A Theory of (Linear-Time) Timed Monitors

Mouloud Amara
Université Paris Cité, CNRS, IRIF, F-75013 Paris, France

Giovanni Bernardi
Université Paris Cité, CNRS, IRIF, F-75013 Paris, France

Mohammed Aristide Foughali
Université Paris Cité, CNRS, IRIF, F-75013 Paris, France

Adrian Francalanza
Dept. of Computer Science, ICT, University of Malta, Msida, Malta

Abstract

Runtime Verification (RV) is gaining popularity due to its scalability and ability to analyse block-box systems. *Monitoring* is at the heart of RV; a logical formula ϕ, formalising some property of interest, is typically translated into a monitor that checks whether the system under scrutiny satisfies ϕ during its execution. A logical formula ϕ is violation (resp. satisfaction) *monitorable* iff there exists a monitor for ϕ that is both sound and complete w.r.t. its violation (resp. satisfaction). The *monitorability* problem is thus concerned with determining the largest subset of a logic L that is monitorable. Although this problem has been solved for expressive untimed logics, it remains open for timed logics, where formulae can express both the order of events and the quantity of time separating them. This paper solves the monitorability problem for T^{lin}, a new expressive (linear-time) timed μ-calculus that we propose. First, we show that T^{lin} is strictly more expressive than MTL, the de facto timed extension of LTL. Second, we identify MT^{lin}, the largest monitorable fragment of T^{lin}: we characterise its largest subsets of formulae that are violation monitorable, satisfaction monitorable, and *complete monitorable* (both satisfaction and violation monitorable). To wit, this is the first work that answers the monitorability question for such an expressive timed logic.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Timed logics, Runtime Verification, Monitorability

Digital Object Identifier 10.4230/LIPIcs.ECOOP.2025.1

Related Version *Full Version*: `https://hal.science/hal-05043055/document`

Supplementary Material *Software*: `https://zenodo.org/records/15283069`
Software (ECOOP 2025 Artifact Evaluation approved artifact):
`https://doi.org/10.4230/DARTS.11.2.8`

Funding This work has received support under the program "Investissement d'Avenir" launched by the French Government and implemented by ANR, with the reference "ANR-18-IdEx-000" as part of its program "Émergence". Other fundings include the ANR-JST Grant CyPhAI as well as the ANR project MAVEriQ.

1 Introduction

Runtime Verification (RV) is a lightweight verification technique [30, 10], where the system at hand is *instrumented* with *monitors* [20, 21]. A monitor M is a software component that corresponds to a logical formula ϕ expressing some property of interest. At runtime, M *incrementally* analyses the execution of the system until it reaches a *verdict* w.r.t. the satisfaction or violation of ϕ [30, 2]. RV can mitigate two major drawbacks of exhaustive, offline techniques such as model checking. First, by trading exhaustiveness for scalability,

RV does not suffer from the state-space explosion problem. Second, provided that actions of interest are visible to monitors, RV is suitable for systems where no model is known beforehand [19], e.g., black-box and self-adaptive systems [33].

Due to the lack of exhaustiveness, the *monitorability* problem becomes a central issue [39, 10, 3]. Intuitively, a formula ϕ is monitorable iff its violation or satisfaction by the system can be detected by a monitor. Consider the following formula, expressed in the (action-based) Linear-time Temporal Logic (LTL) interpreted over infinite executions [38]:

$$\varphi \;=\; G\,(b \Rightarrow F\,c)$$

Formula φ expresses a classical "leads to" property, requiring that "every action b must be followed by an action c in future", with known algorithms to model-check it offline [31, 13, 47]. However, this formula is *not* monitorable, since any satisfaction verdict after a c may be contradicted by a future b not followed by a c (and dually, any violation verdict after a b may be contradicted by a future c). In more technical terms, a formula ϕ is said to be satisfaction (resp. violation) monitorable [2, 19] iff there exists a monitor for ϕ that is both sound and satisfaction (resp. violation) complete (alternative definitions of monitorability are discussed in Section 6). Formula ϕ is said *monitorable* iff it is satisfaction or violation monitorable, and *complete monitorable* if it is both satisfaction and violation monitorable. The *monitorability problem* is thus concerned with determining the largest monitorable subset of a given logic L.

Monitorability has been studied for the modal μ-calculus, also known as the Hennessy-Milner Logic with recursion (μ-HML) [28, 29]. Notably, the *maximally-expressive monitorable subset* of μ-HML, i.e., a subset within which any monitorable formula in μ-HML is expressible, has been identified for both linear-time and branching-time semantics [2, 19]. However, μ-HML is unable to express *timed properties*. Consider φ', a timed version of φ, written in the Metric Temporal Logic MTL [27] (the de facto timed extension of LTL):

$$\varphi' \;=\; G\,(b \Rightarrow F_{[0,5]}\,c)$$

Formula φ' expresses a *bounded-response* property, requiring that every occurrence of b must be followed by an occurrence of c *within 5 time units*. Timed properties are extremely important in real-time systems (see e.g., [35, 17]). Even though the body of research on RV for timed systems is extensive (more in Section 6), there is, to the best of our knowledge, no work that concretely tackles the monitorability problem in the timed setting.

Contributions. Our contribution is twofold. First, we propose T^{lin}, a new timed μ-calculus with a linear-time interpretation[1], and prove that T^{lin} is strictly more expressive than MTL (without past operators, Theorem 3.6). Second, we precisely identify VT^{lin}, ST^{lin} and CT^{lin}, the largest (i.e., maximally-expressive) fragments of T^{lin} that are violation monitorable, satisfaction monitorable and complete monitorable, respectively. Accordingly, $MT^{lin} = VT^{lin} \cup ST^{lin}$ is the largest monitorable fragment of T^{lin} (Theorem 5.43, Theorem 5.45, Theorem 5.44). For practical reasons, we also provide automatic translators (i) from MTL to T^{lin} and (ii) from MT^{lin} to monitors. To the best of our knowledge, this is the first work that solves the monitorability problem for an expressive timed logic, providing a full characterisation of its largest monitorable fragment.

[1] Note that a number of timed μ-calculi exist (see [14] for an overview), yet outside of this paper's scope due to their branching-time semantics.

Outline. The paper is organised as follows. After overviewing the preliminaries in Section 2, we discuss our core contributions in Section 3 through Section 5. Section 3 defines our new logic T^{lin} and proves that it is more expressive than MTL, providing along the way an automatic translation from the latter to the former. Section 4 lays out the theoretical foundations for timed monitors and instrumentation. We then tackle the monitorability problem in Section 5. After setting up the framework (Section 5.1), we provide a compiler from T^{lin} formulae to sound monitors (Section 5.2). We then identify CT^{lin}, VT^{lin} and ST^{lin} (both supersets of CT^{lin}), fragments of complete-, violation- and satisfaction-monitorable formulae in T^{lin}, respectively (Section 5.3, Section 5.4). In Section 5.5, we prove the maximality of CT^{lin}, VT^{lin} and ST^{lin}, and conclude accordingly that $MT^{lin} = VT^{lin} \cup ST^{lin}$ is the largest monitorable fragment of T^{lin}. Section 6 compares our contributions with related work and Section 7 concludes with future work directions.

Full version. An extended version of this paper with all proofs, more examples and further details, is available online (see Full Version above).

2 Preliminaries

In the remainder of this paper, let Σ denote a *non-empty, finite set* of *actions*.

2.1 Timed Labelled Transition Systems

The following definition is inspired from [1].

▶ **Definition 2.1.** *A Timed Labelled Transition System (TLTS) P over Σ is a tuple $\langle S_P, \text{LAB}, \longrightarrow_P \rangle$ where S_P is a set of states, $\text{LAB} = \Sigma \cup \mathbb{R}_{\geq 0} \cup \{\tau\}$ a set of labels, and $\longrightarrow_P \subseteq S_P \times \text{LAB} \times S_P$ the transition relation satisfying:*

1. *Time determinism: for every $p, p', p'' \in S_P$ and $d \in \mathbb{R}_{\geq 0}$, if $(p, d, p') \in \longrightarrow_P$ and $(p, d, p'') \in \longrightarrow_P$ then $p' = p''$,*
2. *Time additivity: for every $p, p'' \in S_P$ and $d_1, d_2 \in \mathbb{R}_{\geq 0}$, $(p, d_1 + d_2, p'') \in \longrightarrow_P$ iff $(p, d_1, p') \in \longrightarrow_P$ and $(p', d_2, p'') \in \longrightarrow_P$ from some $p' \in S_P$,*
3. *Zero delay: for every $p, p' \in S_P$, $(p, 0, p') \in \longrightarrow_P$ iff $p' = p$.*

A transition labelled $\alpha \in \Sigma$ (resp. $\alpha \in \mathbb{R}_{\geq 0}$) is a called an *action* transition (resp. delay transition). A τ transition is a non-observable transition labelled with the special symbol τ. We use a, a_i, b, c and d, d_i to denote, respectively, an element of Σ and $\mathbb{R}_{\geq 0}$. We write $p_1 \xrightarrow{\alpha_1} p_2 \xrightarrow{\alpha_2} p_3 \ldots$ to denote a *sequence of transitions* with $(p_1, \alpha_1, p_2), (p_2, \alpha_2, p_3) \ldots \in \longrightarrow_P$.

Runs. A *run* in the TLTS P is a sequence of labels (in $\text{LAB}^\omega \cup \text{LAB}^* = \text{LAB}^\infty$) corresponding to a sequence of transitions. If there exists an infinite transition sequence $p_1 \xrightarrow{\alpha_1} p_2 \xrightarrow{\alpha_2} p_3 \xrightarrow{\alpha_3} \ldots$, we say that run $\theta = \alpha_1 \alpha_2 \alpha_3 \ldots \in \text{LAB}^\omega$ is *infinite* from $p_1 \in S_P$.

Weak runs. Let $\mathcal{L} = \text{LAB} \setminus \{\tau\}$. To define weak transitions, we write $p \Longrightarrow p'$ instead of $p \, (\xrightarrow{\tau})^* \, p'$ which is the reflexive-transitive closure of the relation $\xrightarrow{\tau}$, and following [49], we write $p \xRightarrow{\alpha} p'$ for $\alpha \in \mathcal{L}$ in the following cases,

(a) $p \xRightarrow{a} p'$ if $p \Longrightarrow \cdot \xrightarrow{a} \cdot \Longrightarrow p'$,

(b) $p \xRightarrow{d} p'$ if there exists $n \geq 0$ s.t. $p \Longrightarrow \cdot \xrightarrow{d_0} \cdot \Longrightarrow \cdot \xrightarrow{d_1} \ldots \xrightarrow{d_n} \cdot \Longrightarrow p'$ where $d = \sum_{i \leq n} d_i$,

with the convention that if the set $d_1, \ldots, d_n$ is empty, then $\sum_{1 \leq i \leq n} d_i = 0$. Following [19], for $\alpha_1, \ldots, \alpha_n \in \mathcal{L}$ and $p, p' \in S_P$, we write sequences of transitions $p \xrightarrow{\alpha_1}\!\!\!\!\rightarrow \ldots \xrightarrow{\alpha_n}\!\!\!\!\rightarrow p'$ as $p \xrightarrow{\theta}\!\!\!\!\rightarrow p'$ where $\theta = \alpha_1 \ldots \alpha_n$. Such θ is called a weak run (in $\mathcal{L}^*$). We say that a weak finite run $\theta' \in \mathcal{L}^*$ *corresponds to* a finite run $\theta \in \text{LAB}^*$ iff removing τ transitions then summing consecutive delays in θ gives θ'. For instance, the weak finite run $5\,a\,3\,b\,2$ corresponds to e.g., finite runs $2\,\tau\,3\,a\,3\,b\,\tau\,2$ and $5\,a\,3\,b\,1\,\tau\,1$. The *empty weak run* is $\epsilon = 0$.

Reachability and inevitability. In TLTS P, a state $p' \in S_P$ is reachable from state $p \in S_P$ along a finite weak run $\theta \in \mathcal{L}^*$ iff $p \xrightarrow{\theta}\!\!\!\!\rightarrow p'$. We say that p' is *inevitably reachable* from p along θ, denoted $p \xrightarrow{\theta}\!\!\!\!\rightarrow_{inev} p'$, if p' is the only reachable state from p along θ, that is:

$$p \xrightarrow{\theta}\!\!\!\!\rightarrow p' \text{ and } \forall p'' \in S_P : p \xrightarrow{\theta}\!\!\!\!\rightarrow p'' \text{ implies } p' = p''$$

Similarly, we write $p \xrightarrow{\theta}\!\!\!\!\!\!\not\rightarrow p'$ if p' is not reachable from p along θ, $p \xrightarrow{\theta}\!\!\!\!\!\!\not\rightarrow$ if there exists a state p' reachable from p along θ, and $p \xrightarrow{\theta}\!\!\!\!\!\!\not\rightarrow$ if there is no reachable state from p along θ.

2.2 Timed Words

As in e.g., [12, 22], we interpret timed logics over infinite timed words.

▶ **Definition 2.2** (Timed words). *A timed word π over Σ is a possibly infinite sequence $(a_1, t_1)(a_2, t_2)...$ of timed events $(a_i, t_i) \in \Sigma \times \mathbb{R}_{\geq 0}$, where t_i are monotonically increasing timestamps, i.e., $\forall i \in \mathbb{N}$: $t_i \leq t_{i+1}$. An empty timed word contains no timed events.*

We denote $T\Sigma^\omega$ and $T\Sigma^*$ the set of infinite (omega) and finite timed words, respectively, and $T\Sigma^\infty$ the set of all timed words ($T\Sigma^\infty = T\Sigma^\omega \cup T\Sigma^*$). We denote by $|\pi|$ the *size* of a timed word $\pi \in T\Sigma^\infty$. The size of a finite timed word $\pi = (a_1, t_1)...(a_n, t_n) \in T\Sigma^*$ equals n. The size of an infinite timed word $\pi \in T\Sigma^\omega$ equals ∞. We use the notation $a_i(\pi)$ (resp. $t_i(\pi)$) for the action (resp. the timestamp) of the timed event at position i of $\pi \in T\Sigma^\infty$, with $i \in \mathbb{N}$ and $i \leq |\pi|$. For a timed event (a_i, t_i), we say that t_i is associated with a_i. The equality between two timed words $\pi, \pi' \in T\Sigma^\infty$ is defined in the usual way. For some $\pi \in T\Sigma^\infty$, we write π^i (resp. ${}^i\pi$), $i \leq |\pi|$, for the suffix of π starting at position i (resp. the prefix of π ending at i). Concatenation is defined through the partial function $Concat : T\Sigma^* \times T\Sigma^\infty \rightharpoonup T\Sigma^\infty$, defined only for couples $(\pi, \pi') \in T\Sigma^* \times T\Sigma^\infty$ such that $\pi' \in T\Sigma^\omega_{\vec{\pi}}$, where $T\Sigma^\omega_{\vec{\pi}}$ is defined as follows, w.r.t. to some finite timed word π: $T\Sigma^\omega_{\vec{\pi}} = \{\pi' \in T\Sigma^\omega \mid t_1(\pi') \geq t_{|\pi|}(\pi)\}$. We use the shorthand notation $\pi.\pi'$ instead of $Concat(\pi, \pi')$. For any timed word $\pi' \in T\Sigma^\infty$, we define the set of its prefixes $Pf(\pi') = \{\pi \in T\Sigma^* \mid \exists \pi'' \in T\Sigma^\omega_{\vec{\pi}} : \pi.\pi'' = \pi'\}$. Dually, we define for any finite timed word $\pi' \in T\Sigma^*$ its infinite *extensions* $Ext(\pi') = \{\pi \in T\Sigma^\omega \mid \pi' \in Pf(\pi)\}$.

Note that a timed word corresponds necessarily to a weak run in $\mathcal{L}^*$, with each timestamp recording the *absolute time* of occurrence of the action associated with it. For example, the finite timed word $(a, 0)(b, 3)(a, 3)(c, 7.2)$ results from the finite weak run $a\,3\,b\,a\,4.2\,c$. We will formalise the function to retrieve a finite weak run from a finite timed word in Section 5.1.

▶ Remark 2.3. Note that, in the timed systems literature, every execution in a TLTS corresponds to an infinite *time divergent* run/timed word (see e.g., [23, 8]), i.e., all runs/timed words diverge in time in the future. We do not make this assumption here since our results up to and including Section 4 do not depend on it. Instead, we formalise, justify and enforce it in Section 5.1 for monitorability purposes.

$$\phi, \psi \quad := \quad a \mid \neg a \mid \phi \wedge \psi \mid \phi \vee \psi \mid \bigcirc_I \phi \mid \phi \, \mathcal{U}_I \, \psi \mid G_I \, \phi$$

Figure 1 MTL syntax.

2.3 Intervals

We now define $\mathcal{I}$, the set of dense-time intervals with natural bounds. Hereafter, we refer to an element of $\mathcal{I}$ simply as "interval".

▶ **Definition 2.4** (Interval syntax). *The set $\mathcal{I}$ is given by the grammar $I \in \mathcal{I} ::= \sqsubset p, q \sqsupset$ where $\sqsubset \in \{[, (\}$, $\sqsupset \in \{],)\}$, $p \in \mathbb{N}$ and $q \in \mathbb{N} \cup \{\infty\}$. As per standard notation, if $q = \infty$ then $\sqsupset =)$. p (resp. q) is the left (resp. right) bound of I, we write accordingly $\mathcal{LB}(I) = p$ (resp. $\mathcal{RB}(I) = q$). I is said left closed if $\sqsubset = [$, right closed if $\sqsupset =]$, and bounded if $\mathcal{RB}(I) \neq \infty$.*

▶ **Definition 2.5** (Interval Semantics). *$[\![I]\!]$, the meaning of an interval $I = \sqsubset p, q \sqsupset \in \mathcal{I}$, is a subset of $\mathbb{R}_{\geq 0}$ defined as follows: $[\![I]\!] = \{x \in \mathbb{R}_{\geq 0} \mid p \prec_l x \prec_r q\}$ where*

- *$\prec_l = \leq$ if $\sqsubset = [$ and $\prec_l = <$ otherwise,*
- *$\prec_r = \leq$ if $\sqsupset =]$ and $\prec_r = <$ otherwise.*

Hereafter, we (i) consider only non-empty intervals, i.e., in the set $\{I \in \mathcal{I} \mid [\![I]\!] \neq \varnothing\}$ and (ii) use I to denote both the syntax and semantics of an interval I.

2.4 Metric Temporal Logic (MTL)

MTL [27] is a timed logic with linear-time semantics; it is the de facto timed extension of LTL. Note that we do not consider past operators.

Syntax and Semantics. The syntax of MTL is given in Figure 1. It includes the timed *Next* operator ($\bigcirc_I$) and the timed *Until* ($\mathcal{U}_I$) operator. As per standard notation, $\top = a \vee \neg a$, $\bot = a \wedge \neg a$, and $F_I \, \phi = \top \, \mathcal{U}_I \, \phi$ (denoting "Eventually ϕ within I"). As in [40], we add the timed *Globally* (G_I) operator to the syntax to guarantee that negations appear only before propositions, without loss of expressivity[2]. The *point-wise* semantics of MTL is given by the truth value of $\pi, i \models \phi$, denoting "$\pi \in T\Sigma^\omega$ satisfies formula ϕ at position i", with the convention that $\pi \models \phi$ iff $\pi, 1 \models \phi$ [6, 36, 45, 32]. Here, we define the semantics of a formula ϕ as the set $\{\pi \mid \pi \models \phi\}$ (Figure 2). In the remainder of this paper, we omit the subscript I of temporal operators if $I = [0, \infty)$. Accordingly, if all operators appearing in a formula ϕ are subscript free, then ϕ is an untimed MTL formula (i.e., $\phi \in$ LTL). For example, $F_{[0,\infty)} \, a$, with $a \in \Sigma$, is simply written as $F \, a$ and is therefore untimed.

▶ **Example 2.6.** Let $b, c \in \Sigma$. $[\![b \, \mathcal{U}_I \, c]\!]$ contains all infinite timed words where b is satisfied in every position from the start, until c becomes satisfied within interval I (relative to the first timestamp in π). In other words, an infinite timed word π is in $[\![b \, \mathcal{U}_I \, c]\!]$ iff $\pi \in [\![b \, \mathcal{U} \, c]\!]$ *and*

[2] That is, any formula expressed within the minimal syntax in e.g., [32] (excluding past operators), allowing negations of formulae and without G_I, can be written in our syntax in Figure 1 using the following standard equalities until negations are pushed to the level of propositions. For timed *Until*: $\neg(\phi \, \mathcal{U}_I \, \psi) = (G_I \, \neg\psi) \vee \neg\psi \, \mathcal{U}_I \, (\neg\psi \wedge \neg\phi)$, and for timed *Next* (see [25]): $\neg \bigcirc_I \phi = \bigvee_{a \in \Sigma} \bigcirc_{[0, \mathcal{LB}(I)\sqsupset} a \vee \bigcirc_I \neg\phi \vee \bigvee_{a \in \Sigma} \bigcirc_{\sqsubset \mathcal{RB}(I), \infty)} a$ where $\sqsupset =)$ if I is left closed and $\sqsupset =]$ otherwise, $\sqsubset = ($ if I is right closed and $\sqsubset = [$ otherwise.

$$
\begin{array}{lcl}
[\![a]\!] & = & \{\pi \mid a_1(\pi) = a\} \\
[\![\neg a]\!] & = & \{\pi \mid a_1(\pi) \neq a\} \\
[\![\phi \wedge \psi]\!] & = & [\![\phi]\!] \cap [\![\psi]\!] \\
[\![\phi \vee \psi]\!] & = & [\![\phi]\!] \cup [\![\psi]\!] \\
[\![\bigcirc_I \phi]\!] & = & \{\pi \mid (t_2(\pi) - t_1(\pi)) \in I \text{ and } \pi^2 \in [\![\phi]\!]\} \\
[\![\phi \,\mathcal{U}_I\, \psi]\!] & = & \{\pi \mid \exists i : (t_i(\pi) - t_1(\pi)) \in I \text{ and } \pi^i \in [\![\psi]\!] \text{ and } \forall 1 \leq j < i : \pi^j \in [\![\phi]\!]\} \\
[\![G_I\, \phi]\!] & = & \{\pi \mid \forall i \geq 1 : (t_i(\pi) - t_1(\pi)) \in I \implies \pi^i \in [\![\phi]\!]\}
\end{array}
$$

Figure 2 MTL semantics.

the first c occurs within I, i.e., at a position i of π satisfying $t_i(\pi) - t_1(\pi) \in I$. For instance, for $I = [2,5]$, $\pi = (b,3)(a,4)(c,7)\ldots$, $\pi' = (b,3)(b,4)(c,7)\ldots$ and $\pi'' = (b,2)(c,3)\ldots$, we have: $\pi \notin [\![b\,\mathcal{U}_I\,c]\!]$, $\pi' \in [\![b\,\mathcal{U}_I\,c]\!]$ and $\pi'' \notin [\![b\,\mathcal{U}_I\,c]\!]$. In particular, $\pi'' \notin [\![b\,\mathcal{U}_I\,c]\!]$ even though $\pi'' \in [\![b\,\mathcal{U}\,c]\!]$, because the first c occurs "too early" with timestamp 3 and $3 - 2 \notin [2,5]$.

3 The logic T^{lin}

We present T^{lin} (Section 3.1), our novel extension of (untimed) modal μ-calculus with linear-time semantics [15] [2, Sect.2] to the dense-time setting, and prove that T^{lin} is strictly more expressive than MTL (Section 3.2).

3.1 T^{lin} Syntax and Semantics

We first introduce a syntactic characterisation of timing constraints which we call *guards*. In the sequel, let $\mathcal{D}$ be an infinite set of variables taking values in $\mathbb{R}_{\geq 0}$, that we call *timestamp variables*, and let $\mathcal{V}$ be the set of valuation functions $v \in \mathcal{D} \to \mathbb{R}_{\geq 0}$.

▶ **Definition 3.1** (Guards). *The set $\mathcal{B}(\mathcal{D})$ of guards over $\mathcal{D}$ is given by the syntax*

$$g \;:=\; x \prec n \;\mid\; (x-y) \prec n \;\mid\; n \prec x \;\mid\; n \prec (x-y) \;\mid\; g \wedge g \;\mid\; \overline{g}$$

where $n \in \mathbb{N}$, $\prec \in \{<, \leq\}$, $x, y \in \mathcal{D}$ and $\overline{g}$ is a compact notation for $\neg g$.

A valuation v *satisfies* a guard g, denoted $v \models g$, iff replacing each variable x by $v(x)$ in g gives an expression that evaluates to true. For example, $v \models (x-y) < 2$ iff $v(x) - v(y) < 2$. We write $v \not\models g$ for $\neg(v \models g)$. If I is an interval, then we write $(x-y) \in I$ as an abbreviation for the guard $\mathcal{LB}(I) \prec_l (x-y) \wedge (x-y) \prec_r \mathcal{RB}(I)$ (where $\prec_l$ and $\prec_r$ are as defined in Definition 2.5), and $(x-y) \notin I$ for $\overline{((x-y) \in I)}$. We write $v[x := m]$, $x \in \mathcal{D}$ and $m \in \mathbb{R}_{\geq 0}$, to denote the valuation that maps x to m and every $y \in \mathcal{D} \setminus \{x\}$ to $v(y)$. We denote by $[x_1 = m_1, \ldots, x_n = m_n]$ the current valuations of some variables of interest $x_1, \ldots, x_n$ when valuation v is understood from the context.

Our T^{lin} consists of a set of *formulae* with linear-time timed semantics. Thus, before defining T^{lin}, we first define TT^{lin}, the set of *terms* that can be used to write a formula ($T^{lin} \subset TT^{lin}$). The set TT^{lin} is generated by the grammar in Figure 3, with $a \in \Sigma$, X an element of a countable set of recursion variables VAR, $x \in \mathcal{D}$, and $g \in \mathcal{B}(\mathcal{D})$. Syntactically $\mathsf{min}(X, \phi)$ and $\mathsf{max}(X, \phi)$ are binders of recursion variable X in ϕ, and $x \,\underline{\mathtt{in}}\, \phi$ binds timestamp variable $x \in \mathcal{D}$ in ϕ. The free recursion (resp. timestamp) variables of a term ϕ are given by the function $fv : TT^{lin} \longrightarrow \mathcal{P}(\text{VAR})$ (resp. $fv_{\text{TIME}} : TT^{lin} \longrightarrow \mathcal{P}(\mathcal{D})$), defined in the expected way. A term in which no variable is free is *closed*.

ϕ	$::=$	tt	*true*	$\mid$	ff	*false*
	$\mid$	X	*variable*	$\mid$	$x \,\underline{\mathtt{in}}\, \phi$	*timestamp*
	$\mid$	$\phi \wedge \psi$	*conjunction*	$\mid$	$\phi \vee \psi$	*disjunction*
	$\mid$	$\langle a \rangle \phi$	*diamond*	$\mid$	$[a]\phi$	*box*
	$\mid$	$\langle g \rangle \phi$	*diamond on time*	$\mid$	$[g]\phi$	*box on time*
	$\mid$	$\mathsf{min}(X, \phi)$	*min. fixpoint*	$\mid$	$\mathsf{max}(X, \phi)$	*max. fixpoint*

Figure 3 Syntax of TT^{lin}.

Compared to the untimed version of linear-time μ-calculi in [15] [2, Sect.2], TT^{lin} provides three new syntactic constructs to handle dense time: $x \,\underline{\mathtt{in}}\, \phi$, $[g]\phi$ and $\langle g \rangle \phi$. These are best discussed when introducing formulae later in this subsection.

$$
\begin{array}{lcllcl}
[\![\mathsf{tt}, \rho]\!]_v & = & T\Sigma^\omega & [\![\mathsf{ff}, \rho]\!]_v & = & \emptyset \\
[\![\phi \vee \psi, \rho]\!]_v & = & [\![\phi, \rho]\!]_v \cup [\![\psi, \rho]\!]_v & [\![\phi \wedge \psi, \rho]\!]_v & = & [\![\phi, \rho]\!]_v \cap [\![\psi, \rho]\!]_v \\
[\![\langle a \rangle \phi, \rho]\!]_v & = & \langle \cdot a \cdot \rangle ([\![\phi, \rho]\!]_v) & [\![[a]\phi, \rho]\!]_v & = & [\cdot a \cdot] ([\![\phi, \rho]\!]_v) \\
[\![x \,\underline{\mathtt{in}}\, \phi, \rho]\!]_v & = & \{\pi \mid \pi \in [\![\phi, \rho]\!]_{v[x:=t_1(\pi)]}\} & [\![X, \rho]\!]_v & = & \rho(X) \\
[\![\mathsf{min}(X, \phi), \rho]\!]_v & = & \bigcap \{S \mid [\![\phi, \rho[X \to S]]\!]_v \subseteq S\} & [\![\langle g \rangle \phi, \rho]\!]_v & = & \langle \cdot g \cdot \rangle (v, [\![\phi, \rho]\!]_v) \\
[\![\mathsf{max}(X, \phi), \rho]\!]_v & = & \bigcup \{S \mid S \subseteq [\![\phi, \rho[X \to S]]\!]_v\} & [\![[g]\phi, \rho]\!]_v & = & [\cdot g \cdot] (v, [\![\phi, \rho]\!]_v)
\end{array}
$$

Figure 4 Linear time semantics of TT^{lin}.

The semantics of terms over infinite timed words is given in Figure 4, where we have used the auxiliary functions $\langle \cdot a \cdot \rangle(-)$, $[\cdot a \cdot](-)$, $\langle \cdot g \cdot \rangle(v, -)$, $[\cdot g \cdot](v, -) : \mathcal{P}(T\Sigma^\omega) \longrightarrow \mathcal{P}(T\Sigma^\omega)$ defined by the following equalities:

$$
\begin{array}{lll}
\langle \cdot a \cdot \rangle(Z) & = \{\pi \mid a_1(\pi) = a \wedge \pi^2 \in Z\} & [\cdot a \cdot](Z) = \{\pi \mid a_1(\pi) = a \Rightarrow \pi^2 \in Z\} \\
\langle \cdot g \cdot \rangle(v, Z) & = \{\pi \mid v \models g \wedge \pi \in Z\} & [\cdot g \cdot](v, Z) = \{\pi \mid v \models g \Rightarrow \pi \in Z\}
\end{array}
\tag{1}
$$

and where environment ρ is an element of Υ, the set of partial functions $\textsc{Var} \rightharpoonup \mathcal{P}(T\Sigma^\omega)$, and v a valuation in $\mathcal{V}$. The notation $[\![\phi, \rho]\!]_v$, with $fv(\phi) \subseteq dom(\rho)$, denotes the set of timed words that satisfy ϕ w.r.t. the environment ρ and the valuation v.

Properties of the semantics. Let $dom(f)$ denote the domain of definition of a partial function f. For any $\phi \in TT^{lin}$, $v \in \mathcal{V}$, $X \in \textsc{Var}$ and $\rho \in \Upsilon$ s.t. $fv(\phi) \subseteq dom(\rho) \cup \{X\}$ we let

$$
\begin{array}{lcl}
\wr \phi \int_{v,X,\rho} & : & \mathcal{P}(T\Sigma^\omega) \longrightarrow \mathcal{P}(T\Sigma^\omega) \\
\wr \phi \int_{v,X,\rho}(Z) & = & [\![\phi, \rho[X \mapsto Z]]\!]_v
\end{array}
\tag{2}
$$

In other words $\wr \phi \int_{v,X,\rho} = \lambda y. [\![\phi, \rho[X \mapsto y]]\!]_v$.

Since environments are functions from logical variables to complete lattice, we order them in the standard pointwise way to obtain a complete lattice $\langle \textsc{Var} \to \mathcal{P}(T\Sigma^\omega), \leq \rangle$, where infinitary least upper bounds and greatest lower bounds are defined as usual. We prove that the function given in Equation (2) is monotone w.r.t. the pointwise ordering of environments, therefore Knaster-Tarski's fixpoint theorem ensures that the largest and least fixpoints of every function $\wr \phi \int_{v,X,\rho}$ exist, and are the semantics of the terms defined via $\mathsf{min}(X, -)$

and $\mathsf{max}(X, -)$. As we further demonstrate that the semantic function is continuous w.r.t. directed sets over the CPO $\langle \mathcal{P}(T\Sigma^\omega), \subseteq \rangle$, we can reason on terms defined via $\mathsf{min}(X, -)$ (resp. $\mathsf{max}(X, -)$) using Kleene fixpoint theorem and its formal dual. These strong results are crucial to prove a number of statements in this paper, in particular Proposition 3.7.

Formulae. We say that a variable $X \in \textsc{Var}$ is *guarded* in a term ϕ if all occurrences of X in ϕ are in the scope of at least one modality (i.e., $\langle a \rangle -$ or $[a]-$, $a \in \Sigma$). We say that a term ϕ is *guarded* if for all its subterms $\mathsf{min}(X, \psi_1)$ and $\mathsf{max}(Y, \psi_2)$, X is guarded in ψ_1 and Y is guarded in ψ_2. For example, the term $\mathsf{min}(X, \langle a \rangle x \,\underline{\mathtt{in}}\, [x \leq 2] X \vee Y)$ is guarded, and the term $\mathsf{min}(X, \langle a \rangle x \,\underline{\mathtt{in}}\, [x \leq 2] X \vee X)$ is not, because the rightmost occurrence of X is not in the scope of a modality. The term $\mathsf{min}(X, x \,\underline{\mathtt{in}}\, [x < 3] X)$ is not guarded, because neither $x \,\underline{\mathtt{in}}$ nor $[x < 3]$ is a modality. We may now define T^{lin}, the set of formulae in TT^{lin}.

▶ **Definition 3.2.** *The set of formulae T^{lin} is defined as the set of closed and guarded terms in TT^{lin}: $T^{lin} = \{\phi \in TT^{lin} \mid \phi \text{ is guarded and } fv(\phi) = fv_{\mathit{TIME}}(\phi) = \emptyset\}$*

Since elements of T^{lin} are closed, their semantics does not depend on valuations and environments. As a consequence, we simply write $[\![\phi]\!]$ instead of $[\![\phi, \rho]\!]_v$ to denote the semantics of any $\phi \in T^{lin}$ in any environment $\rho \in \Upsilon$ and w.r.t. to any $v \in \mathcal{V}$.

▶ Remark 3.3. Note that we work up to α-conversion of bound timestamp and recursion variables. For clarity, we assume these to be distinct and pick fresh ones when manipulating formulae. We also assume each recursion variable to be bound at most once, e.g., the formula $\mathsf{min}(X, \langle a \rangle X \vee \langle b \rangle X) \vee \mathsf{max}(X, \langle c \rangle X)$ can be expressed as $\mathsf{min}(X, \langle a \rangle X \vee \langle b \rangle X) \vee \mathsf{max}(Y, \langle c \rangle Y)$.

Let us now give two examples of the semantics of a formula $\phi \in T^{lin}$, as to explain concretely the meaning of the new constructs $x \,\underline{\mathtt{in}}\, \phi$, $[g]\phi$ and $\langle g \rangle \phi$, relying on Figure 4. In the remainder of this paper, we write $\langle \Sigma \rangle \phi$ to denote a term $\bigvee_{a \in \Sigma} \langle a \rangle \phi$.

▶ **Example 3.4.** Let $\phi \in T^{lin}$ be the following formula:

$$\phi = x \,\underline{\mathtt{in}}\, [x \leq 5] \langle b \rangle \mathtt{tt}$$

The semantics of this formula equals the set of all infinite timed words π that satisfy the following: either $t_1(\pi) > 5$ or $t_1(\pi) \leq 5$ *and* $a_1(\pi) = b$. That is, ϕ expresses the property: "if the first timestamp is smaller than 5, then the action associated with it must be a b".

▶ **Example 3.5.** Let us discuss a more complex example. The following untimed formula, given in [4, Ex. 4.2], expresses the property "every other action is a b":

$$\phi_{even(b)} = \mathsf{max}(X, \langle \Sigma \rangle \langle b \rangle X)$$

Recalling that an untimed formula is implicitly timed with the interval $[0, \infty)$, $\phi_{even(b)}$ contains all timed words π satisfying the following: at every even position i, $a_i(\pi) = b$. We want a timed version of this formula, i.e., to express $\phi_{even(b),I}$, the generalisation of $\phi_{even(b)}$ to any interval $I \in \mathcal{I}$. More precisely, the semantics of $\phi_{even(b),I}$ must contain every infinite timed word π that satisfies the following: for every even position $i \in \mathbb{N}_{>1}$, i.e., $i \bmod 2 = 0$, if $t_i(\pi) - t_1(\pi)$ is within interval I, then $a_i(\pi)$ must be a b, and $a_i(\pi)$ can be any action otherwise. For this, we use the constructs $x \,\underline{\mathtt{in}}\, -$ and $\langle g \rangle -$ to build up the following formula, where y^a denotes a distinct timestamp variable when ranging over actions in Σ:

$$\phi_{even(b),I} = x_s \,\underline{\mathtt{in}}\, \mathsf{max}(X, \langle \Sigma \rangle (x_e \,\underline{\mathtt{in}}\, \langle x_e - x_s \in I \rangle \langle b \rangle X \vee \bigvee_{a \in \Sigma} y^a \,\underline{\mathtt{in}}\, \langle y^a - x_s \notin I \rangle \langle a \rangle X))$$

Let us explain through evaluating whether some infinite timed word π belongs to the semantics of $\phi_{even(b),I}$. The $x_s \,\underline{\texttt{in}}\, -$ is used outside of the maximal-fixpoint operator, allowing timestamp variable x_s (for x "start") to record the first timestamp in π. Then, each recursion step contains two consecutive actions (as in $\phi_{even(b)}$), but this time with two disjoint recursion "paths" (the terms $x_e \,\underline{\texttt{in}}\langle x_e - x_s \in I\rangle\langle b\rangle X$ and $\bigvee_{a\in\Sigma} y^a \,\underline{\texttt{in}}\langle y^a - x_s \notin I\rangle\langle a\rangle X$). The first path requires the second action to be a b occurring within I, and the second allows the second action to be any action in Σ occurring outside of I. To assess whether an action occurs e.g., in I, the first recursion path uses a timestamp variable x_e (for x "end"), updated with the timestamp of every other action in π, and requires the difference between x_e and x_s to belong to I ($x_e \,\underline{\texttt{in}}\langle x_e - x_s \in I\rangle$). Therefore, the only condition for π to be outside of the semantics of $\phi_{even(b),I}$ is the existence of some even position at which the action is not a b *and* occurs within I. For instance, for $I = [3,5]$ and $\pi = (a,3)(c,4)(a,7)(c,9)\ldots$, $\pi' = (a,3)(b,4)(b,7)(b,8)(c,10)\ldots$ and $\pi'' = (a,2)(b,5)(c,6)(c,7)\ldots$ we have:

- $\pi \in [\![\phi_{even(b),[3,5]}]\!]$ because there is no timed action at an even position that happens within $[3,5]$: at even position 2 we have $4-3 \notin [3,5]$, at the next even position 4 we have $9-3 \notin [3,5]$, and since at this position the right bound of the interval is already exceeded then actions at subsequent even positions will occur outside of I,
- $\pi' \in [\![\phi_{even(b),[3,5]}]\!]$ because the only even position with actions occurring within I is position 4 and we have $a_4(\pi') = b$,
- $\pi'' \notin [\![\phi_{even(b),[3,5]}]\!]$ because at position 4 an action occurs within I ($7-2 \in [3,5]$) and such action is not a b.

In general: $\pi \in [\![\phi_{even(b),I}]\!]$ iff $\forall i \in \mathbb{N}_{>1} : i \bmod 2 = 0 \wedge t_i(\pi) - t_1(\pi) \in I \implies a_i(\pi) = b$, which coincides with the generalisation of $\phi_{even(b)}$ to any interval $I \in \mathcal{I}$. Note that when $I = [0,\infty)$, we obtain exactly the untimed version $\phi_{even(b)}$ using the fact that $x_e - x_s \in [0,\infty)$ is always true and $y^a - x_s \notin [0,\infty)$, for every y^a, is always false.

3.2 T^{lin} is strictly more expressive than MTL

▶ **Theorem 3.6.**

1. $\forall \phi \in MTL\ \exists \psi \in T^{lin} : [\![\phi]\!] = [\![\psi]\!]$,
2. $\exists \psi \in T^{lin}\ \forall \phi \in MTL : [\![\psi]\!] \neq [\![\phi]\!]$.

Proof.

Proving Item 1 of Theorem 3.6. We proceed through proposing the function $\text{TO}\,T^{lin} : MTL \longrightarrow T^{lin}$ that transforms any MTL formula into an equivalent T^{lin} formula (Figure 5).

▶ **Proposition 3.7.** *For every $\phi \in MTL$ we have $[\![\phi]\!] = [\![\text{TO}\,T^{lin}(\phi)]\!]$.*

Proving Item 2 of Theorem 3.6. It suffices to find a formula $\psi \in T^{lin}$ that is not expressible in MTL. As a matter of fact, the formula $\phi_{even,I}$ that we used as an example for T^{lin} (Example 3.5) is not expressible in MTL, as states the following proposition[3]:

▶ **Proposition 3.8.** *For any $I \in \mathcal{I}$, the following formula is not expressible in MTL*

$$\phi_{even(b),I} = x_s \,\underline{\texttt{in}}\, \mathsf{max}(X, \langle\Sigma\rangle(x_e \,\underline{\texttt{in}}\langle x_e - x_s \in I\rangle\langle b\rangle X \vee \bigvee_{a\in\Sigma} y^a \,\underline{\texttt{in}}\langle y^a - x_s \notin I\rangle\langle a\rangle X))$$

[3] We recall that elements of $\mathcal{I}$ cannot be empty.

$$
\begin{array}{lcl}
\textsc{to}\, T^{lin}(a) & = & \langle a \rangle \mathsf{tt} \\
\textsc{to}\, T^{lin}(\neg a) & = & [a] \mathsf{ff} \\
\textsc{to}\, T^{lin}(\phi \wedge \psi) & = & \textsc{to}\, T^{lin}(\phi) \wedge \textsc{to}\, T^{lin}(\psi) \\
\textsc{to}\, T^{lin}(\phi \vee \psi) & = & \textsc{to}\, T^{lin}(\phi) \vee \textsc{to}\, T^{lin}(\psi) \\
\textsc{to}\, T^{lin}(\bigcirc_I \phi) & = & x_s \,\underline{\mathtt{in}}\, \langle \Sigma \rangle (x_e \,\underline{\mathtt{in}}\, \langle x_e - x_s \in I \rangle \textsc{to}\, T^{lin}(\phi)) \\
\textsc{to}\, T^{lin}(\phi \,\mathcal{U}_I\, \psi) & = & x_s \,\underline{\mathtt{in}}\, \mathsf{min}(Y, (x_e \,\underline{\mathtt{in}}\, \langle x_e - x_s \in I \rangle \textsc{to}\, T^{lin}(\psi)) \vee \\
 & & \qquad (\textsc{to}\, T^{lin}(\phi) \wedge \langle \Sigma \rangle Y)) \\
\textsc{to}\, T^{lin}(G_I\, \phi) & = & x_s \,\underline{\mathtt{in}}\, \mathsf{max}(Y, (x_e \,\underline{\mathtt{in}}\, [x_e - x_s \in I] \textsc{to}\, T^{lin}(\phi)) \wedge \langle \Sigma \rangle Y)
\end{array}
$$

Figure 5 From MTL to T^{lin}.

The proof of Proposition 3.8 follows a standard result on the lack of expressivity of temporal logics (even when timed) w.r.t. formulae that involve *counting* such as $\phi_{even,I}$ (see [48, 6]). This concludes the proof of Item 2 of Theorem 3.6, and therefore the proof of Theorem 3.6. ◀

Accordingly, T^{lin} is strictly more expressive than MTL. As we will see in Section 5, $\phi_{even(b),I}$ is monitorable for violations in the general case (and also for satisfactions whenever I is bounded), which means that T^{lin} allows for a larger set of monitorable formulae than MTL.

4 Monitoring a System

In this section, we formally define timed monitors (Section 4.1), as well as the instrumentation operator (Section 4.2). In the sequel, we refer to a timed monitor simply as monitor, and to a system that we want to monitor as a *system under monitoring* (SUM).

4.1 Monitors Syntax and Semantics

$$
\begin{array}{lcl}
M, N & ::= & E \mid a.M \mid g.M \mid M + N \mid \mathsf{rec}\ X.M \mid X \mid \textsc{let}\ x\ \textsc{in}\ M \\
E & ::= & \mathsf{END} \mid \mathsf{NO} \mid \mathsf{YES}
\end{array}
$$

Figure 6 Syntax of m-terms in TMon.

We first define TMon, the set of monitor terms, hereafter abbreviated m-terms (Figure 6, where $x \in \mathcal{D}$, $a \in \Sigma$, $g \in \mathcal{B}(\mathcal{D})$ and $X \in \textsc{Var}$). $\mathsf{YES}, \mathsf{NO}, \mathsf{END}$ are special m-terms called *verdicts*, corresponding respectively to acceptance, rejection and termination (i.e., an inconclusive outcome). By abuse of notation, we write E to denote $L(E)$, the set of m-terms generated by the grammar of Figure 6 starting from the nonterminal symbol E [26].

Notions of guardedness and free variables are defined similarly as for TT^{lin} (Section 3.1). In an m-term, $\mathsf{rec}\ X.M$ (resp. $\textsc{let}\ x\ \textsc{in}\ M$) is a binder of the recursion variable X (resp. timestamp variable x). The free recursion (resp. timestamp) variables of an m-term M are given by the function $mfv : \textsc{TMon} \longrightarrow \mathcal{P}(\textsc{Var})$ (resp. $mfv_{time} : \textsc{TMon} \longrightarrow \mathcal{P}(\mathcal{D})$), defined

in the expected way. A recursion variable X is guarded in an m-term M if every occurrence of X in M is in the scope of at least one modality $(a.)$, and $M \in \text{TMon}$ is guarded if for all its subterms $\mathsf{rec}\ X.N$, X is guarded in N. We may now define the set of monitors Mon.

▶ **Definition 4.1.** *The set of monitors* Mon *is defined as the guarded m-terms in* TMon *closed vis-à-vis recursion variables:* $\text{Mon} = \{M \in \text{TMon} \mid \ M \text{ is guarded and } mfv(M) = \emptyset\}$.

For example, $\mathsf{rec}\ X.(a.\text{let}\ x\ \text{in}\ g.X + Y) \notin \text{Mon}$ because it is not closed w.r.t. recursion variable Y, whereas $\mathsf{rec}\ X.\text{let}\ y\ \text{in}\ (x - y \leq 2).a.X \in \text{Mon}$.

▶ Remark 4.2. Note that, compared to that of formulae (Definition 3.2), Definition 4.1 is more relaxed as monitors are allowed to have free timestamp variables. The practical reason behind this is illustrated later in this subsection (Example 4.6).

▶ Remark 4.3. When manipulating monitors, we assume the same conventions as in Remark 3.3; in particular we assume fresh variables are picked at each recursion step.

The semantics of *all* monitors in Mon is given by the TLTS $\mathbb{M} = \langle S_{\mathbb{M}}, \mathcal{L}, \longrightarrow_{\mathbb{M}} \rangle$. A state $m \in S_{\mathbb{M}}$ is of the form (M, v_c, v_t) where M is a monitor (in Mon), $v_c \in \mathbb{R}_{\geq 0}$ the value of *monitor time* (to record the "global" time) and $v_t \in \mathcal{V}$ a valuation (to update timestamp variables and evaluate guards). The rules of the transition relation $\longrightarrow_{\mathbb{M}} \subseteq S_{\mathbb{M}} \times \mathcal{L} \times \mathbb{M}$ are given by[4] Figure 7, where $a \in \Sigma$ and $d \in \mathbb{R}_{\geq 0}$. The rule mVer states that a verdict state, i.e., a state of the form $(M \in E, -, -)$ can perform a transition with any visible action and reach a state preserving M, i.e., verdicts are *persistent*, a central property of monitors [2]. Rule mTime corresponds to a delay transition. The remaining rules are better explained through examples.

▶ Remark 4.4. In practice, in order to monitor a formula $\phi \in T^{lin}$, we start with a *closed* monitor M (i.e., $M \in \text{Mon}$ *and* $mfv_{time}(M) = \emptyset$) at initial state $m_0 = (M, 0, \mathbf{0})$. As the results in this subsection are more generic and do not rely on these assumptions, the latter will be introduced, justified and enforced in Section 5. Without loss of generality, we still use initial states of the form $(M, 0, \mathbf{0})$ for illustration purposes.

▶ **Example 4.5.** Let $\Sigma = \{b, c\}$, $M = \text{let}\ x\ \text{in}\ ((x > 5).\mathsf{YES} + (x \leq 5).(b.\mathsf{YES} + c.\mathsf{NO}))$, and let $m_0 = (M, 0, \mathbf{0})$ be a state of $\mathbb{M}$ (the TLTS of all monitors, see above). Say M performs, from m_0, the finite run $\theta = 6\,c$. This will give the following (unique) trace: $(M, 0, \mathbf{0}) \xrightarrow{6} (M, 6, \mathbf{0}) \xrightarrow{c} (\mathsf{YES}, 6, [x = 6])$, where, from left to right:

- the delay transition $d = 6$ is allowed by rule mTime,
- the action transition c is allowed by rule mLet. This is because the state $((x > 5).\mathsf{YES} + (x \leq 5).(a.\mathsf{YES} + b.\mathsf{NO}), 6, [x = 6])$ can perform any action in Σ and reach $(\mathsf{YES}, 6, [x = 6])$ following, respectively, rules mSelL, mGuard and MVer (Figure 7).

▶ **Example 4.6.** Let $\Sigma = \{b, c\}$ and the monitor

$$M = \text{let}\ x\ \text{in}\ (\mathsf{rec}\ X.(\text{let}\ y\ \text{in}\ ((y - x \leq 5).(b.X + c.\mathsf{NO}) + (y - x > 5).\mathsf{YES})))$$

and let

$$N = \mathsf{rec}\ X.(\text{let}\ y\ \text{in}\ ((y - x \leq 5).(b.X + c.\mathsf{NO}) + (y - x > 5).\mathsf{YES}))$$

i.e., N is the subterm of M without the binder of timestamp variable x. Intuitively, the syntax of M indicates that it accepts a run (becomes YES) iff every action seen within 5 time units, relative to the global time at which the first action occurs, is a b (otherwise it

[4] We formally prove that these rules are well defined over $S_{\mathbb{M}} \times \mathcal{L} \times S_{\mathbb{M}}$.

$$\textsc{mAct}\ \frac{}{(a.M, v_c, v_t) \xrightarrow{a} (M, v_c, v_t)} \qquad \textsc{mVer}\ \frac{M \in E}{(M, v_c, v_t) \xrightarrow{a} (M, v_c, v_t)}$$

$$\textsc{mLet}\ \frac{(M, v_c, v_t[x := v_c]) \xrightarrow{a} (M', v_c, v_t')}{(\text{LET } x \text{ IN } M, v_c, v_t) \xrightarrow{a} (M', v_c, v_t')} \qquad \textsc{mTime}\ \frac{}{(M, v_c, v_t) \xrightarrow{d} (M, v_c + d, v_t)}$$

$$\textsc{mSelL}\ \frac{(M, v_c, v_t) \xrightarrow{a} (M', v_c, v_t')}{(M + N, v_c, v_t) \xrightarrow{a} (M', v_c, v_t')} \qquad \textsc{mRec}\ \frac{(M[^{\text{rec } X.M}/_X], v_c, v_t) \xrightarrow{a} (M', v_c, v_t')}{(\text{rec } X.M, v_c, v_t) \xrightarrow{a} (M', v_c, v_t')}$$

$$\textsc{mSelR}\ \frac{(N, v_c, v_t) \xrightarrow{a} (N', v_c, v_t')}{(M + N, v_c, v_t) \xrightarrow{a} (N', v_c, v_t')} \qquad \textsc{mGuard}\ \frac{(M, v_c, v_t) \xrightarrow{a} (M', v_c, v_t') \wedge v_t \models g}{(g.M, v_c, v_t) \xrightarrow{a} (M', v_c, v_t')}$$

Figure 7 SOS rules for monitors.

rejects). Let us apply the rules of $\longrightarrow_{\mathbb{M}}$, starting at $(M, 0, \mathbf{0})$, to see whether M accepts the run $\theta = 2\,b\,6\,c\,\ldots$ as it should: $(M, 0, \mathbf{0}) \xrightarrow{2} (M, 2, \mathbf{0}) \xrightarrow{b} (M', 2, [x = 2, y = 2]) \xrightarrow{6} (M', 8, [x = 2, y = 2]) \xrightarrow{c} (\mathsf{YES}, 8, [x = 2, y = 8]) \ldots$ where

$$M' = \text{LET } y \text{ IN } ((y - x \leq 5).(b.N + c.\mathsf{NO}) + (y - x > 5).\mathsf{YES})$$

applying, respectively (from left to right) rule mTime, mLet, mTime and mLet. Notice now that, in M', there is no longer the binder of timestamp variable x, and therefore x is free in M'. This is because LET x IN in M, outside of the recursion, is intended to evaluate constantly to the value of global time corresponding to the first action. This simplification is handy to deal with monitors with both recursion and guards, and justifies the relaxed nature of Definition 4.1, allowing monitors to have free timestamp variables.

Parallel monitors. In practice, a formula is often monitored through composing monitors in parallel [2]. The syntax of m-terms is extended accordingly with two parallel composition operators: $\otimes$ for *parallel conjunction* and $\oplus$ for *parallel disjunction*, adapted from [2] to the (linear-time) timed setting. The extended syntax thereof is given in Figure 8, with $\odot \in \{\otimes, \oplus\}$.

M, N	$::= E$	$\mid a.M$	$\mid g.M$	$\mid M + N$
		$\mid \text{rec } X.M$	$\mid X$	$\mid \text{LET } x \text{ IN } M$
		$\mid$	$M \otimes N$	(par. conj.)
		$\mid$	$M \oplus N$	(par. disj.)
E	$::= \mathsf{VYES}$	$\mid \mathsf{VNO}$	$\mid \mathsf{VEND}$	
VYES	$::= \mathsf{YES}$	$\mid \mathsf{VYES} \oplus M$	$\mid \mathsf{VYES} \otimes \mathsf{VYES}$	
VNO	$::= \mathsf{NO}$	$\mid \mathsf{VNO} \otimes M$	$\mid \mathsf{VNO} \oplus \mathsf{VNO}$	
VEND	$::= \mathsf{END}$	$\mid \mathsf{VEND} \odot \mathsf{VEND}$		

Figure 8 Syntax of TMon (extended with parallel composition).

Accordingly, we extend the set of verdicts by grouping acceptance, rejection and termination verdicts in the set VYES, VNO and VEND respectively. By abuse of notation, we write $E \in \{\mathsf{VYES}, \mathsf{VNO}, \mathsf{VEND}\}$ to denote $L(E)$, the set of m-terms generated by the grammar of Figure 8 starting from the nonterminal symbol E. Each of the latter is defined recursively using the corresponding verdict from a *regular* (i.e., non-parallel) m-term in the expected way. For example, an acceptance verdict VYES is reached in a parallel disjunction of two regular m-terms whenever a YES verdict is reached by one of them. The set of monitors Mon is therefore extended accordingly (by applying Definition 4.1 to TMon, as extended to parallel composition). Thanks to lifting verdict composition rules to the syntax level, the extension of the semantics of monitors in Figure 7 to parallel monitors becomes rather simple; we only need one additional rule, stating that monitors in a composition perform actions together:

$$\textsc{mPar}\ \frac{M \odot N \notin E \quad (M, v_c, v_t) \xrightarrow{a} (M', v_c, v_t') \quad (N, v_c, v_t) \xrightarrow{a} (N', v_c, v_t'')}{(M \odot N, v_c, v_t) \xrightarrow{a} (M' \odot N', v_c, v_t' \overset{v_t}{\sqcup} v_t'')}$$

where

$$v_t' \overset{v_t}{\sqcup} v_t''(x) = \begin{cases} v_t'(x) \text{ if } v_t'(x) \neq v_t(x) \\ v_t''(x) \text{ if } v_t''(x) \neq v_t(x) \\ v_t(x) \text{ otherwise} \end{cases}$$

Recall that bound timestamp variables in M and N are disjoint, and therefore the first two cases are disjoint. $\mathbb{M} = \langle S_{\mathbb{M}}, \mathcal{L}, \longrightarrow_{\mathbb{M}} \rangle$, the TLTS describing the semantics of all monitors in Mon is therefore extended to parallel monitors, with the transition relation $\longrightarrow_{\mathbb{M}}$ defined using the rules in Figure 7 and the rule mPar above.

In the sequel, we use the shorthand notation $N + A.M$ to denote the selection between monitor N and $a.M$ for all actions in $A \subseteq \Sigma$, with the convention $N + A.M = N$ if A is empty. Moreover, we write $\overline{A}$, $A \subseteq \Sigma$, to denote the set $\Sigma \setminus A$ and similarly $\overline{a}$, $a \in \Sigma$ to denote $\Sigma \setminus \{a\}$. Using these compact notations, we can write e.g., for $\Sigma = \{a_1, a_2, a_3, a_4\}$, $a_2.M + \overline{a_2}.\mathsf{NO}$ instead of $a_2.M + a_1.\mathsf{NO} + a_3.\mathsf{NO} + a_4.\mathsf{NO}$.

Verdict persistence. As mentioned earlier, all monitors in Mon are verdict persistent.

▶ **Lemma 4.7** (Verdict presistence). *For all $(M, v_c, v_t), (N, v_c', v_t') \in S_{\mathbb{M}}$ and finite weak run $\theta \in \mathcal{L}^*$, if $M \in E$ then: $(M, v_c, v_t) \xrightarrow{\theta}\!\!\!\!\rightarrow (N, v_c', v_t')$ implies $N = M$.*

Verdict states. Thanks to verdict persistence, once a monitor reaches a *verdict state*, i.e., a state whose first component is a verdict, the information within such state become irrelevant regardless of what the monitor will perform in the future, at the exception of the verdict itself (which will remain unchanged). Therefore, we may denote by yes, no, end any verdict state in $(M, v_c, v_t) \in S_{\mathbb{M}}$ such that $M \in \mathsf{VYES}$, $M \in \mathsf{VNO}$, $M \in \mathsf{VEND}$, respectively. Strictly speaking, yes, no, end may be viewed as equivalent classes grouping states with acceptance, rejection and termination verdicts, respectively. In order to simplify the writing, we abuse notation by defining an equality instead. For all $(M, v_c, v_t), (N, v_c', v_t') \in S_{\mathbb{M}}$:

$$(M, v_c, v_t) = (N, v_c', v_t') = \mathsf{yes} \text{ iff } M \in \mathsf{VYES} \text{ and } N \in \mathsf{VYES}$$

$$(M, v_c, v_t) = (N, v_c', v_t') = \mathsf{no} \text{ iff } M \in \mathsf{VNO} \text{ and } N \in \mathsf{VNO}$$

$$(M, v_c, v_t) = (N, v_c', v_t') = \mathsf{end} \text{ iff } M \in \mathsf{VEND} \text{ and } N \in \mathsf{VEND}$$

For example, we write $m \xrightarrow{\theta}\!\!\!\!\rightarrow \mathsf{yes}$ if there exists a state (N, v_c, v_t) with $N \in \mathsf{VYES}$ reachable from m along θ, and similarly $m \xrightarrow{\theta}\!\!\!\!\rightarrow_{inev} \mathsf{yes}$ if all states (N, v_c, v_t) reachable along θ from m satisfy $N \in \mathsf{VYES}$.

4.2 Instrumentation

The notion of a monitor "performing an action" is in fact misleading, since a monitor is rather a *passive entity* that *follows* an SUM. We need an operator to compose a monitor with an SUM accordingly; this is known as *instrumentation*. We define next the rules of the instrumentation operator $\lhd$, adapted to the timed settings from [2]. Hereafter, recall $\mathbb{M} = \langle S_{\mathbb{M}}, \mathcal{L}, \longrightarrow_{\mathbb{M}} \rangle$ the TLTS describing the behaviour of all monitors, and let $P = \langle S_P, \text{LAB}, \longrightarrow_P \rangle$ be an arbitrary TLTS describing the behaviour of an SUM. That is, we do not assume any particular form of states in S_P or rules in $\longrightarrow_P$, since the SUM can be e.g., a blackbox system.

▶ **Definition 4.8** (Instrumentation). *We define the* monitored system *using the instrumentation operator $\lhd$ as follows: $\mathbb{M} \lhd P = \langle S_{\mathbb{M}} \times S_P, \text{LAB}, \longrightarrow_{\lhd} \rangle$. The transition relation $\longrightarrow_{\lhd}$ is defined in Figure 9, where $m, m' \in S_{\mathbb{M}}$, $p, p' \in S_P$, $\alpha \in \mathcal{L}$ and $a \in \Sigma$. To save notations, we use $m \lhd p$ for $(m, p) \in S_{\mathbb{M}} \times S_P$.*

$$\frac{p \xrightarrow{\alpha}_P p' \quad m \xrightarrow{\alpha}_{\mathbb{M}} m'}{m \lhd p \xrightarrow{\alpha}_{\lhd} m' \lhd p'}\ \textsc{iMon} \qquad \frac{p \xrightarrow{\tau}_P p'}{m \lhd p \xrightarrow{\tau}_{\lhd} m \lhd p'}\ \textsc{iAsy}$$

$$\frac{p \xrightarrow{a}_P p' \quad m \xrightarrow{a}\!\!\!\!\!\!\not\rightarrow_{\mathbb{M}} \quad m = (N, v_c, v_t)}{m \lhd p \xrightarrow{a}_{\lhd} (\mathsf{END}, v_c, v_t) \lhd p'}\ \textsc{iTer}$$

Figure 9 SOS rules for Instrumentation.

Rule iMon states that the monitor must follow any action or delay transition performed by the SUM, whereas rule iAsy corresponds to the SUM performing a τ transition. Finally, if the SUM may perform an action a and the monitor may not, rule iTer allows the SUM to perform a while forcing the monitor to reach a verdict state end.

▶ Remark 4.9. Note that iTer is defined only for actions and not for time delays. This is because a monitor can always perform time delays (rule mTime, Figure 7) and therefore it can always follow the SUM if it performs a delay.

Preserving the SUM behaviours. We prove, through strong timed bisimulation [1], that the instrumentation operator *preserves* the behaviours of the SUM, i.e., instrumenting any SUM P with any monitor M preserves *exactly* all the behaviours of P. Accordingly, given a finite run $\theta \in \text{LAB}^*$, performed by the SUM, we say that a monitor *consumes* the weak run $\theta' \in \mathcal{L}^*$ corresponding to θ (Section 2.1).

▶ **Example 4.10.** Let us consider the monitor M, starting at state $m_0 = (M, 0, \mathbf{0})$ (Example 4.5), and an SUM performing a finite run $\theta = 2\,\tau\,4\,\tau\,c$ from state p to state p'. We have then $m_0 \lhd p \xrightarrow{\theta} \mathsf{yes} \lhd p'$, which corresponds to M consuming $\theta' = 6\,c$, the weak finite run corresponding to θ. Informally, the monitor M checks the following property at runtime: "if the first consumed action occurs within 5 time units, then such action must be a b".

5 Monitorability of T^{lin}

In this section, we present the monitorability results on T^{lin}. In particular, we characterise its largest violation-, satisfaction- and complete-monitorable fragments.

5.1 Setting the framework

5.1.1 Abstracting Away the Instrumentation Operator

To alleviate notations, we abstract away the instrumentation operator through considering only *reactive* monitors, i.e., monitors that can follow any run from any state of any SUM (i.e., consume any run in $\mathcal{L}^*$). This choice should introduce no loss of generality: in Section 5.2, we show how it is rather straightforward to synthesise a reactive monitor that intuitively "corresponds" to (verifies online) any formula in T^{lin}.

▶ **Definition 5.1** (Monitor Reactivity (extended from [2])). *A monitor $M \in \textsc{Mon}$ is reactive iff* $\forall \theta \in \mathcal{L}^*, v_c \in \mathbb{R}_{\geq 0}, v_t \in \mathcal{V} : (M, v_c, v_t) \xrightarrow{\theta}\!\!\!\!\rightarrow$

Since we are in a linear-time setting, i.e., the set of behaviours of an SUM equals the set of its infinite runs, all behaviours of all SUMs simply collapse into all infinite runs in $\textsc{Lab}^\omega$. We write accordingly $m \xrightarrow{\theta}\!\!\!\!\rightarrow m'$ to denote a reactive monitor consuming weak run θ. This way, we can simply abstract away SUMs and the instrumentation operator, the former replaced with the sum of all infinite runs in $\textsc{Lab}^\omega$ and the latter collapsing into simple consumption of weak runs that is always possible thanks to the reactivity assumption.

5.1.2 Monitor Initial State

Intuitively, in order to assess whether M can monitor ϕ, monitoring should start at state $m_0 = (M, 0, \mathbf{0})$. That is, monitor M is *initialised* at state m_0, at which its time (monitor time) starts. Since monitor time may only increase (through consuming delays), it refers to the relative time since the monitoring of ϕ started. Examples of a monitor M starting at state m_0 have been already discussed within the previous section, e.g., in Example 4.5.

5.1.3 Timed Words, Weak Runs and Time Divergence

First, we make an important, yet natural assumption on infinite timed words, namely *time divergence*. To explain this, let us consider the following infinite timed word: $\pi = (a_1, \frac{1}{2}), (a_2, \frac{3}{4}), (a_3, \frac{7}{8}), (a_4, \frac{15}{16}), \ldots$ One can notice that π corresponds to a weak run with an initial delay of $\frac{1}{2}$, and each subsequent delay equalling half of the previous delay. This results in the timestamps of π increasing but never beyond absolute time 1 (since $\sum_{k=1}^{\infty}(\frac{1}{2^k}) = 1$). The infinite timed word π is *time convergent*, and it is clear to see that there is no "real" system that can produce such timed word; for the simple reason that time always diverges in reality (see e.g., [23, 8][18, Sect.2]). An infinite timed word π is time divergent iff its timestamps are unbounded, formally [23, Sect. 4], [7, Def. 1]:

$$\forall x \in \mathbb{R}, \exists i \in \mathbb{N} : t_i(\pi) > x \tag{3}$$

We restrict ourselves accordingly to the set $T\Sigma^\omega_{div} = \{\pi \in T\Sigma^\omega \mid \pi \text{ satisfies (3) }\}$ henceforth.

Second, as we will see later on, central notions on monitorability, such as monitor soundness and completeness, gravitate around the ability to reach a verdict, w.r.t. whether an infinite timed word $\pi \in T\Sigma^\omega_{div}$ is in the semantics of the monitored formula, based on

consuming a finite prefix of π. We need therefore to define the notion of consuming a finite timed word. Since (i) we already defined the notion of consuming a weak run (Section 4), and (ii) a finite timed word necessarily corresponds to a finite weak run in $\mathcal{L}^*$ (Section 2.2), we simply need to define a function to retrieve the latter from the former.

▶ **Definition 5.2** (From finite timed words to finite weak runs)**.**

$$\textsc{Trn} : T\Sigma^* \longrightarrow \mathcal{L}^*$$

where $\textsc{Trn}(\pi) = d_1 a_1 d_2 a_2 \ldots$ *with* $d_i = t_i - t_{i-1}$ *and* $t_0 = 0$*, and the convention that* $\textsc{Trn}(\pi)$ *gives the empty weak run* ϵ *if* π *is an empty timed word.*

In the remainder of this paper, by abuse of terminology, we often say a monitor "consumes (the finite timed word) π" instead of "consumes (the finite weak run) $\textsc{Trn}(\pi)$"

▶ **Example 5.3.** Consider the finite timed word $\pi = (a, 1.7)(b, 1.7)(b, 5.7)$. The corresponding weak run is obtained using Definition 5.2: $\textsc{Trn}(\pi) = 1.7\, a\, 0\, b\, 4\, b$.

▶ Remark 5.4. Note that time divergence does not pose any particular restriction on finite timed words, i.e., $\{Pf(\pi) \mid \pi \in T\Sigma^\omega_{div}\} = T\Sigma^*$. In words, any finite timed word (in $T\Sigma^*$) can be retrieved as a prefix of a time-divergent infinite timed word, see [8].

5.1.4 Monitor Soundness and Completeness

We now define the central monitor properties of *soundness* and *completeness* extended from [2, Definition 4.1] to our timed setting.

▶ **Definition 5.5** (Monitor Soundness)**.** *A reactive monitor* M *is sound for a formula* $\phi \in T^{lin}$ *if, for every finite timed word* $\pi \in T\Sigma^*$*:*

- $m_0 \overset{\textsc{Trn}(\pi)}{\longrightarrow\!\!\!\!\!\rightarrow}$ yes *implies* $(\forall \pi' \in Ext(\pi) : \pi' \in [\![\phi]\!])$
- $m_0 \overset{\textsc{Trn}(\pi)}{\longrightarrow\!\!\!\!\!\rightarrow}$ no *implies* $(\forall \pi' \in Ext(\pi) : \pi' \notin [\![\phi]\!])$

Sound monitors have the important property of *consistency*.

▶ **Definition 5.6** (Consistent monitors)**.** *A reactive monitor* M *is consistent iff* $\forall \pi \in T\Sigma^*$:

$m_0 \overset{\textsc{Trn}(\pi)}{\longrightarrow\!\!\!\!\!\rightarrow}$ yes *implies* $m_0 \overset{\textsc{Trn}(\pi)}{\not\longrightarrow\!\!\!\!\!\rightarrow}$ no *and* $m_0 \overset{\textsc{Trn}(\pi)}{\longrightarrow\!\!\!\!\!\rightarrow}$ no *implies* $m_0 \overset{\textsc{Trn}(\pi)}{\not\longrightarrow\!\!\!\!\!\rightarrow}$ yes

▶ **Lemma 5.7.** *If* M *is sound for some* $\phi \in T^{lin}$ *then* M *is consistent.*

▶ **Definition 5.8** (Monitor Completenness)**.** *Let* $M \in \textsc{Mon}$ *be a reactive monitor and* $\phi \in T^{lin}$ *a formula.* M *is:*

- *violation complete for* ϕ *iff for all* $\pi \in T\Sigma^\omega_{div}$*,* $\pi \notin [\![\phi]\!]$ *implies* $\exists \pi' \in Pf(\pi) : m_0 \overset{\textsc{Trn}(\pi')}{\longrightarrow\!\!\!\!\!\rightarrow}$ no
- *satisfaction complete for* ϕ *iff for all* $\pi \in T\Sigma^\omega_{div}$*,* $\pi \in [\![\phi]\!]$ *implies* $\exists \pi' \in Pf(\pi) : m_0 \overset{\textsc{Trn}(\pi')}{\longrightarrow\!\!\!\!\!\rightarrow}$ yes
- *complete for* ϕ *if it is both violation and satisfaction complete for* ϕ*.*

Strong completeness. In practice, it is desirable to have *reliable* monitors, i.e., monitors exhibiting a stronger property than violation/satisfaction completeness. Let us explain this through an example.

▶ **Example 5.9.** Let $\Sigma = \{b, c\}$ and consider the monitor

$$M' = \text{LET } x \text{ IN } ((x > 5).\mathsf{YES} + (x \leq 5).(b.\mathsf{YES} + c.\mathsf{NO} + b.\mathsf{END} + c.\mathsf{END}))$$

a slight modification of the monitor M in Example 4.5. M' is reactive, and it is rather easy to see that M' is sound and complete for the following T^{lin} formula:

$$\phi = x \,\underline{\texttt{in}}[x \leq 5]\langle b\rangle \mathsf{tt}$$

stating that if the first timestamp is smaller than 5, then the action must be a b. For soundness, if verdict state no is reached, then the first timed event the monitor saw was (c, t) with $t \leq 5$, and it is guaranteed that any infinite extension of the finite timed word (c, t) will be outside the semantics of ϕ. (The case where yes is reached is analogous). For violation completeness, any infinite timed word $\pi \notin [\![\phi]\!]$ starts necessarily with a timed event (c, t) with $t \leq 5$, and given any prefix π' of π of the form (c, t), $t \leq 5$, no is reachable from $(M', 0, \mathbf{0})$ along π'. (The reasoning for satisfaction completeness is analogous). In practice, however, M' is not a reliable monitor for ϕ, as it may reach end instead of no after consuming e.g., $(c, 2)$ (and dually, to reach end instead of yes after consuming e.g., $(b, 3)$). We propose a stronger definition of completeness next.

▶ **Definition 5.10** (Strong Completenness)**.** *Let $M \in \text{MON}$ be a reactive monitor and $\phi \in T^{lin}$ a formula. M is:*

- *strongly violation complete for ϕ iff for all $\pi \in T\Sigma^{\omega}_{div}$, $\pi \notin [\![\phi]\!]$ implies $\exists \pi' \in Pf(\pi)$: $m_0 \overset{\text{TRN}(\pi')}{\longrightarrow\!\!\!\!\!\rightarrow}_{inev} \mathsf{no}$*
- *strongly satisfaction complete for ϕ iff for all $\pi \in T\Sigma^{\omega}_{div}$, $\pi \in [\![\phi]\!]$ implies $\exists \pi' \in Pf(\pi)$: $m_0 \overset{\text{TRN}(\pi')}{\longrightarrow\!\!\!\!\!\rightarrow}_{inev} \mathsf{yes}$*

M is strongly complete for ϕ if it is both strongly violation and strongly satisfaction complete for ϕ.

In brief, the key element in the above definition is the inevitability of verdicts. That is, if a strongly satisfaction- (resp. violation-) complete monitor for some $\phi \in T^{lin}$ is given an infinite timed word π that is in (resp. outside of) the semantics of ϕ, then it will inevitably reach verdict yes (resp. no) after consuming a finite prefix of π. Therefore, such monitor has the important feature of reliability in practice as discussed in Example 5.9 above.

▶ **Example 5.11.** In Example 5.9, the monitor M' is neither strongly satisfaction nor strongly violation complete for ϕ.

▶ Remark 5.12. In this paper, the notion of violation- or satisfaction-monitorable formulae will still rely on the existence of a monitor that obeys soundness (Definition 5.5) and the weaker version of completeness (Definition 5.8). Strong (satisfaction/violation) completeness (Definition 5.10) will be a property of *our monitors*, which we synthesise automatically from T^{lin}, for formulae within the satisfaction/violation/complete monitorable fragments.

5.2 Sound monitor synthesis

In this subsection, we define a compiler from T^{lin} to monitors and prove that, for every formula $\phi \in T^{lin}$, our compiler outputs a monitor that is sound for ϕ (Proposition 5.15).

The compiler $(\!|-|\!) : T^{lin} \longrightarrow \text{MON}$ is defined in Figure 10 (note that monitors generated by $(\!|-|\!)$ are necessarily closed since $(\!|-|\!)$ is applied only to formulae, i.e., elements of T^{lin}). The rules of $(\!|-|\!)$ are rather simple and straightforward, operating on the syntax of the formula

$(\!\|\mathsf{tt}\|\!)$	$=$	YES	$(\!\|\mathsf{ff}\|\!)$	$=$	NO
$(\!\|x \,\underline{\mathtt{in}}\, \phi\|\!)$	$=$	LET x IN $(\!\|\phi\|\!)$	$(\!\|X\|\!)$	$=$	X
$(\!\|[a]\phi\|\!)$	$=$	$a.(\!\|\phi\|\!) + \overline{a}.\mathsf{YES}$	$(\!\|\langle a\rangle\phi\|\!)$	$=$	$a.(\!\|\phi\|\!) + \overline{a}.\mathsf{NO}$
$(\!\|[g]\phi\|\!)$	$=$	$g.(\!\|\phi\|\!) + \overline{g}.\mathsf{YES}$	$(\!\|\langle g\rangle\phi\|\!)$	$=$	$g.(\!\|\phi\|\!) + \overline{g}.\mathsf{NO}$
$(\!\|\phi \wedge \psi\|\!)$	$=$	$(\!\|\phi\|\!) \otimes (\!\|\psi\|\!)$	$(\!\|\phi \vee \psi\|\!)$	$=$	$(\!\|\phi\|\!) \oplus (\!\|\psi\|\!)$
$(\!\|\mathsf{max}(X,\phi)\|\!)$	$=$	rec $X.(\!\|\phi\|\!)$	$(\!\|\mathsf{min}(X,\phi)\|\!)$	$=$	rec $X.(\!\|\phi\|\!)$

Figure 10 $(\!|-|\!)$ rules from T^{lin} to monitors, defined recursively on subterms of formulae.

provided as an input. For example, if the formula is $[a]\langle b\rangle\mathsf{tt}$, whose semantics is "if the first action is a, then the second must be a b" (Section 3.1), then we want a monitor that consumes the first action, and becomes verdict YES if such action is not a, and monitors that the second action is a b otherwise. We obtain thus the monitor $(\!|[a]\langle b\rangle\mathsf{tt}|\!) = a.(b.\mathsf{YES} + \overline{b}.\mathsf{no}) + \overline{a}.\mathsf{YES}$, which corresponds to the description above (under the rules defined in Section 4.1).

▶ **Example 5.13.** The monitor M in Example 4.5 is compiled from the formula: $x \,\underline{\mathtt{in}}[x \leq 5]\langle b\rangle\mathsf{tt}$ over $\Sigma = \{b, c\}$. More complex examples will follow within the next subsections.

▶ **Lemma 5.14.** *For all $\phi \in T^{lin}$, $(\!|\phi|\!)$ is reactive.*

▶ **Proposition 5.15.** *For all $\phi \in T^{lin}$, $(\!|\phi|\!)$ is sound for ϕ.*

5.3 Complete Monitorable Fragment

In this subsection, we identify CT^{lin}, a subset of T^{lin}, and prove its complete monitorability (Theorem 5.26).

▶ **Definition 5.16** (Complete-monitorable formulae)**.** *A formula $\phi \in T^{lin}$ is complete monitorable iff there exists a reactive monitor M that is sound and complete for it. A fragment $\Gamma \subseteq T^{lin}$ is complete monitrorable iff each $\phi \in \Gamma$ is complete monitorable.*

It has been shown in [2] that the recursion-free fragment of untimed linear-time μ-calculus (μ-HML) is its largest complete-monitorable fragment. We show that this result does not hold in the timed setting, i.e., the maximal set of complete-monitorable formulae in T^{lin} is strictly larger than that of μ-HML. To explain this, let us first prove a handy lemma. Recall that $\mathcal{D}$ is the set of timestamp variables and $\mathcal{V}$ all valuation functions over $\mathcal{D}$ (Section 3.1).

▶ **Lemma 5.17.** *Let $y \in \mathcal{D}$ and $n \in \mathbb{N}$. For all $\pi \in T\Sigma^{\omega}_{div}, x \in \mathcal{D} \setminus \{y\}, v \in \mathcal{V}, g \in \mathcal{B}(\mathcal{D})$:*
1. *$\exists i \in \mathbb{N}_{>0}$ such that $v[x := t_i(\pi)] \not\models (x - y \prec n) \wedge g$,*
2. *$\forall i \in \mathbb{N}_{>0}$: $v[x := t_i(\pi)] \not\models (x - y \prec n) \wedge g$ implies $\forall j \geq i$: $v[x := t_i(\pi)] \not\models (x - y \prec n) \wedge g$.*

Let us depict the consequences of the above lemma through a few examples. To ease the explanation, we sometimes use MTL formulae as a support.

▶ **Example 5.18.** Consider the formula $\phi = F\, b$ (Section 2.4), meaning "action b will be seen in the future" (recall that this is the special case of $F_I\, b$ with $I = [0, \infty)$). Clearly, there is no violation-complete monitor for ϕ, because there is no way for a monitor to say no

based on a finite timed word not containing b (since a b can occur later on). In contrast, ϕ is monitorable for satisfactions: it suffices for the monitor to see a b to say yes. It follows that $\phi' = \text{TO}\,T^{lin}(\phi) = \mathsf{min}(Y, \langle b\rangle \mathtt{tt} \vee \langle\Sigma\rangle Y)$ (recalling that $F\,b$ is syntactic sugar for $\top\,\mathcal{U}_I\,b$, then using the function $\text{TO}\,T^{lin}$, Section 3.2) is satisfaction monitorable, but not violation monitorable. Now, in a (time) *constrained* setting, i.e., whenever we use a bounded interval I (with $\mathcal{RB}(I) \in \mathbb{N}$, Definition 2.4), ϕ becomes both satisfaction and violation monitorable, and therefore complete monitorable. Consider the example $\phi_c = F_I\,b$ with $I = [5, 8]$. Its compilation into T^{lin} gives the following formula:

$$\phi'_c = x_s \,\underline{\mathtt{in}}\, \mathsf{min}(Y, (x_e \,\underline{\mathtt{in}} \langle x_e - x_s \in [5,8]\rangle \langle b\rangle \mathtt{tt}) \vee \langle\Sigma\rangle Y)$$

which means "eventually b will occur within [5,8]". The guard $x_e - x_s \in [5, 8]$ is of the form $x - y \prec n \wedge g$ where $y = x_s$ (constant), $n = 8$ (constant), $x = x_e$, $\prec = \leq$, $g = x_e - x_s \geq 5$, and x_e diverges after a finite number of recursions (because time diverges and Y is guarded by $\langle\Sigma\rangle$). Therefore, by Lemma 5.17, $x_e - x_s \in [5, 8]$ will become false after a finite number of recursions, and will remain false indefinitely afterwards. It follows that the monitor can safely say no when the guard becomes false (which means that the consumed prefix until then did not contain any action b between 5 and 8).

▶ **Example 5.19.** Let us now consider the more complex *leads to* formula $\psi = G\,(b \Rightarrow F\,c)$ stipulating that "whenever an action b occurs, an action c follows in the future". As explained in Section 1, this formula cannot be monitored (neither for satisfactions nor for violations). Now consider the *constrained bounded response* $\psi_c = G_I\,(b \Rightarrow F_{I'}\,c)$ with $I = [1, 3]$ and $I' = [2, 5]$; i.e., in which we constrain both temporal operators with bounded intervals. The compilation of ψ_c into T^{lin} (Figure 5) gives the following formula:

$$\psi'_c = x_s \,\underline{\mathtt{in}}\, \mathsf{max}(Y, (x_e \,\underline{\mathtt{in}} [x_e - x_s \in [1,3]]([b]\mathtt{ff} \vee \zeta)) \wedge (\langle\Sigma\rangle Y))$$

with:

$$\zeta = x'_s \,\underline{\mathtt{in}}\, \mathsf{min}(X, (x'_e \,\underline{\mathtt{in}} \langle x'_e - x'_s \in [2,5]\rangle \langle c\rangle \mathtt{tt}) \vee (\langle\Sigma\rangle X))$$

The constrained bounded response formula ψ'_c is also complete monitorable. Indeed, if no b occurs between 1 and 3, the monitor can safely say yes (when the right bound of $[1, 3]$ is exceeded). Otherwise, for every b occurring between 1 and 3, the monitor checks whether a c occurs between 2 and 5 (relative to the occurrence of such b), and says no if it is not the case. Since $[2, 5]$ is bounded, the monitor can decide as soon as it consumes an action more than 5 time units since the last occurrence of b between 1 and 3.

▶ **Example 5.20.** Let us now consider the formulae given in Example 3.5, not expressible in MTL. The formula

$$\phi_{even(b)} = \mathsf{max}(X, \langle\Sigma\rangle\langle b\rangle X)$$

is violation monitorable (a monitor can say no as soon as it sees an action at an even position different than b), but is not satisfaction monitorable. Now, following the same reasoning as in the previous examples (applying Lemma 5.17), the formula

$$\phi_{even(b),[3,5]} = x_s \,\underline{\mathtt{in}}\, \mathsf{max}(X, \langle\Sigma\rangle(x_e \,\underline{\mathtt{in}} \langle x_e - x_s \in [3,5]\rangle \langle b\rangle X \vee \bigvee_{a\in\Sigma} y^a \,\underline{\mathtt{in}} \langle y^a - x_s \notin I\rangle \langle a\rangle X))$$

is complete monitorable.

Syntactic characterisation. In order to characterise a sufficiently expressive syntax for complete-monitorable formulae, we remark the following. Formulae ϕ'_c (Example 5.18) is equivalent to the following formulae:

$$x_s \,\underline{\mathtt{in}}\, \mathsf{min}(Y, (x_e \,\underline{\mathtt{in}}\langle x_e - x_s \in [5,8]\rangle\langle b\rangle\mathtt{tt}) \vee \bigvee_{a\in\Sigma} y^a \,\underline{\mathtt{in}}\langle y^a - x_s \leq 8\rangle\langle a\rangle Y)$$

because when the monitor time exceeds the right bound of $[5,8]$, then there is no need to check if $\langle b\rangle\mathtt{tt}$ is satisfied, and therefore recursion can be prevented (and the monitor can say no at this stage). Formula ψ'_c in Example 5.19 can be transformed similarly to the following equivalent formula:

$$x_s \,\underline{\mathtt{in}}\, \mathsf{max}(Y, (x_e \,\underline{\mathtt{in}}[x_e - x_s \in [1,3]]([b]\mathtt{ff} \vee \zeta')) \wedge \bigvee_{a\in\Sigma} (y^a \,\underline{\mathtt{in}}[y^a - x_s \leq 3]\langle a\rangle Y))$$

where:

$$\zeta' = x'_s \,\underline{\mathtt{in}}\, \mathsf{min}(X, (x'_e \,\underline{\mathtt{in}}\langle x'_e - x'_s \in [2,5]\rangle\langle c\rangle\mathtt{tt}) \vee \bigvee_{a\in\Sigma} (z^a \,\underline{\mathtt{in}}\langle z^a - x'_s \leq 5\rangle\langle a\rangle X))$$

As for formula $\phi_{even(b),[3,5]}$ in Example 5.20, it can be rewritten similarly as follows:

$$x_s \,\underline{\mathtt{in}}\, \mathsf{max}(X, \langle\Sigma\rangle(x_e \,\underline{\mathtt{in}}\langle x_e - x_s \in [3,5]\rangle\langle b\rangle X \vee \bigvee_{a\in\Sigma} y^a \,\underline{\mathtt{in}}(\langle y^a - x_s < 3\rangle\langle a\rangle X \vee \langle y^a - x_s > 5\rangle\langle a\rangle\mathtt{tt})))$$

since there is no need to recurse after the right bound of $[3,5]$ is exceeded (in such case, the timed word is in the semantics of the formula).

More generally, in a complete-monitorable formula ϕ, every subterm $\mathsf{min}(X,\phi)$ or $\mathsf{max}(X,\psi)$ is *constrained*. That is, in every subterm of ϕ, every occurrence of recursion variable X is preceded by $x \,\underline{\mathtt{in}}\langle g\rangle\langle a\rangle$ or $x \,\underline{\mathtt{in}}[g]\langle a\rangle$ such that g becomes eventually false, and $a \in \Sigma$. To simplify the proofs, we use a straightforward syntactic manipulation in order to require recursion variables to be immediately preceded by $x \,\underline{\mathtt{in}}\langle g\rangle$ or $x \,\underline{\mathtt{in}}[g]$: every $x \,\underline{\mathtt{in}}\langle g\rangle\langle a\rangle$ preceding a recursion variable becomes $x \,\underline{\mathtt{in}}\langle g\rangle\langle a\rangle x' \,\underline{\mathtt{in}}\langle g'\rangle$ where in g' the timestamp variable x is replaced with x'. For example, the formula

$$\zeta' = x'_s \,\underline{\mathtt{in}}\, \mathsf{min}(X, (x'_e \,\underline{\mathtt{in}}\langle x'_e - x'_s \in [2,5]\rangle\langle c\rangle\mathtt{tt}) \vee \bigvee_{a\in\Sigma} (z^a \,\underline{\mathtt{in}}\langle z^a - x'_s \leq 5\rangle\langle a\rangle X))$$

above is semantically equivalent to

$$\zeta'' = x'_s \,\underline{\mathtt{in}}\, \mathsf{min}(X, (x'_e \,\underline{\mathtt{in}}\langle x'_e - x'_s \in [2,5]\rangle\langle c\rangle\mathtt{tt}) \vee \bigvee_{a\in\Sigma} (z^a \,\underline{\mathtt{in}}\langle z^a - x'_s \leq 5\rangle\langle a\rangle y^a \,\underline{\mathtt{in}}\langle y^a - x'_s \leq 5\rangle X))$$

In order to syntactically specify that some guard becomes unsatisfied after a finite number of recursions, we introduce cLVar, an infinite subset of logical variables Var, such that a variable that belongs to cLVar must appear in a particular form within the scope of a fixpoint operator.

▶ **Definition 5.21** (Complete-monitorable fragment). *A formula $\phi \in T^{lin}$ is in the set $CT^{lin} \subset T^{lin}$ of complete-monitorable formulae iff ϕ is generated by the grammar in Figure 11 (with $n \in \mathbb{N}$, $g \in \mathcal{B}(\mathcal{D})$, $x, y \in \mathcal{D}$ and $X \in$ cLVar)* and *for all $\mathsf{min}(X,\psi)$, $\mathsf{max}(X,\psi)$ subterms of ϕ, y is free in ψ[5]. A subterm $\mathsf{max}(X,\phi)$ (resp. $\mathsf{min}(X,\phi)$) of a formula in CT^{lin} is called a constrained max (resp. constrained min).*

$$
\begin{array}{lcllllll}
\phi, \psi & ::= & \mathsf{tt} & \mid \mathsf{ff} & \mid \phi \vee \psi & \mid \phi \wedge \psi & \mid x \,\underline{\mathtt{in}}\, \phi \\
 & & \mid \langle a \rangle \phi & \mid [a]\phi & \mid \langle g \rangle \phi & \mid [g]\phi & \\
 & & \mid \mathsf{max}(X, \phi) & \mid \mathsf{min}(X, \phi) & \mid x \,\underline{\mathtt{in}}\, [g_x] X & \mid x \,\underline{\mathtt{in}}\, \langle g_x \rangle X & \\
g_x & ::= & x - y \prec n & \mid g_x \wedge g &
\end{array}
$$

Figure 11 Grammar to generate formulae (closed and guarded terms) of CT^{lin}.

To prove that CT^{lin} is complete monitorable, we show that $(\!|\phi|\!)$ is sound and *strongly* complete for every $\phi \in CT^{lin}$. Strong completeness allows us to both guarantee completeness *and* show that our monitors (synthesised by $(\!|-|\!)$) are reliable for CT^{lin}.

▶ **Proposition 5.22.** *For all $\phi \in CT^{lin}$, $(\!|\phi|\!)$ is sound for ϕ.*

Proof. Since $CT^{lin} \subset T^{lin}$, we conclude by Proposition 5.15. ◀

▶ **Proposition 5.23.** *For all $\phi \in CT^{lin}$, $(\!|\phi|\!)$ is strongly satisfaction complete for ϕ.*

▶ **Proposition 5.24.** *For all $\phi \in CT^{lin}$, $(\!|\phi|\!)$ is strongly violation complete for ϕ.*

▶ **Example 5.25.** Consider the following formula:

$$\phi = x_s \,\underline{\mathtt{in}}\, \mathsf{min}(X, (x \,\underline{\mathtt{in}}\, \langle x - x_s \in (3,7] \rangle \langle b \rangle y \,\underline{\mathtt{in}}\, \langle y - x \in [0,2) \rangle \langle c \rangle \mathsf{tt}) \vee \bigvee_{a \in \Sigma} z^a \,\underline{\mathtt{in}}\, \langle z^a - x_s \leq 7 \rangle \langle a \rangle X)$$

easily expressible in CT^{lin} (see above), which expresses the following property "there exists an occurrence of b within interval $(3,7]$ that is immediately followed by a c within the interval $[0,2)$"; where "immediately followed by" means that there are no other actions occurring in between. In other words, the semantics of ϕ contains all infinite timed words π such that there exists a position i with $a_i(\pi) = b$ *and* $t_i(\pi) - t_1(\pi) \in (3,7]$ *and* $a_{i+1}(\pi) = c$ *and* $t_{i+1}(\pi) - t_i(\pi) \in [0,2)$. The compilation of ϕ into a monitor gives:

$$M = (\!|\phi|\!) = \textsc{let}\ x_s\ \textsc{in}\ \mathsf{rec}\ X.((\textsc{let}\ x\ \textsc{in}\ (((x - x_s \in (3,7]).(b.N + \overline{b}.\mathsf{NO})) +$$
$$(x - x_s \notin (3,7]).\mathsf{NO})) \oplus (\oplus_{a \in \Sigma}\ \textsc{let}\ z^a\ \textsc{in}\ (((z^a - x_s \leq 7).(a.X + \overline{a}.\mathsf{NO})) + (z^a - x_s > 7).\mathsf{NO})))$$

where

$$N = \textsc{let}\ y\ \textsc{in}\ (((y - x \in [0,2)).(c.\mathsf{YES} + \overline{c}.\mathsf{NO})) + (y - x \notin [0,2)).\mathsf{NO})$$

and M is sound and strongly complete for ϕ. For e.g., strong completeness, for any $\pi \in T\Sigma^{\omega}_{div}$ in the semantics of ϕ (resp. not in the semantics of ϕ), the monitor will inevitably reach a verdict state yes (resp. no) after consuming a finite prefix of π. For example, for any infinite timed word $\pi_1.\pi$ with $\pi_1 = (a,1)(b,8)(c,9)$ and $\pi \in T\Sigma^{\omega}_{\vec{\pi_1}}$ (resp. $\pi_2.\pi$ with $\pi_2 = (a,2)(b,6)(c,8)$ and $\pi \in T\Sigma^{\omega}_{\vec{\pi_2}}$) we have $m_0 \xrightarrow{\textsc{Trn}(\pi_1)}{}\!\!\!\!\rightarrow_{inev} \mathsf{yes}$ (resp. $m_0 \xrightarrow{\textsc{Trn}(\pi_2)}{}\!\!\!\!\rightarrow_{inev} \mathsf{no}$).

▶ **Theorem 5.26.** *CT^{lin} is complete monitorable.*

Proof. For every $\phi \in CT^{lin}$, $(\!|\phi|\!)$ is sound (Proposition 5.22), satisfaction complete (implied by Proposition 5.23) and violation complete (implied by Proposition 5.24) for ϕ. It follows that CT^{lin} is complete monitorable (Definition 5.16). ◀

[5] This is to guarantee that, within the scope of $\mathsf{max}(Y, \psi)$ or $\mathsf{min}(X, \psi)$, the valuation of y remains constant and so Lemma 5.17 can be applied.

5.4 Monitorabile Fragment

In this subsection, we identify MT^{lin}, a fragment of T^{lin}, and prove its monitorability (Theorem 5.38).

▶ **Definition 5.27** (Monitorable formulae)**.**

- *A formula $\phi \in T^{lin}$ is* violation *(resp.* satisfaction*)* monitorable *iff there exists a reactive monitor M that is sound and violation (resp. satisfaction) complete for ϕ. A fragment $\Gamma \subseteq T^{lin}$ is violation (resp. satisfaction) monitorable iff each $\phi \in \Gamma$ is violation (resp. satisfaction) monitorable.*
- *A formula $\phi \in T^{lin}$ is* monitorable *iff it is violation* or *satisfaction monitorable. A fragment $\Gamma \subseteq T^{lin}$ is monitorable iff each $\phi \in \Gamma$ is monitorable.*

In order to identify a monitorable fragment, let us first proceed with examples.

▶ **Example 5.28.** Consider the following T^{lin} formula:

$$\phi = \mathsf{max}(Y, ([b]\mathsf{ff} \vee \psi) \wedge (\langle\Sigma\rangle Y))$$

with:

$$\psi = x_s \,\underline{\mathtt{in}}\, \mathsf{min}(X, (x_e \,\underline{\mathtt{in}} \langle x_e - x_s \in I\rangle\langle c\rangle\mathsf{tt}) \vee (\langle\Sigma\rangle X))$$

with $I \in \mathcal{I}$ bounded ($\mathcal{RB}(I) \in \mathbb{N}$). Formula ϕ expresses the important bounded response property (in MTL: $G\,(b \Rightarrow F_I\, c)$, Section 1). Notice how every min subformula of ϕ can be rewritten as a constrained min (Section 5.3), whereas the max subformulae are clearly unconstrained. Formula ϕ is violation monitorable: after every occurrence of b, a monitor can check whether a c will follow within I (relative to such occurrence). Since I is bounded, such check always terminates (by Lemma 5.17), and the monitor can say no if no c occurred. However, this formula is not satisfaction monitorable.

▶ Remark 5.29. Bounded response is not *bounded-memory monitorable* in the general case [35] (it is when I is left-closed at 0, see e.g., [17]). Bounded-memory monitorability is out of this paper's scope and is left for future work (Section 7).

▶ **Example 5.30.** Consider now the following T^{lin} formula:

$$\eta = \mathsf{min}(Y, (\langle b\rangle\zeta \vee \langle\Sigma\rangle Y))$$

with:

$$\zeta = x_s \,\underline{\mathtt{in}}\, \mathsf{max}(X, (x_e \,\underline{\mathtt{in}} [x_e - x_s \in I]\langle c\rangle\mathsf{tt}) \wedge (\langle\Sigma\rangle X))$$

with $I \in \mathcal{I}$ bounded. Notice how every max subformula of η can be rewritten as a constrained max (Section 5.3), whereas the min subformulae are clearly not constrained. Formula η is satisfaction monitorable: if a b occurs, a monitor can safely say yes if every action seen within I after such occurrence is a c (this process terminates, again thanks to Lemma 5.17). This formula is however not violation monitorable: any no verdict may be falsified by a future b after which every action within I is a c.

$$\begin{array}{rcllllll} \phi, \psi & ::= & \mathsf{tt} & \mid \mathsf{ff} & \mid \phi \vee \psi & \mid \phi \wedge \psi & \mid x \,\underline{\mathtt{in}}\, \phi \\ & & \mid \langle a \rangle \phi & \mid [a]\phi & \mid \langle g \rangle \phi & \mid [g]\phi & \\ & & \mid \mathsf{min}(X, \phi) & \mid X & & & \end{array}$$

Figure 12 Grammar to generate formulae (closed and guarded terms) of ST^{lin}.

$$\begin{array}{rcllllll} \phi, \psi & ::= & \mathsf{tt} & \mid \mathsf{ff} & \mid \phi \vee \psi & \mid \phi \wedge \psi & \mid x \,\underline{\mathtt{in}}\, \phi \\ & & \mid \langle a \rangle \phi & \mid [a]\phi & \mid \langle g \rangle \phi & \mid [g]\phi & \\ & & \mid \mathsf{max}(Y, \phi) & \mid Y & & & \end{array}$$

Figure 13 Grammar to generate formulae (closed and guarded terms) of VT^{lin}.

Syntactic characterisation. The intuition is that both violation- and satisfaction-monitorable formulae will share the syntax of CT^{lin} and differ for (unconstrained) minimal and maximal fixpoint operators. To simplify the syntax, we prove that the semantics of constrained min and constrained max coincide. That is, for every constrained formula $\mathsf{min}(X, \phi)$, we have the equality $[\![\mathsf{min}(X, \phi)]\!] = [\![\mathsf{max}(X, \phi)]\!]$ (and dually for constrained max formulae). Accordingly, for violation-monitorable formulae, we do not need constrained min in their syntax (Figure 13). This is because constrained min formulae can be rewritten as constrained max formulae, and a constrained max is a special case of max. For example, in the bounded response property (Example 5.28), and after transforming ψ into a constrained min (Section 5.3), the minimal fixpoint operator can be replaced with the maximal one without changing the semantics. Dually, we do not need constrained max in the syntax of satisfaction-monitorable formula (Figure 12).

▶ **Definition 5.31** (Monitorable fragment). *The set of monitorable formulae $MT^{lin} \subset T^{lin}$ is the union of the set of violation-monitorable formulae (Figure 13) and satisfaction-monitorable formulae (Figure 12), i.e., $MT^{lin} = VT^{lin} \cup ST^{lin}$.*

▶ **Proposition 5.32.** *For all $\phi \in MT^{lin}$, $(\!|\phi|\!)$ is sound for ϕ.*

Proof. Since $MT^{lin} \subset T^{lin}$, we conclude by Proposition 5.15. ◀

▶ **Proposition 5.33.** *For all $\phi \in ST^{lin}$, $(\!|\phi|\!)$ is strongly satisfaction complete for ϕ.*

▶ **Theorem 5.34.** *ST^{lin} is satisfaction monitorable.*

Proof. Direct from Proposition 5.32, Proposition 5.33 and Definition 5.27. ◀

▶ **Proposition 5.35.** *For all $\phi \in VT^{lin}$, $(\!|\phi|\!)$ is strongly violation complete for ϕ.*

▶ **Example 5.36.** The formula $\phi_{even(b)} = \mathsf{max}(X, \langle\Sigma\rangle\langle b\rangle X)$ (Example 3.5) is violation monitorable. Its compilation by $(\!|-|\!)$ gives the monitor

$$M = (\!|\phi_{even(b)}|\!) = \mathsf{rec}\ X.(\oplus_{a \in \Sigma}\ a.(b.X + \overline{b}.\mathsf{NO}) + \overline{a}.\mathsf{NO})$$

and it is rather straightforward to see that M is sound and inevitably reaches verdict no (from m_0) when given any infinite timed word outside of the semantics of $\phi_{even(b)}$. Indeed, for any infinite timed word π, we have $\pi \notin [\![\phi_{even(b)}]\!]$ iff there exists a position $i \in \mathbb{N}_{>1}$, $i \bmod 2 = 0$ such that $a_i(\pi) \neq b$. For any such π and the smallest such i, we have $m_0 \overset{\text{TRN}(^i\pi)}{\longrightarrow\!\!\!\!\!\rightarrow}_{inev} \mathsf{no}$.

$$
\begin{array}{llll}
\|M \in \mathsf{VYES}\| = \mathsf{tt} & \|M \in \mathsf{VNO}\| = \|M \in \mathsf{VEND}\| = \mathsf{ff} \\
\|\textsc{let}\ x\ \textsc{in}\ M\| = x\,\underline{\mathtt{in}}\,\|M\| & \|M + N\| = \|M\| \vee \|N\| \\
\|M \oplus N\| = \|M\| \vee \|N\| & \|M \otimes N\| = \|M\| \wedge \|N\| \\
\|\alpha.M\| = \langle\alpha\rangle\|M\| & \|g.M\| = \langle g\rangle\|M\| \\
\|X\| = X & \|\mathsf{rec}\ X.M\| = \mathsf{min}(X, \|M\|)
\end{array}
$$

Figure 14 Compiler from reactive monitors to ST^{lin}, defined recursively on subterms of monitors.

$$
\begin{array}{llll}
\langle\langle M \in \mathsf{VYES}\rangle\rangle = \langle\langle M \in \mathsf{VEND}\rangle\rangle = \mathsf{tt} & \langle\langle M \in \mathsf{VNO}\rangle\rangle = \mathsf{ff} \\
\langle\langle\textsc{let}\ x\ \textsc{in}\ M\rangle\rangle = x\,\underline{\mathtt{in}}\langle\langle M\rangle\rangle & \langle\langle M + N\rangle\rangle = \langle\langle M\rangle\rangle \wedge \langle\langle N\rangle\rangle \\
\langle\langle M \oplus N\rangle\rangle = \langle\langle M\rangle\rangle \vee \langle\langle N\rangle\rangle & \langle\langle M \otimes N\rangle\rangle = \langle\langle M\rangle\rangle \wedge \langle\langle N\rangle\rangle \\
\langle\langle\alpha.M\rangle\rangle = [\alpha]\langle\langle M\rangle\rangle & \langle\langle g.M\rangle\rangle = [g]\langle\langle M\rangle\rangle \\
\langle\langle X\rangle\rangle = X & \langle\langle\mathsf{rec}\ X.M\rangle\rangle = \mathsf{max}(X, \langle\langle M\rangle\rangle)
\end{array}
$$

Figure 15 Compiler from reactive monitors to VT^{lin}, defined recursively on subterms of monitors.

▶ **Theorem 5.37.** *VT^{lin} is violation monitorable.*

Proof. Direct from Proposition 5.32, Proposition 5.35 and Definition 5.27. ◀

▶ **Theorem 5.38.** *MT^{lin} is monitorable.*

Proof. Direct from Theorem 5.34, Theorem 5.37, Definition 5.31 and Definition 5.27. ◀

5.5 Maximality

▶ **Definition 5.39** (Largest monitorable fragment)**.** *A fragment $\Gamma \subset T^{lin}$ is said:*

- *The largest satisfaction- (resp. violation-, complete-) monitorable fragment of T^{lin} iff (i) Γ is satisfaction (resp. violation, complete) monitorable and (ii) for every satisfaction- (resp. violation-, complete-) monitorable formula $\psi \in T^{lin}$, there exists $\phi \in \Gamma$ such that $[\![\phi]\!] = [\![\psi]\!]$,*
- *The largest monitorable fragment of T^{lin} iff (i) Γ is monitorable and (ii) for every monitorable formula $\psi \in T^{lin}$, there exists $\phi \in \Gamma$ such that $[\![\phi]\!] = [\![\psi]\!]$.*

In order to prove that MT^{lin} is the largest monitorable fragment of T^{lin}, we follow an approach inspired from [2]. First, we prove that for every reactive monitor M, if M is satisfaction (resp. violation) complete for two formulae in T^{lin}, then such formulae are semantically equivalent (Lemma 5.40). Second, we define two functions: $\|-\|$ (Figure 14), that maps any reactive monitor M to a formula in ST^{lin}, and $\langle\langle-\rangle\rangle$ (Figure 15) that maps any reactive monitor M to a formula in VT^{lin}; then prove that any reactive M is sound and satisfaction complete for its image by $\|-\|$ (i.e., $\|M\|$), and sound and violation complete for $\langle\langle M\rangle\rangle$ (Lemma 5.41). We conclude therefore that any formula $\phi \in T^{lin}$ that is satisfaction

(resp. violation) monitorable, through the existence of a monitor M that is satisfaction (resp. violation) complete for ϕ, is necessarily equivalent to $\|M\| \in ST^{lin}$ (resp. $\langle\langle M \rangle\rangle \in VT^{lin}$, Proposition 5.42). Accordingly, ST^{lin} is the largest satisfaction-monitorable fragment of T^{lin}, VT^{lin} the largest violation-monitorable fragment of T^{lin}, and $MT^{lin} = VT^{lin} \cup ST^{lin}$ the largest monitorable fragment of T^{lin} (Theorem 5.43, Theorem 5.44). We also prove that CT^{lin} is the largest complete-monitorable fragment of T^{lin} (Theorem 5.45).

▶ **Lemma 5.40.** *For every reactive monitor M and formulae $\phi, \psi \in T^{lin}$:*

1. *M is sound and satisfaction complete for ϕ and ψ implies $[\![\phi]\!] = [\![\psi]\!]$,*
2. *M is sound and violation complete for ϕ and ψ implies $[\![\phi]\!] = [\![\psi]\!]$.*

▶ **Lemma 5.41.** *For every reactive monitor M, if M is consistent then:*

1. *M is sound and satisfaction complete for $\|M\|$*
2. *M is sound and violation complete for $\langle\langle M \rangle\rangle$*

▶ **Proposition 5.42.** *For every formula $\phi \in T^{lin}$:*

1. *if ϕ is satisfaction monitorable, then there exists $\psi \in ST^{lin}$ such that $[\![\phi]\!] = [\![\psi]\!]$,*
2. *if ϕ is violation monitorable, then there exists $\psi \in VT^{lin}$ such that $[\![\phi]\!] = [\![\psi]\!]$.*

Proof.

Item 1. Assume $\phi \in T^{lin}$ is satisfaction monitorable. By definition, there exists a reactive monitor M that is sound and satisfaction complete for ϕ. By Lemma 5.7, we have M is consistent. Now, by Lemma 5.41 (Item 1), we deduce M is sound and satisfaction complete for $\|M\| \in ST^{lin}$, and finally we conclude $[\![\phi]\!] = [\![\|M\|]\!]$ by Lemma 5.40 (Item 1).

Item 2. Analogous. ◀

▶ **Theorem 5.43.**

1. *ST^{lin} is the largest satisfaction-monitorable fragment of T^{lin},*
2. *VT^{lin} is the largest violation-monitorable fragment of T^{lin}.*

Proof.

Item 1. Direct from Definition 5.39, Theorem 5.34, and Proposition 5.42 (Item 1).

Item 2. Direct from Definition 5.39, Theorem 5.37, and Proposition 5.42 (Item 2). ◀

▶ **Theorem 5.44.** *MT^{lin} is the largest monitorable fragment of T^{lin}.*

Proof. Direct from Definition 5.39, Theorem 5.43 and Definition 5.31. ◀

▶ **Theorem 5.45.** *CT^{lin} is the largest complete-monitorable fragment of T^{lin}.*

▶ Remark 5.46. An important remark here is the fact that MT^{lin} reflects precisely the effect of bounded intervals on monitorability. At a first glance, it may seem that the existence of a bounded interval in the syntax of a formula ϕ is sufficient for ϕ to be monitorable; however, this is generally false. Take for example the MTL formula $\psi = G\,(c \Rightarrow F_{[2,5]}\,(b \wedge F\,c))$. Although $[2, 5]$ is bounded, ψ is not monitorable. For satisfactions, the reasoning is similar to bounded response. For violations, take any infinite timed word π starting with $(c, 1)(b, 4)(b, 8)$ and satisfying $\forall i \in \mathbb{N}_{>3} : a_i(\pi) = b$, then we have $\pi \notin [\![\psi]\!]$. Yet, there is no way for a monitor to say no based on a finite prefix of π. Without a surprise, the compilation of ψ into T^{lin} falls outside of MT^{lin} as it uses both unconstrained min and max subformulae.

6 Related Work

The state of the art on RV of timed properties mainly centres around the study of LTL extensions such as MTL [27] and MITL, the fragment of MTL excluding singular intervals [5]. The use of a declarative logic to describe system behaviour creates a clear delineation between what can be specified and how this can then be verified algorithmically. This is in line with our approach and is more conducive to the study of monitorability. Baldor *et al.* [9] were the first to propose a transition based approach to monitoring of MITL formulae and establish complexity results for their proposed synthesis. Ho *et al.* [24] split unbounded and bounded parts of a dense-time MTL formula to permit a simpler construction for the bounded parts to obtain space-bounded monitors. Neither of these works consider monitorability issues.

The influential work by Bauer *et al.* [12] introduces the idea of an n-valued semantics for a timed version of LTL called TLTL, defined over *trace prefixes* by incorporating inconclusive outcome, typically denoted by "?". This is used as a definition of the expected monitor behaviour when analysing a trace prefix online, which then guides a method of obtaining executable monitors from timed properties. The main drawback of the approach is that monitor synthesis disregards monitorability aspects; it generates monitors for *any* timed property, without information on whether such property is monitorable or not. The recent work on monitoring MITL by Grosen *et al.* [22] also suffers from this drawback as a result of adopting a three-valued semantics. Despite this, they propose new symbolic method of synthesising monitors for MITL properties via a translation into timed Büchi automata. Pinisetty *et al.* [37] consider the monitoring of properties expressed as timed ω-automata. However, they ascribe an n-valued semantics to prefixes of infinite runs as discussed above. The work focuses on how prior knowledge on the system (also expressed as a timed automaton) can be used for sound predictive monitoring.

All of the above works take into account in one sense or another infinite executions; the infinitary nature of traces poses the main complication when considering the monitorability of linear-time properties [2]. Popular work on synthesising monitors for timed LTL properties with finitary semantics, such as that of Thati and Rosu [45] and of Basin *et al.* [11] sidestep any monitorability issues since every formula becomes monitorable. Basin *et al.* consider a first-order version of MTL called MFOTL to describe properties involving data ranging over an infinite domain. Ulus *et al.* [46] describe a monitor algorithm for Timed Regular Expressions based on the union of two-dimensional zones for finite words. Raszyk *et al.* [41] extend this work to an augmented monitoring setup that uses multiple analysis points (i.e., multi-head monitors) that relates to our parallel monitoring setup. Lima *et al.* [32] augment monitoring verdicts of MFOTL with explanations of why the monitored execution led to violations/satisfactions. Roussanaly *et al.* [43] study a decentralised monitor synthesis procedure for timed regular expressions that is similar to our parallel monitors. Note that a number of works study *quantitative monitoring* of STL [34], a version of MITL interpreted over continuous signals, proposing thereby a robustness measure instead of yes/no verdicts. Such works sidestep monitorability problems either through *approximate* measures under infinitary semantics (see e.g., [16]) or more precise measures under finitary semantics [42].

In summary, our approach is complementary to the above efforts. On the one hand, our work is the first that proposes through monitorability results for an expressive timed logic. As shown in Section 3.2, MTL is less expressive than our T^{lin} which both in general and w.r.t. to monitorability; we have shown that there exists monitorable formulae in T^{lin} that are not expressible in MTL (Section 5, Section 5.4). On the other hand, works such as [24] and [22] focus on efficiency of monitoring algorithms rather than monitorability issues.

Studies on monitorability generally consider one of two definitions; the one due to Pnueli and Zaks [39] or that due to Schneider [44]. Aceto *et al.* [4] have shown that the latter definition is strictly stronger and have systematised the various variants in terms of soundness and completeness criteria similar to those we used. In particular, the definition of monitorability assumed in our work is the stronger variant due to Schneider, in part because it allowed us to establish maximality results for our logical fragments; at the time of writing, the methods for establishing maximal monitorable fragments for Pnueli and Zaks monitorability are not well-established.

7 Conclusion

In this paper, we solved the monitorability problem for an expressive timed logic. We proposed T^{lin}, a modal timed μ-calculus with linear-time semantics and proved that it is strictly more expressive than the well-known and popular MTL. Then, we precisely identified the maximal fragments of violation-monitorable, satisfaction-monitorable, complete-monitorable, and monitorable formulae of T^{lin}. To the best of our knowledge, this is the first work that solves the monitorability problem for such expressive timed logic.

A natural direction of future work is to synthesise monitors as timed automata (TA). Such monitors should be simpler than Mon and can be efficiently implemented in TA-based tools. Another direction is the study of bounded-memory monitorability. At least two research axes in this direction are worth investigating. First, the identification of the fragments of bounded-memory monitorable formulae of T^{lin} under the same setting as in this paper (linear time, time-divergent infinite words). Second, analogous to the first axis, the characterisation of the weakest conditions on infinite timed words to enable bounded-memory monitorability for some properties that are not monitorable in bounded memory in the general case.

References

1 L. Aceto, A. Ingólfsdóttir, K. G. Larsen, and J. Srba. *Reactive Systems: Modelling, Specification and Verification*. Cambridge University Press, 2007.

2 Luca Aceto, Antonis Achilleos, Adrian Francalanza, Anna Ingólfsdóttir, and Karoliina Lehtinen. Adventures in monitorability: from branching to linear time and back again. *Proc. ACM Program. Lang.*, 3(POPL):52:1–52:29, 2019. doi:10.1145/3290365.

3 Luca Aceto, Antonis Achilleos, Adrian Francalanza, Anna Ingólfsdóttir, and Karoliina Lehtinen. An operational guide to monitorability. In *International Conference on Software Engineering and Formal Methods (SEFM)*, pages 433–453. Springer, 2019. doi:10.1007/978-3-030-30446-1_23.

4 Luca Aceto, Antonis Achilleos, Adrian Francalanza, Anna Ingólfsdóttir, and Karoliina Lehtinen. An operational guide to monitorability with applications to regular properties. *Software and Systems Modeling*, 20:335–361, 2021. doi:10.1007/s10270-020-00860-z.

5 Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996. doi:10.1145/227595.227602.

6 Rajeev Alur and Thomas A Henzinger. Real-time logics: complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993. doi:10.1006/inco.1993.1025.

7 Rajeev Alur and Thomas A Henzinger. A really temporal logic. *Journal of the ACM (JACM)*, 41(1):181–203, 1994. doi:10.1145/174644.174651.

8 Rajeev Alur and Thomas A Henzinger. Modularity for timed and hybrid systems. In *International Conference on Concurrency Theory (CONCUR)*, pages 74–88. Springer, 1997. doi:10.1007/3-540-63141-0_6.

9 Kevin Baldor and Jianwei Niu. Monitoring dense-time, continuous-semantics, metric temporal logic. In *International Conference on Runtime Verification (RV)*, pages 245–259. Springer, 2012. doi:10.1007/978-3-642-35632-2_24.

10 Ezio Bartocci, Yliès Falcone, Adrian Francalanza, and Giles Reger. Introduction to runtime verification. *Lectures on Runtime Verification: Introductory and Advanced Topics*, pages 1–33, 2018. doi:10.1007/978-3-319-75632-5_1.

11 David A. Basin, Felix Klaedtke, Samuel Müller, and Eugen Zalinescu. Monitoring metric first-order temporal properties. *J. ACM*, 62(2):15:1–15:45, 2015. doi:10.1145/2699444.

12 Andreas Bauer, Martin Leucker, and Christian Schallhart. Runtime verification for LTL and TLTL. *ACM Trans. Softw. Eng. Methodol.*, 20(4):14:1–14:64, 2011. doi:10.1145/2000799.2000800.

13 Edmund Clarke, Orna Grumberg, and Kiyoharu Hamaguchi. Another look at LTL model checking. In *International Conference on Computer Aided Verification (CAV)*, pages 415–427. Springer, 1994. doi:10.1007/3-540-58179-0_72.

14 Rance Cleaveland, Jeroen JA Keiren, and Peter Fontana. An expressive timed modal mu-calculus for timed automata. In *International Conference on Quantitative Evaluation of Systems and Formal Modeling and Analysis of Timed Systems (QEST+FORMATS)*, pages 160–178. Springer, 2024. doi:10.1007/978-3-031-68416-6_10.

15 Christian Dax, Martin Hofmann, and Martin Lange. A proof system for the linear time μ-calculus. In S. Arun-Kumar and Naveen Garg, editors, *International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 273–284. Springer, 2006. doi:10.1007/11944836_26.

16 Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, pages 92–106. Springer, 2010. doi:10.1007/978-3-642-15297-9_9.

17 Mohammed Foughali, Saddek Bensalem, Jacques Combaz, and Félix Ingrand. Runtime verification of timed properties in autonomous robots. In *International Conference on Formal Methods and Models for System Design (MEMOCODE)*, pages 1–12. IEEE, 2020. doi:10.1109/MEMOCODE51338.2020.9315156.

18 Mohammed Foughali, Pierre-Emmanuel Hladik, and Alexander Zuepke. Compositional verification of embedded real-time systems. *J. Syst. Archit.*, 142:102928, 2023. doi:10.1016/j.sysarc.2023.102928.

19 A. Francalanza, L. Aceto, and A. Ingólfsdóttir. Monitorability for the Hennessy-Milner logic with recursion. *Formal Methods Syst. Des.*, 51(1), 2017. doi:10.1007/s10703-017-0273-z.

20 Adrian Francalanza. A theory of monitors. *Inf. Comput.*, 281:104704, 2021. doi:10.1016/j.ic.2021.104704.

21 Adrian Francalanza, Luca Aceto, Antonis Achilleos, Duncan Paul Attard, Ian Cassar, Dario Della Monica, and Anna Ingólfsdóttir. A foundation for runtime monitoring. In *RV*, volume 10548 of *Lecture Notes in Computer Science*, pages 8–29. Springer, 2017. doi:10.1007/978-3-319-67531-2_2.

22 Thomas Møller Grosen, Sean Kauffman, Kim Guldstrand Larsen, and Martin Zimmermann. Monitoring timed properties (revisited). In *International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, pages 43–62. Springer, 2022. doi:10.1007/978-3-031-15839-1_3.

23 Thomas A Henzinger. Sooner is safer than later. *Information Processing Letters*, 43(3):135–141, 1992. doi:10.1016/0020-0190(92)90005-G.

24 Hsi-Ming Ho, Joël Ouaknine, and James Worrell. Online monitoring of metric temporal logic. In *International Conference on Runtime Verification (RV)*, pages 178–192. Springer, 2014. doi:10.1007/978-3-319-11164-3_15.

25 Ullrich Hustadt, Ana Ozaki, and Clare Dixon. Theorem proving for pointwise metric temporal logic over the naturals via translations. *Journal of Automated Reasoning*, 64(8):1553–1610, 2020. doi:10.1007/s10817-020-09541-4.

26 Donald Ervin Knuth. Semantics of context-free languages. *Mathematical systems theory*, 2:127–145, 1968. doi:10.1007/BF01692511.

27 Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-time systems*, 2(4):255–299, 1990. doi:10.1007/BF01995674.

28 Dexter Kozen. Results on the propositional μ-calculus. *Theoretical computer science*, 27(3):333–354, 1983. doi:10.1016/0304-3975(82)90125-6.

29 Kim G Larsen. Proof systems for satisfiability in Hennessy-Milner logic with recursion. *Theoretical Computer Science*, 72(2-3):265–288, 1990. doi:10.1016/0304-3975(90)90038-J.

30 Martin Leucker and Christian Schallhart. A brief account of runtime verification. *The journal of logic and algebraic programming*, 78(5):293–303, 2009. doi:10.1016/j.jlap.2008.08.004.

31 Orna Lichtenstein and Amir Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Symposium on Principles of Programming Languages (POPL)*, pages 97–107, 1985. doi:10.1145/318593.318622.

32 Leonardo Lima, Andrei Herasimau, Martin Raszyk, Dmitriy Traytel, and Simon Yuan. Explainable online monitoring of metric temporal logic. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 473–491. Springer, 2023. doi:10.1007/978-3-031-30820-8_28.

33 Frank D Macías-Escrivá, Rodolfo Haber, Raul Del Toro, and Vicente Hernandez. Self-adaptive systems: A survey of current approaches, research challenges and applications. *Expert Systems with Applications*, 40(18):7267–7279, 2013. doi:10.1016/j.eswa.2013.07.033.

34 Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT)*, volume 3253, pages 152–166. Springer, 2004. doi:10.1007/978-3-540-30206-3_12.

35 Oded Maler, Dejan Nickovic, and Amir Pnueli. On synthesizing controllers from bounded-response properties. In *International Conference on Computer Aided Verification (CAV)*, pages 95–107. Springer, 2007. doi:10.1007/978-3-540-73368-3_12.

36 Joël Ouaknine and James Worrell. On metric temporal logic and faulty turing machines. In *International Conference on Foundations of Software Science and Computation Structures*, pages 217–230. Springer, 2006. doi:10.1007/11690634_15.

37 Srinivas Pinisetty, Thierry Jéron, Stavros Tripakis, Yliès Falcone, Hervé Marchand, and Viorel Preoteasa. Predictive runtime verification of timed properties. *Journal of Systems and Software*, 132:353–365, 2017. doi:10.1016/j.jss.2017.06.060.

38 Amir Pnueli. The temporal logic of programs. In *Annual Symposium on Foundations of Computer Science (SFCS)*, pages 46–57. ieee, 1977. doi:10.1109/SFCS.1977.32.

39 Amir Pnueli and Aleksandr Zaks. PSL model checking and run-time verification via testers. In *International Symposium on Formal Methods (FM)*, pages 573–586. Springer, 2006. doi:10.1007/11813040_38.

40 Vasumathi Raman, Mehdi Maasoumy, and Alexandre Donzé. Model predictive control from signal temporal logic specifications: A case study. In *International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*, pages 52–55, 2014. doi:10.1145/2593458.2593472.

41 Martin Raszyk, David Basin, Srđan Krstić, and Dmitriy Traytel. Multi-head monitoring of metric temporal logic. In *International Symposium on Automated Technology for Verification and Analysis (ATVA)*, pages 151–170. Springer, 2019. doi:10.1007/978-3-030-31784-3_9.

42 Neha Rino, Mohammed Foughali, and Eugene Asarin. Efficiently computable distance-based robustness for a practical fragment of STL. In *International Conference on Quantitative Evaluation of Systems and Formal Modeling and Analysis of Timed Systems (QEST+FORMATS)*, pages 179–195. Springer, 2024. doi:10.1007/978-3-031-68416-6_11.

43 Victor Roussanaly and Yliès Falcone. Decentralised runtime verification of timed regular expressions. In *International Symposium on Temporal Representation and Reasoning (TIME)*, pages 6:1–6:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.TIME.2022.6.

44 Fred B. Schneider. Enforceable security policies. *ACM Trans. Inf. Syst. Secur.*, 3(1):30–50, 2000. doi:10.1145/353323.353382.

45 Prasanna Thati and Grigore Roşu. Monitoring algorithms for metric temporal logic specifications. *Electronic Notes in Theoretical Computer Science*, 113:145–162, 2005. doi:10.1016/j.entcs.2004.01.029.

46 Dogan Ulus, Thomas Ferrère, Eugene Asarin, and Oded Maler. Online timed pattern matching using derivatives. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 736–751. Springer, 2016. doi:10.1007/978-3-662-49674-9_47.

47 Moshe Y Vardi. An automata-theoretic approach to linear temporal logic. *Logics for concurrency: structure versus automata*, pages 238–266, 2005.

48 Pierre Wolper. Temporal logic can be more expressive. In *Annual Symposium on Foundations of Computer Science (SFCS)*, pages 340–348. IEEE Computer Society, 1981. doi:10.1109/SFCS.1981.44.

49 W. Yi. Real-time behaviour of asynchronous agents. In *International Conference on Concurrency Theory (CONCUR)*, 1990. doi:10.1007/BFb0039080.