Runtime Verification of Temporal Properties over Out-of-order Data Streams

David Basin[1], Felix Klaedtke[2], and Eugen Zălinescu[3]

[1] ETH Zurich, Department of Computer Science
[2] NEC Laboratories Europe, Heidelberg
[3] Technische Universität München

Abstract. We present a monitoring approach for verifying systems at runtime. Our approach targets systems whose components communicate with the monitors over unreliable channels, where messages can be delayed or lost. In contrast to prior works, whose property specification languages are limited to propositional temporal logics, our approach handles an extension of the real-time logic MTL with freeze quantifiers for reasoning about data values. We present its underlying theory based on a new three-valued semantics that is well suited to soundly and completely reason online about event streams in the presence of message delay or loss. We also evaluate our approach experimentally. Our prototype implementation processes hundreds of events per second in settings where messages are received out of order.

1 Introduction

Verifying systems at runtime can be accomplished by instrumenting system components so that they inform monitors about the actions they perform. The monitors update their states according to the information received and check whether the properties they are monitoring are fulfilled or violated. Various runtime-verification approaches exist for different kind of systems and property specification languages, see for example [2, 5, 7, 10, 17, 18, 21].

Many of these specifications languages are based on temporal logics or finite-state machines, which describe the correct system behavior in terms of *infinite* streams of system actions. However, at any point in time, a monitor has only partial knowledge about the system's behavior. In particular, a monitor can at best only be aware of the previously performed actions, which correspond to a finite prefix of the infinite action stream. When communication channels are unreliable, a monitor's knowledge about the previously performed actions may even be incomplete since messages can be lost or delayed and thus received out of order. Nevertheless, a monitor should output a verdict promptly when the monitored property is fulfilled or violated. Moreover, the verdict should remain correct when some of the monitor's knowledge gaps are subsequently closed.

Many runtime-verification approaches rely on an extension of the standard Boolean semantics of the linear-time temporal logic LTL with a third truth value, proposed by Bauer et al. [9]. Namely, a formula evaluates to the Boolean truth

value b on a finite stream of performed actions σ if the formula evaluates to b on all infinite streams that extend σ; otherwise, the formula's truth value is unknown on σ. This semantics, however, only accounts for settings where monitors are always aware of all previously performed actions. It is insufficient to reason soundly and completely about system behavior at runtime when, for example, unreliable channels are used to inform the monitors about the performed actions.

In this paper, we present an extension of the propositional real-time logic MTL [1,16], which we name MTL$^{\downarrow}$. First, MTL$^{\downarrow}$ comprises a freeze quantifier [15] for reasoning about data values in action streams. The freeze quantifier $\downarrow$ can be seen as a restricted version of the first-order quantifiers $\exists$ and $\forall$. More concretely, at a position of the action stream, the formula $\downarrow x.\, \varphi$ uniquely binds a data value of the action at that position to the logical variable x.

Second, we equip MTL$^{\downarrow}$ with a new three-value semantics that is well suited for settings where system components communicate with the monitors over unreliable channels. Specifically, we define the semantics of MTL$^{\downarrow}$'s connectives over the three truth values t, f, and $\bot$. We interpret these truth values as in Kleene logic and conservatively extend the logic's standard Boolean semantics, where t and f stand for "true" and "false" respectively, and the third truth value $\bot$ stands for "unknown" and accounts for the monitor's knowledge gaps. The models of MTL$^{\downarrow}$ are finite words where knowledge gaps are explicitly represented. Intuitively, a finite word corresponds to a monitor's knowledge about the system behavior at a given time and the knowledge gaps may result from message delays, losses, crashed components, and the like. Critically in our setting, reasoning is monotonic with respect to the partial order on truth values, where $\bot$ is less than t and f, and t and f are incomparable. This monotonicity property guarantees that closing knowledge gaps does not invalidate previously obtained Boolean truth values.

Third, we present an online algorithm for verifying systems at runtime with respect to MTL$^{\downarrow}$ specifications. Our algorithm is based on, and extends, the algorithm for MTL by Basin et al. [6] to additionally handle the freeze quantifier. The algorithm's output is sound and complete for MTL$^{\downarrow}$'s three-valued semantics and with respect to the monitor's partial knowledge about the performed actions at each point in time.

Our algorithm works roughly as follows. It receives messages from the system components describing the actions they perform. As with the algorithm in [6], no assumptions are made on the order in which messages are received. The algorithm updates its state for each received message. This state comprises a graph structure for reasoning about the system behavior, i.e., computing verdicts about the monitored property's fulfillment. The graph's nodes store the truth values of the subformulas at the different times for the data values to which quantified variables are frozen. In each update, the algorithm propagates data values down to the graph's leaves and propagates Boolean truth values for subformulas up along the graph's edges. When a Boolean truth value is propagated to a root node of the graph, the algorithm outputs a verdict.

Our main contribution is a runtime-verification approach that makes no assumptions about message delivery. It handles a significantly richer specification

language than previous approaches, namely, an extension of the real-time logic MTL with a quantifier for reasoning about the data processed by the monitored system. Furthermore, our approach guarantees sound and complete reasoning with partial knowledge about system behavior. Finally, we experimentally evaluate the performance of a prototype implementation of our approach, illuminating its current capabilities, tradeoffs, and performance limitations.

The remainder of this paper is structured as follows. In Section 2, we introduce relevant notation and terminology. In Section 3, we extend MTL with the freeze quantifier and give the logic's semantics. In Section 4, we describe our monitoring approach, including its algorithmic details. In Section 5, we report on our experimental evaluation. Finally, in Sections 6 and 7, we discuss related work and draw conclusions. Further details are given in the appendixes.

2 Preliminaries

In this section, we introduce relevant notation and terminology.

Intervals. An *interval* I is a nonempty subset of $\mathbb{Q}_{\geq 0}$ such that if $a, b \in I$ then $c \in I$, for any $c \in \mathbb{Q}_{\geq 0}$ with $a \leq c \leq b$. We use standard notation and terminology for intervals. For example, $(a, b]$ denotes the interval that is left-open with bound a and right-closed with bound b. Note that an interval I with cardinality $|I| = 1$ is a singleton $\{\tau\} = [\tau, \tau]$, for some $\tau \in \mathbb{Q}_{\geq 0}$. An interval I is *unbounded* if its right bound is ∞, and *bounded* otherwise. Let $I - J := \{\tau - \tau' \mid \tau \in I \text{ and } \tau' \in J\} \cap \mathbb{Q}_{\geq 0}$.

Partial Functions. For a partial function $f : A \nrightarrow B$, let $\mathrm{def}(f) := \{a \in A \mid f(a) \text{ is defined}\}$. If $\mathrm{def}(f) = \{a_1, \ldots, a_n\}$, for some $n \in \mathbb{N}$, we also write $[a_1 \mapsto f(a_1), \ldots, a_n \mapsto f(a_n)]$ for f, when f's domain A and its codomain B are irrelevant or clear from the context. Note that $[\,]$ denotes the partial function that is undefined everywhere. Furthermore, for partial functions $f, g : A \nrightarrow B$, we write $f \sqsubseteq g$ if $\mathrm{def}(f) \subseteq \mathrm{def}(g)$ and $f(a) = g(a)$, for all $a \in \mathrm{def}(f)$. We write $f[a \mapsto b]$ to denote the update of a partial function $f : A \nrightarrow B$ at $a \in A$, i.e., $f[a \mapsto b]$ equals f, except that a is mapped to b if $b \in B$, and $a \notin \mathrm{def}(f[a \mapsto b])$ if $b \notin B$.

Truth Values. Let $\mathsf{3}$ be the set $\{\mathsf{t}, \mathsf{f}, \bot\}$, where t (true) and f (false) denote the standard Boolean values, and $\bot$ denotes the truth value "unknown." Table 1 shows the truth tables of some standard logical operators over $\mathsf{3}$. Observe that these operators coincide with their Boolean counterparts when restricted to the set $\mathsf{2} := \{\mathsf{t}, \mathsf{f}\}$. We partially order the elements in $\mathsf{3}$ by their knowledge: $\bot \prec \mathsf{t}$ and $\bot \prec \mathsf{f}$, and t and f are incomparable as they carry the same amount of knowledge. Note that $(\mathsf{3}, \prec)$ is a lower semilattice where $\curlywedge$ denotes the meet. We remark that the operators in Table 1 are monotonic. This ensures that reasoning is monotonic in knowledge. Intuitively, when closing a knowledge gap, represented by $\bot$, with t or f, we never obtain a truth value that disagrees with the previous one.

Table 1. Truth tables for three-valued logical operators (strong Kleene logic).

$\neg$	
t	f
f	t
$\bot$	$\bot$

$\vee$	t	f	$\bot$
t	t	t	t
f	t	f	$\bot$
$\bot$	t	$\bot$	$\bot$

$\wedge$	t	f	$\bot$
t	t	f	$\bot$
f	f	f	f
$\bot$	$\bot$	f	$\bot$

$\rightarrow$	t	f	$\bot$
t	t	f	$\bot$
f	t	t	t
$\bot$	t	$\bot$	$\bot$

Timed Words. Let Σ be an alphabet. A *timed word* over Σ is an infinite word $(\tau_0, a_0)(\tau_1, a_1)\ldots \in (\mathbb{Q}_{\geq 0} \times \Sigma)^\omega$, where the sequence of τ_is is strictly monotonic and nonzeno, that is, $\tau_i < \tau_{i+1}$, for every $i \in \mathbb{N}$, and for every $t \in \mathbb{Q}_{\geq 0}$, there is some $i \in \mathbb{N}$ such that $\tau_i > t$.

3 Metric Temporal Logic Extensions

In this section, we extend the propositional real-time logic MTL [1, 16] with a freeze quantifier [15]. The logic's three-valued semantics conservatively extends the standard Boolean semantics and accounts for knowledge gaps during monitoring.

3.1 Syntax

Let P be a finite set of predicate symbols, where $\iota(p)$ denotes the arity of $p \in P$. Furthermore, let V be a set of variables and R a finite set of registers. The syntax of the real-time logic MTL$^{\downarrow}$ is given by the grammar:

$$\varphi \ ::= \ \mathsf{t} \mid p(x_1, \ldots, x_{\iota(p)}) \mid \downarrow^r x.\, \varphi \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \,\mathsf{U}_I\, \varphi\,,$$

where $p \in P$, $x, x_1, x_2 \ldots, x_{\iota(p)} \in V$, $r \in R$, and I is an interval. For the sake of brevity, we limit ourselves to the future fragment and omit the temporal connective for "next." A formula is *closed* if each variable occurrence is bound by a freeze quantifier. A formula is *temporal* if the connective at the root of the formula's syntax tree is U_I. We denote by $\mathit{Sub}(\varphi)$ the set of φ's subformulas.

We employ standard syntactic sugar. For example, $\varphi \rightarrow \psi$ abbreviates $(\neg\varphi)\vee\psi$, and $\Diamond_I\, \varphi$ ("eventually") and $\Box_I\, \varphi$ ("always") abbreviate $\mathsf{t}\,\mathsf{U}_I\, \varphi$ and $\neg\Diamond_I\neg\varphi$, respectively. The nonmetric variants of the temporal connectives are also easily defined, e.g., $\Box\,\varphi := \Box_{[0,\infty)}\,\varphi$. Finally, we use standard conventions concerning the connectives' binding strength to omit parentheses. For example, $\neg$ binds stronger than $\wedge$, which binds stronger than $\vee$, and the connectives $\neg$, $\vee$, etc. bind stronger than the temporal connectives, which bind stronger than the freeze quantifier. To simplify notation, we omit the superscript r in formulas like $\downarrow^r x.\, \varphi$ whenever $r \in R$ is irrelevant or clear from the context.

Example 1. Before defining the logic's semantics, we provide some intuition. The following formula formalizes the policy that whenever a customer executes a transaction that exceeds some threshold (e.g. \$2,000) then this customer must not execute any other transaction for a certain period of time (e.g. 3 days).

$$\Box \downarrow^{cid} c. \downarrow^{tid} t. \downarrow^{sum} a.\, \mathit{trans}(c, t, a) \wedge a \geq 2000 \rightarrow \Box_{(0,3]} \downarrow^{tid} t'. \downarrow^{sum} a'.\, \neg \mathit{trans}(c, t', a')$$

We assume that the predicate symbol *trans* is interpreted as a singleton relation or the empty set at any point in time. For instance, the interpretation $\{(Alice, 42, 99)\}$ of *trans* at time τ describes the action of *Alice* executing a transaction with identifier 42 with the amount \$99 at time τ. When the interpretation is the empty set, no transaction is executed. We further assume that when the interpretation of the predicate symbol *trans* is nonempty, the registers *cid*, *tid*, and *sum* store (a) the transaction's customer, (b) the transaction identifier, and (c) the transferred amount, respectively. If the interpretation is the empty set, the registers store a dummy value, representing undefinedness.

The variables c, t, a, t', and a' are frozen to the respective register values. For example, c is frozen to the value stored in the register *cid* at each point in time and is used to identify later transactions from this customer. Furthermore, note that, e.g., the variables t and t' are frozen to values stored in the registers *tid* at different times. The freeze quantifier can be seen as a weak form of the standard first-order quantifiers [15]. Since a register stores exactly one value at any time, it is irrelevant whether we quantify existentially or universally over a register's value. □

3.2 Semantics

MTL$^{\downarrow}$'s models under the three-valued semantics are finite words (see Definition 2 below). Such a model represents a monitor's partial knowledge about the system behavior at a given point in time. This is in contrast to the models for the standard Boolean semantics for MTL, which are infinite timed words and capture the complete system behavior in the limit.

Definition 2. *Let D be the* data domain*, a nonempty set of values with $\bot \notin D$.* Observations *are finite words with letters of the form (I, σ, ϱ), where I is an interval, $\sigma : P \nrightarrow 2^{\bigcup_{\iota \in \mathbb{N}} D^\iota}$, and $\varrho : R \nrightarrow D$. We define observations inductively.*

- *The word $\big([0,\infty), [\,], [\,]\big)$ of length 1 is an observation.*
- *If w is an observation, then the word obtained by applying one of the following transformations to w is an observation.*
  - *(T1) Some letter (I, σ, ϱ) of w, where $|I| > 1$, is replaced by the three-letter word $\big(I \cap [0, \tau), \sigma, \varrho\big)\big(\{\tau\}, \sigma, \varrho\big)\big(I \cap (\tau, \infty), \sigma, \varrho\big)$, where $\tau \in I$ and $\tau > 0$. If $\tau = 0$, then (I, σ, ϱ) is replaced by $\big(\{\tau\}, \sigma, \varrho\big)\big(I \cap (\tau, \infty), \sigma, \varrho\big)$.*
  - *(T2) Some letter (I, σ, ϱ) of w, where $|I| > 1$ and I is bounded, is removed.*
  - *(T3) Some letter (I, σ, ϱ) of w, where $|I| = 1$, is replaced by (I, σ', ϱ'), where $\sigma \sqsubseteq \sigma'$ and $\varrho \sqsubseteq \varrho'$, and $\sigma \neq \sigma'$ or $\varrho \neq \varrho'$.*

For an observation w of length $n \in \mathbb{N}$, let $pos(w) := \{0, \ldots, n-1\}$. We call $i \in pos(w)$ a *time point* in w if the interval I_i of the letter at position i in w is a singleton. In this case, the element of I_i is the *timestamp* of the time point i, denoted by $\mathrm{ts}_w(i)$. We note that for any letter (I, σ, ϱ) of an observation, if $|I| > 1$ then $\sigma = \varrho = [\,]$.

Example 3. A monitor's initial knowledge is represented by the observation $w_0 = \big([0,\infty), [\,], [\,]\big)$. Suppose a transaction of \$99 with identifier 42 from *Alice*

is executed at time 3.0. The monitor's initial knowledge w_0 is then updated by (T1) and (T3) to $w_1 = \big([0,3.0),[\,],[\,]\big)\big(\{3.0\},\sigma,\varrho\big)\big((3.0,\infty),[\,],[\,]\big)$, where $\sigma(\mathit{trans}) = \{(\mathit{Alice},42,99)\}$ and $\varrho = [\mathit{cid} \mapsto \mathit{Alice}, \mathit{tid} \mapsto 42, \mathit{sum} \mapsto 99]$. If the monitor also receives the information that no action has occurred in the interval $[0,3.0)$, then its updated knowledge is represented by $\big(\{3.0\},\sigma,\varrho\big)\big((3.0,\infty),[\,],[\,]\big)$, obtained from w_1 by (T2). The information that no action has occurred in an interval can be communicated explicitly or implicitly by the monitored system to the monitor, for instance, by attaching a sequence number to each action. See [6] for details. Finally, note that the interval of the last letter of any observation is always unbounded. This reflects that a monitor is unaware of what it will observe in the future. □

Definition 4. *The observation w'* refines *the observation w, written $w \sqsubset_1 w'$, iff w' is obtained from w by one of the transformations (T1), (T2), or (T3). The reflexive-transitive closure of $\sqsubset_1$ is $\sqsubseteq$.*

MTL$^{\downarrow}$'s three-valued semantics is defined by a function $\varphi \mapsto [\![w,i,\nu \models\!\!\approx \varphi]\!] \in \mathsf{3}$, for a given observation w, time point $i \in \mathbb{N}$, and partial valuation $\nu : V \rightharpoonup D$. We define this function inductively over the formula structure. For a predicate symbol $p \in P$, we write in the following $p(\bar{x})$ instead of $p(x_1,\ldots,x_{\iota(p)})$. Furthermore, we abuse notation by abbreviating, e.g., $\nu(x_1),\ldots,\nu(x_n)$ as $\nu(\bar{x})$, for a partial valuation $\nu : V \rightharpoonup D$ and variables $x_1,\ldots,x_n$. Also, the notation $\bar{x} \in \mathrm{def}(\nu)$ means that $x \in \mathrm{def}(\nu)$, for each x occurring in $\bar{x}$. Finally, we identify the logic's constant symbol t with the Boolean value $\mathsf{t} \in \mathsf{3}$, and the connectives $\neg$ and $\vee$ with the corresponding three-valued logical operators in Table 1.

$$
\begin{aligned}
[\![w,i,\nu \models\!\!\approx \mathsf{t}]\!] &:= \mathsf{t}\\
[\![w,i,\nu \models\!\!\approx p(\bar{x})]\!] &:= \begin{cases} \mathsf{t} & \text{if } \bar{x} \in \mathrm{def}(\nu),\ p \in \mathrm{def}(\sigma_i), \text{ and } \nu(\bar{x}) \in \sigma_i(p)\\ \mathsf{f} & \text{if } \bar{x} \in \mathrm{def}(\nu),\ p \in \mathrm{def}(\sigma_i), \text{ and } \nu(\bar{x}) \notin \sigma_i(p)\\ \bot & \text{otherwise} \end{cases}\\
[\![w,i,\nu \models\!\!\approx \downarrow^r x.\,\varphi]\!] &:= [\![w,i,\nu[x \mapsto \varrho_i(r)] \models\!\!\approx \varphi]\!]\\
[\![w,i,\nu \models\!\!\approx \neg\varphi]\!] &:= \neg[\![w,i,\nu \models\!\!\approx \varphi]\!]\\
[\![w,i,\nu \models\!\!\approx \varphi \vee \psi]\!] &:= [\![w,i,\nu \models\!\!\approx \varphi]\!] \vee [\![w,i,\nu \models\!\!\approx \psi]\!]\\
[\![w,i,\nu \models\!\!\approx \varphi\, \mathsf{U}_I\, \psi]\!] &:= \bigvee_{j \in pos(w), j \geq i} \Big(\mathrm{tp}_w(j) \wedge \mathrm{tc}_{w,I}(j,i) \wedge [\![w,j,\nu \models\!\!\approx \psi]\!] \wedge \\
&\qquad\qquad \bigwedge_{i \leq k < j} \big(\mathrm{tp}_w(k) \rightarrow [\![w,k,\nu \models\!\!\approx \varphi]\!]\big)\Big)
\end{aligned}
$$

The auxiliary functions $\mathrm{tp}_w : \mathit{pos}(w) \to \mathsf{3}$ and $\mathrm{tc}_{w,I} : \mathit{pos}(w) \times \mathit{pos}(w) \to \mathsf{3}$, are defined as follows, where I_k denotes the interval at position $k \in \mathit{pos}(w)$ in w.

$$
\mathrm{tp}_w(j) := \begin{cases} \mathsf{t} & \text{if } j \text{ is a time point in } w\\ \bot & \text{otherwise} \end{cases}
$$

$$
\mathrm{tc}_{w,I}(i,j) := \begin{cases} \mathsf{t} & \text{if } \tau - \tau' \in I, \text{ for all } \tau \in I_i \text{ and } \tau' \in I_j\\ \mathsf{f} & \text{if } \tau - \tau' \notin I, \text{ for all } \tau \in I_i \text{ and } \tau' \in I_j\\ \bot & \text{otherwise} \end{cases}
$$

We comment on the semantics of $\varphi \,\mathsf{U}_I\, \psi$. The auxiliary functions account for the positions in w that are not time points. For example, at position i, for a position $j \leq i$ to be a "valid anchor" for the formula, j must be a time point (in this case $\mathrm{tp}_w(j) = \mathsf{t}$). Otherwise, the truth value $\bot$ is used to express that it is not yet known whether the interval at position j in w will contain a time point. Note that using the truth value f would be incorrect since a refinement of w might contain a time point with a timestamp in I_j. Furthermore, $\mathrm{tc}_{w,I}(i,j)$ is used to account for the metric constraint of the temporal connective. In particular, $\mathrm{tc}_{w,I}(i,j)$ is $\bot$ if it is unknown in w whether the formula's metric constraint is always satisfied or never satisfied for the positions i and j. Finally, suppose that φ's truth value is f at a position k between j and i. If the interval I_k at position k is not a singleton, the function $\mathrm{tp}_w(k)$ "downgrades" this value to $\bot$, since it will be irrelevant in refinements of w that do not contain any time points with timestamps in I_k.

Note that it may be the case that $[\![w, i, \nu \models\!\!\approx \varphi]\!] \in \mathsf{2}$ when i is not a time point in w (i.e., I_i is not a singleton). A trivial example is when $\varphi = \mathsf{t}$. In a refinement of w, it might turn out that there are no time points with timestamps in I_i, and hence a monitor should not output a verdict for the specification φ at position i in w. We address this artifact by downgrading (with respect to the partial order $\prec$) a Boolean truth value $[\![w, i, \nu \models\!\!\approx \varphi]\!]$ to $\bot$ when i is not a time point. To this end, we introduce the following variant of the semantics.

Definition 5. *For a formula φ, an observation w, $\tau \in \mathbb{Q}_{\geq 0}$, and ν a partial valuation, we define $[w, \tau, \nu \models\!\!\approx \varphi] := [\![w, i, \nu \models\!\!\approx \varphi]\!]$, provided that τ is the timestamp of some time point $i \in pos(w)$ in w, and $[w, \tau, \nu \models\!\!\approx \varphi] := \bot$, otherwise.*

3.3 Properties

The following theorem states that MTL$^{\downarrow}$'s three-valued semantics is monotonic in $\sqsubseteq$ (on observations and partial valuations) and $\preceq$ (on truth values). This property is crucial for monitoring since it guarantees that a verdict output for an observation stays valid for refined observations.

Theorem 6. *Let φ be a formula, μ and ν partial valuations, u and v observations, and $\tau \in \mathbb{Q}_{\geq 0}$. If $u \sqsubseteq v$ and $\mu \sqsubseteq \nu$ then $[u, \tau, \mu \models\!\!\approx \varphi] \preceq [v, \tau, \nu \models\!\!\approx \varphi]$.*

A similar theorem shows that MTL$^{\downarrow}$'s three-valued semantics conservatively extends the standard Boolean semantics (see Appendix A for details). Intuitively speaking, if a formula φ evaluates to a Boolean value for an observation at time $\tau \in \mathbb{Q}_{\geq 0}$, then φ has the same Boolean value at time τ for any timed word[4] that refines the observation. Formally, a timed word w' *refines* an observation w, $w \sqsubseteq w'$ for short, if for every $j \in \mathbb{N}$, there is some $i \in pos(w)$, such that $\tau_j \in I_i$, $\sigma_i \sqsubseteq \sigma'_j$, and $\varrho_i \sqsubseteq \varrho'_j$, where $(I_\ell, \sigma_\ell, \varrho_\ell)$ and $(\tau_k, \sigma'_k, \varrho'_k)$, for $\ell \in pos(w)$ and $k \in \mathbb{N}$, are the letters of w and w', respectively.

We investigate next the decision problem that underlies monitoring.

[4] We assume here that the timed words are over the alphabet Σ that consists of the pairs (σ, ϱ), where (i) σ is a total function over P with $\sigma(p) \subseteq D^{\iota(p)}$ for $p \in P$, and (ii) ϱ is a total function over R with $\varrho(r) \in D$ for $r \in R$.

Theorem 7. *For an arbitrary formula φ, observation w, partial valuation ν, time $\tau \in \mathbb{Q}_{\geq 0}$, and truth value $b \in \mathsf{2}$, the question of whether $[w, \tau, \nu \models\!\!\approx \varphi]$ equals b is* PSPACE*-complete.*

In a propositional setting, the corresponding decision problem can be solved in polynomial time using dynamic programming, where the truth values at the positions of an observation are propagated up the formula structure. Note that the truth value of a proposition at a position is given by the observation's letter at that position. This is in contrast to $\mathrm{MTL}^{\downarrow}$, where atomic formulas can have free variables and their truth values at the positions in an observation w may depend on the data values stored in the registers and frozen to these variables at different time points of w. Before truth values are propagated up, the bindings of variables to data values must be propagated down.

4 Monitoring Algorithm

In this section, we present an online algorithm that computes verdicts for $\mathrm{MTL}^{\downarrow}$ specifications. To support scalable monitoring, the computation is incremental in that, when refining an observation according to the transformations (T1)–(T3), the results from previous computations are reused, including the propagated data values and Boolean values. We also define correctness requirements for monitoring and establish the algorithm's correctness.

4.1 Correctness Requirements

We define when a sequence of observations is valid for representing a monitor's knowledge over time. We assume that the monitor receives in the limit infinitely many messages containing information about the system behavior. This assumption is invalid if the system ever terminates. Nevertheless, we make this assumption to simplify matters and it is easy to adapt the definitions and results to the general case.

Definition 8. *The infinite sequence $\bar{w} = (w_i)_{i \in \mathbb{N}}$ of observations is* valid *if $w_0 = ([0, \infty), [\,], [\,])$ and $w_i \sqsubsetneq w_{i+1}$, for all $i \in \mathbb{N}$.*

Let M be a monitor and $\bar{w}$ a valid sequence of observations. In the following, we view w_i as the input to M at iteration i. For the input w_i, M outputs a set of *verdicts*, which is a finite set of pairs (τ, b) with $\tau \in \mathbb{Q}_{\geq 0}$ and $b \in \mathsf{2}$. We denote this set by $M(w_i)$. Note that in practice, M would receive at iteration $i > 0$ a message that describes just the differences between w_{i-1} and w_i. Furthermore, the w_is can be understood as abstract descriptions of M's states over time, representing M's knowledge about the system behavior, where w_0 represents M's initial knowledge. Also note that if the timed word v is the system behavior in the limit, then $w_i \sqsubseteq v$, for all $i \in \mathbb{N}$, assuming that components do not send bogus messages. However, for every $i \in \mathbb{N}$, there are infinitely many timed words u with $w_i \sqsubseteq u$. Since messages sent to the monitor can be lost, it can even be the case that there are timed words u with $u \neq v$ and $w_i \sqsubseteq u$, for all $i \in \mathbb{N}$.

Definition 9. *Let M be a monitor, φ a formula, and $\bar{w}$ a valid observation sequence.*
- *M is* observationally sound *for $\bar{w}$ and φ if for all partial valuations ν and $i \in \mathbb{N}$, if $(\tau, b) \in M(w_i)$ then $[w_i, \tau, \nu \models\!\!\approx \varphi] = b$.*
- *M is* observationally complete *for $\bar{w}$ and φ if for all partial valuations ν, $i \in \mathbb{N}$, and $\tau \in \mathbb{Q}_{\geq 0}$, if $[w_i, \tau, \nu \models\!\!\approx \varphi] \in \mathsf{2}$ then $(\tau, b) \in \bigcup_{j \leq i} M(w_j)$, for some $b \in \mathsf{2}$.*

We say that M is observationally sound *if M is observational sound for all valid observation sequences and formulas φ. The definition of M being* observationally complete *is analogous.*

It follows from Theorem 7 that there exist monitors for MTL$^{\downarrow}$ that are both observationally sound and complete. This is in contrast to correctness requirements that demand that a monitor outputs a verdict as soon as the specification has the same Boolean value on every extension of the monitor's current knowledge. It is easy to see that, for a given specification language, such monitoring is at least as hard as checking satisfiability for the language. The propositional fragment of MTL$^{\downarrow}$ is already undecidable [20]. Thus monitors satisfying such strong requirements do not exist for MTL$^{\downarrow}$. For LTL, such stronger requirements are standardly formalized using a three-valued runtime-verification semantics, as introduced by Bauer et al. [10], and adopted by other runtime-verification approaches, e.g. [8]. See Appendix A.2 for a formal definition of these requirements in our setting.

Example 10. Consider the formula $\varphi = \Box(p \wedge \Diamond \neg p)$. Under the classical Boolean semantics, φ is logically equivalent to f, however not under our semantics. For example, $[\![w, 0, \nu \models\!\!\approx \varphi]\!] = \bot$, for $w = \big([0, \infty), [\,], [\,]\big)$ and any valuation ν. Given a valid observation sequence $\bar{w}$, an observationally sound and complete monitor for $\bar{w}$ and φ will first output the verdict $(0, \mathsf{f})$ for the minimal i such that w_i contains a letter that assigns p to false. □

4.2 Monitoring Algorithm

We sketch the algorithm's state, its main procedure, and its main data structure. We provide further algorithmic details in Appendix B.

4.2.1 Monitor State. Before explaining the algorithm, we first rephrase the MTL$^{\downarrow}$'s semantics such that it is closer to the representation used by the monitor. Given an $i \in \mathbb{N}$, a position $j \in pos(w_i)$, and a subformula γ of φ, we denote by Φ_i^{γ, J_j}, where J_k is the interval of the kth letter of w_i, the propositional formula:

$$\Phi_i^{\gamma, J_j} := \begin{cases} \gamma^{J_j} & \text{if } \gamma \text{ is atomic} \\ \neg\alpha^{J_j} & \text{if } \gamma = \neg\alpha \\ \alpha^{J_j} \vee \beta^{J_j} & \text{if } \gamma = \alpha \vee \beta \\ \alpha^{J_j} & \text{if } \gamma = \downarrow^r x.\, \alpha \\ \bigvee_{k \geq j} \left(tp^{J_k} \wedge tc_\gamma^{J_k, J_j} \wedge \beta^{J_k} \wedge \bigwedge_{j \leq h < k} (tp^{J_h} \rightarrow \alpha^{J_h}) \right) & \text{if } \gamma = \alpha \,\mathsf{U}_I\, \beta, \end{cases}$$

where α^K, tp^K, and $tc_\psi^{H,K}$ denote atomic propositions, for each proper subformula α of φ, each temporal subformula ψ of φ, and all intervals H, K of letters in w_i. Next, we define, for any partial valuation ν, the substitution θ_i^ν of Boolean values for these atomic propositions as follows:

$$\begin{array}{rll} \theta_i^\nu(\alpha^{J_j}) := [\![w_i, j, \nu \models\!\!\approx \alpha]\!] & \text{if } [\![w_i, j, \nu \models\!\!\approx \alpha]\!] \in 2, \\ \theta_i^\nu(tp^{J_j}) := \mathrm{tp}_{w_i}(j) & \text{if } \mathrm{tp}_{w_i}(j) \in 2, \\ \theta_i^\nu(tc_{\alpha \mathsf{U}_I \beta}^{J_j,J_k}) := \mathrm{tc}_{w_i,I}(j,k) & \text{if } \mathrm{tc}_{w_i,I}(j,k) \in 2, \end{array}$$

and θ_i^ν is undefined otherwise. In what follows, the symbol $\equiv$ denotes semantic equivalence between propositional formulas. It is easy to see that

$$\theta_i^\mu(\Phi_i^{\gamma,J_j}) \equiv [\![w_i, j, \nu \models\!\!\approx \gamma]\!] \quad \text{iff} \quad [\![w_i, j, \nu \models\!\!\approx \gamma]\!] \in 2,$$

where $\mu = \nu[x \mapsto \varrho_j(r)]$ if $\gamma = \downarrow^r x.\ \alpha$ and $\mu = \nu$ otherwise, with ϱ_j being the third component of the jth letter of w_i. Note that the formula $\theta_i^\mu(\Phi_i^{\gamma,J_j})$ tells us more than the truth value $[\![w_i, j, \nu \models\!\!\approx \gamma]\!]$. Indeed, when $\theta_i^\mu(\Phi_i^{\gamma,J_j}) \not\equiv b$, for each $b \in 2$, then we know not only that $[\![w_i, j, \nu \models\!\!\approx \gamma]\!] = \bot$, but we also know what the causes of uncertainty are, namely the direct subformulas α of γ and indexes k with $[\![w_i, k, \mu \models\!\!\approx \alpha]\!] = \bot$.

The monitor maintains as state between its iterations a variant of the propositional formulas $\theta_i^\mu(\Phi_i^{\gamma,J_j})$. The reason for using variants is that it is not algorithmically convenient to transform $\theta_i^\mu(\Phi_i^{\gamma,J})$ into $\theta_{i+1}^\mu(\Phi_{i+1}^{\gamma,K})$, where K is an interval (of a letter) in w_{i+1} that originates from the interval J in w_i. Such a transformation is needed for obtaining an incremental monitoring algorithm that reuses information already computed at previous iterations.

The formulas that the monitors maintains, denoted $\Psi_i^{\gamma,J_j,\nu}$, can be obtained from the formulas $\theta_i^\mu(\Phi_i^{\gamma,J_j})$ as follows. When γ is a nontemporal formula, then $\Psi_i^{\gamma,J_j,\nu}$ equals $\theta_i^\mu(\Phi_i^{\gamma,J_j})$. When γ is a temporal formula $\alpha \mathsf{U}_I \beta$, then, to each disjunct for index k in Φ_i^{γ,J_j}, we add the subformula $(tp^{J_k} \vee \alpha^{J_k})$ as a conjunct. This is sound, based on the equivalence $tp^{J_k} \equiv tp^{J_k} \wedge (tp^{J_k} \vee \alpha^{J_k})$. Furthermore, the monitor treats the subformulas $(tp^{J_k} \wedge \beta^{J_k})$, $(tp^{J_h} \to \alpha^{J_h})$, and $(tp^{J_k} \vee \alpha^{J_k})$ in a special way: they are not simplified in $\Psi_i^{\gamma,J,\nu}$ when they are still needed to obtain $\Psi_{i+1}^{\gamma,K,\nu}$. That is, even if one the atomic propositions q of these subformulas could be instantiated (i.e. $q \in \mathrm{def}(\theta_i^\mu)$) this is not always done, as explained in the next section. Instead, these three types of subformulas are represented in $\Psi_i^{\gamma,J_j,\nu}$ by the atomic propositions $\bar{\beta}^{J_k}$, $\bar{\alpha}^{J_h}$, and $\bar{\bar{\alpha}}^{J_k}$, respectively.

Example 11. We illustrate here the definitions of the propositional formulas $\Phi_i^{\gamma,J,\nu}$ and $\Psi_i^{\gamma,J,\nu}$ for temporal formulas γ. We also suggest why variants of the formulas $\theta_i^\mu(\Phi_i^{\gamma,J_j})$ are needed.

Let $\gamma = p \mathsf{U} q$, where p and q are 0-ary predicates. Assume that in w_1 we have the intervals $L = [0, \tau_1)$, $N = \{\tau_1\}$, and $R = (\tau_1, \infty)$, and in w_2 we have the intervals $L_1 = [0, \tau_0)$, $L_2 = \{\tau_0\}$, $L_3 = (\tau_0, \tau_1)$, N, and R, with $\tau_0 \in L$. Assume

also that neither p nor q holds at τ_1. Then

$$\theta_1^{[]}(\Phi_1^{\gamma,L}) \equiv tp^L \wedge q^L \qquad \theta_2^{[]}(\Phi_2^{\gamma,L_2}) \equiv q^{L_2} \vee (tp^{L_3} \wedge q^{L_3} \wedge p^{L_2})$$
$$\Psi_1^{\gamma,L,[]} = \bar{q}^L \wedge \bar{\bar{p}}^L \qquad \Psi_2^{\gamma,L_2,[]} = \bar{q}^{L_2} \vee (\bar{q}^{L_3} \wedge \bar{\bar{p}}^{L_3} \wedge \bar{\bar{p}}^{L_2})$$

Note that p^L is not an atomic proposition of $\Phi_1^{\gamma,L}$, while $\bar{\bar{p}}^L$ is an atomic proposition of $\Psi_1^{\gamma,L,[]}$. This last fact allows the monitoring algorithm to obtain $\Psi_2^{\gamma,L_2,[]}$ from $\Psi_1^{\gamma,L,[]}$, by introducing the needed new propositions $\bar{p}^{L_2}$, $\bar{p}^{L_3}$, and $\bar{\bar{p}}^{L_2}$. □

To recapitulate, the monitor's state at iteration i consists of propositional formulas $\Psi_i^{\gamma,J,\nu}$, one for each subformula γ of φ, interval J occurring in a letter of w_i, where i is the current iteration, and partial valuation ν that is *relevant* for the current subformula and position corresponding to J in w_i. Intuitively, a valuation ν is relevant for ψ and a position $j \in pos(w_i)$, if $[\![w_i, j, \nu \mid\approx \psi]\!]$ is reached when unfolding the formula that defines $[\![w_i, k, [\,] \mid\approx \varphi]\!]$, for some $k \in pos(w_i)$.[5] For instance, $[\,]$ is relevant for φ and any $j \in pos(w_i)$. Furthermore, if ν is relevant for $\downarrow^r x.\, \psi$ and j, then $\nu[x \mapsto \varrho_j(r)]$ is relevant for ψ and j.

Example 12. Let $\varphi := \downarrow^r x.\, \Diamond_{(0,1]}\, p(x)$. For brevity, we treat the temporal connective $\Diamond_{(0,1]}$ as a primitive. Also, for readability, we let $\alpha := \Diamond_{(0,1]}\, p(x)$ and $\beta := p(x)$. Consider an observation w_1 that has the same interval structure as in the previous example and the second letter is $(\tau_1, \sigma, \varrho)$ with $\varrho(r) = d$ for some data value d and $p \notin \mathrm{def}(\sigma)$. The monitor's state for w_1 consists of the formulas:

$$\Psi_1^{\varphi,K,[]} = \alpha^K, \text{ for any } K \in \{L, N, R\}, \qquad \Psi_1^{\alpha,L,[]} = \bar{\beta}^L \vee \bar{\beta}^N \vee \bar{\bar{\beta}}^R,$$
$$\Psi_1^{\beta,K,[]} = \beta^K, \text{ for any } K \in \{L, N, R\}, \qquad \Psi_1^{\alpha,N,[x\mapsto d]} = \bar{\bar{\beta}}^R,$$
$$\Psi_1^{\beta,R,[x\mapsto d]} = \beta^R, \qquad \Psi_1^{\alpha,R,[]} = \bar{\bar{\beta}}^R.$$

Note that there are two relevant valuations for β and position 2 (which is the position of the interval R in w_1), namely $[\,]$ and $[x \mapsto d]$. This follows from the definition and it corresponds to the fact that $\bar{\bar{\beta}}^R$ is an atomic proposition of a formula both of the form $\Psi_1^{\alpha,K,[]}$ (namely, when $K \in \{L, R\}$) and of the form $\Psi_1^{\alpha,K,[x\mapsto d]}$ (namely, when $K = N$). □

4.2.2 Main Procedure. The monitor's pseudocode is shown in Listing 1. After initializing the monitor's state, the monitor loops. In each loop iteration, the monitor receives a message, updates its state according to the information extracted from the message, and outputs the computed verdicts.

We assume that each received message describes a new time point in an observation, i.e., a letter of the form $(\{\tau\}, \sigma, \varrho)$. Furthermore, we assume that each received message m contains information that identifies the *component* that

[5] We consider here that the formulas defining the semantics are first simplified. E.g., assuming that $[\![w_i, j, \nu \mid\approx \alpha\, \mathsf{U}_I\, \beta]\!]$ is reached, $k \in pos(w_i)$, and $k \geq j$, if $\mathrm{tc}_{w_i,I}(k, j) = \mathsf{f}$, then $[\![w_i, k, \nu \mid\approx \beta]\!]$ is not reached, otherwise (i.e. $\mathrm{tc}_{w_i,I}(k, j) \neq \mathsf{f}$) it is reached.

Listing 1.

```
procedure Monitor(φ)
  Init(φ)
  loop
    m ← NewMessage()
    τ, σ, ϱ, comp, seq_num := Parse(m)
    J, new := Split(τ, comp, seq_num)
    NewTimePoint(φ, J, new)
    foreach ⇓^r x. ψ in Sub(φ) with r ∈ def(ϱ) do
      PropagateDown(ψ, {τ}, x, ϱ(r))
    foreach Ψ^{p(x̄),{τ},ν} ≠ nil with x̄ ∈ def(φ), p ∈ def(σ) do
      b := (ν(x̄) ∈ σ(p))
      Ψ^{p(x̄),{τ},ν} := b
      PropagateUp(p(x̄), {τ}, b)
    NewVerdicts()
```

has sent the message to the monitor and a *sequence number*, i.e., the number of messages, including m, that the component has sent to the monitor so far. Using this information, the monitor can detect *complete* intervals, i.e., the nonsingleton intervals that do not contain the timestamp of any message that the monitor processes in later iterations. Thus, the received messages describe the "deltas" of a valid observation sequence (cf. Section 4.1), where the next observation is obtained from the previous one by applying transformation (T1), followed by (T3), possibly followed by several applications of (T2).

With the procedure NewMessage, the monitor receives a new message, for instance over a channel or a log file. Next, the monitor parses the message to recover the corresponding letter $(\{\tau\}, \sigma, \varrho)$, the component, and the sequence number. Afterwards, using the procedure Split, the monitor determines the interval J that is split (namely, the one where $\tau \in J$) and the resulting new, incomplete intervals, stored in the sequence new. Concretely, the intervals in new consist of those intervals among $J \cap [0, \tau)$, $\{\tau\}$, and $J \cap (\tau, \infty)$ that are not complete. Note that new contains at least the singleton $\{\tau\}$. The detection of complete intervals by the Split procedure is done in the same manner as in [6].

The remaining pseudocode updates the monitor's state to reflect the new observation. It first transforms formulas $\Psi^{\gamma,K,\nu}$ so that they reflect the interval structure of the new observation, with NewTimePoint. Afterwards, the monitor propagates the new data values down (the formula φ's syntax tree) with PropagateDown, and propagates newly obtained Boolean values up with PropagateUp. The procedures NewTimePoint and PropagateUp are conceptually similar to analogous procedures given in [6], although the formulas $\Psi^{\gamma,K,\nu}$ were implicit in [6]. We outline next these three procedures and give their pseudocode in Appendix B. Finally, the monitor reports the verdicts computed during the current iteration by calling the procedure NewVerdicts.

In the rest of the section, we use the convention that whenever γ or ν are not specified in a formula $\Psi^{\gamma,J_j,\nu}$ then we assume they are an arbitrary subformula of φ and respectively an arbitrary partial valuation that is relevant for γ and j.

Adding a New Time Point. The procedure NewTimePoint builds new formulas $\Psi^{\gamma,K,\nu}$ with $K \in$ new from the corresponding formulas $\Psi^{\gamma,J,\nu}$. It also updates all

formulas $\Psi^{\gamma,J,\nu}$ such that they use atomic propositions α^K with $K \in \mathsf{new}$ instead of α^J. For nontemporal formulas γ, the update is straightforward. For instance, if $\gamma = \alpha \vee \beta$ and $\Psi^{\gamma,J,\nu} = \beta^J$, then $\Psi^{\gamma,K,\nu} = \beta^K$, for each $K \in \mathsf{new}$. For temporal formulas γ, the update is more involved, although it can be performed easily by applying well-suited substitutions. To illustrate the kind of updates that are needed, suppose for example that $\gamma = \alpha \,\mathsf{U}_I\, \beta$ and that $\Psi^{\gamma,J',\nu}$, for some $J' < J$, contains the atomic proposition $\bar{\alpha}^J$. Then $\Psi^{\gamma,J',\nu}$ is updated by replacing $\bar{\alpha}^J$ with the conjunct $\bigwedge_{K \in \mathsf{new}} \bar{\alpha}^K$. Finally, we note that formulas $\Psi^{\gamma,K,\nu}$ with $K \neq J$ and without atomic propositions α^J need not be updated.

Downward Propagation. Whenever a variable x is frozen to a data value at time τ, the procedure PropagateDown updates the monitor's state to account for this fact. Concretely, this value is propagated according to the semantics through partial valuations to atomic formulas $p(\bar{y})$. The propagation is performed by starting from formulas $\Psi^{\downarrow^{\tau} x.\psi,\{\tau\},\mu}$ and recursively visiting formulas $\Psi^{\alpha,K,\nu}$ with α a subformula of ψ. For each visited formula, a new formula $\Psi^{\alpha,K,\nu[x \mapsto \varrho(r)]}$ is created, where the new formula is simply a copy of $\Psi^{\alpha,K,\nu}$. Note that the old formula $\Psi^{\alpha,K,\nu}$ may still be relevant in the future. For instance, suppose a value d is propagated from $\Psi^{\Diamond_I \beta,\{\tau\},\nu}$ to $\Psi^{\beta,K,\nu}$, copying it to $\Psi^{\beta,K,\nu[x \mapsto d]}$, and suppose also that $\bar{\beta}^K$ is an atomic proposition in $\Psi^{\Diamond_I \beta,J',\nu}$. Then $\Psi^{\beta,K,\nu}$ might be used again later when another data value d' is propagated downwards from $\Psi^{\Diamond_I \beta,\{\tau'\},\nu}$ with $\tau' \in J'$, to copy it to $\Psi^{\beta,K,\nu[x \mapsto d']}$.

Upward Propagation. The procedure PropagateUp performs the following update of the monitor's state. When a formula $\Psi^{\alpha,K,\mu}$ simplifies to a Boolean value b, then this Boolean value is propagated up the syntax tree of φ as follows: α^K is instantiated to b in every formula $\Psi^{\gamma,J',\nu}$ that has α^K as an atomic proposition, except when γ is itself an atom of φ. The formula is then simplified (using rules like $z \vee \mathsf{t} \equiv \mathsf{t}$) and if it simplifies to a Boolean value then propagation continues recursively. Note that γ is a parent of α. When $\Psi^{\varphi,\{\tau'\},[]}$ is simplified to a Boolean value b', then (τ', b') is marked as a new verdict. Propagation starts from the atoms of φ. The Boolean value t is propagated from the atom t only once, in the Init procedure. For an atom $\alpha = p(\bar{x})$, the monitor sets $\Psi^{p(\bar{x}),\{\tau\},\mu}$ to a Boolean value, if possible, according to the semantics, for all relevant valuations μ.

Recall that for temporal formulas $\gamma = \alpha \,\mathsf{U}_I\, \beta$, the formula $\Psi^{\gamma,J',\nu}$ contains atomic propositions of the form $\bar{\alpha}^K$, $\bar{\bar{\alpha}}^K$, and $\bar{\beta}^K$ instead of α^K and β^K. These atomic propositions are treated specially: they are not instantiated when K is not a singleton and the value b to be propagated is t for β formulas and f for α formulas (otherwise they are instantiated). This behavior corresponds to the meaning of these atomic proposition given in Section 4.2.1. For instance, $\bar{\beta}^K$ stands for $tp^K \vee \beta^K$ and thus it is not instantiated to t in $\Psi^{\gamma,J',\nu}$ when K is not a singleton even when $\Psi^{\gamma,K,\nu} = \mathsf{t}$, because the existence of a time point in K is not guaranteed: it might turn out that K is a complete interval. The propagation will be done later for singletons $\{\tau'\}$ with $\tau' \in K$, if and when a message with timestamp τ' arrives.

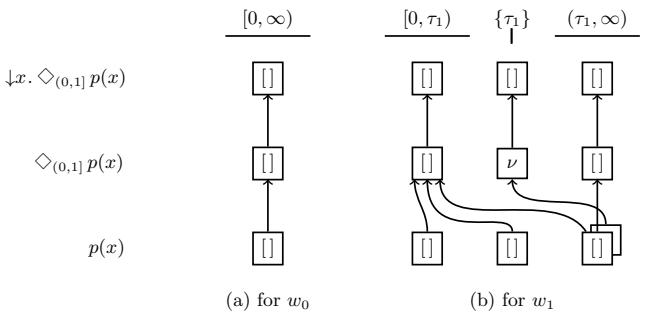


Fig. 1. Graph structures.

4.2.3 Data Structure. We have not yet described the data structure used in our pseudocode, which is needed for an efficient implementation. The data structure that we use is similar to that described in [6]. Namely, it is a directed acyclic graph. The graph's *nodes* are tuples of the form (ψ, J, ν), where ψ is a subformula of φ, J an interval, and ν a partial valuation. Each node (γ, J, ν) stores the associated propositional formula $\Psi^{\gamma,J,\nu}$. Nodes are linked via *triggers*: a trigger of a node (α, K, μ) points to a node (γ, J, ν) if and only if α^K, $\bar{\alpha}^K$, or $\bar{\bar{\alpha}}^K$ is an atomic proposition of $\Psi^{\gamma,J,\nu}$, γ is a nonatomic formula, and $\mu = \nu[x \mapsto \varrho^i_j(r)]$ if $\gamma = \downarrow^r x.\, \alpha$ and $\mu = \nu$ otherwise. Triggers are actually bidirectional: for any (outgoing) trigger there is a corresponding ingoing trigger.

This data structure allows us to directly access, given a formula $\Psi^{\alpha,K,\mu}$, all the formulas $\Psi^{\gamma,J,\nu}$ that have α^K as an atomic proposition. Also, conversely, for any formula $\Psi^{\gamma,J,\nu}$ the data structure allows us to directly access the formula $\Psi^{\alpha,K,\mu}$ for any atomic proposition α^K of $\Psi^{\gamma,J,\nu}$. These two operations are used for upward and downward propagation respectively. We note also that a node for which the associated propositional formula has simplified to a Boolean value that has been propagated can be deleted.

Figure 1 illustrates the data structure at the end of iterations 0 and 1, that is, corresponding to the observations w_0 and w_1, for the setting in Example 12. A box in the figure corresponds to a node of the graph structure, where the node's formula is given by the row, the interval by the column of the box, and the valuation by the content of the box. The valuation of the hidden box in the lower right corner is $\nu = [x \mapsto d]$, the same as the box in the middle of Figure 1(b). Arrows correspond to triggers.

4.2.4 Correctness. The following theorem establishes the monitor's correctness. We refer to Appendix C for its proof.

Theorem 13. *Let $\bar{w}$ be the valid observation sequence derived from the messages received by* Monitor. *Furthermore, let φ be a closed* MTL$^{\downarrow}$ *formula.* Monitor(φ) *is observationally complete and sound for $\bar{w}$ and φ.*

$$\square \downarrow^{cid} c.\, \downarrow^{tid} t.\, \downarrow^{sum} a.\, trans(c,t,a) \wedge a > 2000 \rightarrow \diamondsuit_{[0,3]}\, report(t) \tag{P1}$$

$$\square \downarrow^{cid} c.\, \downarrow^{tid} t.\, \downarrow^{sum} a.\, trans(c,t,a) \wedge a > 2000 \rightarrow \square_{(0,3]} \downarrow^{tid} t'.\, \downarrow^{sum} a'.\, trans(c,t',a') \rightarrow a' \leq 2000 \tag{P2}$$

$$\square \downarrow^{cid} c.\, \downarrow^{tid} t.\, \downarrow^{sum} a.\, trans(c,t,a) \wedge a > 2000 \rightarrow \left((\downarrow^{tid} t'.\, \downarrow^{sum} a'.\, trans(c,t',a') \rightarrow t = t') \,\mathsf{W}\, report(t) \right) \tag{P3}$$

$$\square \downarrow^{cid} c.\, \downarrow^{tid} t.\, \downarrow^{sum} a.\, trans(c,t,a) \wedge a > 2000 \rightarrow \square_{[0,6]} \downarrow^{tid} t'.\, \downarrow^{sum} a'.\, trans(c,t',a') \rightarrow \diamondsuit_{[0,3]}\, report(t') \tag{P4}$$

Fig. 2. $\mathrm{MTL}^{\downarrow}$ formulas used in the experimental evaluation.

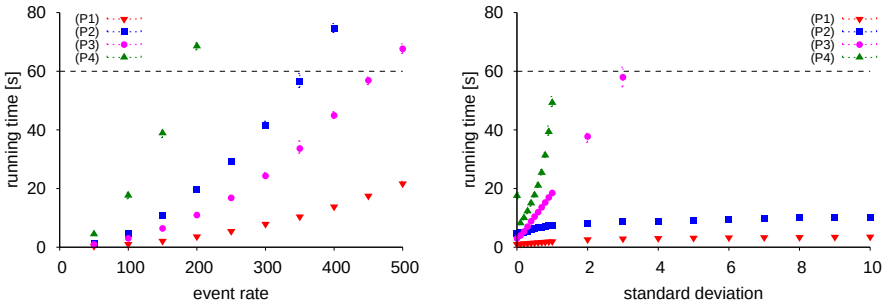
An important property class in monitoring are safety properties. We note that our monitor is not limited to formulas of this class, and the monitor is sound and observationally complete for any formula. For instance, for the formula $\varphi = \square \diamondsuit p$, which states that p is true infinitely often, the monitor will never output a verdict, as expected. It is nevertheless observationally complete for any valid observation sequence $\bar{w}$, since $[w_i, \tau, \nu \models\!\!\approx \varphi] = \bot$, for any $i \in \mathbb{N}$, $\tau \in \mathbb{Q}_{\geq 0}$, and partial valuation ν.

Besides correctness requirements, time and space requirements are also important. Concerning time requirements, recall that the underlying decision problem is PSPACE-complete, see Theorem 7. Concerning space requirements, note that space cannot be bounded even in the setting without message loss and with in-order delivery. Indeed, consider the formula $\square \downarrow x.\, p(x) \rightarrow \square_{(0,\infty)} \neg p(x)$ stating that the parameter of p events are fresh at each time point. Any monitor must store the parameters seen. Further investigation of the time and space complexity of the monitoring procedure is left for future work.

5 Experiments

We have implemented our monitor in a prototype tool, written in the programming language Go. Our tool either reads messages from a log file or over a UDP socket. Our experimental evaluation focuses on the prototype's performance in settings with different message orderings.

Setup. We monitor the formulas in Figure 2, which vary in their temporal requirements and the data involved. They express compliance policies from the banking domain and are variants of policies that have been used in previous case studies [5]. Furthermore, we synthetically generate log files. Each log spans over 60 time units (i.e., a minute) and contains one event per time point. The number of events in a log is determined by the *event rate*, which is the approximate number of events per time unit (i.e., a second). For each time point i, with $0 \leq i < 60$, the number of events with a timestamp in the time interval $[i, i+1)$ is randomly chosen within 10% of the event rate. The events and their parameters are randomly chosen such that the number of violations is in a provided range. For instance, a log with event rate 100 comprises approximately 6000 events. Finally, we use a standard desktop computer with a 2.8Ghz Intel Core i7 CPU, 8GB of RAM, and the Linux operating system.



(a) in-order (b) out-of-order

Fig. 3. Running times (where each data point shows the mean over five logs together with the minimum and maximum, which are very close to the mean).

In-order Delivery. In our first setting, messages are received ordered by their timestamps and are never lost. Namely, all events of the log are processed in the order of their timestamps. Figure 3(a) shows the running times of our prototype tool for different event rates. Note that each log spans 60 seconds and a running time below 60 seconds essentially means that the events in the log could have been processed online. The dashed horizontal lines mark this border.

Out-of-order Delivery. In our second setting, messages can arrive out of order but they are not lost. We control the degree of message arrival disruption as follows. For the events in a generated log file, we choose their *arrival times*, which provide the order in which the monitor processes them. The arrival time of an event is derived from the event's timestamp by offsetting it by a random delay with respect to the normal distribution with a mean of 10 time units and a chosen standard deviation. In particular, for an event's timestamp τ and for a standard deviation $\sigma > 0$, it holds that an arrival time τ' is in the interval $[\tau+10-\sigma, \tau+10+\sigma]$ with probability 0.68 and in $[\tau+10-2\sigma, \tau+10+2\sigma]$ with probability 0.95. For the degenerate case $\sigma = 0$, the reordered log is identical to the original log. We remark that the mean value does not impact the event reordering because it does not influence the difference between arrival times.

Figure 3(b) shows the prototype's running times on logs with the fixed event rate 100 for different deviations. For instance, for (P1), the logs are processed in around 1 second when $\sigma = 0$ and in 3.5 seconds when $\sigma = 10$.

Interpretation. The running times are nonlinear in the event rate for all four formulas. This is expected from Theorem 7. The growth is caused by the data values occurring in the events. A log with a higher event rate contains more different data values and the monitor's state must account for those. As expected, (P1) is the easiest to monitor. It has only one block of freeze quantifiers. Note that (P1)–(P3) have two temporal connectives, where one is the outermost connective $\Box$, which is common to all formulas, whereas (P4) has an additional

nesting of temporal connectives. The time window is also larger than in (P1) and (P2).

Also expected, the running times increase when messages are received out of order. Again, (P4) is worst. For (P1) and (P2), however, the growth rate decreases for larger standard deviations. This is because, as the standard deviation increases, all the events within the relevant time window for a given time point arrive at the monitor in an order that is increasingly close to the uniformly random one. The running times thus stabilize. Due to the larger time window, this effect does not take place for (P4). The running times for (P3) increase more rapidly than for (P1) and (P2) because of the data values and the "continuation formula" of the derived unbounded temporal connective W.

To put the experimental results in perspective, we carried out two additional experiments. First, we conducted similar experiments on formulas with their freeze quantifiers removed and further transformed into propositional formulas, as described in Appendix D.2. We make similar observations in the propositional setting. However, in the propositional setting the running times increase linearly with respect to the event rate and logs are processed several orders of magnitude faster. Overall, one pays a price at runtime for the expressivity gain given by the freeze quantifier. Second, we compared our prototype with the MONPOLY tool [3]. MONPOLY's specification language is, like $\text{MTL}^{\downarrow}$, a point-based real-time logic. It is richer than $\text{MTL}^{\downarrow}$ in that it admits existential and universal quantification over domain elements. However, MONPOLY specifications are syntactically restricted in that temporal future connectives must be bounded (except for the outermost connective $\Box$). Thus, (P3) does not have a counterpart in MONPOLY's specification language. MONPOLY handles the counterparts of (P1), (P2), and (P4) significantly faster, up to three orders of magnitude. Comparing the performance of both tools should, however, be taken with a grain of salt. First, MONPOLY only handles the restrictive setting where messages must be received in-order. Second, MONPOLY outputs violations for specifications with (bounded) future only after all events in the relevant time window are available, whereas our prototype outputs verdicts promptly.[6] Finally, while MONPOLY is optimized, our prototype is not.

In summary, our experimental evaluation shows that one pays a high price to handle an expressive specification language together with message delays. Nevertheless, our prototype's performance is sufficient to monitor systems that generate hundreds of events per second, and the prototype can be used as a starting point for a more efficient implementation.

[6] For instance, for the formula $\Box_{[0,3]}\, p$, if p does not hold at time point i with timestamp τ, then our prototype outputs the corresponding verdict directly after processing the time point i, whereas MONPOLY reports this violation at the first time point with a timestamp larger than $\tau + 3$.

6 Related Work

Runtime verification is a well-established approach for checking at runtime whether a system's execution fulfills a given specification. Various monitoring algorithms exist, e.g., [2,5,10,18]. They differ in the specifications they can handle (some of the specification languages account for data values) and they make different assumptions on the monitored systems. A commonly made assumption is that a monitor has always complete knowledge about the system behavior up to the current time. Only a few runtime-verification approaches exist that relax this assumption. Note that this assumption is, for instance, not met in distributed systems whose components communicate over unreliable channels.

Closest to our work is the runtime-verification approach by Basin et al. [6]. We use the same system model and our monitoring algorithm extends their monitoring algorithm for the propositional real-time logic MTL. Namely, our algorithm handles the more expressive specification language $\text{MTL}^{\downarrow}$ and handles data values. Furthermore, we present a semantics for $\text{MTL}^{\downarrow}$ that is based on three truth values and uses observations instead of timed words. This enables us to cleanly state correctness requirements and establish stronger correctness guarantees for the monitoring algorithm. Basin et al.'s completeness result [6] is limited in that it assumes that all messages are eventually received. Finally, Basin et al. [6] do not evaluate their monitoring algorithm experimentally.

Colombo and Falcone [11] propose a runtime-verification approach, based on formula rewriting, that also allows the monitor to receive messages out of order. Their approach only handles the propositional temporal logic LTL with the three-valued semantics proposed by Bauer et al. [9]. In a nutshell, their approach unfolds temporal connectives as time progresses and special propositions act as placeholders for subformulas. The subsequent assignment of these placeholders to Boolean truth values triggers the reevaluation and simplification of the formula. Their approach only guarantees soundness but not completeness, since the simplification rules used for formula rewriting are incomplete. Finally, its performance with respect to out-of-order messages is not evaluated.

The monitoring approaches by Garg et al. [13] and Basin et al. [4], both targeting the auditing of policies on system logs, also account for knowledge gaps, i.e., logs that may not contain all the actions performed by a system. Both approaches handle rich policy specification languages with first-order quantification and a three-valued semantics. Garg et al.'s approach [13], which is based on formula rewriting, is however, not suited for online use since it does not process logs incrementally. It also only accounts for knowledge gaps in a limited way, namely, the interpretation of a predicate symbol cannot be partially unknown, e.g., for certain time periods. Furthermore, their approach is not complete. Basin et al.'s approach [4], which is based on their prior work [5], can be used online. However, the problem of how to incrementally output verdicts as prior knowledge gaps are resolved is not addressed, and thus it does not deal with out-of-order events. Moreover, the semantics of the specification language handled does not

reflect a monitor's partial view about the system behavior. Instead, it is given for infinite data streams that represent system behavior in the limit.

Several dedicated monitoring approaches for distributed systems have been developed [7, 19, 21]. These approaches only handle less expressive specification languages, namely, the propositional temporal logic LTL or variants thereof. Furthermore, none of them handles message loss or out-of-order delivery of messages, problems that are inherent to such systems because of crashing components and nonuniform delays in message delivery.

A similar extension of MTL with the freeze quantifier is defined by Feng et al. [12]. Their analysis focuses on the computational complexity of the path-checking problem. However, they use a finite trace semantics, which is less suitable for runtime verification. Out-of-order messages are also not considered.

Temporal logics with additional truth values have also been considered in model checking finite-state systems. Closest to our three-valued semantics is the three-valued semantics for LTL by Goidefroid and Piterman [14], which is based on infinite words, not observations (Definition 2). Similar to (T3) of Definition 2, a proposition with the truth value $\bot$ at a position can be refined by `t` or `f`. In contrast, their semantics does not support refinements that add and delete letters, cf. (T1) and (T2) of Definition 2.

7 Conclusion

We have presented a runtime-verification approach to checking real-time specifications given as $\text{MTL}^{\downarrow}$ formulas. Our approach handles the practically-relevant setting where messages sent to the monitors can be delayed or lost, and it provides soundness and completeness guarantees. Although our experimental evaluation is promising, our approach does not yet scale to monitor systems that generate thousands or even millions of events per second. This requires additional research, including algorithmic optimizations. We plan to do this in future work, as well as to deploy and evaluate our approach in realistic, large-scale case studies.

Acknowledgments. This work was partly performed within the 5G-ENSURE project (www.5gensure.eu) and received funding from the EU Framework Programme for Research and Innovation Horizon 2020 under grant agreement no. 671562. David Basin acknowledges support from the Swiss National Science Foundation grant Big Data Monitoring (167162).

References

1. R. Alur and T. A. Henzinger. Logics and models of real time: A survey. In *Proceedings of the 1991 REX Workshop on Real Time: Theory in Practice*, volume 600 of *Lect. Notes Comput. Sci.*, pages 74–106. Springer, 1992.
2. H. Barringer, A. Goldberg, K. Havelund, and K. Sen. Rule-based runtime verification. In *Proceedings of the 5th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI)*, volume 2937 of *Lect. Notes Comput. Sci.*, pages 44–57. Springer, 2004.

3. D. Basin, M. Harvan, F. Klaedtke, and E. Zălinescu. MONPOLY: Monitoring usage-control policies. In *Proceedings of the 2nd International Conference on Runtime Verification (RV)*, volume 7186 of *Lect. Notes Comput. Sci.*, pages 360–364. Springer, 2012.
4. D. Basin, F. Klaedtke, S. Marinovic, and E. Zălinescu. Monitoring compliance policies over incomplete and disagreeing logs. In *Proceedings of the 3rd International Conference on Runtime Verification (RV)*, volume 7687 of *Lect. Notes Comput. Sci.*, pages 151–167. Springer, 2013.
5. D. Basin, F. Klaedtke, S. Müller, and E. Zălinescu. Monitoring metric first-order temporal properties. *J. ACM*, 62(2), 2015.
6. D. Basin, F. Klaedtke, and E. Zălinescu. Failure-aware runtime verification of distributed systems. In *Proceedings of the 35th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 45 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 590–603. Schloss Dagstuhl – Leibniz Center for Informatics, 2015.
7. A. Bauer and Y. Falcone. Decentralised LTL monitoring. In *Proceedings of the 18th International Symposium on Formal Methods (FM)*, volume 7436 of *Lect. Notes Comput. Sci.*, pages 85–100. Springer, 2012.
8. A. Bauer, J. Küster, and G. Vegliach. The ins and outs of first-order runtime verification. *Form. Methods Syst. Des.*, 46(3):286–316, 2015.
9. A. Bauer, M. Leucker, and C. Schallhart. Comparing LTL semantics for runtime verification. *J. Logic Comput.*, 20(3):651–674, 2010.
10. A. Bauer, M. Leucker, and C. Schallhart. Runtime verification for LTL and TLTL. *ACM Trans. Softw. Eng. Meth.*, 20(4), 2011.
11. C. Colombo and Y. Falcone. Organising LTL monitors over distributed systems with a global clock. *Form. Methods Syst. Des.*, 49(1):109–158, 2016.
12. S. Feng, M. Lohrey, and K. Quaas. Path checking for MTL and TPTL over data words. In *Proceedings of the 19th International Conference on Development in Language Theory (DLT)*, volume 9168 of *Lect. Notes Comput. Sci.*, pages 326–339. Springer, 2015.
13. D. Garg, L. Jia, and A. Datta. Policy auditing over incomplete logs: Theory, implementation and applications. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)*, pages 151–162. ACM Press, 2011.
14. P. Goidefroid and N. Piterman. LTL generalized model checking revisited. *Int. J. Softw. Tools Technol. Trans.*, 13(6):571–584, 2011.
15. T. A. Henzinger. Half-order modal logic: how to prove real-time properties. In *Proceedings of the 9th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 281–296. ACM Press, 1990.
16. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Syst.*, 2(4):255–299, 1990.
17. O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Proceedings of the Joint International Conferences on Formal Modelling and Analysis of Timed Systems (FORMATS) and on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT)*, volume 3253 of *Lect. Notes Comput. Sci.*, pages 152–166. Springer, 2004.
18. P. O. Meredith, D. Jin, D. Griffith, F. Chen, and G. Roşu. An overview of the MOP runtime verification framework. *Int. J. Softw. Tools Technol. Trans.*, 14(3):249–289, 2012.
19. M. Mostafa and B. Bonakdarbour. Decentralized runtime verification of LTL specifications in distributed systems. In *Proceedings of the 29th IEEE International*

Parallel and Distributed Processing Symposium (IPDPS). IEEE Computer Society, 2015.
20. J. Ouaknine and J. Worrell. On metric temporal logic and faulty turing machines. In *Proceedings of the 9th International Conference on Foundations of Software Science and Computation Structures (FOSSACS)*, volume 3921 of *Lect. Notes Comput. Sci.*, pages 217–230. Springer, 2006.
21. K. Sen, A. Vardhan, G. Agha, and G. Roşu. Efficient decentralized monitoring of safety in distributed systems. In *Proceedings of the 26th International Conference on Software Engineering (ICSE)*, pages 418–427. IEEE Computer Society, 2004.

A Additional MTL$^{\downarrow}$ Details

A.1 Standard Boolean Semantics

MTL$^{\downarrow}$ with a Boolean semantics has the same syntax as MTL$^{\downarrow}$ with a three-valued semantics as defined in Section 3. We use the same syntactic sugar and conventions from Section 3. The Boolean semantics for MTL$^{\downarrow}$ is defined over timed words.

To define MTL$^{\downarrow}$'s Boolean semantics, we introduce the following notation and terminology. Recall that D—the *data domain*—is a nonempty set of values. Furthermore, let Σ be the set of the pairs (σ, ϱ), where σ is a total function over P with $\sigma(p) \subseteq D^{\iota(p)}$ for $p \in P$ and ϱ is a total function over R with $\varrho(r) \in D$ for $r \in R$. In the following, timed words are always over the alphabet Σ. Note that the letter at position $i \in \mathbb{N}$ of a timed word (over Σ) is of the form $(\tau_i, \sigma_i, \varrho_i)$, where σ_i interprets the predicate symbols at time τ_i and ϱ_i determines the values stored in the registers in R at time τ_i. As for observations, we call the positions $i \in \mathbb{N}$ of a timed word w the *time points* of w. Furthermore, we call τ_i the *timestamp* of the time point $i \in \mathbb{N}$.

Remark 14. Observations generalize the notion of a finite prefix of a timed word. Let $w = (\tau_1, \sigma_1, \varrho_1)(\tau_2, \sigma_2, \varrho_2) \dots$ be a timed word. For the prefix of length $n \in \mathbb{N}$ of w, we define the word w_n as $(\{\tau_1\}, \sigma_1, \varrho_1) \dots (\{\tau_n\}, \sigma_n, \varrho_n)(I, [\,], [\,])$, with $I = \mathbb{Q}_{\geq 0} \setminus \big(\bigcup_{1 \leq i \leq n} [0, \tau_i]\big)$. That is, we transform the timestamps of the prefix into singletons and attach a last letter, which can be seen as a placeholder for the remaining letters in w. The words w_n for $n \in \mathbb{N}$ are observations. Obviously, $w_0 = ([0, \infty), [\,], [\,])$ is an observation. For $n > 0$, we obtain w_n from w_{n-1} by applying the transformation (T1) with the timestamp τ_n on w_{n-1}'s last letter, then applying (T2) to delete the letter with the interval (τ_{n-1}, τ_n), and finally applying (T3) on the letter with interval $\{\tau_n\}$ to populate it with σ_n and ϱ_n.

MTL$^{\downarrow}$'s Boolean semantics is defined inductively over the formula structure. In particular, similar to the function $\varphi \mapsto [\![w, i, \nu \mathrel{\mathbb{P}} \varphi]\!]$ defined in Section 3, we define a function $\varphi \mapsto [\![w, i, \nu \models p(\bar{x})]\!] \in \mathsf{2}$, for a given timed word w, a time point $i \in \mathbb{N}$, and a valuation $\nu : V \rightharpoonup D$. Let $w = (\tau_0, \sigma_0, \varrho_0)(\tau_1, \sigma_1, \varrho_1) \dots$.

$$
\begin{aligned}
[\![w, i, \nu \models \mathsf{t}]\!] &:= \mathsf{t} \\
[\![w, i, \nu \models p(\bar{x})]\!] &:= \begin{cases} \mathsf{t} & \text{if } \nu(\bar{x}) \in \sigma_i(p) \\ \mathsf{f} & \text{otherwise} \end{cases} \\
[\![w, i, \nu \models \downarrow^{r} x.\, \varphi]\!] &:= [\![w, i, \nu[x \mapsto \varrho_i(r)] \models \varphi]\!] \\
[\![w, i, \nu \models \neg\varphi]\!] &:= \neg [\![w, i, \nu \models \varphi]\!] \\
[\![w, i, \nu \models \varphi \vee \psi]\!] &:= [\![w, i, \nu \models \varphi]\!] \vee [\![w, i, \nu \models \psi]\!] \\
[\![w, i, \nu \models \varphi \,\mathsf{U}_I\, \psi]\!] &:= \textstyle\bigvee_{j \in \{\ell \in \mathbb{N} \mid \tau_\ell - \tau_i \in I\}} \big([\![w, j, \nu \models \psi]\!] \wedge \bigwedge_{i \leq k < j} [\![w, k, \nu \models \varphi]\!] \big)
\end{aligned}
$$

Note that we abuse notation here and identify the logic's constant symbol t with the Boolean value t, and the connectives $\neg$ and $\vee$ with the corresponding logical operators.

Definition 15. *For $\tau \in \mathbb{Q}_{\geq 0}$, a timed word w, a valuation ν, and a formula φ, we define $[w, \tau, \nu \models \varphi] := [\![w, j, \nu \models \varphi]\!]$, provided that there is some time point j in w with timestamp τ, and $[w, \tau, \nu \models \varphi] := \bot$, otherwise.*

The following theorem shows that the semantics of MTL$^{\downarrow}$ on observations conservatively extends the logic's semantics on timed words. In particular, if a formula evaluates to a Boolean value for an observation for a given time $\tau \in \mathbb{Q}_{\geq 0}$, it has the same Boolean value on any timed word that refines the observation.

Theorem 16. *Let φ be a formula, μ a partial valuation, ν a total valuation, u an observation, v a timed word, and $\tau \in \mathbb{Q}_{\geq 0}$. If $u \sqsubseteq v$ and $\mu \sqsubseteq \nu$ then $[u, \tau, \mu \mathrel{\overline{\models}} \varphi] \preceq [v, \tau, \nu \models \varphi]$.*

Proof. Let $(I_i, \sigma_i, \varrho_i)$ and $(\tau_j, \sigma'_j, \varrho'_j)$, for $i \in pos(u)$ and $j \in \mathbb{N}$ be the letters of u and v, respectively. As $u \sqsubseteq v$, we have that there is a function $\pi : \mathbb{N} \to pos(u)$ such that (R1) $\tau_j \in I_{\pi(j)}$, (R2) $\sigma_{\pi(j)} \sqsubseteq \sigma'_j$, and (R3) $\varrho_{\pi(j)} \sqsubseteq \varrho'_j$, for every $j \in \mathbb{N}$. It is easy to see that π is monotonic.

We prove by structural induction on φ that for any time point $i' \in \mathbb{N}$ and partial valuations μ and ν with $\mu \sqsubseteq \nu$, it holds that $[\![u, \pi(i'), \mu \mathrel{\overline{\models}} \varphi]\!] \preceq [\![v, i', \nu \models \varphi]\!]$. The theorem's statement easily follows from this property. For the reminder of the proof, we fix an arbitrary time point $i' \in \mathbb{N}$ and arbitrary partial valuations μ and ν with $\mu \sqsubseteq \nu$. Let $i = \pi(i')$.

When $[\![u, i, \mu \mathrel{\overline{\models}} \varphi]\!] = \bot$ the statement clearly holds. Hence, it suffices to show that $[\![u, i, \mu \mathrel{\overline{\models}} \varphi]\!] = [\![v, i', \nu \models \varphi]\!]$, provided that $[\![u, i, \mu \mathrel{\overline{\models}} \varphi]\!] \in \mathsf{2}$.

Base cases. The case $\varphi = \mathsf{t}$ is trivial. Consider the case $\varphi = p(\bar{x})$, for some $p \in P$. As $[\![u, i, \nu \mathrel{\overline{\models}} p(\bar{x})]\!] \in \mathsf{2}$, it holds that $\bar{x} \in \mathrm{def}(\nu)$ and $p \in \mathrm{def}(\sigma_i)$. It follows from the theorem's premise that $\mu(\bar{x}) = \nu(\bar{x})$, and from (R2) that $\sigma_i(p) = \sigma'_{i'}(p)$. Thus $[\![u, i, \nu \mathrel{\overline{\models}} p(\bar{x})]\!] = [\![v, i', \nu \models p(\bar{x})]\!]$.

Inductive cases. The cases where φ is of the form $\neg\alpha$ or $\alpha \vee \beta$ are straightforward and omitted. The remaining cases are as follows.

First, assume that φ is of the form $\downarrow^{r} x \,.\, \psi$. Let $\eta = \mu[x \mapsto \varrho_i(r)]$ and $\eta' = \nu[x \mapsto \varrho'_{i'}(r)]$. By (R3), we have that if $r \in \mathrm{def}(\varrho_i)$ then $r \in \mathrm{def}(\varrho'_{i'})$ and $\varrho_i(r) = \varrho'_{i'}(r)$, and thus $\eta(x) = \eta'(x)$. Furthermore, if $r \notin \mathrm{def}(\varrho_i)$, then $x \notin \mathrm{def}(\eta)$. This shows that $\eta \sqsubseteq \eta'$. It follows from the induction hypothesis that $[\![u, i, \eta \mathrel{\overline{\models}} \psi]\!] \preceq [\![v, i', \eta' \models \psi]\!]$. We conclude that $[\![u, i, \mu \mathrel{\overline{\models}} \varphi]\!] = [\![v, i', \nu \models \varphi]\!]$.

Finally, assume that φ is of the form $\alpha \,\mathsf{U}_I\, \beta$. We consider first the case $[\![u, i, \mu \mathrel{\overline{\models}} \alpha \,\mathsf{U}_I\, \beta]\!] = \mathsf{t}$. By definition, there is a $j \geq i$ such that $\mathrm{tp}_u(j) = \mathsf{t}$, $\mathrm{tc}_{u,I}(i, j) = \mathsf{t}$, $[\![u, j, \mu \mathrel{\overline{\models}} \beta]\!] = \mathsf{t}$, and $\mathrm{tp}_u(k) \to [\![u, k, \mu \mathrel{\overline{\models}} \alpha]\!] = \mathsf{t}$, for all k with $i \leq k < j$. As j is a time point in u, there is a time point j' in v such that $\pi(j') = j$. From (R1) we have that $\tau_{j'} = \mathrm{ts}_u(j)$. As $\mathrm{tc}_{u,I}(i, j) = \mathsf{t}$ and $I_j = \{\tau_{j'}\}$, we have that $\tau_{j'} - \tau \in I$, for all $\tau \in I_i$. From (R1), we have that $\tau_{i'} \in I_i$. Thus $\tau_{j'} - \tau_{i'} \in I$ (I1). From the induction hypothesis, we have that $[\![u, j, \mu \mathrel{\overline{\models}} \beta]\!] \preceq [\![v, j', \nu \models \beta]\!]$. Hence $[\![v, j', \nu \models \beta]\!] = \mathsf{t}$ (I2). From the induction hypothesis, we also have that $[\![u, \pi(k'), \mu \mathrel{\overline{\models}} \alpha]\!] \preceq [\![v, k', \nu \models \alpha]\!]$, for any $k' \in \mathbb{N}$. Let $k' \in \mathbb{N}$ such that $i' \leq k' < j'$, and let $k = \pi(k')$. From the monotonicity of π we have that $i \leq k \leq j$. Since j is a time point in u we also have that $k < j$. As $\mathrm{tp}_u(k) \to [\![u, k, \mu \mathrel{\overline{\models}} \alpha]\!] = \mathsf{t}$ and tp is never f by definition, we have

that $[\![u, k, \mu \approx\!\!\!\!\approx \alpha]\!] = \mathsf{t}$. Then $[\![v, k', \mu \models \alpha]\!] = \mathsf{t}$ (I3). Summing up, from (I1), (I2), (I3), and as k' was chosen arbitrarily, we obtain that $[\![v, i', \nu \models \alpha \,\mathsf{U}_I\, \beta]\!] = \mathsf{t}$.

The case $[\![u, i, \mu \approx\!\!\!\!\approx \alpha \,\mathsf{U}_I\, \beta]\!] = \mathsf{f}$ is as follows. Note that each disjunct in the definition of $[\![u, i, \mu \approx\!\!\!\!\approx \alpha \,\mathsf{U}_I\, \beta]\!]$ is f. We fix an arbitrary $j' \geq i'$ and let $j = \pi(j')$. It holds that $\mathrm{tp}_u(j) \wedge \mathrm{tc}_{u,I}(i,j) \wedge [\![u, j, \mu \approx\!\!\!\!\approx \beta]\!] \wedge \bigwedge_{i \leq k < j} (\mathrm{tp}_u(k) \rightarrow [\![u, k, \mu \approx\!\!\!\!\approx \alpha]\!]) = \mathsf{f}$. Since $\mathrm{tp}_u(j) \neq \mathsf{f}$, one of the remaining conjuncts must be f.

(1) If $\mathrm{tc}_{u,I}(i,j) = \mathsf{f}$, then $\tau' - \tau'' \notin I$, for all $\tau'' \in I_i$ and $\tau' \in I_j$. From (R1), $\tau_{i'} \in I_i$ and $\tau_{j'} \in I_j$. It follows that $\tau_{j'} - \tau_{i'} \notin I$.
(2) If $[\![u, j, \mu \approx\!\!\!\!\approx \beta]\!] = \mathsf{f}$, then $[\![v, j', \nu \models \beta]\!] = \mathsf{f}$, by induction hypothesis.
(3) If $\mathrm{tp}_u(k) \rightarrow [\![u, k, \mu \approx\!\!\!\!\approx \alpha]\!] = \mathsf{f}$, for some k with $i \leq k < j$, then $\mathrm{tp}_u(k) = \mathsf{t}$ and $[\![u, k, \mu \approx\!\!\!\!\approx \alpha]\!] = \mathsf{f}$. It follows as before that there is a k' with $i' \leq k' < j'$ such that $[\![v, k', \nu \models \alpha]\!] = \mathsf{f}$.

We have thus obtained that either $\tau_{i'} - \tau_{j'} \notin I$ or one of the conjuncts of $[\![v, j', \nu \models \beta]\!] \wedge \bigwedge_{i' \leq k' < j'} [\![v, k', \nu \models \alpha]\!]$ is f. In other words, if j' is such that $\tau_{j'} - \tau_{i'} \in I$, then $[\![v, j', \nu \models \beta]\!] \wedge \bigwedge_{i' \leq k' < j'} [\![v, k', \nu \models \alpha]\!] = \mathsf{f}$. As j' was chosen arbitrarily, we conclude that $[\![v, i', \nu \models \alpha \,\mathsf{U}_I\, \beta]\!] = \mathsf{f}$. □

A.2 Correctness Requirements

In this section, we formulate similar monitoring requirements as in Section 4.1, formulating them this time with respect to the Boolean MTL$^{\downarrow}$ semantics. We then argue that these requirements are too strong.

For an observation w, we define $U_w := \{v \mid v \text{ a timed word with } w \sqsubseteq v\}$. Intuitively, U_w contains all the timed words that are compatible with the reported system behavior that a monitor received so far, represented by w.

Definition 17. *Let M be a monitor, φ a formula, and $\bar{w}$ a valid observation sequence.*

- *M is* sound *for $\bar{w}$ and φ if for all valuations ν and $i \in \mathbb{N}$, whenever $(\tau, b) \in M(w_i)$ then $\bigwedge_{v \in U_{w_i}} [v, \tau, \nu \models \varphi] = b$.*
- *M is* complete *for $\bar{w}$ and φ if for all valuations ν, $i \in \mathbb{N}$, and $\tau \in \mathbb{Q}_{\geq 0}$, whenever $\bigwedge_{v \in U_{w_i}} [v, \tau, \nu \models \varphi] \in \mathsf{2}$ then $(\tau, b) \in \bigcup_{j \leq i} M(w_j)$, for some $b \in \mathsf{2}$.*

We say that M is sound *if M is sound for all valid observation sequences $\bar{w}$ and formulas φ. The definition of M being* complete *is analogous.*

The correctness requirements in Definition 17 are related to the use of a three-valued "runtime-verification" semantics for a specification language, as introduced by Bauer et al. [10] for LTL and adopted by other runtime-verification approaches (e.g. [8]). Intuitively speaking, both a sound and complete monitor and a monitor implementing a three-valued "runtime-verification" semantics output a verdict as soon as the specification has the same Boolean value on every extension of the monitor's current knowledge. However, Bauer et al. [10] make no distinction between a monitor's soundness and its completeness. Distinguishing these two requirements separates concerns and this is important, as explained next. Ideally, a monitor is both sound and complete. However, achieving both of these properties can be hard or even impossible for non-trivial specification languages, as we now explain, when relying on the standard Boolean semantics.

Remark 18. For a specification language, having a sound and complete monitor M for a specification language is at least as hard as checking satisfiability for this language. For $\text{MTL}^{\downarrow}$, a closed formula φ is satisfiable iff $(0, \mathsf{f}) \notin M(w_1)$, assuming that 0 is always the timestamp of the first time point of a timed word and the observation w_1 is obtained from the monitor's initial knowledge w_0 by (T1) for the timestamp 0. Note that $w_0 = ([0, \infty), [\,], [\,])$, U_{w_0} is the set of all timed words, and $U_{w_1} = U_{w_0}$. The propositional fragment of $\text{MTL}^{\downarrow}$ is already undecidable [20]. There are fragments that are decidable but the complexity is usually high. Recall that for LTL, checking satisfiability is already PSPACE-complete.[7]

Some monitoring approaches try to compensate for this complexity burden with a pre-processing step. For instance, the monitoring approach of Bauer et al. [10] translates an LTL formula into an automaton prior to monitoring. The resulting automaton can be directly used for sound and complete monitoring in environments where messages are neither delayed nor lost. However, there are no obvious extensions for handling out-of-order message delivery. Furthermore, not every specification language has such a corresponding automaton model and, for the ones where translations are known, the automaton construction can be very costly. For LTL, the size of the automaton is already in the worst case doubly exponential in the size of the formula [10].

In contrast the correctness requirements from Definition 9 are weaker and achievable. This is due to the three-valued semantics, based on Kleene logic, for $\text{MTL}^{\downarrow}$ over observations. Furthermore, note that the three-valued semantics for $\text{MTL}^{\downarrow}$ conservatively extends the standard Boolean semantics, as shown by Theorem 16.

Theorem 16 allows us to prove that the completeness requirement from Definition 9 is indeed a weaker notion than completeness requirement from Definition 17, while the soundness requirement from Definition 17 offers the same correctness guarantees as the one from Definition 9.

Theorem 19. *Let M be a monitor. If M is observationally sound, then M is sound. If M is complete, then M is observationally complete.*

Proof. First, let M be an observationally sound monitor. Let φ be a formula and $\bar{w}$ a valid observation sequence. Furthermore, let ν be a total valuation, $i \in \mathbb{N}$, $\tau \in \mathbb{Q}_{\geq 0}$, and $b \in \mathsf{2}$ such that $(\tau, b) \in M(i)$. Then, by definition, $[w_i, \tau, \nu \models\!\!\approx \varphi] = b$. Now, let $v \in U_{w_i}$. We have that $w_i \sqsubseteq v$. Then from Theorem 16 we have that $[v, \tau, \nu \models \varphi] = b$. As v was chosen arbitrarily, we get $\bigwedge_{v \in U_{w_i}} [v, \tau, \nu \models \varphi] = b$. Thus M is a sound monitor.

Now, let M be a complete monitor. Let φ be a formula and $\bar{w}$ a valid observation sequence. Furthermore, let ν be a partial valuation, $i \in \mathbb{N}$, $\tau \in \mathbb{Q}_{\geq 0}$ such that $[w_i, \tau, \nu \models \varphi] = b'$ for some $b' \in \mathsf{2}$. Let ν' be a total valuation with

[7] In contrast to model checking, runtime verification is often advertised as a "lightweight" verification technique. In terms of complexity classes, the problem of soundly and completely monitoring finite-state systems with respect to LTL specifications is however at least as hard as the corresponding model-checking problem, which is PSPACE-complete.

$\nu \sqsubseteq \nu'$. Also, let $v \in U_{w_i}$. Then, as $w_i \sqsubseteq v$, from Theorem 16 we obtain that $[v, \tau', \nu \models \varphi] = b'$. As v was chosen arbitrarily, we get $\bigwedge_{v \in U_{w_i}} [v, \tau, \nu' \models \varphi] = b'$. As M is complete, it follows, by definition, that there is $b \in \mathsf{2}$ and $j \leq i$ such that $(\tau, b) \in M(w_j)$. Thus M is an observationally complete monitor. □

A.3 Proof of Theorem 6

The following lemma characterizes the $\sqsubseteq$ relation on observations.

Lemma 20. *Let w and w' be observations with letters $(I_i, \sigma_i, \varrho_i)$ and respectively $(I'_j, \sigma'_j, \varrho'_j)$, for $i \in pos(w)$ and $j \in pos(w')$. If $w \sqsubseteq w'$ then there is a monotonic function $\pi : pos(w') \to pos(w)$ with the following properties.*
(R1) $I'_j \subseteq I_{\pi(j)}$, for all $j \in pos(w')$.
(R2) $\sigma_{\pi(j)} \sqsubseteq \sigma'_j(p)$, for all $j \in pos(w')$ and $p \in P$.
(R3) $\varrho_{\pi(j)} \sqsubseteq \varrho'_j$, for all $j \in pos(w')$ and $r \in R$.

Proof (sketch). If $w = w'$ then take π to be the identity. If w' is obtained from w using one of the transformations, that is, if $w \sqsubset w'$, then, for each transformation it is easy to construct a function π satisfying the stated properties. If $w \sqsubsetneq w'$, then there is a sequence $(w_i)_{0 \leq i \leq n}$ of observations, with $n \geq 1$, such that $w = w_0 \sqsubset w_1 \sqsubset \cdots \sqsubset w_n = w'$. From the previous observation there is a sequence of functions $\pi_i : pos(w_i) \to pos(w_{i-1})$, with $1 \leq i \leq n$, each satisfying the stated properties. Then it is easy to see that their composition $\pi = \pi_1 \circ \cdots \circ \pi_n$ also satisfies these properties. □

The proof of Theorem 6 is similar to that of Theorem 16 and is thus omitted. We just note that the omitted proof uses properties (R1) to (R3) from Lemma 20, which correspond to the ones given in the proof of Theorem 16.

A.4 Proof of Theorem 7

We first show that the problem is PSPACE-hard by reducing the satisfiability problem for quantified Boolean logic (QBL) to it. For simplicity, we assume that MTL$^{\downarrow}$ comprises the temporal past-time connectives $\blacklozenge$ ("once") and $\blacksquare$ ("historically"), which are the counterparts of $\Diamond$ and $\Box$. Their semantics is as expected. A slightly more involved reduction without these temporal connectives is possible.

Let α be a closed QBL formula over propositions $p_1, \ldots, p_n$. We define the set P of predicate symbols as $\{P_1, \ldots, P_n\}$, where each predicate symbol has arity 1. Moreover, let $R := \{r\}$ and $D := \{0, 1\}$, and let w be the observation $(\{0\}, \sigma, \varrho_0)(\{1\}, \sigma, \varrho_1)(\{3\}, [\,], [\,])((3, \infty), [\,]\,[\,])$, with $\sigma(P_i) = \{1\}$, for each $i \in \{1, \ldots, n\}$, and $\varrho_i(r) = i$, for $i \in \{0, 1\}$. Finally, we translate the QBL formula α to an MTL$^{\downarrow}$ formula α^* as follows.

$$p_i^* := P_i(x_i) \qquad (\neg\beta)^* := \neg\beta^* \qquad (\beta \vee \gamma)^* := \beta^* \vee \gamma^*$$
$$(\exists p_i.\, \beta)^* := \blacklozenge \Diamond_{[0,1]} {\downarrow} x_i.\, \beta^* \qquad (\forall p_i.\, \beta)^* := \blacksquare \Box_{[0,1]} {\downarrow} x_i.\, \beta^*$$

It is easy to see that α is satisfiable iff $[w, 0, [\,]) \approx \alpha^*] = \mathsf{t}$.

Listing 2.

```
procedure Init(φ)
  verdicts := ∅
  J := [0, ∞)
  foreach ψ ∈ Sub(φ) # in a bottom−up manner
    case ψ = p(x̄): Ψ^{ψ,J,[]} := ψ^J
    case ψ = ¬α: Ψ^{ψ,J,[]} := ¬α^J
    case ψ = α ∨ β: Ψ^{ψ,J,[]} := α^J ∨ β^J
    case ψ = ↓x. α: Ψ^{ψ,J,[]} := α^J
    case ψ = α U_I β:
      if I = [0, ∞) then tc := t else tc := tc^{J,J}
      Ψ^{ψ,J,[]} := tc ∧ β̄^J ∧ ᾱ^J
  foreach α in Atoms(φ) with α = t do
    PropagateUp(α, J, t)
```

We only sketch the problem's membership in PSPACE. Note that w is finite. If there is no time point in w with timestamp τ, then $[w, \tau, \nu \models\!\!\approx \varphi] = \bot$. Suppose that $i \in pos(w)$ is a time point in w with timestamp τ. A computation of φ's truth value at position i can be easily obtained from the inductive definition of the satisfaction relation $\models\!\!\approx$. This computation can be done in polynomial space when traversing the formula structure depth-first. Note that for a position, a subformula of φ might be visited multiple times with possibly different partial valuations.

B Additional Algorithmic Details

We present in this section the procedures Init, NewTimePoint, PropagateDown, and PropagateUp, which are called from the main procedure. Their pseudo-code is given in the Listings 2 to 5, respectively.

The procedure Init initializes the state of the monitor, which consists of the formulas $\Psi^{\gamma,[0,\infty),[\,]}$, for each subformula γ of the monitored formula φ. These formulas are as defined in Section 4.2.1. The procedure also propagates the Boolean value t from the atoms t of φ.

The procedure NewTimePoint transforms formulas $\Psi^{\gamma,K,\nu}$ so that they reflect the interval structure of the new observation, the one obtained after receiving the current message. Recall that, in the pseudo-code, J is the interval that is split at the current iteration and the sequence new consists of those intervals among $J \cap [0, \tau)$, $\{\tau\}$, and $J \cap (\tau, \infty)$ that are not complete. NewTimePoint creates a new formula $\Psi^{\gamma,K,\nu}$ for each $K \in$ new and each each ν such that the variable $\Psi^{\gamma,J,\nu}$ is defined. The new formula is obtained by applying a substitution which translates atomic propositions α^J into propositional formulas over atomic propositions $\alpha^{K'}$ with $K' \in$ new. For non-temporal formulas this propositional formula is simply α^K. That is the substitution is $[\alpha^J \mapsto \alpha^K]$. For temporal formulas the substitution is obtained in two steps: a first substitution replaces atomic propositions of the form $tc^{H,J}$ with atomic substitutions of the form $tc^{H,K}$, and a second substitution, obtained by calling the procedure RefinementUntil, deals with atomic propositions of the form α^J.

Listing 3.

```
procedure NewTimePoint(φ, J, new)
  foreach ψ ∈ Sub(φ) # in a top−down manner
    foreach ν with Ψ^{ψ,J,ν} ≠ nil do
      foreach K in new do
        case ψ = p(x̄):
          Ψ^{ψ,K,ν} := Apply(Ψ^{ψ,J,ν}, [ψ^J ↦ ψ^K])
        case ψ = ¬α:
          Ψ^{ψ,K,ν} := Apply(Ψ^{ψ,J,ν}, [α^J ↦ α^K])
        case ψ = α ∨ β:
          Ψ^{ψ,K,ν} := Apply(Ψ^{ψ,J,ν}, [α^J ↦ α^K, β^J ↦ β^K])
        case ψ = ↓x. α:
          Ψ^{ψ,K,ν} := Apply(Ψ^{ψ,J,ν}, [α^J ↦ α^K])
        case ψ = α U_I β:
          Ψ^{ψ,K,ν} := Apply(Ψ^{ψ,J,ν}, [tc^{H,J} ↦ tc^{H,K}]_{tc^{H,J} ∈ AP(Ψ^{ψ,J,ν})})
      foreach γ, H, μ with ψ^J ∈ AP(Ψ^{γ,H,μ}) and (γ = ψ U_I _ or γ = ψ U_I _)
        θ := RefinementUntil(γ, H, J, new)
        Ψ^{γ,H,μ} := Apply(Ψ^{γ,H,μ}, θ)

procedure RefinementUntil(α U_I β, H, J, new)
  anchor, continuation := f, t
  for K in new with K ≥ H do
    if Singleton(K) then cont := t else β̄^K
    anchor := anchor ∨ β̄^K ∧ tc^{K,H} ∧ cont ∧ continuation
    continuation := continuation ∧ ᾱ^K
  return [tc^{J,K} ↦ t, β̄^J ↦ anchor, ᾱ^J ↦ continuation, ᾱ̄^J ↦ t]
```

NewTimePoint also transforms some formulas $\Psi^{\gamma,K,\nu}$ with $K \notin \mathsf{new}$, that is for intervals K that occur in the letters of the old observation, that is, the one from the previous iteration. It does that for those formulas that have atomic propositions of the form α^J. Note that then it is necessarily the case that γ is a temporal formula. The required substitution is also computed by the RefinementUntil procedure.

The RefinementUntil procedure produces the necessary substitution to update the atomic propositions $\bar{\alpha}^J$, $\bar{\bar{\alpha}}^J$, and $\bar{\beta}^J$ that may occur in formulas $\Psi^{\gamma,H,\nu}$ for some interval H of the new observation, where $\gamma = \alpha \,\mathsf{U}_I\, \beta$. (Note that before calling RefinementUntil the new formulas $\Psi^{\gamma,K,\nu}$ with $K \in \mathsf{new}$ have already been created.) The substitution of the proposition $\bar{\alpha}^J$ is straightforward. Namely, we replace $\bar{\alpha}^J$ by the conjunction $\bigwedge_{K \in \mathsf{new}, K \geq H} \bar{\alpha}^K$. Next, the atomic propositions $tc^{J,K}$ and $\bar{\bar{\alpha}}^J$ are discarded: they appeared in $\Psi^{\gamma,H,\nu}$ as conjuncts and thus replacing them by t effectively discards them. The substitution of $\bar{\beta}^J$ is more involved. We illustrate it with an example. Suppose that $\mathsf{new} = (K_1, K_2, K_3)$ and that $H \leq K_1$. Note that K_2 is a singleton. In this case, $\bar{\beta}^J$ is substituted by the following formula.

$$\begin{aligned}
&\bar{\beta}^{K_1} \wedge tc^{K_1,H} \wedge \bar{\bar{\alpha}}^{K_1} \vee \\
&\bar{\beta}^{K_2} \wedge tc^{K_2,H} \wedge \bar{\alpha}^{K_1} \vee \\
&\bar{\beta}^{K_3} \wedge tc^{K_3,H} \wedge \bar{\bar{\alpha}}^{K_3} \wedge \bar{\alpha}^{K_2} \wedge \bar{\alpha}^{K_1}
\end{aligned}$$

The application of the computed substitution is performed by the procedure Apply given in Listing 5. Apply actually does more than just applying the

Listing 4.

```
procedure PropagateDown(ψ, J, x, d)
  foreach ν with Ψ^{ψ,J,ν} ≠ nil
    Ψ^{ψ,J,ν[x↦d]} := Ψ^{ψ,J,ν}
    if ψ ∉ Atoms(φ) then
      foreach α^K ∈ AP(Ψ^{ψ,J,ν})
        PropagateDown(α, K, x, d)
```

Listing 5.

```
procedure Apply(Ψ^{ψ,J,ν}, θ)
  f := Substitute(Ψ^{ψ,J,ν}, θ)
  if f ∈ 2 then PropagateUp(ψ, J, ν, f)
  return f

procedure PropagateUp(ψ, J, b)
  γ := Parent(ψ)
  if γ = nil then
    if Singleton(J) then verdicts := verdicts ∪ {(Timestamp(J), b)}
  else if CanPropagateUp(ψ, J, b)
    θ := [ψ^J ↦ b]
    foreach K, μ with ψ^J ∈ AP(Ψ^{γ,K,μ})
      Ψ^{γ,K,μ} := Apply(Ψ^{γ,K,μ}, θ)

procedure CanPropagateUp(ψ, J, b)
  γ := Parent(ψ)
  case γ = nil or γ ≠ _ U_I _: return t
  case γ = _ U_I ψ: return (Singleton(J) or not b)
  case γ = ψ U_I _: return (Singleton(J) or b)
```

substitution given as an argument. First, it also simplifies the resulting formula. The actual application and the simplification are performed by procedure Substitute, which is as expected and thus not detailed further. Second, Apply checks whether the resulting formula is a Boolean constant. If this is the case, then propagation is initiated by calling the PropagateUp procedure.

Note that if $\Psi^{\gamma,J,\nu}$ is already a Boolean constant that has not yet been propagated, because for instance the $\gamma = \bar{\beta}$ and the constant is t, then, since Apply tries again to propagate it, this Boolean value will actually be propagated for the new interval $\{\tau\}$, as it is a singleton and thus corresponds to a time point.

The pseudo-code of the procedures PropagateDown and PropagateUp is straightforward in that it implements downward and upward propagation exactly as described in Section 4.2.2.

C Soundness and Completeness Proof

In this section, we will consider many substitutions from atomic propositions to propositional formulas. Given such a substitution θ and a propositional formula ψ, we denote by $\theta(\psi)$ and $\psi\theta$ the formula obtained by replacing in ψ the atomic propositions p that occur in both ψ and in $\mathrm{def}(\theta_i)$ by $\theta_i(p)$.

C.1 Overview

Let $\bar{w}$ be valid observation sequence and let φ be the monitored formula. We assume that φ is not an atomic formula. We let $(J^i_j, \sigma^i_j, \varrho^i_j)$ be the jth letter of w_i. We drop the superscript i if it is clear from the context. Also, given an interval J in w_i, we denote by $\hat{J}$ the index $j \in pos(w_i)$ such that $J_j = J$, assuming that the iteration i is clear from the context.

We first state a lemma from which correctness follows, and only later prove the lemma. We recall that $\Psi_i^{\psi,J,\nu}$ denotes the the value of the variable $\Psi^{\psi,J,\nu}$ from the pseudo-code at the end of iteration i.

Lemma 21. *For any $i \in \mathbb{N}$, $j \in pos(w_i)$, and $b \in 2$, we have that*

$$\theta_i^{[]}(\Phi_i^{\varphi,J}) = b \quad \textit{iff} \quad \Psi_i^{\varphi,J,[]} = b,$$

where J is the interval of the jth letter of w_i.

We show next how observational soundness and completeness follow from this lemma. We first note that for any iteration $i \in \mathbb{N}$, the variable $\Psi^{\psi,J,\nu}$ is defined at the end of iteration i for some[8] ν if and only if ψ is a subformula of φ and J is an interval in w_i. Next, we note that the global variable verdicts is only updated in the execution of PropagateUp(ψ, J, b) when $\psi = \varphi$ and J is a singleton. Furthermore, by analyzing where it is called from, we see that all calls are preceded by setting $\Psi^{\psi,J,\nu}$ to b. Moreover, ν has to be $[\,]$ because for any defined variable $\Psi^{\psi,J,\nu}$ with $\nu \neq [\,]$ we have that $\psi \neq \varphi$. Indeed, $\Psi^{\psi,J,\nu}$ is only used with a "new" ν in PropagateDown and this procedure is never called for $\psi = \varphi$. We thus obtain that if some tuple (τ, b) is added to verdicts then $\Psi^{\varphi,\{\tau\},[]} = b$.

Observational soundness follows directly from the lemma, the above stated properties of θ_i^{ν} and Φ_i, and the observations made in the previous paragraph. Note that, since φ is closed, then $[\![w_i, j, \nu \models\!\!\approx \varphi]\!] = [\![w_i, j, [\,] \models\!\!\approx \varphi]\!]$, for any partial valuation ν.

Consider now completeness. Say that $[\![w_i, j, \nu \models\!\!\approx \varphi]\!] = b \in 2$ and j is a time point with timestamp τ. Let $J = \{\tau\}$. Then $[\![w_i, j, [\,] \models\!\!\approx \varphi]\!] = b$ and thus $\theta_i^{[]}(\Phi_i^{\varphi,J}) = b$. By the lemma, we have that $\Psi_i^{\varphi,J,[]} = b$. Then there is an iteration $i' \leq i$ when $\Psi_{i'}^{\varphi,J',[]}$ has been set to b, where J' is the interval in $w_{i'}$ from which J originates. Let i'' be the first iteration when J is a letter of w_k for some k. At this iteration $\Psi_{i''}^{\varphi,J,[]}$ is set to $\Psi_{i'}^{\varphi,J',[]}$ (see the NewTimePoint procedure). Clearly $i' \leq i'' \leq i$. The setting of $\Psi_{i''}^{\psi,J,[]}$ to a new value is preceded by a call to Apply, and since this new value is a Boolean value, Apply calls PropagateUp which adds (τ, b) to verdicts.

C.2 The Key Lemma

We state next the key lemma of the proof, from which Lemma 21 follows. In order to state this more general lemma, we first introduce some additional notation.

[8] We will later, in Lemma 22, also characterize for which ν is $\Psi_i^{\psi,J,\nu}$ defined.

C.2.1 Additional Notation. Given an $i \in \mathbb{N}$, a subformula ψ of φ, and a $j \in pos(w_i)$, we define inductively the set $Val_i(\psi, j)$ of *relevant valuations* for ψ at iteration i and position j. For $i = 0$, we set $Val_i(\psi, 0) := \{[]\}$, for any subformula ψ of φ, and for $i > 0$ we let

$$Val'_i(\psi, j) := \begin{cases} \{[]\} & \text{if } \psi = \varphi, \\ Val_i(\gamma, j) & \text{if } \gamma = \neg\psi,\ \gamma = \psi \vee \psi', \text{ or } \gamma = \psi' \vee \psi, \\ \{\nu[x \mapsto \varrho^i_j(r)] \mid \nu \in Val_i(\gamma, j)\} & \text{if } \gamma = \downarrow^x r.\ \psi \\ \cup_{h \in A} Val_i(\gamma, h) & \text{if } \gamma = \psi\ \mathsf{U}_I\ \psi' \\ \cup_{k \in B} Val_i(\gamma, k) & \text{if } \gamma = \psi'\ \mathsf{U}_I\ \psi \end{cases}$$

and

$$Val_i(\psi, j) := \{\nu \in Val'_i(\psi, j) \mid [\![w_i, j, \nu \models\!\!\approx \psi]\!] \notin \mathsf{2}\},$$

where γ is the "parent" of ψ (i.e. the subformula of φ that has ψ as a direct subformula) and

$$A := \{h \in \mathbb{N} \mid k < h \leq j \text{ where } k \in \mathbb{N} \text{ is such that } k \leq j \text{ and } \mathrm{tc}_{w_i,I}(j,k) \neq \mathsf{f}\},$$
$$B := \{k \in \mathbb{N} \mid k \leq j \text{ and } \mathrm{tc}_{w_i,I}(j,k) \neq \mathsf{f}\}.$$

We denote by AP_i the set of the atomic propositions used by the formulas $\Phi_i^{\gamma,J}$. We let $\overline{AP}_i$ be the set of atomic propositions obtained from AP_i be removing the atomic propositions of the form tp^J and replacing atomic propositions of the form β^J and α^J where $\alpha\ \mathsf{U}_I\ \beta$ is an subformula of φ for some I, by the atomic propositions $\bar{\beta}^J$ and respectively $\bar{\alpha}^J$ and $\bar{\bar{\alpha}}^J$. Note that $\overline{AP}_i$ represents the set of atomic propositions of the formulas $\Psi_i^{\gamma,J,\nu}$ from the pseudo-code.

We also let $\hat{\delta}_i^\nu$ be the following substitution:

$$\hat{\delta}_i^\nu(\bar{\beta}^J) := \begin{cases} tp^J \wedge \beta^J & \text{if } J \text{ not a singleton and } \Psi_i^{\beta,J,\nu} \notin \mathsf{2} \\ tp^J & \text{if } J \text{ not a singleton and } \Psi_i^{\beta,J,\nu} = \mathsf{t} \\ \beta^J & \text{if } J \text{ is a singleton and } \Psi_i^{\beta,J,\nu} \notin \mathsf{2} \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$\hat{\delta}_i^\nu(\bar{\alpha}^J) := \begin{cases} tp^J \rightarrow \alpha^J & \text{if } J \text{ not a singleton and } \Psi_i^{\alpha,J,\nu} \notin \mathsf{2} \\ \neg tp^J & \text{if } J \text{ not a singleton and } \Psi_i^{\alpha,J,\nu} = \mathsf{f} \\ \alpha^J & \text{if } J \text{ is a singleton and } \Psi_i^{\alpha,J,\nu} \notin \mathsf{2} \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$\hat{\delta}_i^\nu(\bar{\bar{\alpha}}^J) := \begin{cases} tp^J \vee \alpha^J & \text{if } J \text{ not a singleton and } \Psi_i^{\alpha,J,\nu} \notin \mathsf{2} \\ tp^J & \text{if } J \text{ not a singleton and } \Psi_i^{\alpha,J,\nu} = \mathsf{f} \\ \alpha^J & \text{if } J \text{ is a singleton and } \Psi_i^{\alpha,J,\nu} \notin \mathsf{2} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Note that, for instance when J is a singleton and $\Psi_i^{\alpha,J,\nu} = \mathsf{t}$, there is no need to define $\hat{\delta}_i^\nu(\bar{\alpha}^J)$, because in this case $\bar{\alpha}^J$ is not an atomic proposition of $\Psi_i^{\alpha \mathsf{U}_I \beta,K,\nu}$: that atomic proposition has already been instantiated.

C.2.2 The Lemma and its Proof. The following key lemma states the main invariant of the monitoring algorithm.

Lemma 22. *For any non-atomic subformula ψ of φ, $i \in \mathbb{N}$, $j \in pos(w_i)$, partial valuation $\nu \in Val_i(\psi, j)$, we have that*

$$\theta_i^{\mu}(\Phi_i^{\psi,J}) \equiv \hat{\delta}_i^{\mu}(\Psi_i^{\psi,J,\nu}),$$

where J is the interval of the jth letter of w_i and $\mu = \nu[x \mapsto \varrho_j^i(r)]$ if $\psi = \downarrow^r x.\, \alpha$ and $\mu = \nu$ otherwise.

We first note that Lemma 21 follows easily from Lemma 22. Indeed, we just need to show that, $\hat{\delta}_i^{\mu}(\Psi_i^{\varphi,J,[]}) = b$ iff $\Psi_i^{\varphi,J,[]} = b$, for any $b \in \mathsf{2}$ where μ is an in the lemma statement. The right to left direction is trivial, while the left to right direction also follows easily: if $\Psi_i^{\varphi,J,[]} \notin \mathsf{2}$ then it contains at least an atomic proposition, and then $\hat{\delta}_i^{[]}(\Psi_i^{\varphi,J,[]})$ also contains an atomic proposition, by the definition of $\hat{\delta}_i^{\mu}$, and thus $\hat{\delta}_i^{\mu}(\Psi_i^{\varphi,J,[]}) \notin \mathsf{2}$ too.

We devote the rest of this section to the proof of Lemma 22. We start with an assumption and continue with a series of definitions.

We assume in the pseudo-code that formula rewriting (that is, applying substitutions) and propagations are separated into two distinct phases. Note that this can be easily achieved by postponing propagations, that is, by storing all tuples (φ, J, f) with $f \in \mathsf{2}$ that occur during the execution of Apply, and by performing the corresponding propagations at the end of the NewTimePoint procedure. We choose not to present this two-phase version of the pseudo-code in order to keep the pseudo-code more compact.

Let new_i be the value of new at iteration $i > 0$. Given an interval $J \in \mathsf{new}_i$, we let ζ_i^J be the substitution computed by the pseudo-code at some iteration $i > 0$ during the formula rewriting phase. Note that there is no formula rewriting phase at iteration $i = 0$. For $i > 0$, for instance when α is such that its parent is $\psi = \neg\alpha$, we have that $\zeta_i^J(\alpha^{J'}) = \alpha^J$ when J is a new interval in w_i obtained by splitting the interval J' in w_{i-1}.[9]

We let $\bar{\xi}_i^{\nu}$ be the substitution computed by the pseudo-code at iteration i during the propagation phase for the partial valuation ν. Formally, $\bar{\xi}_i^{\nu}$ is a partial valuation over $\overline{AP}_i$ defined as follows:

$$\begin{aligned}
\xi_i^{\nu}(p(\bar{x})^J) &:= \mathsf{t} && \text{if } \bar{x} \in \mathrm{def}(\varphi),\ p \in \mathrm{def}(\sigma), \text{ and } \nu(\bar{x}) \in \sigma(p),\\
\xi_i^{\nu}(p(\bar{x})^J) &:= \mathsf{f} && \text{if } \bar{x} \in \mathrm{def}(\varphi),\ p \in \mathrm{def}(\sigma), \text{ and } \nu(\bar{x}) \notin \sigma(p),\\
\bar{\xi}_i^{\nu}(\gamma^J) &:= b && \text{if } \gamma \text{ is non-atomic and } \Psi_i^{\gamma,J,\nu} = b \in \mathsf{2},\\
\bar{\xi}_i^{\nu}(\bar{\beta}^J) &:= \begin{cases} \mathsf{t} & \text{if } J \text{ is a singleton and } \Psi_i^{\beta,J,\nu} = \mathsf{t},\\ \mathsf{f} & \text{if } \Psi_i^{\beta,J,\nu} = \mathsf{f}, \end{cases}\\
\bar{\xi}_i^{\nu}(\bar{\alpha}^J) := \bar{\xi}_i^{\nu}(\bar{\bar{\alpha}}^J) &:= \begin{cases} \mathsf{t} & \text{if } \Psi_i^{\alpha,J,\nu} = \mathsf{t},\\ \mathsf{f} & \text{if } J \text{ is a singleton and } \Psi_i^{\alpha,J,\nu} = \mathsf{f}, \end{cases}
\end{aligned}$$

[9] Note that in the case of temporal subformulas, this is the composition of two substitutions: the unnamed one and the substitution θ computed by RefinementUntil.

$$\bar{\xi}_i^{\nu}(tc_{\alpha \mathsf{U}_I \beta}^{K,J}) := \begin{cases} \mathsf{t} & \text{if } (K-J) \subseteq I, \\ \mathsf{f} & \text{if } (K-J) \cap I = \emptyset. \end{cases}$$

For any $i > 0$, subformula ψ of φ, and interval J in w_i, we have

$$\Psi_i^{\psi,J,\nu} := \bar{\xi}_i^{\mu}(\zeta_i^J(\Psi_{i-1}^{\psi,J',\nu})), \tag{1}$$

where J' is the interval in w_{i-1} from which J originates and $\mu = \nu[x \mapsto \varrho_j^i(r)]$ if $\psi = \downarrow^r x.\, \alpha$ and $\mu = \nu$ otherwise.

We define next the substitution ξ_i^{ν} over AP_i and which is a variant of $\bar{\xi}_i^{\nu}$:

$$\begin{aligned} \xi_i^{\nu}(\gamma^J) &:= \bar{\xi}_i^{\nu}(\gamma^J) \quad \text{if } \gamma \in Sub(\varphi), \\ \xi_i^{\nu}(tp^J) &:= \mathsf{t} \qquad\quad \text{if } J \text{ is a singleton}, \\ \xi_i^{\nu}(tc_{\psi}^{K,J}) &:= \bar{\xi}_i^{\nu}(tc_{\psi}^{K,J}). \end{aligned}$$

We also let $\bar{\delta}_i$ be the following substitution:

$$\begin{aligned} \bar{\delta}_i(\bar{\beta}^J) &:= tp^J \wedge \beta^J, \\ \bar{\delta}_i(\bar{\alpha}^J) &:= tp^J \rightarrow \alpha^J, \\ \bar{\delta}_i(\bar{\bar{\alpha}}^J) &:= tp^J \vee \alpha^J. \end{aligned}$$

Note that we do not have that $\hat{\delta}_i^{\nu} = \xi_i^{\nu} \circ \bar{\delta}_i$. Indeed, if $\Psi_i^{\beta,J,\nu} = \mathsf{f}$ then $\bar{\beta}^J\, \hat{\delta}_i^{\nu} = \bar{\beta}^J$, while $\bar{\beta}^J\, \bar{\delta}_i\, \xi_i^{\nu} = \mathsf{f}$. However, as we will prove later, we have that $\hat{\delta}_i^{\nu} \circ \bar{\xi}_i^{\nu} = \xi_i^{\nu} \circ \bar{\delta}_i$.

For $i > 0$ and partial valuation ν, we define the substitution $\theta_{i-1,i}^{\nu}$ such that it behaves as θ_{i-1}^{ν}, but it acts however on the atomic propositions of formulas $\Phi_i^{\gamma,J,\nu}$. In other words, in contrast to θ_i^{ν}, it does not take into account the new interpretations received at iteration i. Formally, $\theta_{i-1,i}^{\nu}$ is a partial function over AP_i defined by

$$\begin{aligned} \theta_{i-1,i}^{\nu}(\alpha^J) &:= \theta_{i-1}^{\nu}(\alpha^{J'}) \\ \theta_{i-1,i}^{\nu}(tp^J) &:= \theta_{i-1}^{\nu}(tp^{J'}) \\ \theta_{i-1,i}^{\nu}(tc^{H,K}) &:= \theta_{i-1}^{\nu}(tc^{H',K'}), \end{aligned}$$

where J', H', and K' are the intervals in w_{i-1} from which the intervals J, H, and respectively K originate.

Finally, we let $\theta_{i,\Delta}$ be the substitution defined over AP_i by

$$\begin{aligned} \theta_{i,\Delta}^{\nu}(\alpha^J) &:= \theta_i^{\nu}(\alpha^J) && \text{if } i = 0 \text{ or } \theta_{i-1}^{\nu} \text{ is undefined on } \alpha^{J'}, \\ \theta_{i,\Delta}^{\nu}(tp^J) &:= \theta_i^{\nu}(tp^J) && \text{if } i = 0 \text{ or } \theta_{i-1}^{\nu} \text{ is undefined on } tp^{J'}, \\ \theta_{i,\Delta}^{\nu}(tc^{H,K}) &:= \theta_i^{\nu}(tc^{H,K}) && \text{if } i = 0 \text{ or } \theta_{i-1}^{\nu} \text{ is undefined on } tc^{H',K'}, \end{aligned}$$

where, when $i > 0$, J', H', and K' are the intervals in w_{i-1} from which the intervals J, H, and respectively K originate.

Clearly, we have that

$$\theta_i^{\nu} = \theta_{i-1,i}^{\nu}\, \theta_{i,\Delta}^{\nu}, \quad \text{for } i > 0, \tag{2}$$

Table 2. Summary of substitutions used in the proof.

notation	domain	intuition
θ_i^ν	AP_i	$\gamma^J \mapsto [\![w_i, \hat{J}, \nu \models\!\!\approx \gamma]\!]$ if in 2
$\theta_{i-1,i}^\nu$	AP_i	"extension" of θ_{i-1}^ν from A_{i-1} to AP_i
$\theta_{i,\Delta}^\nu$	AP_i	a substitution such that $\theta_i^\nu = \theta_{i-1,i}^\nu \, \theta_{i,\Delta}^\nu$
$\bar{\zeta}_i^J$	$\overline{AP}_{i-1}$	the substitution of the rewriting phase
ζ_i^J	AP_{i-1}	variant of $\bar{\zeta}_i^J$ over AP_{i-1} such that $\bar{\zeta}_i^J \, \bar{\delta}_i \equiv \bar{\delta}_{i-1} \, \zeta_i^J$
$\bar{\xi}_i^\nu$	$\overline{AP}_i$	the substitution of the propagation phase
ξ_i^ν	AP_i	$\gamma^J \mapsto \Psi_i^{\gamma,J,\nu}$ if in 2
$\bar{\delta}_i$	$\overline{AP}_i$	e.g. $\bar{\beta}^J \mapsto tp^J \wedge \beta^J$
$\hat{\delta}_i^\nu$	$\overline{AP}_i$	variant of $\bar{\delta}_i$, taking propagations into account

$$\theta_0^\nu = \theta_{0,\Delta}^\nu. \tag{3}$$

The following easy to prove equivalence will also be useful:

$$\Phi_i^{\psi,J} \theta_i^\mu = b \quad \text{iff} \quad \psi^J \theta_i^\nu = b, \tag{4}$$

for any formula ψ that is a subformula of φ and any $b \in 2$, and where $\mu = \nu[x \mapsto \varrho_j^i(r)]$ if $\psi = \downarrow^r x.\, \alpha$ and $\mu = \nu$ otherwise.

We summarize in Table 2 all substitutions used in the proof together with their informal meaning.

In the remaining proof, we will use equivalences between substitutions, with the following meaning. Given two substitutions θ and θ' from atomic propositions to propositional formulas, we write that $\theta \equiv \theta'$ iff $\mathrm{def}(\theta) = \mathrm{def}(\theta')$ and $p\theta \equiv p\theta'$, for any $p \in \mathrm{def}(\theta)$.

Next, we prove Lemma 22 by nested induction, the outer induction being on the iteration i and the inner induction being a structural induction on ψ.

Outer base case: $i = 0$. We have a single letter in w_0 with the interval $J = [0, \infty)$. Let $\bar{\Psi}_0^{\psi,J,[]}$ be the value of $\Psi^{\psi,J,[]}$ at the point of execution of the Init procedure (thus at iteration 0) between the two **foreach** loops, that is, before propagation of the t atoms. We thus have that

$$\Psi_0^{\gamma,J,[]} = \bar{\Psi}_0^{\gamma,J,[]} \, \bar{\xi}_0^{[]}. \tag{5}$$

We also have that

$$\bar{\Psi}_0^{\gamma,J,[]} \bar{\delta}_0 \equiv \Phi_0^{\gamma,J}. \tag{6}$$

This is easy to check by inspecting the Init procedure. For instance, for formulas $\gamma = \alpha \, \mathsf{U}_I \, \beta$, we have that $\bar{\Psi}_0^{\gamma,J} = tc^{J,J} \wedge \bar{\beta}^J \wedge \bar{\alpha}^J$ and thus

$$\bar{\Psi}_0^{\gamma,J} \bar{\delta}_0 = tc^{J,J} \wedge (tp^J \wedge \beta^J) \wedge (tp^J \vee \alpha^J) \equiv tp^J \wedge tc^{J,J} \wedge \beta^J = \Phi_0^{\gamma,J}.$$

This case then follows from the following sequence of equivalences:

$$\begin{aligned}
\Psi_0^{\gamma,J,[]}\,\hat{\delta}_0^{[]} \equiv\ & \text{[by (5): } \Psi_0^{\gamma,J,[]} = \bar{\Psi}_0^{\gamma,J,[]}\,\bar{\xi}_0\text{]}\\
\bar{\Psi}_0^{\gamma,J}\,\bar{\xi}_0^{[]}\,\hat{\delta}_0 \equiv\ & \text{[by (8): } \bar{\xi}_i^{\nu}\,\hat{\delta}_i^{\nu} \equiv \bar{\delta}_i\,\xi_i^{\nu}\text{]}\\
\bar{\Psi}_0^{\gamma,J,[]}\,\bar{\delta}_0\,\xi_0^{[]} \equiv\ & \text{[by (6): } \bar{\Psi}_0^{\gamma,J,[]}\,\bar{\delta}_0 \equiv \Phi_0^{\gamma,J}\text{]}\\
\Phi_0^{\gamma,J}\,\xi_0^{[]} \equiv\ & \text{[by (3) and (IH'): } \xi_0^{[]} \equiv \theta_0^{[]}\text{]}\\
\Phi_i^{\gamma,J}\,\theta_0^{[]}
\end{aligned}$$

We postpone the proof of the not yet justified equivalences (namely the 2nd and 4th), as similar ones are also used in the inductive case, and are proved together.

Outer inductive case: $i > 0$. We assume that the equivalence from the lemma statement holds for $i-1$:

$$\Psi_{i-1}^{\gamma,J',\nu}\,\hat{\delta}_{i-1}^{\nu} \equiv \Phi_{i-1}^{\gamma,J'}\,\theta_{i-1}^{\nu}, \tag{IH}$$

where J' is the intervals in w_{i-1} from which J originates. The inductive case follows from the following sequence of equivalences:

$$\begin{aligned}
\Psi_i^{\gamma,J,\nu}\,\hat{\delta}_i^{\mu} \equiv\ & \text{[by (1): } \Psi_i^{\gamma,J,\nu} = \Psi_{i-1}^{\gamma,J',\nu}\,\bar{\zeta}_i^{J}\,\bar{\xi}_i^{\mu}\text{]}\\
\Psi_{i-1}^{\gamma,J',\nu}\,\bar{\zeta}_i^{J}\,\bar{\xi}_i^{\mu}\,\hat{\delta}_i^{\mu} \equiv\ & \text{[by (8): } \bar{\xi}_i^{\mu}\,\hat{\delta}_i^{\mu} \equiv \bar{\delta}_i\,\xi_i^{\mu}\text{]}\\
\Psi_{i-1}^{\gamma,J',\nu}\,\bar{\zeta}_i^{J}\,\bar{\delta}_i\,\xi_i^{\mu} \equiv\ & \text{[by (7): } \xi_i^{\mu} \equiv \xi_i^{\mu}\,\xi_i^{\mu}\text{]}\\
\Psi_{i-1}^{\gamma,J',\nu}\,\bar{\zeta}_i^{J}\,\bar{\delta}_i\,\xi_i^{\mu}\,\xi_i^{\mu} \equiv\ & \text{[by (12): } \bar{\zeta}_i^{J}\,\bar{\delta}_i \equiv \bar{\delta}_{i-1}\,\zeta_i^{J}\text{]}\\
\Psi_{i-1}^{\gamma,J',\nu}\,\bar{\delta}_{i-1}\,\zeta_i^{J}\,\xi_i^{\mu}\,\xi_i^{\mu} \equiv\ & \text{[by (10): } \zeta_i^{J}\,\xi_i^{\mu} \equiv \xi_{i-1}\,\zeta_i^{J}\text{]}\\
\Psi_{i-1}^{\gamma,J',\nu}\,\bar{\delta}_{i-1}\,\xi_{i-1}\,\zeta_i^{J}\,\xi_i^{\mu}\,\xi_i^{\mu} \equiv\ & \text{[by (9): } \Psi_{i-1}^{\gamma,J',\nu}\,\bar{\delta}_{i-1}\,\xi_{i-1} \equiv \Psi_{i-1}^{\gamma,J',\nu}\,\hat{\delta}_{i-1}\text{]}\\
\Psi_{i-1}^{\gamma,J',\nu}\,\hat{\delta}_{i-1}\,\zeta_i^{J}\,\xi_i^{\mu} \equiv\ & \text{[by (IH): } \Psi_{i-1}^{\gamma,J',\nu}\,\hat{\delta}_{i-1} \equiv \Phi_{i-1}^{\gamma,J'}\,\theta_{i-1}^{\mu}\text{]}\\
\Phi_{i-1}^{\gamma,J'}\,\theta_{i-1}^{\mu}\,\zeta_i^{J}\,\xi_i^{\mu} \equiv\ & \text{[by (IH'): } x\,\xi_i^{\mu} \equiv x\,\theta_{i,\Delta}^{\mu}\text{, for any } q \in AP(\Phi_{i-1}^{\gamma,J'}\,\theta_{i-1}^{\mu}\,\zeta_i^{J})\text{]}\\
\Phi_{i-1}^{\gamma,J'}\,\theta_{i-1}^{\mu}\,\zeta_i^{J}\,\theta_{i,\Delta}^{\mu} \equiv\ & \text{[by (11): } \theta_{i-1}^{\mu}\,\zeta_i^{J} \equiv \zeta_i^{J}\,\theta_{i-1,i}^{\mu}\text{]}\\
\Phi_{i-1}^{\gamma,J'}\,\zeta_i^{J}\,\theta_{i-1,i}^{\mu}\,\theta_{i,\Delta}^{\mu} \equiv\ & \text{[by (13): } \Phi_{i-1}^{\gamma,J'}\,\zeta_i^{J} \equiv \Phi_i^{\gamma,J}\text{]}\\
\Phi_i^{\gamma,J}\,\theta_{i-1,i}^{\mu}\,\theta_{i,\Delta}^{\mu} \equiv\ & \text{[by (2): } \theta_{i-1,i}^{\mu}\,\theta_{i,\Delta}^{\mu} \equiv \theta_i^{\mu}\text{]}\\
\Phi_i^{\gamma,J}\,\theta_i^{\mu}
\end{aligned}$$

where $\mu = \nu[x \mapsto \varrho_j^i(r)]$ if $\psi = \downarrow^r x.\alpha$ and $\mu = \nu$ otherwise, and ζ_i^J is a substitution depending on $\bar{\zeta}_i^J$.

We now complete the proof by proving the not yet justified equivalences occurring in the above two sequences of equivalences. We start with the following statement. For any $q \in AP_i$ such that $q = tp^K$ or $q = tc^{H,K}$, or $q = \psi^K$ with ψ a direct subformula of γ, the following holds:

$$q\,\xi_i^{\mu} \equiv q\,\theta_{i,\Delta}^{\mu}. \tag{IH'}$$

The case when q is one of the atomic propositions $p(\bar{x})^J$ with $p(\bar{x})$, tp^J, and $tc^{H,K}$, follows directly from the definitions of ξ_i^{μ} and $\theta_{i,\Delta}^{\mu}$. So let $q = \psi^K$. We have that $\psi^K\,\xi_i^{\mu}$ equals $\Psi_i^{\psi,K,\mu}$ if $\Psi_i^{\psi,K,\mu} \in 2$ and equals ψ^K otherwise.

Suppose first that $\Psi_i^{\psi,K,\mu} = b$, for some $b \in 2$. Then, by the inner induction hypothesis, we have that $\Phi_i^{\psi,K}\theta_i^{\eta} \equiv \Psi_i^{\psi,K,\mu}\hat{\delta}_i^{\eta} = b$, where $\eta = \mu[x \mapsto \varrho_j^i(r)]$ if $\psi = \downarrow^r x.\,\alpha$ and $\eta = \nu$ otherwise. From (4), we know that $\bar{\Phi}_i^{\psi,K}\theta_i^{\eta} \equiv \psi^K\theta_i^{\mu}$, and thus we obtain that $\psi^K\theta_i^{\mu} = b$.

Suppose now that $\Psi_i^{\psi,K,\mu} \notin 2$. Then, reasoning similarly to the previous case, we obtain that $\psi^K\theta_i^{\mu} \notin 2$. Thus $\psi^K\theta_i^{\mu} = \psi^K$ and hence also $\psi^K\theta_{i,\Delta}^{\mu} = \psi^K$.

C.2.3 Remaining Details. The following two equivalences follow directly from the definitions of the four involved substitutions.

$$\xi_i^{\nu} \equiv \xi_i^{\nu}\,\xi_i^{\nu} \tag{7}$$

$$\bar{\xi}_i^{\nu}\,\hat{\delta}_i^{\nu} \equiv \bar{\delta}_i\,\xi_i^{\nu} \tag{8}$$

Furthermore, for any $i \in \mathbb{N}$, J interval in w_i, partial evaluations ν and subformula γ of φ, the following equivalence holds:

$$\Psi_i^{\gamma,J,\nu}\,\hat{\delta}_i^{\mu} \equiv \Psi_i^{\gamma,J,\nu}\bar{\delta}_i\,\xi_i^{\mu}. \tag{9}$$

Indeed, let $\bar{\Psi}_i^{\gamma,J,\nu} := \Psi_{i-1}^{\gamma,J',\nu}\bar{\zeta}^J$, for $i > 0$. From (1) and (5), we get that, for any $i \geq 0$, we need to prove that $\bar{\Psi}_i^{\gamma,J,\nu}\,\bar{\xi}_i^{\nu}\,\hat{\delta}_i^{\nu} \equiv \bar{\Psi}_i^{\gamma,J,\nu}\,\bar{\xi}_i^{\nu}\,\bar{\delta}_i\,\xi_i^{\nu}$. This follows directly from equivalence (8) and the equivalence $\bar{\xi}_i^{\nu} \equiv \bar{\xi}_i^{\nu}\,\bar{\xi}_i^{\nu}$.

Lemma 23. *For any $i > 0$ and J in new_i, there is a substitution ζ_i^J such that, for any formula γ and partial valuation ν, the following equivalences hold:*

$$\zeta_i^J\,\xi_i^{\nu} \equiv \xi_{i-1}^{\nu}\,\zeta_i^J \tag{10}$$

$$\zeta_i^J\,\theta_{i-1,i}^{\nu} \equiv \theta_{i-1}^{\nu}\,\zeta_i^J \tag{11}$$

$$\bar{\zeta}_i^J\,\bar{\delta}_i \equiv \bar{\delta}_{i-1}\,\zeta_i^J \tag{12}$$

$$\Phi_i^{\gamma,J} \equiv \Phi_{i-1}^{\gamma,J'}\,\zeta_i^J, \tag{13}$$

where J' is the interval in w_{i-1} from which J originates.

Proof. Let ψ be a proper subformula of φ and γ its parent. For readability, in this proof we drop the index i from $\bar{\zeta}_i^J$ and ζ_i^J. If ψ is a direct subformula of a non-temporal subformula of φ, then $\bar{\zeta}^J(\psi^{J'}) = \psi^J$. In this case we let $\zeta^J(\psi^{J'}) := \psi^J$. It is easy to check that the four equivalences hold in this case.

We consider now the case when γ is a temporal formula, with $\gamma = \alpha\,\mathsf{U}_I\,\beta$. We first note that for atomic propositions $\bar{\beta}^{J'}$, $\bar{\alpha}^{J'}$, and $\bar{\bar{\alpha}}^{J'}$, the substitution $\bar{\zeta}_i^J$ depends on whether J is L, N, or R, where (L, N, R) are the intervals obtained by splitting J. Thus, we make a case distinction based on the value of J. We only consider one case, namely when $J = N$, the other ones being treated similarly. In this case we have the following equalities:

$$\bar{\zeta}^N(\bar{\beta}^{J'}) = (\bar{\beta}^N \wedge tc^{N,N}) \vee (\bar{\beta}^R \wedge tc^{R,N} \wedge \bar{\bar{\alpha}}^R \wedge \bar{\alpha}^N),$$

$$\begin{aligned}
\bar{\zeta}^N(\bar{\alpha}^{J'}) &= \bar{\alpha}^N \wedge \bar{\alpha}^R,\\
\bar{\zeta}^N(\bar{\bar{\alpha}}^{J'}) &= \mathsf{t},\\
\bar{\zeta}^N(tc^{J',J'}) &= \mathsf{t},\\
\bar{\zeta}^N(tc^{H,J'}) &= tc^{H,N}, \text{ for } H > J',
\end{aligned}$$

and the substitution ζ^N is defined as follows

$$\begin{aligned}
\zeta^N(\beta^{J'}) &:= (\beta^N \wedge tp^N \wedge tc^{N,N}) \vee\\
&\qquad (\beta^R \wedge tp^R \wedge tc^{R,N} \wedge (tp^R \vee \alpha^R) \wedge (tp^N \to \alpha^N)),\\
\zeta^N(\alpha^{J'}) &:= (tp^N \to \alpha^N) \wedge (tp^R \to \alpha^R),\\
\zeta^N(tp^{J'}) &:= \mathsf{t},\\
\zeta^N(tc^{J',J'}) &:= \mathsf{t},\\
\bar{\zeta}^N(tc^{H,J'}) &:= tc^{H,N}, \text{ for } H > J'.
\end{aligned}$$

By just using the definitions, it is easy to check that the equivalences (10), (11), and (12) hold. For instance, we check next that $\bar{\beta}^{J'}\,\bar{\zeta}^J\,\bar{\delta}_i \equiv \bar{\beta}^{J'}\,\bar{\delta}_{i-1}\,\zeta^J$:

$$\begin{aligned}
\bar{\delta}_i(\bar{\zeta}^N(\bar{\beta}^{J'})) &= \bar{\delta}_i\big((\bar{\beta}^N \wedge tc^{N,N}) \vee (\bar{\beta}^R \wedge tc^{R,N} \wedge \bar{\bar{\alpha}}^R \wedge \bar{\alpha}^N)\big)\\
&= ((tp^N \wedge \beta^N) \wedge tc^{N,N}) \vee\\
&\qquad ((tp^R \wedge \beta^R) \wedge tc^{R,N} \wedge (tp^R \vee \alpha^R) \wedge (tp^N \to \alpha^N))\\
&= \zeta^N(\beta^{J'}) = \zeta^N(tp^{J'} \wedge \beta^{J'}) = \zeta^N(\bar{\delta}_{i-1}(\bar{\beta}^{J'})).
\end{aligned}$$

For (13), we check next that $\Phi_i^{\gamma,N} \equiv \Phi_{i-1}^{\gamma,J'}\zeta^N$:

$$\begin{aligned}
\Phi_i^{\gamma,N} &\equiv \bigvee_{K \geq N} \big(tp^K \wedge tc^{K,N} \wedge \beta^K \wedge \bigwedge_{N \leq H < K} (tp^H \to \alpha^H)\big)\\
&\equiv \big(tp^N \wedge tc^{N,N} \wedge \beta^N\big) \vee\\
&\quad \big(tp^R \wedge tc^{R,N} \wedge \beta^R \wedge (tp^R \vee \alpha^R) \wedge (tp^N \to \alpha^N)\big) \vee\\
&\quad \bigvee_{K > R} \big(tp^K \wedge tc^{K,N} \wedge \beta^K \wedge\\
&\qquad\qquad \wedge (tp^N \to \alpha^N) \wedge (tp^R \to \alpha^R) \wedge \bigwedge_{R < H < K} (tp^H \to \alpha^H)\big)\\
&\equiv \zeta^N\Big(\big(tp^J \wedge tc^{J,J} \wedge \beta^J\big) \vee\\
&\qquad \bigvee_{K > J} \big(tp^K \wedge tc^{K,J} \wedge \beta^K \wedge (tp^J \to \alpha^J) \wedge \bigwedge_{J < H < K} (tp^H \to \alpha^H)\big)\Big)\\
&\equiv \zeta^N(\Phi_{i-1}^{\gamma,J}).
\end{aligned}$$

□

$$\text{(P1}') \quad \Box\, \mathit{transaction} \wedge \mathit{suspicious} \rightarrow \Diamond_{[0,3]}\, \mathit{report}$$

$$\text{(P2}') \quad \Box\, \mathit{transaction} \wedge \mathit{suspicious} \rightarrow \Box_{(0,3]}\, \mathit{transaction} \rightarrow \neg \mathit{suspicious}$$

$$\text{(P3}') \quad \Box\, \mathit{transaction} \wedge \mathit{suspicious} \rightarrow \big((\mathit{transaction} \rightarrow \Diamond_{[0,3]}\, \mathit{report})\, \mathsf{W}\, \mathit{unflag}\big)$$

$$\text{(P4}') \quad \Box\, \mathit{transaction} \wedge \mathit{suspicious} \rightarrow \Box_{[0,6]}\, \mathit{transaction} \rightarrow \Diamond_{[0,3]}\, \mathit{report}$$

Fig. 4. MTL formulas.

D Additional Evaluation Details

D.1 Compliance Policy Description

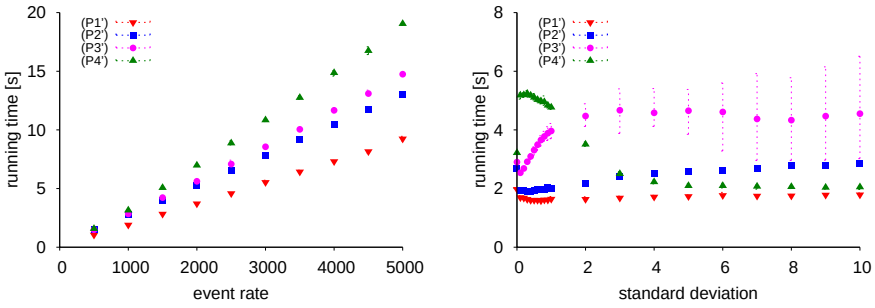
We start by explaining the predicate symbols that model the events that the banking system is assumed to log or transmit to the monitor. The predicate $\mathit{trans}(c, t, a)$ represents the execution of the transaction t of the customer c transferring the amount a of money. The predicate $\mathit{report}(t)$ classifies the transaction t as suspicious. Note that a message sent to the monitor describes an event and the register values. For instance, when executing a transaction, the registers *tid* and *cid* store the identifiers of the transaction and the customer; the amount of the transaction is stored in the register *sum*. For a *report* event, the register *tid* stores the identifier of the transaction whereas the other registers for the customer and the amount store the default value 0.

The formula (P1) requires that a transaction t of a customer c must be reported within at most three time units if the transferred amount a exceeds the threshold of $2,000. The formulas (P2) to (P4) are variants of (P1). (P2) requires that whenever a customer c makes a transaction that exceeds the threshold, then any of c's future transactions (within a specified period of time) must eventually be reported (within a specified time bound). (P3) requires that whenever a customer c makes a transaction t that exceeds the threshold, then c is not allowed to make further transactions until the transaction t is reported. Note that the syntactic sugar W ("weak until") is used here instead of the primitive temporal connective U. We not require that the transaction must eventually be reported. Finally, (P4) requires that whenever a customer c makes a transaction that exceeds the threshold, then any of c's transactions in a given time period must be reported.

D.2 Evaluation in a Propositional Setting

We consider the formulas in Figure 4 for comparing our experimental results in Section 5 with the simpler settings where no data values are involved. These formulas are propositional versions of the MTL$^{\downarrow}$ formula in Figure 2, except (P3$'$), which has an additional temporal connective and accounts for the additional event *unflag*.

Figure 5(a) shows the running times on logs with different event rates for the formulas (P1$'$) to (P4$'$). Figure 5(b) shows the impact when messages are received out of order for logs with an event rate 1000. We remark that some



(a) in-order (b) out-of-order

Fig. 5. Running times in a propositional setting (where each data point shows the mean of five logs together with the minimum and maximum).

care must be taken when comparing these figures with the Figures 3(a) and (b). First, the formulas express different policies. For instance, in (P2′) a report might discharge multiple transactions. Second, the logs for the propositional settings differ from the logs for the formulas (P1) to (P4). In particular, the events in the log files generated for the propositional settings do not account for different customers. Furthermore, we have chosen event rates that are 10 times higher. However, the running times in the propositional setting are significantly faster. In the propositional setting, our prototype usually processes an event in a fraction of a millisecond.