots \cup I_{n-1}$, for some $n \in \mathbb{N}$. The least such $n \in \mathbb{N}$ is the *size* of the signal γ on I. We denote it by $\|\gamma \cap I\|$.

[1] We do not use $\mathbb{R}_{\geq 0}$ as dense time domain because of representation issues. Namely, each element in $\mathbb{Q}_{\geq 0}$ can be finitely represented, which is not the case for $\mathbb{R}_{\geq 0}$. Choosing $\mathbb{Q}_{\geq 0}$ instead of $\mathbb{R}_{\geq 0}$ is without loss of generality for the satisfiability of properties specified in real-time logics like metric interval temporal logic [1].

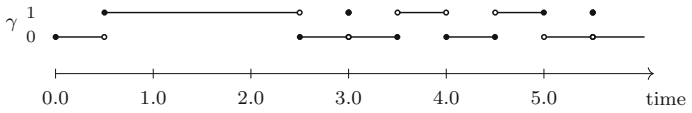

Fig. 1 Example of a signal

We use the term "signal" for such a set γ because its characteristic function $\chi_\gamma : \mathbb{T} \to \{0, 1\}$ represents, for example, the values over time of an input or an output of a sequential circuit. Intuitively, $\tau \in \gamma$ iff the circuit's signal is high at the time $\tau \in \mathbb{T}$. The finite-variability condition imposed on γ prevents switching infinitely often from high to low in finite time. Note that $\|\gamma \cap I\|$ formalizes how often the signal γ is high on the bounded interval I, in particular, $\|\gamma \cap I\| = 0$ iff $\gamma \cap I = \emptyset$.

A signal γ is *stable* on an interval $I \in \mathbb{I}$ if $I \subseteq \gamma$ or $I \cap \gamma = \emptyset$. A signal γ is stable on an interval partition if it is stable on each interval of the partition. The *induced interval partition* $\overline{\mathsf{iip}}(\gamma)$ of a signal γ is the interval partition $\bar{I}$ such that γ is stable on each of the intervals in $\bar{I}$ and, for any other interval partition $\bar{J}$ over which γ is stable, $\bar{J}$ refines $\bar{I}$. We write $\overline{\mathsf{iip}}^1(\gamma)$ for the sequence of intervals I in $\overline{\mathsf{iip}}(\gamma)$ such that $I \cap \gamma \neq \emptyset$. Similarly, we write $\overline{\mathsf{iip}}^0(\gamma)$ for the sequence of intervals I in $\overline{\mathsf{iip}}(\gamma)$ such that $I \cap \gamma = \emptyset$. We assume that the intervals in these sequences are ordered by their margins. Intuitively, $\overline{\mathsf{iip}}^1(\gamma)$ and $\overline{\mathsf{iip}}^0(\gamma)$ are the sequences of maximal intervals on which the signal γ is high and low, respectively. Note that $\|\gamma \cap I\| = |\overline{\mathsf{iip}}^1(\gamma \cap I)|$, for any bounded interval I.

Example 2.1 Figure 1 depicts the signal $\gamma = [0.5, 2.5) \cup \{3.0\} \cup (3.5, 4.0) \cup (4.5, 5.0] \cup \{5.5\}$. The interval margins are marked with dots and circles. A dot signifies that the element belongs to the interval and a circle excludes the element from the interval. Note that an isolated dot denotes an interval consisting of a single element. The size of the signal on $I = [0, 5.5]$ is $\|\gamma \cap I\| = 5$, and on $I' = [0, 5.5)$, $\|\gamma \cap I'\| = 4$. The sequence $\overline{\mathsf{iip}}^1(\gamma)$ consists of the five intervals $[0.5, 2.5)$, $\{3.0\}$, $(3.5, 4.0)$, $(4.5, 5.0]$, and $\{5.5\}$ and $\overline{\mathsf{iip}}^0(\gamma)$ consists of the six intervals $[0, 0.5)$, $[2.5, 3.0)$, $(3.0, 3.5]$, $[4.0, 4.5]$, $(5.0, 5.5)$, and $(5.5, \infty)$.

2.3 Metric temporal logics

We restrict our attention to a syntactically defined safety fragment of metric temporal logic (MTL) in a point-based and an interval-based setting, namely, the past-only fragment. See [2,18] for the syntax and semantics of metric temporal logics that also include temporal future operators. Furthermore, note that temporal future operators like $\Diamond_I$, where the interval I is bounded, can be handled during monitoring by using queues that postpone the evaluation until enough time has elapsed. See [5,6], for such a monitoring algorithm that handles the arbitrary nesting of temporal past and bounded future operators.

Let P be a non-empty set of *propositions*. The syntax of the past-only fragment of metric temporal logic is given by the grammar

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \,\mathsf{S}_I\, \phi,$$

where $p \in P$ and $I \in \mathbb{I}$. Note that S is the temporal past operator "since" of linear temporal logic (LTL) [21] extended with an interval for the metric setting.

We use standard syntactic sugar. For instance, $\phi \vee \psi$ stands for the formula $\neg(\neg\phi \wedge \neg\psi)$, $\phi \to \psi$ stands for $\neg\phi \vee \psi$, $\blacklozenge_I\, \psi$ ("once") stands for $(p \vee \neg p)\, \mathsf{S}_I\, \psi$, for some $p \in P$, and $\phi\, \mathsf{T}_I\, \psi$ ("trigger") stands for $\neg(\neg\phi\, \mathsf{S}_I\, \neg\psi)$. Moreover, we often omit the interval $I = [0, \infty)$ attached to a temporal operator. To reduce the number of parentheses, we employ standard

conventions about operators' binding strength. For instance, $\neg$ binds stronger than $\wedge$ and the Boolean operators bind stronger than temporal ones.

We define next two satisfaction relations, the interval-based satisfaction relation $\models$ and the point-based satisfaction relation $\models^{\bullet}$, where $\hat{\gamma} = (\gamma_p)_{p \in P}$ is a family of signals, $\bar{\tau}$ a time sequence, $\tau \in \mathbb{T}$, and $i \in \mathbb{N}$.

$$\begin{array}{rll} \hat{\gamma}, \tau \models p & \text{iff} & \tau \in \gamma_p \\ \hat{\gamma}, \tau \models \neg\phi & \text{iff} & \hat{\gamma}, \tau \not\models \phi \\ \hat{\gamma}, \tau \models \phi \wedge \psi & \text{iff} & \hat{\gamma}, \tau \models \phi \text{ and } \hat{\gamma}, \tau \models \psi \\ \hat{\gamma}, \tau \models \phi \,\mathsf{S}_I\, \psi & \text{iff} & \text{there is } \tau' \in [0, \tau] \text{ with } \tau - \tau' \in I \text{ such that} \\ & & \hat{\gamma}, \tau' \models \psi \text{ and } \hat{\gamma}, \kappa \models \phi, \text{ for all } \kappa \in (\tau', \tau] \end{array}$$

and

$$\begin{array}{rll} \hat{\gamma}, \bar{\tau}, i \models^{\bullet} p & \text{iff} & \tau_i \in \gamma_p \\ \hat{\gamma}, \bar{\tau}, i \models^{\bullet} \neg\phi & \text{iff} & \hat{\gamma}, \bar{\tau}, i \not\models^{\bullet} \phi \\ \hat{\gamma}, \bar{\tau}, i \models^{\bullet} \phi \wedge \psi & \text{iff} & \hat{\gamma}, \bar{\tau}, i \models^{\bullet} \phi \text{ and } \hat{\gamma}, \bar{\tau}, i \models^{\bullet} \psi \\ \hat{\gamma}, \bar{\tau}, i \models^{\bullet} \phi \,\mathsf{S}_I\, \psi & \text{iff} & \text{there is } i' \in [0, i] \cap \mathbb{N} \text{ with } \tau_i - \tau_{i'} \in I \text{ such that} \\ & & \hat{\gamma}, \bar{\tau}, i' \models^{\bullet} \psi \text{ and } \hat{\gamma}, \bar{\tau}, k \models^{\bullet} \phi, \text{ for all } k \in (i', i] \cap \mathbb{N} \end{array}$$

Note that $\models$ defines the truth value of a formula for every $\tau \in \mathbb{T}$. In contrast, a formula's truth value with respect to $\models^{\bullet}$ is defined at the "sample points" $i \in \mathbb{N}$ to which the "timestamps" $\tau_i \in \mathbb{T}$ from the time sequence $\bar{\tau}$ are attached. For instance, under the interval-based semantics, the formula $\phi \,\mathsf{S}_I\, \psi$ holds at time τ for a family of signals $\hat{\gamma}$ if and only if there is a previous time τ' with $\tau - \tau' \in I$ such that ψ holds at τ' and ϕ holds throughout the interval $(\tau', \tau]$. In contrast, under the point-based semantics, $\phi \,\mathsf{S}_I\, \psi$ holds at the sample point i for a family of signals $\hat{\gamma}$ and a sequence of sample points with timestamps given by the time sequence $\bar{\tau}$ if and only if there is a previous sample point i' with $\tau_i - \tau_{i'} \in I$ such that ψ holds at i' and ϕ holds at each sample point k, where $i' < k \leq i$.

We denote the set of subformulas of a formula ϕ by $\mathsf{sf}(\phi)$. We say that a formula is a temporal formula if it is of the form $\alpha \,\mathsf{S}_I\, \beta$. Note that $p \wedge \blacklozenge\, q$ is not considered a temporal formula. Furthermore, we define $\mathsf{tsf}(\phi)$ as the set of the temporal subformulas of ϕ and $\mathsf{dsf}(\phi)$ as the set of the direct subformulas of ϕ, i.e.,

$$\mathsf{tsf}(\phi) := \{\psi \in \mathsf{sf}(\phi) \mid \psi \text{ is of the form } \psi_1 \,\mathsf{S}_I\, \psi_2\}$$

and

$$\mathsf{dsf}(\phi) := \begin{cases} \emptyset & \text{if } \phi \in P, \\ \{\phi'\} & \text{if } \phi \text{ is of the form } \neg\phi', \\ \{\phi_1, \phi_2\} & \text{if } \phi \text{ is of the form } \phi_1 \wedge \phi_2 \text{ or } \phi_1 \,\mathsf{S}_I\, \phi_2. \end{cases}$$

Finally, the size of the formula ϕ, denoted by $|\phi|$, is the number of nodes in ϕ's parse tree.

3 Point-based versus interval-based time models

We first point out several shortcomings of a point-based time model in Sect. 3.1. In Sect. 3.2, we then present a class of formulas for which the point-based and the interval-based time models coincide.

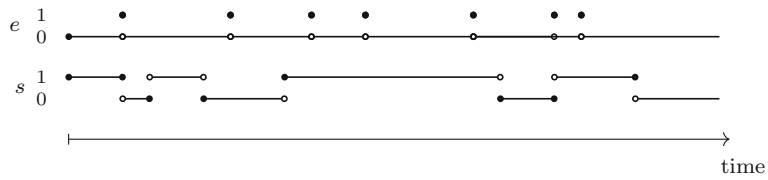

Fig. 2 Event and state signals

3.1 State variables and system events

Atomic proposition $p \in P$ can model both state variables and system events. These are different kinds of entities, and both may be involved when formalizing properties of system behaviors. One distinguishing feature is that events happen at single points in time whereas the value of a state variable remains constant for some amount of time. In the following, we distinguish between these two entities.

Let P be the disjoint union of the proposition sets S and E. We call propositions in S *state propositions* and propositions in E *event propositions*. Semantically, a signal $\gamma \subseteq \mathbb{T}$ is an *event signal* if $\gamma \cap I$ is finite, for every bounded interval I, and a signal $\gamma \subseteq \mathbb{T}$ is a *state signal* if for every bounded left- and right-open interval I, the sets $\gamma \cap I$ and $(\mathbb{T} \backslash \gamma) \cap I$ are finite unions of non-singular intervals. A family of signals $\hat{\gamma} = (\gamma_p)_{p \in S \cup E}$ is *consistent* with S and E if γ_p is a state signal, for all $p \in S$, and γ_p is an event signal, for all $p \in E$. Figure 2 depicts consistent signals for the event and state propositions e and s, respectively. Note that the signal in Fig. 1 is neither an event signal nor a state signal.

The point-based semantics is often motivated by the study of real-time systems whose behavior is determined by system events. Intuitively, a time sequence $\bar{\tau}$ records the points in time when events occur and the signal γ_p for a proposition $p \in E$ consists of the points in time when the event p occurs. The following examples, however, demonstrate that the point-based semantics can be less intuitive than the interval-based semantics.

Example 3.1 A state proposition $p \in S$ can often be mimicked by the formula $\neg f \,\mathsf{S}\, s$ with corresponding event propositions $s, f \in E$ representing "start" and "finish." For the state signal γ_p, let γ_s and γ_f be the event signals where γ_s and γ_f consist of the points in time of γ_p when the Boolean state variable starts and respectively stops holding. Then $(\gamma_s, \gamma_f), \tau \models \neg f \,\mathsf{S}\, s$ iff $\gamma_p, \tau \models p$, for any $\tau \in \mathbb{T}$, under the assumption that $I \cap \gamma_p$ is the finite union of left-closed and right-open intervals, for every bounded left-closed and right-open interval I.

However, replacing p by $\neg f \,\mathsf{S}\, s$ does not always capture the essence of a Boolean state variable when using the point-based semantics. Consider the formula $\blacklozenge_{[0,1]}\, p$ containing the state proposition p and let $\gamma_p = [0, 5)$ be a state signal. Moreover, let (γ_s, γ_f) be the family of corresponding event signals for the event propositions s and f, i.e., $\gamma_s = \{0\}$ and $\gamma_f = \{5\}$. For a time sequence $\bar{\tau}$ with $\tau_0 = 0$ and $\tau_1 = 5$, we have that $(\gamma_s, \gamma_f), \bar{\tau}, 1 \not\models \blacklozenge_{[0,1]}(\neg f \,\mathsf{S}\, s)$ but $\gamma_p, \tau_1 \models \blacklozenge_{[0,1]}\, p$. Note that $\bar{\tau}$ only contains timestamps when an event occurs. An additional sample point between τ_0 and τ_1 with, e.g., the timestamp 4 would result in identical truth values at time 5.

Even when restricted to events, the point-based semantics can be unintuitive.

Example 3.2 Consider the (event) signals $\gamma_p = \{\tau \in \mathbb{T} \mid \tau = 2n, \text{ for some } n \in \mathbb{N}\}$ and $\gamma_q = \emptyset$ for the (event) propositions p and q. One might expect that these signals satisfy the

formula $p \rightarrow \blacklozenge_{[0,1]} \neg q$ at every point in time. However, for a time sequence $\bar{\tau}$ with $\tau_0 = 0$ and $\tau_1 = 2$, $\hat{\gamma}, \bar{\tau}, 1 \not\models^{\bullet} p \rightarrow \blacklozenge_{[0,1]} \neg q$. The reason is that in the point-based semantics, the temporal operator $\blacklozenge_I$ requires the existence of a previous point in time that also occurs in the time sequence $\bar{\tau}$.

As another example consider the formula $\blacklozenge_{[0,1]} \blacklozenge_{[0,1]} p$. One might expect that it is logically equivalent to $\blacklozenge_{[0,2]} p$. However, this is not the case in the point-based semantics. To see this, consider a time sequence $\bar{\tau}$ with $\tau_0 = 0$ and $\tau_1 = 2$. Then $\hat{\gamma}, \bar{\tau}, 1 \not\models^{\bullet} \blacklozenge_{[0,1]} \blacklozenge_{[0,1]} p$ and $\hat{\gamma}, \bar{\tau}, 1 \models^{\bullet} \blacklozenge_{[0,2]} p$ if $\tau_0 \in \gamma_p$.

Remark 3.1 These examples suggest that adding additional sample points restores a formula's intended meaning, which usually stems from having the interval-based semantics in mind. However, a drawback of this approach for monitoring is that each additional sample point increases the workload of a point-based monitoring algorithm, since it is invoked for each sample point. Moreover, in the dense time domain, adding sample points does not always make the two semantics coincide. For instance, for $\gamma_p = [0, 1)$ and $\tau \geq 1$, we have that $\gamma_p, \tau \not\models \neg p \,\mathsf{S}\, p$ and $\gamma_p, \bar{\tau}, i \models^{\bullet} \neg p \,\mathsf{S}\, p$, for every time sequence $\bar{\tau}$ with $\tau_0 < 1$ and every $i \in \mathbb{N}$.

3.2 Event-relativized formulas

In the following, we identify a class of formulas for which the point-based and the interval-based semantics coincide. For formulas in this class, a point-based monitoring algorithm can be used to soundly monitor properties given by formulas interpreted using the interval-based semantics. We assume that the propositions are typed, i.e., $P = S \cup E$, where S contains the state propositions and E the event propositions, and a family of signals $\hat{\gamma} = (\gamma_p)_{p \in S \cup E}$ is consistent with S and E. Moreover, we assume without loss of generality that there is always at least one event signal γ in $\hat{\gamma}$ that is the infinite union of singular intervals, e.g., γ is the signal of a clock event that regularly occurs over time.

We inductively define the sets $rel_\forall$ and $rel_\exists$ for formulas in negation normal form. Recall that a formula is in negation normal form if negation only occurs directly in front of propositions. A logically-equivalent negation normal form of a formula can always be obtained by eliminating double negations and by pushing negations inwards, where we consider the Boolean connective $\vee$ and the temporal operator T_I as primitives.

$$
\begin{aligned}
\neg p &\in rel_\forall \quad \text{if} \quad p \in E \\
\phi_1 \wedge \phi_2 &\in rel_\forall \quad \text{if} \quad \phi_1 \in rel_\forall \text{ and } \phi_2 \in rel_\forall \\
\phi_1 \vee \phi_2 &\in rel_\forall \quad \text{if} \quad \phi_1 \in rel_\forall \text{ or } \phi_2 \in rel_\forall
\end{aligned}
$$

$$
\begin{aligned}
p &\in rel_\exists \quad \text{if} \quad p \in E \\
\phi_1 \wedge \phi_2 &\in rel_\exists \quad \text{if} \quad \phi_1 \in rel_\exists \text{ or } \phi_2 \in rel_\exists \\
\phi_1 \vee \phi_2 &\in rel_\exists \quad \text{if} \quad \phi_1 \in rel_\exists \text{ and } \phi_2 \in rel_\exists
\end{aligned}
$$

Definition 3.1 A formula ϕ is *event-relativized* if $\alpha \in rel_\forall$ and $\beta \in rel_\exists$, for every subformula of ϕ of the form $\alpha \,\mathsf{S}_I\, \beta$ or $\beta \,\mathsf{T}_I\, \alpha$. Furthermore, formula ϕ *strongly event-relativized* if ϕ is event-relativized and $\phi \in rel_\forall \cup rel_\exists$.

Intuitively, a formula ϕ is event-relativized if the direct subformulas of the temporal formulas of ϕ are relativized (or guarded) by event propositions. The event-relativized fragment is analogous to the guarded fragment of first-order logic, where quantified variables must be guarded by atomic formulas. Indeed, a formula of the form $\phi \,\mathsf{S}_I\, \psi$ hides an existential

quantification behind ψ and a universal quantification behind ϕ, which explains the notation behind the $rel_\forall$ and $rel_\exists$ sets. Also note that a formula of the form $(p \rightarrow \phi)\,\mathsf{S}_I\,(q \wedge \psi)$, where p and q are event propositions, is event-relativized. In contrast, the formulas in Examples 3.1 and 3.2 are not event-relativized.

The following theorem relates the interval-based semantics and the point-based semantics for event-relativized formulas.

Theorem 3.1 *Let $\hat{\gamma} = (\gamma_p)_{p \in S \cup E}$ be a family of consistent signals and $\bar{\tau}$ the time sequence listing the occurrences of events in $\hat{\gamma}$, i.e., $\bar{\tau}$ is the time sequence obtained by linearly ordering the set $\cup_{p \in E} \gamma_p$. For an event-relativized formula ϕ and every $i \in \mathbb{N}$, it holds that*

$$\hat{\gamma}, \tau_i \models \phi \quad \textit{iff} \quad \hat{\gamma}, \bar{\tau}, i \models^{\bullet} \phi.$$

Furthermore, if ϕ is strongly event-relativized, then it also holds that (a) $\hat{\gamma}, \tau \not\models \phi$ if $\phi \in rel_\exists$ and (b) $\hat{\gamma}, \tau \models \phi$ if $\phi \in rel_\forall$, for all $\tau \in \mathbb{T} \backslash \{\tau_i \mid i \in \mathbb{N}\}$.

Proof The proof is by induction on the structure of the formula ϕ in negation normal form. Let T be the set $\mathbb{T} \backslash \{\tau_i \mid i \in \mathbb{N}\}$.

Base case: $\phi = p$ with $p \in P$. If p is a state proposition then there is nothing to prove. Assume that p is an event proposition. By definition, p is strongly event-relativized, in particular, $p \in rel_\exists$. In the interval-based semantics for $\tau \in \mathbb{T}$, it holds that $\hat{\gamma}, \tau \models p$ iff $\tau \in \gamma_p$. Since p is an event proposition, $\tau = \tau_i$, for some $i \in \mathbb{N}$. It follows that $\hat{\gamma}, \tau_i \models p$ iff $\gamma_p, \bar{\tau}, i \models^{\bullet} p$. Note that $\hat{\gamma}, \tau \not\models p$ when $\tau \in T$.

Base case: $\phi = \neg p$ with $p \in P$. If p is a state proposition then there is nothing to prove. Assume that p is an event proposition. By definition, $\neg p$ is strongly event-relativized, in particular, $\neg p \in rel_\forall$. In the interval-based semantics for $\tau \in \mathbb{T}$, it holds that $\hat{\gamma}, \tau \models \neg p$ iff $\tau \notin \gamma_p$. Since p is an event proposition, if $\tau = \tau_i$ then $\tau_i \notin \gamma_p$, for all $i \in \mathbb{N}$. It follows that $\hat{\gamma}, \tau_i \models \neg p$ iff $\gamma_p, \bar{\tau}, i \models^{\bullet} \neg p$. Note that $\hat{\gamma}, \tau \not\models \neg p$ when $\tau \in T$.

Step case: $\phi = \phi_1 \wedge \phi_2$. We have the following equivalences: $\hat{\gamma}, \tau_i \models \phi_1 \wedge \phi_2$ iff (by the interval-based semantics) $\hat{\gamma}, \tau_i \models \phi_1$ and $\hat{\gamma}, \tau_i \models \phi_2$ iff (by the induction hypothesis) $\hat{\gamma}, \bar{\tau}, i \models^{\bullet} \phi_1$ and $\hat{\gamma}, \bar{\tau}, i \models^{\bullet} \phi_2$ iff (by the point-based semantics) $\hat{\gamma}, \bar{\tau}, i \models^{\bullet} \phi_1 \wedge \phi_2$.

If $\phi \in rel_\exists$ then by definition, $\phi_1 \in rel_\exists$ or $\phi_2 \in rel_\exists$. Without loss of generality, assume $\phi_1 \in rel_\exists$. By the induction hypothesis, $\hat{\gamma}, \tau \not\models \phi_1$, for all $\tau \in T$. It follows that $\hat{\gamma}, \tau \not\models \phi$, for all $\tau \in T$.

If $\phi \in rel_\forall$ then by definition, $\phi_1 \in rel_\forall$ and $\phi_2 \in rel_\forall$. By the induction hypothesis, $\hat{\gamma}, \tau \models \phi_1$, for all $\tau \in T$ and $\hat{\gamma}, \tau \models \phi_2$, for all $\tau \in T$. It follows that $\hat{\gamma}, \tau \models \phi$, for all $\tau \in T$.

Step case: $\phi = \phi_1 \vee \phi_2$. This case is dual to the previous case. We therefore omit it.

Step case: $\phi = \phi_1\,\mathsf{S}_I\,\phi_2$. Since ϕ is not strongly event-relativized, we need only show that $\hat{\gamma}, \tau_i \models \phi$ iff $\hat{\gamma}, \bar{\tau}, i \models^{\bullet} \phi$. We have the equivalence

$$\begin{aligned} \hat{\gamma}, \tau_i \models \phi_1\,\mathsf{S}_I\,\phi_2 \quad \text{iff} \quad & \text{there is some } \tau \in [0, \tau_i) \text{ with } \tau_i - \tau \in I, \\ & \hat{\gamma}, \tau \models \phi_2, \text{ and } \hat{\gamma}, \kappa \models \phi_1, \text{ for all } \kappa \in (\tau, \tau_i]. \end{aligned}$$

Since $\phi_2 \in rel_\exists$, there is some $j \in \mathbb{N}$ with $\tau_j = \tau$. From the induction hypothesis and the fact that $\phi_1 \in rel_\forall$, we conclude that

$$\begin{aligned} \hat{\gamma}, \tau_i \models \phi_1\,\mathsf{S}_I\,\phi_2 \quad \text{iff} \quad & \text{there is some } j \leq i \text{ with } \tau_i - \tau_j \in I, \\ & \hat{\gamma}, \bar{\tau}, j \models^{\bullet} \phi_2, \text{ and } \hat{\gamma}, \bar{\tau}, k \models^{\bullet} \phi_1, \text{ for all } k \in \{j+1, \ldots, i\}. \end{aligned}$$

In the point-based semantics, the right-hand side is by definition equivalent to $\hat{\gamma}, \bar{\tau}, i \models^{\bullet} \phi_1\,\mathsf{S}_I\,\phi_2$.

Step case: $\phi = \phi_1\,\mathsf{T}_I\,\phi_2$. This case is dual to the previous case. We therefore omit it. □

The definition of event-relativized formulas and Theorem 3.1 straightforwardly extend to richer real-time logics that also contain temporal future operators and are first-order. We point out that most formulas that we encountered when formalizing security policies in such a richer temporal logic are strongly event-relativized [4,6]. We also remark that the fragment of event-relativized formulas is not the maximal one for which the two semantics coincide. For instance, the two semantics of the formula $p \vee \neg p$ coincide, however the formula is not event-relativized.

From Theorem 3.1, it follows that the interval-based semantics can simulate the point-based one by using a fresh event proposition sp with its signal $\gamma_{sp} = \{\tau_i \mid i \in \mathbb{N}\}$, for a time sequence $\bar{\tau}$. We then event-relativize a formula ϕ with the proposition sp, i.e., subformulas of the form $\psi_1 \,\mathsf{S}_I\, \psi_2$ are replaced by $(sp \rightarrow \psi_1) \,\mathsf{S}_I\, (sp \wedge \psi_2)$ and $\psi_1 \,\mathsf{T}_I\, \psi_2$ by $(sp \wedge \psi_1) \,\mathsf{T}_I\, (sp \rightarrow \psi_2)$.

4 Point-based monitoring

In this section, we present and analyze our monitoring algorithm for the point-based setting. It iteratively computes the truth values of a formula ϕ at the sample points $i \in \mathbb{N}$ for a given time sequence $\bar{\tau}$ and a family of signals $\hat{\gamma} = (\gamma_p)_{p \in P}$. We assume that $\bar{\tau}$ and $\hat{\gamma}$ are given incrementally, i.e., in the $(i+1)$st iteration, the monitor obtains the timestamp τ_i and the signal values at τ_i. Concretely, the monitor receives the snapshots $\Gamma_i := \{p \in P \mid \tau_i \in \gamma_p\}$, for $i \in \mathbb{N}$, i.e., Γ_i is the set of propositions that hold at time τ_i. Note that in the point-based setting we need not consider the value of signals at times between two successive sample points.

We present our monitoring algorithm in pseudo code, written in a functional-programming style using pattern matching. We write $\langle\rangle$ to denote the empty sequence, $+\!\!+$ for sequence concatenation, and $x :: L$ for the sequence with head x and tail L. To simplify matters, we assume throughout this section, without loss of generality, that the temporal subformulas of a formula ϕ occur only once in ϕ. Moreover, we let P be the set of propositions that occur in ϕ.

Each iteration of the monitor is performed by executing the procedure $\mathsf{step}^\bullet$. At sample point $i \in \mathbb{N}$, $\mathsf{step}^\bullet$ takes as arguments the formula ϕ, the snapshot Γ_i, and i's timestamp τ_i. It computes the truth value of ϕ at i recursively over ϕ's structure. For efficiency, $\mathsf{step}^\bullet$ maintains for each subformula ψ of the form $\psi_1 \,\mathsf{S}_I\, \psi_2$ a sequence L_ψ of timestamps. These sequences are initialized by the procedure $\mathsf{init}^\bullet$ and updated by the procedure $\mathsf{update}^\bullet$. These three procedures are given in Fig. 3 and are described next.

The base case of $\mathsf{step}^\bullet$, where ϕ is a proposition, and the cases for the Boolean connectives $\neg$ and $\wedge$ are straightforward. The only involved case is where ϕ is of the form $\phi_1 \,\mathsf{S}_I\, \phi_2$. Here, $\mathsf{step}^\bullet$ first updates the sequence L_ϕ and then computes ϕ's truth value at the sample point $i \in \mathbb{N}$.

Before we describe how we update the sequence L_ϕ, we describe the elements that are stored in L_ϕ and how we obtain from them ϕ's truth value. After the update of L_ϕ by $\mathsf{update}^\bullet$, the sequence L_ϕ stores the timestamps τ_j with $\tau_i - \tau_j \in {}^{\leq}I$ (i.e., the timestamps that satisfy the time constraint now, at sample point i, or that may satisfy it in the future, at some sample point $i' > i$) at which ϕ_2 holds and from which ϕ_1 continuously holds up to the current sample point i (i.e., ϕ_2 holds at $j \leq i$ and ϕ_1 holds at each $k \in \{j+1, \ldots, i\}$). Moreover, if there are timestamps τ_j and $\tau_{j'}$ with $j < j'$ in L_ϕ with $\tau_i - \tau_j \in I$ and $\tau_i - \tau_{j'} \in I$ then we only keep in L_ϕ the timestamp of the later sample point, i.e., $\tau_{j'}$. Finally, the timestamps in L_ϕ are stored in ascending order. Having L_ϕ at hand, it is easy to determine ϕ's truth

```
step•(φ, Γ, τ)
  case φ = p
    return p ∈ Γ
  case φ = ¬φ'
    return not step•(φ', Γ, τ)
  case φ = φ1 ∧ φ2
    return step•(φ1, Γ, τ) and step•(φ2, Γ, τ)
  case φ = φ1 S_I φ2
    update•(φ, Γ, τ)
    if L_φ = ⟨⟩ then
      return false
    else
      return τ − head(L_φ) ∈ I
```

```
init•(φ)
  for each ψ ∈ sf(φ) with ψ = ψ1 S_I ψ2 do
    L_ψ := ⟨⟩

update•(φ, Γ, τ)
  let φ1 S_I φ2 = φ
      b1 = step•(φ1, Γ, τ)
      b2 = step•(φ2, Γ, τ)
      L = if b1 then drop•(L_φ, I, τ) else ⟨⟩
  in
    if b2 then
      L_φ := L +++ ⟨τ⟩
    else
      L_φ := L
```

Fig. 3 Monitoring in a point-based setting

```
drop•(L, I, τ)
  case L = ⟨⟩
    return ⟨⟩
  case L = κ :: L'
    if τ − κ ⊄ ≤I then
      return drop•(L', I, τ)
    else
      return drop'•(κ, L', I, τ)
```

```
drop'•(κ, L', I, τ)
  case L' = ⟨⟩
    return ⟨κ⟩
  case L' = κ' :: L''
    if τ − κ' ∈ I then
      return drop'•(κ', L'', I, τ)
    else
      return κ :: L'
```

Fig. 4 Auxiliary procedures

value. If L_ϕ is the empty sequence, then obviously ϕ does not hold at sample point i. If L_ϕ is non-empty, then ϕ holds at i iff the first timestamp κ in L_ϕ fulfills the timing constraints given by the interval I, i.e., $\tau_i - \kappa \in I$. Recall that ϕ holds at i iff there is a sample point $j \leq i$ with $\tau_i - \tau_j \in I$ at which ϕ_2 holds and since then ϕ_1 continuously holds.

Initially, L_ϕ is the empty sequence. If ϕ_2 holds at sample point i, then update$^\bullet$ adds the timestamp τ_i to L_ϕ. However, prior to this, it removes the timestamps of the sample points from which ϕ_1 does not continuously hold. Clearly, if ϕ_1 does not hold at i then we can empty the sequence L_ϕ. Otherwise, if ϕ_1 holds at i, we first drop the timestamps for which the distance to the current timestamp τ_i becomes too large with respect to the right margin of I. Afterwards, we drop timestamps until we find the last timestamp τ_j with $\tau_i - \tau_j \in I$. This is done by the procedures drop$^\bullet$ and drop'$^\bullet$ shown in Fig. 4.

The following theorem states the algorithm's correctness.

Theorem 4.1 *Let ϕ be a formula, $\hat{\gamma} = (\gamma_p)_{p \in P}$ be a family of signals, $\bar{\tau}$ be a time sequence, and $n > 0$. The procedure* step$^\bullet(\phi, \Gamma_{n-1}, \tau_{n-1})$ *terminates and returns* **true** *iff $\hat{\gamma}, \bar{\tau}, n-1 \models^\bullet \phi$, whenever* init$^\bullet(\phi)$, step$^\bullet(\phi, \Gamma_0, \tau_0), \ldots,$ step$^\bullet(\phi, \Gamma_{n-2}, \tau_{n-2})$ *were called previously in this order, where $\Gamma_i = \{p \in P \mid \tau_i \in \gamma_p\}$, for $i < n$.*

Proof The recursive procedures step$^\bullet$, drop$^\bullet$, and drop'$^\bullet$ terminate as they are called on formulas and respectively sequences of sizes that are strictly smaller with each call.

The second statement of the theorem follows by induction on the well-ordered set of tuples (i, ψ) with $0 \leq i < n$ and ψ a subformula of ϕ, ordered lexicographically and using the formula size when comparing the formula components. The proved statement is the theorem's statement strengthened with the invariant on the sequences L_ψ stated in the explanation of the step$^\bullet$ procedure. The bases cases are trivial, while the step case distinguishes cases based on the form of ψ. The cases where ψ's main connective is non-temporal are straightforward,

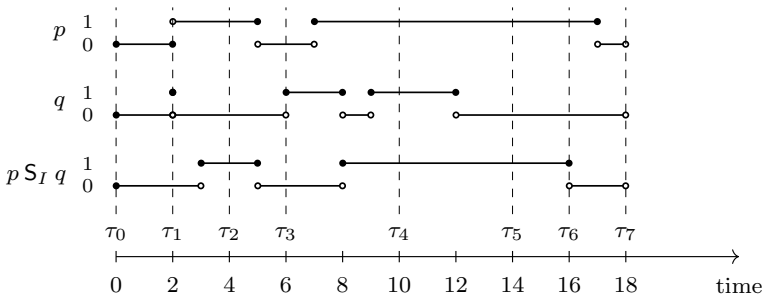

Fig. 5 The signal $\gamma_{p\mathsf{S}_I q}$ given the signals γ_p and γ_q

Table 1 The sequence L_ψ and the return value of the $\mathsf{step}^\bullet$ procedure at the first eight iterations

i	τ_i	L_ψ	$\mathsf{step}^\bullet(\psi, \Gamma_i, \tau_i)$
0	0	$\langle\rangle$	false ($L_\psi = \langle\rangle$)
1	2	$\langle 2\rangle$	false ($2 - 2 \notin [1, 4]$)
2	4	$\langle 2\rangle$	true ($4 - 2 \in [1, 4]$)
3	6	$\langle 6\rangle$	false ($6 - 6 \notin [1, 4]$)
4	10	$\langle 6, 10\rangle$	true ($10 - 6 \in [1, 4]$)
5	14	$\langle 10\rangle$	true ($14 - 10 \in [1, 4]$)
6	16	$\langle\rangle$	false ($L_\psi = \langle\rangle$)
7	18	$\langle\rangle$	false ($L_\psi = \langle\rangle$)

while the proof for the case when ψ is of the form $\psi_1 \mathsf{S}_I \psi_2$ follows the same reasoning steps as in the explanation previously given. □

Example 4.1 We illustrate the algorithm on the formula $\psi = p \mathsf{S}_I q$, with $I = [1, 4]$, and on the snapshots of the signals γ_p and γ_q at the timestamps τ_i, for $0 \leq i < 8$, where both the signals and the snapshots are given in Fig. 5. The dashed vertical lines highlight the sample points. The figure also depicts the signal γ_ψ. For its computation, we refer the reader to Sect. 5 and, in particular, to Example 5.1.

Table 1 gives the contents of the sequence L_ψ at the end of the execution of the $\mathsf{step}^\bullet$ procedure together with the return value of this procedure at the $(i + 1)$st iteration, for $0 \leq i < 8$. After the return value, we also give in parentheses the reason for this decision, namely whether $L_\psi = \langle\rangle$ or alternatively whether $\tau_i - \mathsf{head}(L_\psi) \in I$. Note that at sample point 6 with timestamp $\tau_6 = 16$, the point-based and interval-based semantics do not coincide. Indeed, the point-based algorithm returns false at this sample point because there is no sample point in the interval [16–4, 16–1] where q were satisfied, while the signal γ_ψ is high at the corresponding timestamp, because γ_q is high at time 12 and γ_p is always high in the interval (12, 16].

We conclude this subsection by analyzing the monitor's computational complexity. Observe that we cannot bound the space that is needed to represent the timestamps in the time sequence $\bar{\tau}$. They become arbitrarily large as time progresses. Moreover, since the time domain is dense, they can be arbitrarily close to each other. As a consequence, operations like the subtraction of elements from $\mathbb{T}$ cannot be done in constant time. We return to this point in Sect. 6.2.

In the following, we assume that each $\tau \in \mathbb{T}$ is represented by 2 bitstrings, for the numerator and the denominator. The representation of an interval I consists of the representations of

$\ell(I)$ and $r(I)$ and whether the left margin and right margin is closed or open. We denote the maximum length of these bit strings by $\|\tau\|$ and $\|I\|$, respectively. The operations on elements in $\mathbb{T}$ that the monitoring algorithm performs are subtractions and membership tests. Subtraction $\tau - \tau'$ can be carried out in time $\mathcal{O}(m^2)$, where $m = \max\{\|\tau\|, \|\tau'\|\}$.[2] A membership test $\tau \in I$ can also be carried out in time $\mathcal{O}(m^2)$, where $m = \max\{\|\tau\|, \|I\|\}$.

The following theorem establishes an upper bound on the time complexity of our monitoring algorithm.

Theorem 4.2 *Let ϕ, $\hat{\gamma}$, $\bar{\tau}$, n, and $\Gamma_0, \ldots, \Gamma_{n-1}$ be as in* Theorem 4.1. *Executing the sequence* $\mathsf{init}^\bullet(\phi)$, $\mathsf{step}^\bullet(\phi, \Gamma_0, \tau_0), \ldots, \mathsf{step}^\bullet(\phi, \Gamma_{n-1}, \tau_{n-1})$ *requires* $\mathcal{O}(m^2 \cdot n \cdot |\phi|)$ *time, where* $m = \max(\{\|I\| \mid \alpha\ \mathsf{S}_I\ \beta \in \mathsf{sf}(\phi)\} \cup \{\|\tau_0\|, \ldots, \|\tau_{n-1}\|\})$.

Proof We first analyze the running time of a single iteration. We claim that the running time of $\mathsf{step}^\bullet(\phi, \Gamma_{n-1}, \tau_{n-1})$ is in $\mathcal{O}(|\phi| + m^2 \cdot \sum_{\psi \in \mathsf{tsf}(\phi)} T^n_\psi)$, where $T^1_\psi := 1$ and

$$T^n_{\alpha \mathsf{S}_I \beta} := 1 + \left|\{\tau_j \mid \tau_{n-2} - \ell(I) \leq \tau_j \leq \tau_{n-1} - \ell(I), \text{ for some } j < n\}\right|,$$

for $n > 1$.

Since we traverse ϕ's syntax tree recursively, the running time of one iteration is the sum of all t_ψ, for all occurrences of subformulas ψ of ϕ, where t_ψ denotes the running time for ψ without the running times for its proper subformulas.

We have that $t_\psi \in \mathcal{O}(1)$ for the cases where ψ is of the form p, $\neg\psi'$, or $\psi_1 \wedge \psi_2$. Note that we can assume, without loss of generality, that the membership test $p \in \Gamma$ for the base case in the procedure $\mathsf{step}^\bullet$ can be done in constant time. The reason is that, in the nth iteration, from the set Γ_n (which is given for instance as a list) we can first build a hash table that allows us to check in constant time whether the proposition p is an element of Γ_n. Building (and discarding) such a hash table takes $\mathcal{O}(|P|)$ time. Since $|P| \leq |\phi|$, this time factor is subsumed by the claimed complexity $\mathcal{O}(|\phi| + m^2 \sum_{\psi \in \mathsf{tsf}(\phi)} T^n_\psi)$.

We next analyze the running time for the case where ψ is of the form $\psi_1\ \mathsf{S}_I\ \psi_2$. We first make the following observations about the sequence L_ψ and the elements it contains in the nth iteration.

(1) For each element τ in L_ψ, $\|\tau\| \leq m$, since τ is a timestamp that occurs in the prefix of length n of the time sequence $\bar{\tau}$.
(2) Removing the head and appending an element to L_ψ can be done in $\mathcal{O}(m)$; we assume that L_ψ is implemented as a doubly linked list with pointers to the first and to the last element.
(3) The disequality test $\mathsf{last}(L) \neq \tau$ and the membership tests whether the distance $\tau - \kappa$ is in I or ${}^{\leq}I$ can be performed in time $\mathcal{O}(m^2)$, since the timestamps τ and κ occur in the prefix of the time sequence $\bar{\tau}$, and thus, by assumption, $\|I\| \leq m$, $\|\tau\| \leq m$, and $\|\kappa\| \leq m$.

It follows from these observations that $t_\psi \in \mathcal{O}(m^2 \cdot T)$, where T is number of elements from the sequence L_ψ that are visited by the procedures $\mathsf{drop}^\bullet$ and $\mathsf{drop}'^\bullet$. Note that L_ψ is empty in the first iteration. Suppose that $n > 1$. The procedure $\mathsf{drop}^\bullet$ first traverses the sequence L_ψ up to the first element τ_k such that $\tau_{n-1} - \tau_k \notin I$. Hence all elements τ_j in L_ψ up to and excluding τ_k satisfy $\tau_j \leq \tau_{n-1} - \ell(I)$. Moreover, with the possible exception of the first element of L_ψ, all elements τ_j in L_ψ satisfy $\tau_j \geq \tau_{n-2} - \ell(I)$. Hence $T \leq 1 + T^n_\psi$

[2] Note that $\frac{p}{q} - \frac{p'}{q'} = \frac{p \cdot q' - p' \cdot q}{q \cdot q'}$ and that $\mathcal{O}(m^2)$ is an upper bound on the multiplication of two m bit integers. There are more sophisticated algorithms for multiplication that run in $\mathcal{O}(m \log m \log\log m)$ time [32] and $\mathcal{O}(m \log m 2^{\log^* m})$ time [13], where $\log^* m$ denotes the iterated logarithm of m. For simplicity, we use the quadratic upper bound.

since at most two elements (the first one and the last one visited in L_ψ) may be outside the interval $[\tau_{n-2} - \ell(I), \tau_{n-1} - \ell(I)]$.

We conclude that $\mathsf{step}^\bullet(\phi, \Gamma_{n-1}, \tau_{n-1})$ has the claimed running time $\mathcal{O}(|\phi| + m^2 \cdot \sum_{\psi \in \mathsf{tsf}(\phi)} T_\psi^n)$.

Finally we prove the upper bound on the running time of all n iterations, i.e., for the sequence $\mathsf{init}^\bullet(\phi)$, $\mathsf{step}^\bullet(\phi, \Gamma_0, \tau_0), \ldots, \mathsf{step}^\bullet(\phi, \Gamma_{n-1}, \tau_{n-1})$. Note that the sets $\{\tau_j \mid \tau_{i-1} - \ell(I) \leq \tau_j \leq \tau_i - \ell(I), \text{ for some } j \leq i\}$ and $\{\tau_j \mid \tau_i - \ell(I) \leq \tau_j \leq \tau_{i+1} - \ell(I), \text{ for some } j \leq i+1\}$ have at most one element in common. Thus, $\sum_{1<i\leq n} T_\psi^i \leq n + (n + |\{\tau_j \mid j < n\}|)$. Since $|\{\tau_j \mid j < n\}| \leq n$, this sum of the T_ψ^is is in $\mathcal{O}(n)$. Then, by summing up the running times of all iterations, we obtain that the total running time is in $\mathcal{O}(n \cdot |\phi| + m^2 \cdot n \cdot |\mathsf{tsf}(\psi)|)$, from which the stated upper bound follows. □

We now focus on our algorithm's space complexity. As previously observed, we cannot bound the space needed to represent the timestamps in the time sequence $\bar{\tau}$. Moreover, we cannot in general bound the size of the lists L_ψ. Indeed, consider the time sequence $\bar{\tau}$ with $\tau_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n+1}$, for $n \geq 0$. This sequence diverges. Consider also the formula $\phi = \blacklozenge_{[0,1]}\, p$ and the signal γ_p with $\gamma_p(\tau_n) = 1$ for any $n \geq 0$. Then the number of elements in L_ϕ increases at each iteration and is not bounded.

However, in practice, it is reasonable to assume a bound on the number of sample points per unit of time. Formally, given a $k \in \mathbb{N}$ with $k \geq 1$, we say that a time sequence $\bar{\tau}$ is *k-bounded* if $|\{i \in \mathbb{N} \mid \tau_i \in [\tau, \tau + 1)\}| \leq k$, for any $\tau \in \mathbb{T}$. The following theorem establishes the space complexity of our monitoring algorithm under the assumption that the input time sequence is k-bounded. We note in particular that the space usage is independent from the number n of snapshots.

Theorem 4.3 *Let ϕ, $\hat{\gamma}$, $\bar{\tau}$, n, and $\Gamma_0, \ldots, \Gamma_{n-1}$ be as in* Theorem 4.1*, and such that $\bar{\tau}$ is k-bounded, for some $k \in \mathbb{N}$ with $k \geq 1$. Executing the sequence $\mathsf{init}^\bullet(\phi)$, $\mathsf{step}^\bullet(\phi, \Gamma_0, \tau_0), \ldots, \mathsf{step}^\bullet(\phi, \Gamma_{n-1}, \tau_{n-1})$ requires $\mathcal{O}(m \cdot k \cdot \ell_\phi \cdot |\phi|)$ space, where $\ell_\phi = \max\{\ell(I) \mid \alpha\ \mathsf{S}_I\ \beta \in \mathsf{sf}(\phi)\}$ and $m = \max(\{\|I\| \mid \alpha\ \mathsf{S}_I\ \beta \in \mathsf{sf}(\phi)\} \cup \{\|\tau_0\|, \ldots, \|\tau_{n-1}\|\})$.*

Proof We first note that the space used by the algorithm is dominated by the space required to store the sequences L_ψ, with $\psi \in \mathsf{tsf}(\phi)$. That is, when ignoring this storage space, the algorithm uses $\mathcal{O}(m \cdot |\phi|)$ space. The space used to store these sequences is $\mathcal{O}(m \cdot \sum_{\psi \in \mathsf{tsf}(\phi)} |L_\psi|)$. (Recall that the elements of these sequences are among the timestamps $\tau_0, \ldots, \tau_{n-1}$.) Next, we note that the length of a sequence L_ψ after iteration $j \geq 0$ is upper bounded by $1 + |\{\tau_{j'} \mid \tau_j - \ell(I) \leq \tau_{j'} \leq \tau_j,\ j' \geq 0\}|$. As $\bar{\tau}$ is k-bounded, it follows that the length is upper bounded by $1 + k \cdot \ell(I)$. Summing up these upper bounds for each $\psi \in \mathsf{tsf}(\phi)$, we obtain the stated complexity. □

5 Interval-based monitoring

In this section, we present and analyze our monitoring algorithm for the interval-based setting. Let P be the set of propositions that occur in the given formula ϕ. The algorithm determines for a given family of signals $\hat{\gamma} = (\gamma_p)_{p \in P}$, the truth value of ϕ, for any $\tau \in \mathbb{T}$. In other words, it determines the set $\gamma_{\phi,\hat{\gamma}} := \{\tau \in \mathbb{T} \mid \hat{\gamma}, \tau \models \phi\}$. We simply write γ_ϕ instead of $\gamma_{\phi,\hat{\gamma}}$ when the family of signals $\hat{\gamma}$ is clear from the context. Similar to the point-based setting, the monitor incrementally receives the input $\hat{\gamma}$ and incrementally outputs γ_ϕ. That is, the input and output signals are split into "chunks" by an infinite interval partition $\bar{J}$. Concretely, the

```
step(φ, Δ̂, J)
  case φ = p
    return Δ_p
  case φ = ¬φ'
    let Δ' = step(φ', Δ̂, J)
    in
      return invert(Δ', J)
  case φ = φ1 ∧ φ2
    let Δ1 = step(φ1, Δ̂, J)
        Δ2 = step(φ2, Δ̂, J)
    in
      return intersect(Δ1, Δ2)
  case φ = φ1 S_I φ2
    let (Δ'1, Δ'2) = update(φ, Δ̂, J)
    in
      return merge(combine(Δ'1, Δ'2, I, J))
```

```
init(φ)
  for each ψ ∈ sf(φ) with ψ = ψ1 S_I ψ2 do
    K_ψ := ∅
    Δ_ψ := ⟨⟩
```

```
update(φ, Δ̂, J)
    let φ1 S_I φ2 = φ
        Δ1 = step(φ1, Δ̂, J)
        Δ2 = step(φ2, Δ̂, J)
        Δ'1 = prepend(K_φ, Δ1)
        Δ'2 = concat(Δ_φ, Δ2)
    in
      K_φ := if Δ'1 = ⟨⟩ then ∅ else last(Δ'1)
      Δ_φ := drop(Δ'2, I, J)
      return (Δ'1, Δ'2)
```

Fig. 6 Monitoring in an interval-based setting

input of the $(i+1)$st iteration consists of the formula ϕ that is monitored, the interval J_i of $\bar{J}$, and the family $\hat{\Delta}_i = (\Delta_{i,p})_{p\in P}$ of sequences of intervals $\Delta_{i,p} = \overline{\mathsf{iip}}^1(\gamma_p \cap J_i)$, for propositions $p \in P$. The output of the $(i+1)$st iteration is the sequence $\overline{\mathsf{iip}}^1(\gamma_\phi \cap J_i)$.

5.1 Algorithm description

As in Sect. 4, we assume without loss of generality that the temporal subformulas of a formula ϕ occur only once in ϕ. Observe that the sequence $\overline{\mathsf{iip}}^1(\gamma_p \cap J_i)$ only consists of a finite number of intervals since the signal γ_p of the proposition $p \in P$ satisfies the finite-variability condition and J_i is bounded. Moreover, since γ_p is stable on every interval in $\overline{\mathsf{iip}}(\gamma_p)$ and an interval has a finite representation, the sequence $\overline{\mathsf{iip}}^1(\gamma_p \cap J_i)$ finitely represents the signal chunk $\gamma_p \cap J_i$. Similar observations are valid for the signal chunk $\gamma_\phi \cap J_i$.

Each iteration is performed by the procedure step. To handle the since operator efficiently, step maintains for each subformula ψ of the form $\psi_1 \mathsf{S}_I \psi_2$, a (possibly empty) interval K_ψ and a finite sequence of intervals Δ_ψ. These global variables are initialized by the procedure init and updated by the procedure update. These three procedures are given in Fig. 6 and are described next.

The procedure step computes the signal chunk $\gamma_\phi \cap J_i$ recursively over the formula structure. It utilizes the right-hand sides of the equalities from the following lemma.

Lemma 5.1 *Let $p \in P$ and ϕ', ϕ_1, and ϕ_2 be formulas. Furthermore, let γ_p, $\gamma_{\neg\phi'}$, $\gamma_{\phi_1\wedge\phi_2}$, and $\gamma_{\phi_1 \mathsf{S}_I \phi_2}$ be signals of the corresponding formulas. The following equalities hold.*

$$\gamma_p \cap J_i = \bigcup_{K \in \overline{\mathsf{iip}}^1(\gamma_p \cap J_i)} K \tag{5.1}$$

$$\gamma_{\neg\phi'} \cap J_i = J_i \setminus \left(\bigcup_{K \in \overline{\mathsf{iip}}^1(\gamma_{\phi'} \cap J_i)} K \right) \tag{5.2}$$

$$\gamma_{\phi_1 \wedge \phi_2} \cap J_i = \bigcup_{\substack{K_1 \in \overline{\mathsf{iip}}^1(\gamma_{\phi_1} \cap J_i) \\ K_2 \in \overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap J_i)}} (K_1 \cap K_2) \tag{5.3}$$

$$\gamma_{\phi_1 \mathsf{S}_I \phi_2} \cap J_i = \bigcup_{\substack{K_1 \in \overline{\mathsf{iip}}^1(\gamma_{\phi_1} \cap^{\leq} J_i) \text{ with } K_1 \cap J_i \neq \emptyset \\ K_2 \in \overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap^{\leq} J_i) \text{ with } (K_2 \oplus I) \cap (J_i^{\geq}) \neq \emptyset}} \Big(\big((K_2 \cap {}^{+}K_1) \oplus I \big) \cap K_1 \cap J_i \Big) \tag{5.4}$$

where ${}^{+}K := \{\ell(K)\} \cup K$*, for* $K \in \mathbb{I}$*, i.e., making the interval* K *left-closed.*

Proof The equalities (5.1)–(5.3) are obvious. The equality (5.4) for $\phi_1 \mathsf{S}_I \phi_2$ is less obvious. To prove its inclusion $\subseteq$, assume $\tau \in \gamma_{\phi_1 \mathsf{S}_I \phi_2} \cap J_i$. By the semantics of the since operator, there is a $\tau_2 \in \gamma_{\phi_2}$ with $\tau - \tau_2 \in I$ and $\tau_1 \in \gamma_{\phi_1}$, for all $\tau_1 \in (\tau_2, \tau]$.

- Obviously, $\tau_2 \in K_2$, for some $K_2 \in \overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap^{\leq} J_i)$. By taking the time constraint $\tau - \tau_2 \in I$ into account, we get $(K_2 \oplus I) \cap (J_i^{\geq}) \neq \emptyset$. Note that even the more restrictive constraint $(K_2 \oplus I) \cap J_i \neq \emptyset$ holds. However, we employ the weaker constraint in our implementation as it is useful for later iterations.
- Since $\overline{\mathsf{iip}}(\gamma_{\phi_1})$ is the coarsest interval partition of γ_{ϕ_1}, there is an interval $K_1 \in \overline{\mathsf{iip}}^1(\gamma_{\phi_1})$ with $(\tau_2, \tau] \subseteq K_1$. As $\tau \in J_i$, the constraint $K_1 \cap J_i \neq \emptyset$ holds.

It follows that $\tau \in K_1$ and $\tau_2 \in {}^{+}K_1$, and thus $\tau_2 \in K_2 \cap {}^{+}K_1$. From $\tau - \tau_2 \in I$, we obtain that $\tau \in (K_2 \cap {}^{+}K_1) \oplus I$. Finally, since $\tau \in K_1 \cap J_i$, we have that $\tau \in ((K_2 \cap {}^{+}K_1) \oplus I) \cap K_1 \cap J_i$. The other inclusion $\supseteq$ can be shown similarly. □

The right-hand sides of the equalities (5.1)–(5.3) are directly reflected in our pseudo code. The case where ϕ is a proposition is straightforward. For the case $\phi = \neg\phi'$, we use the procedure invert, shown in Fig. 7, to compute $\overline{\mathsf{iip}}^1(\gamma_\phi \cap J_i)$ from $\Delta' = \overline{\mathsf{iip}}^1(\gamma_{\phi'} \cap J_i)$. This is done by "complementing" Δ' with respect to the interval J_i. For instance, the output of invert($\langle [1, 2]\,(3, 4)\rangle$, $[0, 10)$) is $\langle [0, 1)\,(2, 3]\,[4, 10)\rangle$. For the case $\phi = \phi_1 \wedge \phi_2$, we use the procedure intersect, also shown in Fig. 7, to compute $\overline{\mathsf{iip}}^1(\gamma_\phi \cap J_i)$ from $\Delta_1 = \overline{\mathsf{iip}}^1(\gamma_{\phi_1} \cap J_i)$ and $\Delta_2 = \overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap J_i)$. This procedure returns the sequence of intervals that have a non-empty intersection of two intervals in the input sequences. The elements in the returned sequence are stored in ascending order.

The use of the right-hand side of the equality (5.4) for $\phi = \phi_1 \mathsf{S}_I \phi_2$ for an implementation is less straightforward since the intervals K_1 and K_2 are not restricted to occur in the current chunk J_i. Instead, they are intervals in $\overline{\mathsf{iip}}^1(\gamma_{\phi_1} \cap {}^{\leq}J_i)$ and $\overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap {}^{\leq}J_i)$, respectively, satisfying certain constraints.

```
cons(K, Δ)
  if K = ∅ then
    return Δ
  else
    return K :: Δ

invert(Δ, J)
  case Δ = ⟨⟩
    return ⟨J⟩
  case Δ = K :: Δ'
    return cons(J ∩ <K, invert(Δ', J ∩ (K>)))
```

```
intersect(Δ1, Δ2)
  if Δ1 = ⟨⟩ or Δ2 = ⟨⟩ then
    return ⟨⟩
  else
    let K1 :: Δ1' = Δ1
        K2 :: Δ2' = Δ2
    in
      if K1 ∩ (K2>) = ∅ then
        return cons(K1 ∩ K2, intersect(Δ1', Δ2))
      else
        return cons(K1 ∩ K2, intersect(Δ1, Δ2'))
```

Fig. 7 The auxiliary procedures for the Boolean connectives

```
prepend(K, Δ)
  if K = ∅ then
    return Δ
  else
    case Δ = ⟨⟩
      return ⟨K⟩
    case Δ = K′ :: Δ′
      if adjacent(K, K′) or K ∩ K′ ≠ ∅ then
        return K ∪ K′ :: Δ′
      else
        return K :: Δ
```

```
combine(Δ′1, Δ′2, I, J)
  if Δ′1 = ⟨⟩ or Δ′2 = ⟨⟩ then
    return ⟨⟩
  else
    let K2 :: Δ″2 = Δ′2
    in
      if (K2 ⊕ I) ∩ J = ∅ then
        return ⟨⟩
      else
        let K1 :: Δ″1 = Δ′1
            Δ = if K2^> ∩ +K1 = ∅ then
                  combine(Δ″1, Δ′2, I, J)
                else
                  combine(Δ′1, Δ″2, I, J)
        in
          return (K2 ∩ +K1) ⊕ I) ∩ K1 ∩ J :: Δ
```

```
concat(Δ1, Δ2)
  case Δ1 = ⟨⟩
    return Δ2
  case Δ1 = Δ′1 ++ ⟨K1⟩
    return Δ′1 ++ prepend(K1, Δ2)
```

```
merge(Δ)
  case Δ = ⟨⟩
    return Δ
  case Δ = K :: Δ′
    return prepend(K, merge(Δ′))
```

```
drop(Δ′2, I, J)
  case Δ′2 = ⟨⟩
    return ⟨⟩
  case Δ′2 = K2 :: Δ″2
    let K = (K2 ⊕ I) ∩ (J^>)
    in
      if K = ∅ then
        return drop(Δ″2, I, J)
      else
        return drop′(K, Δ′2, I, J)
```

```
drop′(K, Δ′2, I, J)
  case Δ′2 = ⟨⟩
    return ⟨K⟩
  case Δ′2 = K2 :: Δ″2
    let K′ = (K2 ⊕ I) ∩ (J^>)
    in
      if K ⊆ K′ then
        return drop′(K′, Δ″2, I, J)
      else
        return Δ′2
```

Fig. 8 The auxiliary procedures for the since operator

For computing the signal chunk $\gamma_{\phi_1 \mathsf{S}_I \phi_2} \cap J_i$, the procedure step first determines the subsequences Δ'_1 and Δ'_2 of $\overline{\mathsf{iip}}^1(\gamma_{\phi_1} \cap (^{\leq} J_i))$ and $\overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap (^{\leq} J_i))$ consisting of those intervals K_1 and K_2 appearing in the equality (5.4), respectively.[3] This is done by the procedure update. Afterwards, step computes the sequence $\overline{\mathsf{iip}}^1(\gamma_{\phi} \cap J_i)$ from Δ'_1 and Δ'_2 using the procedures combine and merge, given in Fig. 8. We now explain how merge(combine($\Delta'_1, \Delta'_2, I, J$)) returns the sequence $\overline{\mathsf{iip}}^1(\gamma_{\phi_1 \mathsf{S}_I \phi_2} \cap J_i)$. First, combine($\Delta'_1, \Delta'_2, I, J$) computes a sequence of intervals whose union is $\gamma_{\phi_1 \mathsf{S}_I \phi_2} \cap J_i$. It traverses the ordered sequences Δ'_1 and Δ'_2 and adds the interval $((K_2 \cap {}^+K_1) \oplus I) \cap K_1 \cap J_i$ to the resulting ordered sequence, for K_1 in Δ'_1 and K_2 in Δ'_2. The test $K_2^{>} \cap {}^+K_1 = \emptyset$ determines in which sequence (Δ'_1 or Δ'_2) we advance next: if the test succeeds then $K'_2 \cap {}^+K_1 = \emptyset$ where K'_2 is the successor of K_2 in Δ'_2, and hence we advance in Δ'_1. This is because, if $K_2^{>} \cap {}^+K_1 = \emptyset$ then for all K'_2 following K_2 in Δ'_2 we have $K'_2 \cap {}^+K_1 = \emptyset$; in contrast, if $K_2^{>} \cap {}^+K_1 \neq \emptyset$ then for all K'_1 following K_1 in Δ'_1 we have $K_2 \cap {}^+K'_1 = \emptyset$. The sequence Δ'_2 is not necessarily traversed in its entirety: when $(K_2 \oplus I) \cap J_i = \emptyset$, one need not inspect other elements K'_2 of the sequence Δ'_2, as then $((K'_2 \cap {}^+K_1) \oplus I) \cap K_1 \cap J_i = \emptyset$. The elements in the sequence returned by the combine procedure might be empty, adjacent, or overlapping. The merge procedure removes

[3] In case $r(I) = \infty$, we actually store fewer intervals K_2 in Δ'_2 then those appearing in the equality (5.4). Details are provided when explaining the contents of Δ_ϕ.

empty elements and merges adjacent or overlapping intervals, i.e., it returns the sequence $\overline{\mathsf{iip}}^1(\gamma_{\phi_1 \mathsf{S}_I \phi_2} \cap J_i)$.

Finally, we explain the contents of the variables K_ϕ and Δ_ϕ and how they are updated. We start with K_ϕ. At the $(i+1)$st iteration, for some $i \geq 0$, the following invariant is satisfied by K_ϕ: before the update, the interval K_ϕ is the last interval of $\overline{\mathsf{iip}}^1(\gamma_{\phi_1} \cap {}^{\leq}J_{i-1})$ if $i > 0$ and this sequence is not empty, and K_ϕ is the empty set otherwise. The interval K_ϕ is prepended to the sequence $\overline{\mathsf{iip}}^1(\gamma_{\phi_1} \cap J_i)$ using the prepend procedure from Fig. 8, which merges K_ϕ with the first interval of $\Delta_1 = \overline{\mathsf{iip}}^1(\gamma_{\phi_1} \cap J_i)$ if these two intervals are adjacent. The obtained sequence Δ_1' is the maximal subsequence of $\overline{\mathsf{iip}}^1(\gamma_{\phi_1} \cap {}^{\leq}J_i)$ such that $K_1 \cap J_i \neq \emptyset$, for each interval K_1 in Δ_1'. Thus, after the update, K_ϕ is the last interval of $\overline{\mathsf{iip}}^1(\gamma_{\phi_1} \cap {}^{\leq}J_i)$ if this sequence is not empty, and K_ϕ is the empty set otherwise. Hence the invariant on K_ϕ is preserved at the next iteration.

The following invariant is satisfied by Δ_ϕ at the $(i+1)$st iteration: before the update, the sequence Δ_ϕ is empty if $i = 0$, and otherwise, if $i > 0$, it stores the intervals K_2 in $\overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap {}^{\leq}J_{i-1})$ with $(K_2 \oplus I) \cap (J_{i-1}^{>}) \neq \emptyset$ and $(K_2 \oplus I) \cap (J_{i-1}^{>}) \not\subseteq (K_2' \oplus I) \cap (J_{i-1}^{>})$, where K_2' is the successor of K_2 in $\overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap {}^{\leq}J_{i-1})$. The procedure concat concatenates the sequence Δ_ϕ with the sequence $\Delta_2 = \overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap J_i)$. Since the last interval of Δ_ϕ and the first interval of Δ_2 can be adjacent, concat may have to merge them. Thus, the obtained sequence Δ_2' is a subsequence of $\overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap {}^{\leq}J_i)$ such that $(K_2 \oplus I) \cap (J_i^{\geq}) \neq \emptyset$, for each element K_2. Note that $J_{i-1}^{>} = J_i^{\geq}$. The updated sequence Δ_ϕ is obtained from Δ_2' by removing the intervals K_2 with $(K_2 \oplus I) \cap (J_i^{>}) = \emptyset$, i.e., the intervals that are irrelevant for later iterations. The procedure drop from Fig. 8 removes these intervals. Moreover, if there are intervals K_2 and K_2' in Δ_ϕ with $(K_2 \oplus I) \cap (J_i^{>}) \subseteq (K_2' \oplus I) \cap (J_i^{>})$, then only the interval that occurs later is kept in Δ_ϕ. Note that this inclusion only holds when $r(I) = \infty$. Dropping such intervals K_2 allows us to avoid storing in Δ_ϕ all intervals in $\overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap {}^{\leq}J_i)$. This optimization is performed by the procedure drop′. Thus, after the update, the sequence Δ_ϕ stores the intervals K_2 in $\overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap {}^{\leq}J_i)$ with $(K_2 \oplus I) \cap (J_i^{>}) \neq \emptyset$ and $(K_2 \oplus I) \cap (J_i^{>}) \not\subseteq (K_2' \oplus I) \cap (J_i^{>})$, where K_2' is the successor of K_2 in $\overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap {}^{\leq}J_i)$. Hence the invariant on Δ_ϕ is preserved at the next iteration. Finally, we note that it is safe to drop elements K_2 from Δ_ϕ when $(K_2 \oplus I) \cap (J_i^{>}) \subseteq (K_2' \oplus I) \cap (J_i^{>})$, where K_2' is the successor of K_2 in Δ_ϕ. Indeed, assume that K_2 and K_2' satisfy the stated dropping conditions. We show that at iteration $i+1$, we have $((K_2 \cap^{+} K_1) \oplus I) \cap K_1 \cap J_{i+1} \subseteq ((K_2' \cap^{+} K_1) \oplus I) \cap K_1 \cap J_{i+1}$, for any interval $K_1 \in \overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap {}^{\leq}J_{i+1})$. Let τ be an element of the set on the left-hand side. In particular, $\tau \in (K_2 \oplus I) \cap J_{i+1}$. As $(K_2 \oplus I) \cap (J_i^{>}) \subseteq (K_2' \oplus I) \cap (J_i^{>})$, it follows that $\tau \in (K_2' \oplus I) \cap J_{i+1}$. It is then easy to see that τ is in the right-hand side set as well.

The following theorem states the algorithm's correctness.

Theorem 5.1 *Let ϕ be a formula, $\hat{\gamma} = (\gamma_p)_{p \in P}$ a family of signals, $\bar{J}$ an infinite interval partition, and $n > 0$. The procedure* $\mathsf{step}(\phi, \hat{\Delta}_{n-1}, J_{n-1})$ *terminates and returns the sequence* $\overline{\mathsf{iip}}^1(\gamma_\phi \cap J_{n-1})$*, whenever* $\mathsf{init}(\phi)$, $\mathsf{step}(\phi, \hat{\Delta}_0, J_0), \ldots, \mathsf{step}(\phi, \hat{\Delta}_{n-2}, J_{n-2})$ *were called previously in this order, where $\hat{\Delta}_i = (\Delta_{i,p})_{p \in P}$ with $\Delta_{i,p} = \overline{\mathsf{iip}}^1(\gamma_p \cap J_i)$, for $i < n$.*

As in the case of Theorem 4.1, the proof is an induction over tuples (i, ψ), with $0 \leq i < n$ and ψ subformula of ϕ. The property proved by induction is the theorem's statement strengthened with the invariants on the intervals K_ψ and sequences Δ_ψ stated in the the explanation just given. The proof follows the same structure and reasoning as in the explanation and is thus omitted.

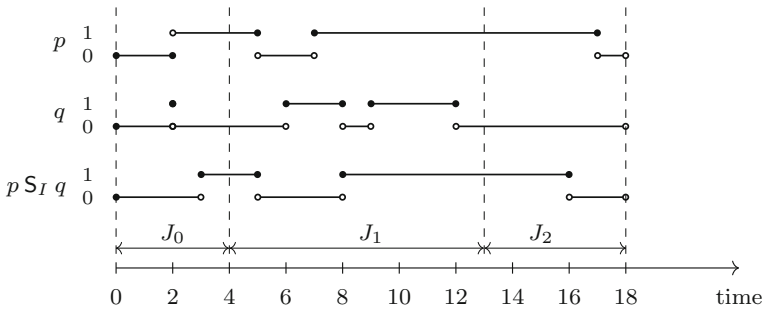

Fig. 9 The first three chunks of the signals γ_p, γ_q, and $\gamma_{p\mathsf{S}_I q}$, with $I = [1, 4]$

Table 2 Values of various variables at the end of the execution of the update procedure

i	J_{i-1}	Δ_1	Δ_2	Δ'_1	Δ'_2	K_ψ	Δ_ψ
0						$\emptyset$	$\langle\rangle$
1	$[0, 4)$	$\langle(2, 4)\rangle$	$\langle[2, 2]\rangle$	$\langle(2, 4)\rangle$	$\langle[2, 2]\rangle$	$(2, 4)$	$\langle[2, 2]\rangle$
2	$[4, 13)$	$\langle[4, 5], [7, 13)\rangle$	$\langle[6, 8], [9, 12]\rangle$	$\langle[1, 5], [7, 13)\rangle$	$[2, 2]::\Delta_2$	$\langle[7, 13)\rangle$	$\langle[9, 12]\rangle$
3	$[13, 18)$	$\langle[13, 16]\rangle$	$\langle\rangle$	$\langle[7, 18]\rangle$	$\langle[9, 12]\rangle$	$[7, 18]$	$\langle\rangle$

Table 3 Values of various expressions during the execution of the combine procedure, where e_i denotes the expression $((K_2 \cap {}^+K_1) \oplus I) \cap K_1 \cap J_{i-1}$

i	J_{i-1}	K_1	K_2	$(K_2 \oplus I) \cap J_{i-1}$	$K_2^{>} \cap {}^+K_1$	e_i
1	$[0, 4)$	$(2, 4)$	$[2, 2]$	$[3, 4)$	$(2, 4]$	$[3, 4)$
2	$[4, 13)$	$(2, 5]$	$[2, 2]$	$[4, 6]$	$(2, 5]$	$[4, 5]$
		$(2, 5]$	$[6, 8]$	$[7, 12]$	$\emptyset$	$\emptyset$
		$[7, 13)$	$[6, 8]$	$[7, 12]$	$(8, 13)$	$[8, 12]$
		$[7, 13)$	$[9, 12]$	$[10, 13)$	$(12, 13)$	$[10, 13)$
3	$[13, 18)$	$[7, 18)$	$[9, 12]$	$[10, 16]$	$(12, 18)$	$[13, 16]$

Example 5.1 We illustrate an execution of the interval-based monitoring algorithm using the same formula $\psi = p\ \mathsf{S}_I\ q$, with $I = [1, 4]$ and the same signals as in Example 4.1. For convenience, we depict them again in Fig. 9. Suppose that the first three chunks received by the algorithm are given by the intervals J_0, J_1, and J_2 as depicted in the figure. Table 2 gives the values of variables Δ_1, Δ_2, Δ'_1, Δ'_2, K_ψ, and Δ_ψ at the end of the execution of the update procedure. Table 3 gives the values of the expressions $(K_2 \oplus I) \cap J_{i-1}$, $K_2^{>} \cap {}^+K_1$, and $((K_2 \cap {}^+K_1) \oplus I) \cap K_1 \cap J_{i-1}$ during the execution of the combine procedure, allowing us to track how the signal γ_ψ is computed. We note that the signal γ_ψ over the first three chunks is the union of the intervals in the e_i column.

5.2 Monitoring complexity

We analyze the monitor's computational complexity. As in the point-based setting, we take the representation size of elements of the time domain $\mathbb{T}$ into account. The basic operations

here in which elements of $\mathbb{T}$ are involved are operations on intervals like checking emptiness (i.e., $I = \emptyset$), extension (e.g. $I^{>}$), and shifting (i.e. $I \oplus J$). The representation size of the resulting interval of the shifting operation $I \oplus J$ is in $\mathcal{O}(\|I\| + \|J\|)$. The time required to carry out the shift operation is in $\mathcal{O}(\max\{\|I\|, \|J\|\}^2)$. All the other basic operations that return an interval do not increase the representation size of the resulting interval with respect to the given intervals. However, the time complexity is quadratic in the representation size of the given intervals whenever the operation needs to compare interval margins.

5.2.1 Time complexity

The following theorem establishes an upper bound on the time complexity of our monitoring algorithm.

Theorem 5.2 *Let ϕ, $\hat{\gamma}$, $\bar{J}$, n, and $\hat{\Delta}_i$ be given as in* Theorem 5.1. *Executing the sequence* $\mathsf{init}(\phi)$, $\mathsf{step}(\phi, \hat{\Delta}_0, J_0), \ldots, \mathsf{step}(\phi, \hat{\Delta}_{n-1}, J_{n-1})$ *requires* $\mathcal{O}(m^2 \cdot (n + \delta \cdot |\phi|) \cdot |\phi|^3)$ *time, where* $m = \max\big(\big\{\|I\| \mid \alpha\ \mathsf{S}_I\ \beta \in \mathsf{sf}(\phi)\big\} \cup \big\{\|J_0\|, \ldots, \|J_{n-1}\|\big\} \cup \bigcup_{p \in P} \big\{\|K\| \mid K \in \overline{\mathsf{inp}}^1(\gamma_p \cap ({}^{<}J_n))\big\}\big)$ *and* $\delta = \sum_{p \in P} \|\gamma_p \cap ({}^{<}J_n)\|$.

We remark that the factor $m^2 \cdot |\phi|^2$ is due to the operations on the margins of intervals. Under the assumption that the representation of elements of the time domain is constant, we obtain the upper bound $\mathcal{O}((n + \delta \cdot |\phi|) \cdot |\phi|)$.

In the following, we prove Theorem 5.2. Let ψ be a subformula of ϕ. We denote by $t_n(\psi)$ the running time of the sequence $\mathsf{init}(\psi), \mathsf{step}(\psi, \hat{\Delta}_0, J_0), \ldots, \mathsf{step}(\psi, \hat{\Delta}_{n-1}, J_{n-1})$. We also define

$$m_\psi := \max\ \{\|I\| \mid \alpha\ \mathsf{S}_I\ \beta \in \mathsf{sf}(\psi)\} \cup \{\|J_0\|, l \ldots, \|J_{n-1}\|\} \cup \bigcup_{\psi' \in \mathsf{sf}(\psi)} \big\{\|K\| \mid K \in \overline{\mathsf{inp}}^1(\gamma_{\psi'} \cap {}^{<}J_n)\big\}.$$

That is, m_ψ is the maximal representation size of some interval that occurs in the first n iterations of the monitoring algorithm when determining the signal for ψ. By inspecting the operations performed on intervals in one iteration, we obtain

$$m_\psi \leq \begin{cases} m_{\psi'} & \text{if } \psi \text{ is of the form } \neg\psi', \\ \max\{m_{\psi_1}, m_{\psi_2}\} & \text{if } \psi \text{ is of the form } \psi_1 \wedge \psi_2, \\ \max\{m_{\psi_1}, m_{\psi_2}\} + \|I\| & \text{if } \psi \text{ is of the form } \psi_1\ \mathsf{S}_I\ \psi_2, \\ m & \text{otherwise.} \end{cases}$$

The inequality $m_\psi \leq m \cdot |\psi|$ follows by induction on the formula structure.

The following lemma, which follows from the equalities (5.2)–(5.4), establishes an upper bound on the number of intervals that are necessary for representing the signal determined by the formula ψ up to some point in time.

Lemma 5.2 *Let J be a bounded interval with $0 \in J$. If ψ is not a proposition then*

$$\|\gamma_\psi \cap J\| \in \mathcal{O}\left(\sum_{\psi' \in \mathsf{dsf}(\psi)} \|\gamma_{\psi'} \cap J\|\right).$$

The following lemma is the key ingredient for proving Theorem 5.2. It establishes an upper bound of the running time for the formula ψ excluding the running times for its proper subformulas.

Lemma 5.3 *If ψ is a proposition then $t_n(\psi) \in \mathcal{O}(n)$. If ψ is not a proposition then*

$$t_n(\psi) - \sum_{\psi' \in \mathsf{dsf}(\psi)} t_n(\psi') \in \mathcal{O}\left(m_\psi^2 \cdot \sum_{\psi' \in \mathsf{dsf}(\psi)} \sum_{i<n} \left(1 + \|\gamma_{\psi'} \cap J_i\|\right) \right).$$

Proof The upper bound when ψ is a proposition is obviously true, since in each iteration we just return Δ_p. In the following, we prove by case distinction the upper bound for the cases when ψ is not a proposition. Here, $|\Delta|$ denotes the length of a sequence Δ.

Case $\psi = \neg\psi'$. For $i < n$, the running time of $\mathsf{step}(\psi, \hat{\Delta}_i, J_i)$, excluding the running time of $\mathsf{step}(\psi', \hat{\Delta}_i, J_i)$, is the running time of $\mathsf{invert}(\Delta', J_i)$. The procedure invert visits sequentially the elements in the sequence Δ'. Each visit requires $\mathcal{O}(m_\psi^2)$ time, since the procedure cons checks whether an interval K is empty before appending it to the inverted sequence. From Theorem 5.1 we know that at the $(i+1)$st iteration, the sequence Δ' equals $\overline{\mathsf{iip}}^1(\gamma_{\psi'} \cap J_i)$. So, the length of Δ' is $|\Delta'| = \|\gamma_{\psi'} \cap J_i\|$. From this, the upper bound for $\neg\psi'$ follows.

Case $\psi = \psi_1 \wedge \psi_2$. Similar arguments as in the case where ψ equals $\neg\psi'$ establish the upper bound for $\psi = \psi_1 \wedge \psi_2$. Note that $\mathsf{intersect}(\Delta_1, \Delta_2)$ runs in time $\mathcal{O}(m_\psi^2 \cdot (1 + \|\gamma_{\psi_1} \cap J_i\| + \|\gamma_{\psi_2} \cap J_i\|))$.

Case $\psi = \psi_1 \; \mathsf{S}_I \; \psi_2$. We first inspect the running time of the $(i+1)$st iteration with $i < n$. The running time of $\mathsf{step}(\psi, \hat{\Delta}_i, J_i)$ without the running times for $\mathsf{step}(\psi_1, \hat{\Delta}_i, J_i)$ and $\mathsf{step}(\psi_2, \hat{\Delta}_i, J_i)$ is the sum of the running times of $\mathsf{prepend}(K_\psi, \Delta_1)$, $\mathsf{concat}(\Delta_\psi, \Delta_2)$, $\mathsf{last}(\Delta_1')$, $\mathsf{drop}(\Delta_2', I, J_i)$, which are called by the procedure update, and of $\mathsf{merge}(\mathsf{combine}(\Delta_1', \Delta_2', I, J_i))$.

The procedures prepend, concat, and last run in time at most $\mathcal{O}(m_\psi^2)$. So, their total running time for all n iterations is in $\mathcal{O}(m_\psi^2 \cdot n)$. The procedures merge, combine, and drop, including the call to drop′, sequentially visit elements of an input sequence. Each such visit requires $\mathcal{O}(m_\psi^2)$ time. Whereas the procedure merge visits all elements, the procedures combine, drop, and drop′ might stop before reaching the input sequence's end.

Before analyzing the procedures drop, combine, and merge, we make the following remarks. Consider the procedure update. We have $|\Delta_1'| \leq 1 + |\Delta_1|$ and $|\Delta_2'| \leq |\Delta_\psi| + |\Delta_2|$, where $|\Delta_\psi|$ refers to the number of elements in Δ_ψ before calling the procedure drop. Moreover, from Theorem 5.1, $|\Delta_1'| \in \mathcal{O}(\|\gamma_{\psi_1} \cap J_i\|)$ and $\Delta_2' \subseteq \overline{\mathsf{iip}}^1(\gamma_{\psi_2} \cap {}^{\leq}J_i)$.

We first consider the call to $\mathsf{drop}(\Delta_2', I, J_i)$. The drop procedure only visits a prefix of the sequence Δ_2', and returns the last visited element appended to the unvisited suffix of Δ_2'. Thus the running time of drop is in $\mathcal{O}(m_\psi^2 \cdot (1 + |\Delta_{\mathit{diff}}|))$, where Δ_ψ' is the value of Δ_ψ after the call to drop and Δ_{diff} is such that $\Delta_2' = \Delta_{\mathit{diff}} +\!\!+ \Delta_\phi'$. In the next iteration, the input of drop will be a subsequence of $\overline{\mathsf{iip}}^1(\gamma_{\psi_2} \cap {}^{\leq}J_{i+1})$ having Δ_ψ' as a prefix. It follows that at most one element in $\overline{\mathsf{iip}}^1(\gamma_{\psi_2} \cap {}^{<}J_n)$ is visited by any two consecutive calls to the drop procedure. Thus the total running time of the drop procedure during the first n iterations is in $\mathcal{O}(m_\psi^2 \cdot (n + \|\gamma_{\psi_2} \cap {}^{<}J_n\|))$. Note that ${}^{<}J_n = {}^{\leq}J_{n-1}$.

We now consider the running time of $\mathsf{combine}(\Delta_1', \Delta_2', I, J_i)$. The combine procedure traverses the sequences Δ_1' and Δ_2', and, similarly to the procedure intersect, it runs in $\mathcal{O}(m_\psi^2 \cdot (1 + |\Delta_1'| + |\Delta_2'|))$. This upper bound does not suffice to obtain the desired upper bound on the total running time, as we do not have that $|\Delta_2'| \in \mathcal{O}(\|\gamma_{\psi_2} \cap J_i\|)$. Nevertheless, we prove below that, while the sequences Δ_2' in two consecutive iterations may have more than one interval in common, at most one such interval is visited by both iterations. This shows that in the first n iterations at most $n + \|\gamma_{\psi_2} \cap {}^{<}J_n\|$ intervals from the sequences Δ_2' are visited.

To show that at most one interval is visited by the combine procedure during each of the $(i+1)$st and $(i+2)$nd iterations, we recall that in the $(i+1)$st iteration, we have $\Delta'_2 = \Delta_{diff} +\!\!+ \Delta'_\psi$, while in the $(i+2)$nd iteration the sequence Δ'_ψ is a prefix of Δ'_2. Hence it is sufficient to show that at most one interval among the ones that combine visits in the $(i+1)$st iteration belongs to Δ'_ψ. Let $used(\Delta'_2) := \{K_2 \in \Delta'_2 \mid (K_2 \oplus I) \cap J_i \neq \emptyset\}$. Note that there is at most one visited element in Δ'_2 that is not in $used(\Delta'_2)$, as in the $(i+1)$st iteration, the combine procedure stops whenever $(K_2 \oplus I) \cap J_i = \emptyset$ and thus no elements following K_2 in Δ'_2 are visited. We show that there is at most one interval in both Δ'_ψ and $used(\Delta'_2)$. Suppose by absurdity that there are two such intervals. Then there are two consecutive such intervals, K_2 and K'_2. By the invariant on Δ'_ψ enforced by the procedure drop′, we have $(K_2 \oplus I) \cap (J_i^{>}) \not\subseteq (K'_2 \oplus I) \cap (J_i^{>})$. Then $K'_2 \oplus I \subseteq J_i^{>}$, and thus $(K'_2 \oplus I) \cap J_i = \emptyset$, which is a contradiction with $K'_2 \in used(\Delta'_2)$.

Finally, the running time of the procedure merge is linear in the length of its argument, that is, the length of the result of combine, which in turn is at most $|\Delta'_1| + |\Delta'_2|$. From the above discussion, it follows that the total running time of merge for the n iterations is in $\mathcal{O}(m_\psi^2 \cdot (n + \|\gamma_{\psi_2} \cap {}^{<}J_n\| + \sum_{i<n}(1 + \|\gamma_{\psi_1} \cap J_i\|)))$. As $\|\gamma_{\psi_2} \cap {}^{<}J_n\| \leq \sum_{i<n} \|\gamma_{\psi_2} \cap J_i\|$, this concludes the proof of the upper bound, where ψ is of the form $\psi_1 \,\mathsf{S}_I\, \psi_2$. □

We now put these ingredients together to establish the upper bound of Theorem 5.2. By Lemma 5.3,

$$t_n(\psi) - \sum_{\psi' \in \mathsf{dsf}(\psi)} t_n\left(\psi'\right) \in \mathcal{O}\left(m_\psi^2 \cdot \sum_{\psi' \in \mathsf{dsf}(\psi)} \sum_{i<n} \left(1 + \|\gamma_{\psi'} \cap J_i\|\right)\right).$$

We obtain

$$t_n(\psi) - \sum_{\psi' \in \mathsf{dsf}(\psi)} t_n\left(\psi'\right) \in \mathcal{O}\left(m_\psi^2 \cdot \sum_{\psi' \in \mathsf{dsf}(\psi)} \left(n + \|\gamma_{\psi'} \cap {}^{<}J_n\|\right)\right),$$

since $\sum_{i<n}(1 + \|\gamma \cap J_i\|) \in \mathcal{O}(n + \|\gamma \cap {}^{<}J_n\|)$, for any signal $\gamma \subseteq \mathbb{T}$. Since ψ has at most two direct subformulas, we obtain

$$t_n(\psi) - \sum_{\psi' \in \mathsf{dsf}(\psi)} t_n\left(\psi'\right) \in \mathcal{O}\left(m_\psi^2 \cdot \left(n + \sum_{\psi' \in \mathsf{dsf}(\psi)} \|\gamma_{\psi'} \cap {}^{<}J_n\|\right)\right).$$

By Lemma 5.2,

$$t_n(\psi) - \sum_{\psi' \in \mathsf{dsf}(\psi)} t_n\left(\psi'\right) \in \mathcal{O}\left(m_\psi^2 \cdot \left(n + |\psi| \cdot \sum_{p \in P} \|\gamma_p \cap {}^{<}J_n\|\right)\right),$$

that is, $t_n(\psi) - \sum_{\psi' \in \mathsf{dsf}(\psi)} t_n(\psi') \in \mathcal{O}\left(m_\psi^2 \cdot (n + |\psi| \cdot \delta)\right)$. Using the inequalities $m_\psi \leq m \cdot |\psi|$ and $|\psi| \leq |\phi|$, we obtain

$$t_n(\psi) - \sum_{\psi' \in \mathsf{dsf}(\psi)} t_n\left(\psi'\right) \in \mathcal{O}\left(m^2 \cdot |\phi|^2 \cdot (n + |\phi| \cdot \delta)\right).$$

Summing up these equations for each subformula ψ of ϕ, we have

$$t_n(\phi) \in \mathcal{O}\left(n \cdot |\phi| + m^2 \cdot |\phi|^2 \cdot (n + |\phi| \cdot \delta) \cdot |\phi|\right),$$

which simplifies to $t_n(\phi) \in \mathcal{O}(m^2 \cdot (n + |\phi| \cdot \delta) \cdot |\phi|^3)$.

5.2.2 Space complexity

We now consider the algorithm's space complexity. Similarly to the point-based setting, the space needed by the algorithm is in general unbounded for two reasons: (1) the size of the representation of intervals grows over time, and (2) the length of the sequences Δ_ψ can be arbitrarily large in general. To illustrate (2), consider the formula $\phi = \blacklozenge_{[0,1]}\, p$, the interval partition $\bar{J}$ with $J_i = [i, i+1)$ for any $i \in \mathbb{N}$, the time sequence $\bar{\tau}$ with $\tau_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n+1}$ for $n \geq 0$, and the input signal $\gamma_p(\tau_n) = 1$, for each $n \in \mathbb{N}$, and $\gamma_p(\tau) = 0$ for each τ that is not an element of the time sequence $\bar{\tau}$. It is easy to see that the number of elements in Δ_ϕ increases at each iteration and is not bounded. However, in practice, signals have a bounded variability, that is, there is a bound on the number of signal changes in any fixed period of time. Formally, we say that a signal γ is *bound in variability by* k, in short *k-bounded*, if $\|\gamma \cap [\tau, \tau+1)\| \leq k$, for any $\tau \in \mathbb{T}$.

The following theorem establishes the space complexity of our monitoring algorithm under the assumption that the input signals are k-bounded. In particular, under this assumption the length of the sequences Δ_ψ is bounded. We note that the space usage is independent of the number n of chunks and from the size of the input signals, namely $\delta = \sum_{p \in P} \|\gamma_p \cap ({}^{<}J_n)\|$.

Theorem 5.3 *Let ϕ, $\hat{\gamma}$, $\bar{J}$, n, and $\hat{\Delta}_i$ be given as in* Theorem 5.1*, and such that each signal in $\hat{\gamma}$ is k-bounded, for some $k \in \mathbb{N}$ with $k \geq 1$. Executing the sequence* $\mathsf{init}(\phi)$, $\mathsf{step}(\phi, \hat{\Delta}_0, J_0), \ldots, \mathsf{step}(\phi, \hat{\Delta}_{n-1}, J_{n-1})$ *requires $\mathcal{O}(m \cdot (s + \ell_\phi \cdot |\phi|) \cdot k \cdot |\phi|^2)$ space, where $s = \max\left\{r(J_i) - \ell(J_i)) \mid 0 \leq i < n\right\}$, $\ell_\phi = \max\left\{\ell(I) \mid \alpha\, \mathsf{S}_I\, \beta \in \mathsf{sf}(\phi)\right\}$, and $m = \max\left(\left\{\|I\| \mid \alpha\, \mathsf{S}_I\, \beta \in \mathsf{sf}(\phi)\right\} \cup \left\{\|J_0\|, \ldots, \|J_{n-1}\|\right\} \cup \bigcup_{p \in P} \left\{\|K\| \mid K \in \overline{\mathsf{iip}}^1(\gamma_p \cap ({}^{<}J_n))\right\}\right)$.*

We remark that the factor $m \cdot |\phi|$ is due to the space required to represent intervals. Under the assumption that the representation of elements of the time domain is constant, we obtain the upper bound $\mathcal{O}((s + \ell_\phi \cdot |\phi|) \cdot k \cdot |\phi|)$.

The remainder of this section is dedicated to proving Theorem 5.3. We start with the following key lemma.

Lemma 5.4 *Let ϕ be a formula, $k \in \mathbb{N}$, with $k \geq 1$, and $\hat{\gamma}$ a family of k-bounded signals. The signal γ_ϕ is $(k \cdot |\phi|)$-bounded.*

Proof We reason by structural induction on ϕ. The base case where ϕ is an atomic proposition is trivial. For the case $\phi = \neg\psi$, we have $\|\gamma_\phi \cap [\tau, \tau+1)\| \leq 1 + \|\gamma_\psi \cap [\tau, \tau+1)\| \leq 1 + k \cdot |\psi| \leq k \cdot (1 + |\psi|) = k \cdot |\phi|$, for any $\tau \in \mathbb{T}$. The first inequality is straightforward, as $|\overline{\mathsf{iip}}^u(\gamma \cap J)| \leq 1 + |\overline{\mathsf{iip}}^v(\gamma \cap J)$, for any signal γ, any bounded interval J, and any $u, v \in \{0, 1\}$ with $u + v = 1$. The second inequality follows from the induction hypothesis. The case $\phi = \phi_1 \wedge \phi_2$ follows similarly from the inequality $|\overline{\mathsf{iip}}^1(\gamma_\phi \cap J)| \leq |\overline{\mathsf{iip}}^1(\gamma_{\phi_1} \cap J)| + |\overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap J)|$.

The remaining case is where $\phi = \phi_1\, \mathsf{S}_I\, \phi_2$. From the induction hypothesis, γ_{ϕ_1} is $(k \cdot |\phi_1|)$-bounded and γ_{ϕ_2} is $(k \cdot |\phi_2|)$-bounded. Informally, γ_ϕ is $(k \cdot |\phi|)$-bounded because any change from low to high in the signal $\gamma_\phi \cap [\tau, \tau+1)$ corresponds to such a change in the signal $\gamma_{\phi_1} \cap [\tau, \tau+1)$ or in the signal $\gamma_{\phi_2} \cap [\tau - \ell(I), \tau + 1 - \ell(I))$. We formalize this observation and use it to establish the upper bound in this case. Note first that the equality (5.4) also holds when J_i is replaced by $[\tau, \tau+1)$, as J_i is an arbitrary bounded interval. Thus, for any

interval K in $\overline{\mathsf{iip}}^1(\gamma_{\phi_1 \mathsf{S}_I \phi_2} \cap [\tau, \tau+1))$, there are intervals $K_1 \in \overline{\mathsf{iip}}^1(\gamma_{\phi_1} \cap [0, \tau+1))$ and $K_2 \in \overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap [0, \tau+1))$ such that $\ell(K) = \ell(((K_2 \cap {}^+K_1) \oplus I) \cap K_1 \cap [\tau, \tau+1))$. From this it follows that $\ell(K) = \tau$, $\ell(K) = \ell(K_2) + \ell(I)$, or $\ell(K) = \ell(K_1) + \ell(I)$. Moreover, the interval K_1 or K_2 for which one of the previous two equalities holds is unique. Therefore, there is at most one interval K such that $\ell(K) = \tau$. Also, if $\ell(K) = \ell(K_1) + \ell(I)$, then $K_1 \cap [\tau, \tau+1) \neq \emptyset$ as otherwise $K = \emptyset$, and if $\ell(K) = \ell(K_2) + \ell(I)$ then $K_2 \cap [\tau - \ell(I), \tau + 1 - \ell(I)) \neq \emptyset$. We have shown the stated observation. It follows that the number of intervals K in $\overline{\mathsf{iip}}^1(\gamma_{\phi_1 \mathsf{S}_I \phi_2} \cap [\tau, \tau+1))$ is upper-bounded by one plus the number of intervals K_1 in $\overline{\mathsf{iip}}^1(\gamma_{\phi_1} \cap [\tau, \tau+1))$ and intervals K_2 in $\overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap [\tau - \ell(I), \tau + 1 - \ell(I)))$. We have that $|\overline{\mathsf{iip}}^1(\gamma_{\phi_1 \mathsf{S}_I \phi_2} \cap [\tau, \tau+1))| \leq 1 + k \cdot |\phi_1| + k \cdot |\phi_2| \leq k \cdot |\phi|$. □

We now turn to the algorithm's space complexity. The space used by the algorithm is dominated by the space required to store the sequences $\overline{\mathsf{iip}}^1(\gamma_\psi \cap J_i)$, for $0 \leq i < n$ and $\psi \in \mathsf{sf}(\phi)$, and the sequences Δ_ψ, for $\psi \in \mathsf{tsf}(\phi)$. We bound the storage space for the two types of sequences.

We first show that the space needed to store the sequences $\overline{\mathsf{iip}}^1(\gamma_\psi \cap J_i)$ at the $(i+1)$st iteration is in $\mathcal{O}(m \cdot |\phi| \cdot (r(J_i) - \ell(J_i)) \cdot k \cdot |\phi|)$. This follows easily by induction on the formula structure from the bounded-variability assumption, and in particular from Lemma 5.4, and from the proof of Theorem 5.2. The factor $m \cdot |\phi|$ represents the upper bound on representation size of the intervals in the sequences $\overline{\mathsf{iip}}^1(\gamma_\psi \cap J_i)$, while the remaining factor represents an upper bound on the length of these sequences. Note too that the sequences $\overline{\mathsf{iip}}^1(\gamma_\psi \cap J_i)$ are not stored between iterations, but only during iterations. It is thus sufficient that, for each $\psi \in \mathsf{sf}(\phi)$, we take the maximum, and not the sum, over their storage space. We thus obtain that the space usage is in $\mathcal{O}(m \cdot |\phi| \cdot \max_{0 \leq i < n}(r(J_i) - \ell(J_i)) \cdot k \cdot |\phi|)$, that is, in $\mathcal{O}(m \cdot |\phi| \cdot s \cdot k \cdot |\phi|)$, over all n iterations.

In contrast to the case of the sequences $\overline{\mathsf{iip}}^1(\gamma_\psi \cap J_i)$, the sequences Δ_ψ are stored between iterations, and thus we must consider the sum of their storage space. We bound next the length of these sequences. Recall that, for $\psi = \psi_1 \, \mathsf{S}_I \, \psi_2$ and the $(i+1)$st iteration with $i > 0$, the sequence Δ_ψ stores (before its update) intervals K_2 in $\overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap {}^{\leq}J_{i-1})$ with $(K_2 \oplus I) \cap (J_{i-1}^{>}) \neq \emptyset$ and $(K_2 \oplus I) \cap (J_{i-1}^{>}) \not\subseteq (K_2' \oplus I) \cap (J_{i-1}^{>})$, where K_2' is the successor of K_2 in $\overline{\mathsf{iip}}^1(\gamma_{\phi_2} \cap {}^{\leq}J_{i-1})$. We show that the second condition implies that $r(K_2') \geq \ell(J_{i-1}) - \ell(I)$, where K_2' is the second interval in Δ_ψ. Assume, by absurdity, that $r(K_2') < \ell(J_{i-1}) - \ell(I)$. Then $\ell(K_2) < \ell(K_2') < \ell(J_{i-1}) - \ell(I)$, that is, $\ell(K_2 \oplus I) < \ell(K_2' \oplus I) < \ell(J_{i-1})$. And as $\ell(J_{i-1}) \leq r(K_2 \oplus I) \leq r(K_2' \oplus I)$, we have that $(K_2 \oplus I) \cap (J_{i-1}^{>}) \subseteq (K_2' \oplus I) \cap (J_{i-1}^{>})$, which is a contradiction. Thus, for each interval K_2 in Δ_ψ except the first one, $r(K_2) \geq \ell(J_{i-1}) - \ell(I)$. (Note that from the first condition it only follows that $r(K_2) \geq \ell(J_{i-1}) - r(I)$, which is always true when $r(I) = \infty$.) Hence, the number of intervals K_2 stored in Δ_ψ is upper bounded by $1 + |\overline{\mathsf{iip}}^1(\gamma_{\psi_2} \cap [\ell(J_{i-1}) - \ell(I), \ell(J_{i-1})])|$, and is thus, by Lemma 5.4, upper bounded by $1 + \ell(I) \cdot k \cdot |\psi_2|$. Therefore the space needed to store the sequences Δ_ψ is in $\mathcal{O}(m \cdot |\phi| \cdot \ell_\phi \cdot k \cdot |\phi|^2)$.

We have shown that the algorithm uses $\mathcal{O}(m \cdot |\phi| \cdot s \cdot k \cdot |\phi|)$ space to store the sequences $\overline{\mathsf{iip}}^1(\gamma_\psi \cap J_i)$ at iteration $(i+1)$st, for $\psi \in \mathsf{sf}(\phi)$ and $0 \leq i < n$, and $\mathcal{O}(m \cdot |\phi| \cdot \ell_\phi \cdot k \cdot |\phi|^2)$ space to store the Δ_ψ sequences, for $\psi \in \mathsf{tsf}(\phi)$. Summing up these upper bounds gives the stated upper bound.

6 Comparison and discussion

6.1 Time complexity

In the following, we compare the time complexity of our two algorithms when monitoring a strongly event-relativized formula ϕ. By Theorem 3.1, the point-based setting and the interval-based setting coincide on this formula class.

First note that the input for the $(i+1)$st iteration of the point-based monitoring algorithm can be easily obtained online from the given signals $\hat{\gamma} = (\gamma)_{p \in S \cup E}$. Whenever an event occurs, we record the time $\tau_i \in \mathbb{T}$, determine the current truth values of the propositions, i.e., $\Gamma_i = \{p \in P \mid \tau_i \in \gamma_p\}$, and invoke the monitor by executing $\mathsf{step}^\bullet(\phi, \Gamma_i, \tau_i)$. The worst-case time complexity of the point-based monitoring algorithm for the first $\dot{n}$ iterations is $\mathcal{O}(\dot{m}^2 \cdot \dot{n} \cdot |\phi|)$, where $\dot{m}$ is according to Theorem 4.2. Note that we use the dot superscript to distinguish parameters associated with the point-based monitoring algorithm from those for the interval-based monitoring algorithm discussed next.

When using the interval-based monitoring algorithm, we are more flexible in that we need not invoke the monitoring algorithm whenever an event occurs. Instead, we can freely split the signals into chunks. Let $\bar{J}$ be a splitting in which the nth interval J_{n-1} is right-closed and $r(J_{n-1}) = \tau_{\dot{n}-1}$. We have the worst-case time complexity of $\mathcal{O}(m^2 \cdot (n + \delta \cdot |\phi|) \cdot |\phi|^3)$, where m and δ are according to Theorem 5.2. We can reduce this upper bound to $\mathcal{O}(m^2 \cdot (n + \delta \cdot |\phi|) \cdot |\phi|)$, since the formula ϕ is strongly event-relativized. Instead of the factor $m^2 \cdot |\phi|^2$ for processing the interval margins in the n iterations, we only have the factor m^2. The reason is that the margins of the intervals in the signal chunks of subformulas of the form $\psi_1 \, \mathsf{S}_I \, \psi_2$ already appear as interval margins in the input. Note that $m \geq \dot{m}$ and that δ is independent of n.

The upper bound of the interval-based monitoring algorithm depends on the number n of how often the procedure step is invoked. The case where $n = 1$ corresponds to the scenario where we use the monitoring algorithm offline (up to time $\tau_{\dot{n}-1}$). Note that for the point-based algorithm, it is irrelevant whether we use it offline or online. To mimic the point-based monitoring algorithm with the interval-based monitoring algorithm, we invoke the procedure step whenever an event occurs. In this case $n = \dot{n}$. An alternative is to invoke the procedure step every d time units, e.g., every 5 s. In the latter case, $n = \lceil \tau_{\dot{n}-1}/d \rceil$.

Even when using the interval-based monitoring algorithm offline and assuming a fixed-size representation of the elements in $\mathbb{T}$, the upper bounds differ by the factors $\dot{n}$ and $\delta \cdot |\phi|$. Since $\delta \geq \dot{n}$, the upper bound of the point-based monitoring algorithm is lower. In fact, the following examples show that the gap between the running times matches our upper bounds and that $\delta \cdot |\phi|$ can be significantly larger than $\dot{n}$. We assume that timestamps and interval margins have a fixed size representation, that is, they are represented in constant space.

Our first example shows that δ can be significantly larger than $\dot{n}$. It also illustrates that the point-based algorithm ignores large portions of the state signals, while the interval-based algorithm processes the entire state signals.

Example 6.1 Let ϕ be the formula $e \wedge s$, where e is an event proposition and s is a state proposition. Note that ϕ is a strongly event-relativized. Let γ_e be the event signal $\mathbb{N}$ and γ_s be the state signal $\cup_{i \in \mathbb{N}} [2i \cdot \frac{n}{2k}, (2i+1) \cdot \frac{n}{2k})$, for some $n, k \in \mathbb{N}$ with $n, k > 0$. The running time of the first n iterations of the point-based algorithm is clearly in $\Theta(n)$. The running time of the first iteration of the interval-based algorithm on the chunk $\gamma_s \cap [0, n]$ is in $\Theta(k)$ since the procedure $\mathsf{intersect}$ traverses the entire sequence $\overline{\mathsf{iip}}^1(\gamma_s \cap [0, n])$, which contains k intervals. The choices of n and k are independent of each other. When choosing k significantly larger than n, the number of intervals in $\overline{\mathsf{iip}}^1(\gamma_s \cap [0, n])$ dominates the running

time of the interval-based algorithm, and not the number of events, which in turn determines the running time of the point-based algorithm.

Our second example shows that $\Omega(|\phi|^2)$ is a lower bound for the worst-case running time of the interval-based algorithm, even when the proposition set P is a singleton. It again illustrates that for event-relativized formulas, the interval-based algorithm processes, for each subformula, intermediate signal chunks that contain information also on what happens when events do not occur. In contrast, this information is (soundly) ignored by the point-based algorithm.

Example 6.2 Let p be an event proposition. We define the event-relativized formulas $\phi_1 := \blacklozenge_{[1,1]}\, p$, and $\phi_i := \phi_{i-1} \vee \blacklozenge_{[i,i]}\, p$, for $i \in \mathbb{N}$ with $i > 0$. In the following, let $k \in \mathbb{N}$ with $k > 0$ and let ϕ be the formula $p \wedge \phi_k$. Note that ϕ has the form $p \wedge ((\blacklozenge_{[1,1]}\, p) \vee (\blacklozenge_{[2,2]}\, p) \vee \cdots \vee (\blacklozenge_{[k,k]}\, p))$ and that $|\phi| \in \Theta(k)$, ignoring the representation sizes of the intervals attached to the temporal operators. Consider the event signal $\gamma_p = [0, 0]$. We have that $\gamma_\phi = \emptyset$ and $\gamma_{\phi_i} = [1, 1] \cup [2, 2] \cup \cdots \cup [i, i]$, for $i \in \mathbb{N}$ with $1 \leq i \leq k$. Let $\bar{J}$ be an interval partition with $r(J_0) > k$. We have $\|\overline{\mathsf{iip}}^1(\gamma_p \cap J_0)\| = 1$, $\|\overline{\mathsf{iip}}^1(\gamma_{\phi_i} \cap J_0)\| = i$, and $\|\overline{\mathsf{iip}}^1(\gamma_{\neg\phi_i} \cap J_0)\| = i + 1$, for each $i \in \mathbb{N}$ with $1 \leq i \leq k$.

The running time of the first iteration of the interval-based algorithm is in $\Theta(k^2)$. It is thus in $\Theta(|\phi|^2)$, as for each subformula ϕ_i (which is $\neg(\neg\phi_{i-1} \wedge \neg \blacklozenge_{[i,i]}\, p)$ when removing the syntactic sugar for the Boolean connective $\vee$) the procedures invert and intersect traverse $\Theta(i)$ intervals, for $1 < i \leq k$. The running time of the point-based algorithm on the signal chunk given by J_0 is in $\Theta(|\phi|)$ since there is only one event, namely, the one that occurs at time 0.

Note that considering disjunction as a primitive and adjusting the implementation accordingly would not change the bound $\Theta(|\phi|^2)$. Finally, we remark that singular intervals attached to temporal operators do not play an essential role in this example: we obtain the same running times when replacing the intervals $[i, i]$ with intervals $[i - \epsilon, i + \epsilon]$, where $\epsilon \in \mathbb{T}$ is sufficiently small.

6.2 Time domains

In the following, we discuss different options for the time domain $\mathbb{T}$ and their impact for the monitoring algorithms we presented. The choice between a dense or a discrete time domain for MTL's semantics, i.e., selecting $\mathbb{T}$ as $\mathbb{N}$ instead of $\mathbb{Q}_{\geq 0}$, has only a minor impact on the monitoring algorithms. The point-based monitoring algorithm can be used without any modifications for the discrete time domain and the interval-based algorithm requires only minor modifications.

For the interval-based algorithm, the operator ${}^+K$ used in the equality (5.4) must be redefined. In a discrete time domain, we extend K by one point in time to the left if it exists, i.e., ${}^+K := K \cup \{k-1 \mid k \in K \text{ and } k > 0\}$. The algorithms' correctness for the time domain $\mathbb{N}$ is similarly shown as in the Theorems 4.1 and 5.1, respectively. Note that the reasoning performed in both algorithms concerning the time domain elements encountered reduces to subtraction and comparison (with 0) of elements in $\mathbb{T}$ and is independent of whether $\mathbb{T}$ is dense or discrete.

In terms of complexity, if we assume a discrete and unbounded time domain, we still cannot assume that the operations on elements from the time domain can be carried out in constant time. But multiplication is no longer needed to compare these elements and thus the operations can be carried out in time linear in the representation size. The worst-case complexity of both algorithms improves accordingly. In practice, however, it is often reasonable to assume that

the time domain elements have a bounded representation, since arbitrarily precise clocks do not exist and a fixed number of bits suffices to encode all timestamps that the monitor will actually encounter. For example, for many applications it suffices to represent timestamps as Unix time, i.e., as 32 or 64 bit integers. The operations performed by our monitoring algorithms on the time domain elements would then be carried out in constant time.

Since clocks have a limited precision, a so-called fictitious-clock semantics [2,29] is often used to reason about the time associated with events; this represents another variant for the time model, which is orthogonal to whether one assumes a dense or a discrete time domain. This semantics formalizes, for example, that if the system event e happens strictly before the event e', but both events fall between two clock ticks, then we can distinguish them by their temporal ordering, but not by time. In a fictitious-clock semantics, we timestamp e and e' with the same clock value and in a trace e appears strictly before e'. To order these two events in a trace, signals must be synchronized. Note that the timestamps from logical clocks [20] only partially order events in general and this might be insufficient to obtain such a total "happens-before" relation. Our point-based monitoring algorithm can be directly used for a fictitious-clock semantics. It iteratively processes a sequence of snapshots $\langle \Gamma_0, \Gamma_1, \ldots \rangle$ together with a sequence of timestamps $\langle \tau_0, \tau_1, \ldots \rangle$, which is increasing but no longer necessarily strictly increasing. In contrast, our interval-based monitoring algorithm does not carry over to a fictitious-clock semantics. In fact, the interval-based semantics assumes that events with equal timestamps $\tau \in \mathbb{T}$ happen simultaneously at time τ.

In the remainder of this section, we revisit and further substantiate Remark 3.1 (page 7) when assuming the discrete time domain $\mathbb{N}$. Let ϕ be a non-event-relativized formula, $\hat{\gamma}$ be a family of event signals, and $\bar{\tau}$ be a time sequence. We cannot apply the analogue of Theorem 3.1 for the discrete time domain, as ϕ is not event-relativized. However, for the time sequence $\bar{\kappa}$ with $\kappa_i = i$ for each $i \in \mathbb{N}$, the point-based semantics and the interval-based semantics coincide. Thus, instead of using the point-based semantics on $\bar{\tau}$, which may not represent ϕ's intended meaning, we can use the point-based semantics on $\bar{\kappa}$, which more naturally represents ϕ's meaning. Indeed, the two examples given in Example 3.2 illustrate that the interval-based semantics is more natural than the point-based semantics, independent of the time domain. However, there is a trade-off between the use of (one of) the more natural semantics and the efficiency of the algorithm chosen for monitoring. We illustrate next this efficiency issue.

We have three options for monitoring $\hat{\gamma}$ with respect to ϕ: (1) using the point-based algorithm over $\bar{\tau}$, (2) using it over $\bar{\kappa}$, or (3) using the interval-based algorithm. Recall from the previous paragraph that option (1) may give less intuitive outputs than options (2) and (3), which give the same outputs. Consider the nth sample point in $\bar{\tau}$, for some $n > 0$. For simplicity, we ignore the representation size of elements of $\mathbb{T}$. Furthermore, we assume that the sample points in $\bar{\tau}$ are those time points when an event occurs, and that the interval-based algorithm is called on the chunks $[0, \tau_0]$ and $[\tau_{i-1} + 1, \tau_i]$, for any $i \geq 1$. We thus have that δ is in $\mathcal{O}(n)$, where δ is according to Theorem 5.2. The three options then have the following upper bounds on their running time: (1) $\mathcal{O}(n \cdot |\phi|)$, (2) $\mathcal{O}(\tau_n \cdot |\phi|)$, and (3) $\mathcal{O}(n \cdot |\phi|^2)$. Note that τ_n can be significantly larger than n. Indeed, assuming that events occur on average every millisecond and that the time unit is one nanosecond, $\tau_n \approx 10^6 n$. Thus, while option (2) is the most attractive choice, as it uses the more natural semantics and the conceptually simpler algorithm, it is not clear that it is actually better than (3) in terms of efficiency.

7 Related work

The monitoring algorithms most closely related to those presented in this article are the ones by Basin et al. [5,6], Thati and Roşu [33], Nickovic and Maler [23,25,26], Baldor and Niu [3], and Ho et al. [17]. They are related as follows to the work presented in this article.

The monitoring algorithm of Basin et al. [5,6] given for the future-bounded fragment of metric first-order temporal logic, which subsumes the past-only fragment of MTL, can be seen as an extension of the presented monitoring algorithm for the point-based setting to future temporal operators and the first-order case. We restricted ourselves here to the propositional case and to the past-only fragment of MTL in order to compare the effect of different time models on monitoring.

Thati and Roşu [33] provide a monitoring algorithm for metric temporal logic with a point-based semantics, which uses formula rewriting. Their algorithm is more general than ours for the point-based setting since it handles temporal past and future operators. Their complexity analysis is based on the assumption that operations involving elements from the time domain can be carried out in constant time. The worst-case complexity of their algorithm on the past-only fragment is worse than ours, since rewriting a formula can generate additional formulas. In particular, their algorithm is not linear in the number of subformulas.

Nickovic and Maler's [23,25,26] monitoring algorithms are for the interval-based setting and have ingredients similar to our algorithm for this setting. These ingredients were first presented by Nickovic and Maler for an offline version of their monitoring algorithms [22] for the fragment of interval metric temporal logic with bounded temporal future operators. Their setting is more general in that their signals are continuous functions and not Boolean values for each point in time. Moreover, their algorithms also handle bounded [26] and unbounded [25] temporal future operators by delaying the evaluation of subformulas. The algorithm in [25] slightly differs from the one in [26]; namely, the algorithm in [25] also handles temporal past operators and before initiating monitoring, it rewrites the given formula to eliminate the temporal operators until and since with timing constraints. However, Maler and Nickovic do not provide algorithmic details for handling the Boolean connectives and the temporal operators. In fact, the worst-case complexity, which is only stated for their offline algorithm [22], seems to be too low even when ignoring representation and complexity issues for elements of the time domain.

After the appearance of the preliminary version of this article as the conference paper [7], Baldor and Niu [3] presented a monitoring algorithm for the interval-based setting, which also deals with temporal future operators. In contrast to our interval-based algorithm, where the input signals are incrementally given in "chunks," their algorithm is triggered by signal changes. In this respect, their algorithm has the flavor of our point-based algorithm. However, additional machinery is needed to soundly reason in the interval-based setting. For example, their algorithm needs to consider the different ways in which a signal can change at a given time. Similar to the above works, their complexity analysis is based on the assumption that operations involving elements from the time domain can be carried out in constant time. Furthermore, as with [22], the stated worst-case time complexity (namely, only linear in the size of the formula) for the general case is too low. For instance, for formulas—similar to the ones in Example 6.2—in the past-only fragment with the temporal operators $\blacklozenge_{[a,b]}$ with $a > 0$, the time complexity of their algorithm can be quadratic in the worst case in the formula size.

After the appearance of the conference version [7] of this article, Ho et al. [17] presented a point-based monitoring algorithm for full metric temporal logic that handles, in particular,

temporal future and past operators. Their algorithm is based on separating metric temporal operators from non-metric operators. They show that such a separation is always possible by rewriting the formula. Moreover, all metric temporal operators are bounded. However, this separation might result in a non-elementary blow-up in the formula size. Automata-theoretic methods are used to handle the non-metric part and are combined with a dynamic programming approach, which is used for checking the metric constraints.

Other monitoring algorithms for real-time logics are given by Kristoffersen et al. [19], Drusinsky [12], and Bauer et al. [8]. Their real-time logics, which are based on a point-based time model, differ from the past-only fragment of metric temporal logic. For instance, in [8] an extension of LTL with the freeze quantifier is considered. Similar to the monitoring algorithm in [33], the algorithm in [19] is based on formula rewriting. However, their rewriting can lead to exponential blow-ups. Another rewriting-based monitoring algorithm for a point-based real-time logic with the freeze quantifier is presented in [10]. The monitoring approach in [8] uses region automata, which are constructed from formulas with temporal future operators of a real-time logic with a three-valued semantics. The size of the constructed region automaton can be exponential in the size of the given formula and the computational complexity of monitoring a trace can be exponential in the worst case. The representation of elements of the time domain is not considered. The monitoring approach in [12] is based on alternating automata. Although the preprocessing step of constructing the automaton is linear in the size of the formula, monitoring is more expensive than using non-deterministic automata as in [8].

Pike et al. [28] presented an approach for real-time monitoring. They obtain monitors from specifications given by so-called stream equations, which can encode the past-only fragment of LTL. The underlying semantics is point-based. Their work is practically motivated and they do not consider real-time logics and computational complexity questions. Reinbacher et al. [30,31] describe and evaluate a framework for monitoring hardware circuits. Properties are specified in a metric temporal logic with a discrete time model. Their monitoring algorithm is tailored towards the application area; in particular, it is designed to be realized in hardware.

We are not aware of any work that compares different time models for runtime verification. The surveys [2,9,27] on real-time logics focus on the expressiveness, satisfiability, and automatic verification of real-time systems. A comparison of a point-based and interval-based time model for temporal databases with a discrete time domain is given by Toman [34]. The work by Furia and Rossi [14] on sampling and the work on digitization [16] by Henzinger et al. are orthogonal to our comparison. These relate fragments of metric interval temporal logic with respect to a discrete and a dense time domain.

8 Conclusions

We have presented, analyzed, and compared monitoring algorithms for real-time logics with point-based and interval-based semantics. Moreover, we have presented a practically relevant fragment for the interval-based setting by distinguishing between state variables and system events, which can be more efficiently monitored in the point-based setting. Our comparison provides a detailed analysis of the trade-offs between the different time models with respect to monitoring. In particular, the interval-based semantics can be more natural than the point-based semantics. In contrast, the point-based algorithm has better worst-case complexity bounds and is conceptually simpler than the interval-based algorithm, which suggests that monitoring in a point-based setting is easier than in an interval-based setting. The meaning

of "easier" is, however, admittedly informal here since we do not provide lower bounds for the corresponding monitoring problems.

Future work is to establish such lower bounds. Thati and Roşu [33] give lower bounds for future fragments of metric temporal logic including the temporal operator referring to the next time point. However, we are not aware of any lower bounds for the past-only fragment. In general, establishing lower bounds for monitoring problems seems to be difficult. Schnoebelen and others (see [11, Section 4.1] and [24, Section 3]) have also identified this as a relevant open problem.

Acknowledgements We thank Kevin Baldor and Jianwei Niu for answering our questions about their work. The Nokia Research Center, Switzerland and the Zurich Information Security and Privacy Center (ZISC) partly supported this work.

References

1. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. J. ACM **43**(1), 116–146 (1996)
2. Alur, R., Henzinger, T.A.: Logics and models of real time: a survey. In: Proceedings of the 1991 REX Workshop on Real-Time: Theory in Practice Lecture Notes in Computer Science, vol. 600, pp. 74–106. Springer, Berlin (1992)
3. Baldor, K., Niu, J.: Monitoring dense-time, continuous-semantics, metric temporal logic. Proceedings of the 3rd International Conference on Runtime Verification (RV). Lecture Notes in Computer Science, vol. 7687, pp. 245–259. Springer, Berlin (2013)
4. Basin, D., Klaedtke, F., Müller, S.: Monitoring security policies with metric first-order temporal logic. In: Proceedings of the 15th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 23–33. ACM Press, New York (2010)
5. Basin, D., Klaedtke, F., Müller, S., Pfitzmann, B.: Runtime monitoring of metric first-order temporal properties. In: Proceedings of the 28th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS). Leibniz International Proceedings in Informatics (LIPIcs), vol. 2, pp. 49–60. Schloss Dagstuhl - Leibniz Center for Informatics (2008)
6. Basin, D., Klaedtke, F., Müller, S., Zălinescu, E.: Monitoring metric first-order temporal properties. J. ACM, **62**(2), 15:1–15:45 (2015)
7. Basin, D., Klaedtke, F., Zălinescu, E.: Algorithms for monitoring real-time properties. Proceedings of the 2nd International Conference on Runtime Verification (RV). Lecture Notes in Computer Science, vol. 7186, pp. 260–275. Springer, Berlin (2012)
8. Bauer, A., Leucker, M., Schallhart, C.: Runtime verification for LTL and TLTL. ACM Trans. Softw. Eng. Methodol. **20**(4), 14:1–14:64 (2011)
9. Bouyer, P.: Model-checking times temporal logics. In: Proceedings of the 5th Workshop on Methods for Modalities (M4M5). Elec. Notes Theo. Computer Science, vol. 231, pp. 323–341. Elsevier, Amsterdam (2009)
10. Chai, M., Schlingloff, H.: A rewriting based monitoring algorithm for TPTL. In: Proceedings of the 22nd International Workshop on Cocnurrency, Specification and Programming (CS&P). CEUR Workshop Proceedings, vol. 1032, pp. 61–72. CEUR-WS.org (2013)
11. Demri, S., Schnoebelen, P.: The complexity of propositional linear temporal logics in simple cases. Inf. Comput. **174**(1), 84–103 (2002)
12. Drusinsky, D.: On-line monitoring of metric temporal logic with time-series constraints using alternating finite automata. J. UCS **12**(5), 482–498 (2006)
13. Fürer, M.: Faster integer multiplication. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC), pp. 55–67. ACM Press, New York (2007)
14. Furia, C.A., Rossi, M.: A theory of sampling for continuous-time metric temporal logic. ACM Trans. Comput. Log. **12**(1), 8:1–8:40 (2010)
15. Goodloe, A., Pike, L.: Monitoring Distributed Real-Time Systems: A Survey and Future Directions. Technical Report NASA/CR-2010-216724, NASA Langley Research Center, Hampton (2010)
16. Henzinger, T.A., Manna, Z., Pnueli, A.: What good are digital clocks? In: Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP). Lecture Notes in Computer Science, vol. 623, pp. 545–558. Springer, Berlin (1992)

17. Ho, H.-M., Ouaknine, J., Worrell, J.: Online monitoring of metric temporal logic. Proceedings of the 5th International Conference on Runtime Verification (RV). Lecture Notes in Computer Science, vol. 8734, pp. 178–192. Springer, Berlin (2014)
18. Koymans, R.: Specifying real-time properties with metric temporal logic. Real Time Syst. **2**(4), 255–299 (1990)
19. Kristoffersen, K. J., Pedersen, C., Andersen, H. R.: Runtime verification of timed LTL using disjunctive normalized equation systems. In: Proceedings of the 3rd Workshop on Runtime Verification (RV). Elec. Notes Theo. Computer Science, vol. 89, pp. 210–225. Elsevier, Amsterdam (2003)
20. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Commun. ACM **21**(7), 558–565 (1978)
21. Lichtenstein, O., Pnueli, A., Zuck, L.: The glory of the past. Proceedings of the Conference on Logics of Programs. Lecture Notes in Computer Science, vol. 193, pp. 196–218. Springer, Berlin (1985)
22. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. Proceedings of the Joint International Conferences on Formal Modelling and Analysis of Timed Systems (FORMATS) and on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT). Lecture Notes in Computer Science, vol. 3253, pp. 152–166. Springer, Berlin (2004)
23. Maler, O., Nickovic, D.: Monitoring properties of analog and mixed-signal circuits. Int. J. Softw. Tools Technol. Trans. **15**(3), 247–268 (2013)
24. Markey, N., Schnoebelen, P.: Model checking a path. Proceedings of the 15th International Conference on Concurrency Theory (CONCUR). Lecture Notes in Computer Science, vol. 3170, pp. 248–262. Springer, Berlin (2004)
25. Nickovic, D.: Checking Timed and Hybrid Properties: Theory and Applications. Ph.D. thesis, Université Joseph Fourier, Grenoble, France (2008)
26. Nickovic, D., Maler, O.: AMT: A property-based monitoring tool for analog systems. Proceedings of the 5th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS). Lecture Notes Computer Science, vol. 4763, pp. 304–319. Springer, Berlin (2007)
27. Ouaknine, J., Worrell, J.: Some recent results in metric temporal logic. Proceedings of the 6th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS). Lecture Notes in Computer Science, vol. 5215, pp. 1–13. Springer, Berlin (2008)
28. Pike, L., Goodloe, A., Morisset, R., Niller, S.: Copilot: A hard real-time runtime monitor. Proceedings of the 1st International Conference on Runtime Verification (RV). Lecture Notes in Computer Science, vol. 6418, pp. 345–359. Springer, Berlin (2010)
29. Raskin, J.-F., Schobbens, P.-Y.: Real-time logics: fictitious clock as an abstraction of dense time. Proceedings of the 3rd International Workshop on Tools and Algorithms for Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 1217, pp. 165–182. Springer, Berlin (1997)
30. Reinbacher, T., Függer, M., Brauer, J.: Real-time runtime verification on chip. Proceedings of the 3rd International Conference on Runtime Verification (RV). Lecture Notes in Computer Science, vol. 7687, pp. 110–125. Springer, Berlin (2013)
31. Reinbacher, T., Függer, M., Brauer, J.: Runtime verification on embedded real-time systems. Form. Methods Syst. Des. **44**(3), 203–239 (2014)
32. Schönhage, A., Strassen, V.: Schnelle Multiplikation großer Zahlen. Computing **7**(3–4), 281–292 (1971)
33. Thati, P., Roşu, G.: Monitoring algorithms for metric temporal logic specifications. In: Proceedings of the 4th Workshop on Runtime Verification (RV). Elec. Notes Theo. Computer Science, vol. 113, pp. 145–162. Elsevier, Amsterdam (2005)
34. Toman, D.: Point vs. interval-based query languages for temporal databases (extended abstract). In: Proceedings of the 15th ACM Symposium on Principles of Database Systems (PODS), pp. 58–67. ACM Press, New York (1996)