Policy Monitoring in First-order Temporal Logic[*]

David Basin, Felix Klaedtke, and Samuel Müller

Department of Computer Science, ETH Zurich, Switzerland

Abstract. We present an approach to monitoring system policies. As a specification language, we use an expressive fragment of a temporal logic, which can be effectively monitored. We report on case studies in security and compliance monitoring and use these to show the adequacy of our specification language for naturally expressing complex, realistic policies and the practical feasibility of monitoring these policies using our monitoring algorithm.

1 Introduction

Runtime monitoring is an approach to verifying system properties at execution time by using an online algorithm to check whether a system trace satisfies a temporal property. Whereas novel application areas such as compliance or business activity monitoring [8, 19, 24] require expressive property specification languages, current monitoring techniques are restricted in the properties they can handle. They either support properties expressed in propositional temporal logics and thus cannot cope with variables ranging over infinite domains [11, 27, 34, 39, 49], do not provide both universal and existential quantification [6, 30, 40, 43] or only in restricted ways [6, 25, 47, 48], do not allow arbitrary quantifier alternation [6,38], cannot handle unrestricted negation [13,38,46], do not provide quantitative temporal operators [38, 43], or cannot simultaneously handle both past and future operators [13, 25, 38–40, 44, 46, 48].

In this paper, we present our recent work [9,10] on runtime monitoring using an expressive safety fragment of metric first-order temporal logic (MFOTL), which overcomes most of the above limitations. The fragment consists of formulae of the form $\Box\,\phi$, where ϕ is bounded, i.e., its temporal operators refer only finitely into the future. As both (metric) past and bounded future operators may be arbitrarily nested, MFOTL supports natural specifications of complex policies. Moreover, the monitors work with infinite structures where relations are either representable by automata, so-called automatic structures [12, 32], or are finite.

We review MFOTL and our monitoring algorithm, present applications, and give performance results. For reasons of space, we only consider here monitoring structures with finite relations. In [10], we also show how to handle automatic structures and provide all definitions, algorithms, and proofs. Further details on our case studies and performance results are given in [9].

The applications we present come from the domain of security and compliance monitoring. An example, from financial reporting, is the requirement: *Every transaction of a customer who has within the last 30 days been involved in a previous suspicious transaction, must be reported as suspicious within two days.* Our examples illustrate MFOTL's suitability for specifying complex, realistic security policies. The class of policies covered constitute safety properties, where compliance can be checked by monitoring system traces. In the domain of security, this encompasses most traditional access-control policies as well as usage-control policies and policies arising in regulatory compliance. As we will see, such policies often combine event and state predicates, relations on data, and complex temporal relationships; all of these aspects can be naturally represented by MFOTL formulae interpreted over a metric, point-based semantics.

To evaluate our monitoring algorithm, we monitored different policies on synthetic data streams. Our experiments indicate that our approach is practically feasible with modest computing and storage requirements. Indeed, given that events can be processed in the order of milliseconds, the efficiency is such that our monitors can also be used online to detect policy violations.

2 Monitoring Metric First-order Temporal Properties

We first introduce metric first-order temporal logic (MFOTL), an extension of propositional metric temporal logic [33]. Afterwards, we describe our monitoring algorithm from [10] for a safety fragment of MFOTL.

2.1 Metric Temporal First-order Logic

Syntax and Semantics. Let $\mathbb{I}$ be the set of nonempty intervals over $\mathbb{N}$. We often write an interval in $\mathbb{I}$ as $[b, b') := \{a \in \mathbb{N} \mid b \leq a < b'\}$, where $b \in \mathbb{N}$, $b' \in \mathbb{N} \cup \{\infty\}$, and $b < b'$. A *signature* S is a tuple (C, R, ι), where C is a finite set of constant symbols, R is a finite set of predicates disjoint from C, and the function $\iota : R \to \mathbb{N}$ associates each predicate $r \in R$ with an arity $\iota(r) \in \mathbb{N}$. In the following, let $S = (C, R, \iota)$ be a signature and V a countably infinite set of variables, assuming $V \cap (C \cup R) = \emptyset$.

The *(MFOTL) formulae* over the signature S are given by the grammar

$$\phi ::= t_1 \approx t_2 \mid t_1 \prec t_2 \mid r(t_1, \ldots, t_{\iota(r)}) \mid \neg\phi \mid \phi \wedge \phi \mid \exists x.\phi \mid {\bullet}_I \phi \mid {\bigcirc}_I \phi \mid \phi \math:{\sf S}_I \phi \mid \phi \math:{\sf U}_I \phi \,,$$

where $t_1, t_2, \ldots$ range over the elements in $V \cup C$, and r, x, and I range over the elements in R, V, and $\mathbb{I}$, respectively.

To define MFOTL's semantics, we need the following notions. A *(first-order) structure* $\mathcal{D}$ over S consists of a domain $|\mathcal{D}| \neq \emptyset$ and interpretations $c^{\mathcal{D}} \in |\mathcal{D}|$ and $r^{\mathcal{D}} \subseteq |\mathcal{D}|^{\iota(r)}$, for each $c \in C$ and $r \in R$. A *temporal (first-order) structure* over S is a pair $(\bar{\mathcal{D}}, \bar{\tau})$, where $\bar{\mathcal{D}} = (\mathcal{D}_0, \mathcal{D}_1, \ldots)$ is a sequence of structures over S and $\bar{\tau} = (\tau_0, \tau_1, \ldots)$ is a sequence of natural numbers (i.e., time stamps), where:
1. The sequence $\bar{\tau}$ is monotonically increasing (i.e., $\tau_i \leq \tau_{i+1}$, for all $i \geq 0$) and makes progress (i.e., for every $i \geq 0$, there is some $j > i$ such that $\tau_j > \tau_i$).
2. $\bar{\mathcal{D}}$ has constant domains, i.e., $|\mathcal{D}_i| = |\mathcal{D}_{i+1}|$, for all $i \geq 0$. We denote the domain by $|\bar{\mathcal{D}}|$ and require that $|\bar{\mathcal{D}}|$ is strict linearly ordered by a relation $<$.

$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models t \approx t'$	iff	$v(t) = v(t')$
$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models t \prec t'$	iff	$v(t) < v(t')$
$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models r(t_1, \ldots, t_{\iota(r)})$	iff	$\big(v(t_1), \ldots, v(t_{\iota(r)})\big) \in r^{\mathcal{D}_i}$
$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \neg\phi$	iff	$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \not\models \phi$
$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi \wedge \psi$	iff	$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$ and $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \psi$
$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \exists x. \phi$	iff	$(\bar{\mathcal{D}}, \bar{\tau}, v[x/d], i) \models \phi$, for some $d \in |\bar{\mathcal{D}}|$
$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \bullet_I \phi$	iff	$i > 0, \ \tau_i - \tau_{i-1} \in I$, and $(\bar{\mathcal{D}}, \bar{\tau}, v, i-1) \models \phi$
$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \bigcirc_I \phi$	iff	$\tau_{i+1} - \tau_i \in I$ and $(\bar{\mathcal{D}}, \bar{\tau}, v, i+1) \models \phi$
$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi \, \mathsf{S}_I \, \psi$	iff	for some $j \le i, \ \tau_i - \tau_j \in I, \ (\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \psi$,
		and $(\bar{\mathcal{D}}, \bar{\tau}, v, k) \models \phi$, for all $k \in [j+1, i+1)$
$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi \, \mathsf{U}_I \, \psi$	iff	for some $j \ge i, \ \tau_j - \tau_i \in I, \ (\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \psi$,
		and $(\bar{\mathcal{D}}, \bar{\tau}, v, k) \models \phi$, for all $k \in [i, j)$

Fig. 1. Semantics of MFOTL.

3. Each constant symbol $c \in C$ has a rigid interpretation, i.e., $c^{\mathcal{D}_i} = c^{\mathcal{D}_{i+1}}$, for all $i \ge 0$. We denote c's interpretation by $c^{\bar{\mathcal{D}}}$.

A *valuation* is a mapping $v : V \to |\bar{\mathcal{D}}|$. We abuse notation by applying a valuation v also to constant symbols $c \in C$, with $v(c) = c^{\bar{\mathcal{D}}}$. For a valuation v, the variable vector $\bar{x} = (x_1, \ldots, x_n)$, and $\bar{d} = (d_1, \ldots, d_n) \in |\bar{\mathcal{D}}|^n$, $v[\bar{x}/\bar{d}]$ is the valuation mapping x_i to d_i, for $1 \le i \le n$, and the other variables' valuation is unaltered.

The semantics of MFOTL, $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$, is given in Figure 1, where $(\bar{\mathcal{D}}, \bar{\tau})$ is a temporal structure over the signature S, with $\bar{\mathcal{D}} = (\mathcal{D}_0, \mathcal{D}_1, \ldots)$, $\bar{\tau} = (\tau_0, \tau_1, \ldots)$, v a valuation, $i \in \mathbb{N}$, and ϕ a formula over S. Note that the temporal operators are augmented with intervals and a formula of the form $\bullet_I \phi$, $\bigcirc_I \phi$, $\phi \, \mathsf{S}_I \, \psi$, or $\phi \, \mathsf{U}_I \, \psi$ is only satisfied in $(\bar{\mathcal{D}}, \bar{\tau})$ at the time point i if it is satisfied within the bounds given by the interval I of the respective temporal operator, which are relative to the current time stamp τ_i.

Terminology and Notation. We use standard syntactic sugar such as $\blacksquare_I \phi :=$ $\neg(\textit{true} \, \mathsf{S}_I \, \neg\phi)$ and $\Box_I \phi := \neg(\textit{true} \, \mathsf{U}_I \, \neg\phi)$, where $\textit{true} := \exists x. x \approx x$. We also use non-metric operators like $\Box \phi := \Box_{[0,\infty)} \phi$. We omit parentheses where possible, e.g., unary operators (temporal and Boolean) bind stronger than binary ones.

We call formulae with no temporal operators *first-order*. A formula α is *bounded* if the interval I of every temporal operator U_I occurring in α is finite. The outermost connective (i.e., Boolean connective, quantifier, or temporal operator) occurring in a formula α is called the *main connective* of α. A formula that has a temporal operator as its main connective is a *temporal* formula. The set $tsub(\alpha)$ of *immediate* temporal subformulae of α is: (i) $tsub(\beta)$, if $\alpha = \neg\beta$ or $\alpha = \exists x. \beta$, (ii) $tsub(\beta) \cup tsub(\gamma)$, if $\alpha = \beta \wedge \gamma$, (iii) $\{\alpha\}$, if α is a temporal formula, and (iv) $\emptyset$ otherwise. For instance, for $\alpha := ((\bigcirc \beta) \, \mathsf{S} \, \gamma) \wedge \bullet \beta'$, we have $tsub(\alpha) = tsub((\bigcirc \beta) \, \mathsf{S} \, \gamma) \cup tsub(\bullet \beta') = \{(\bigcirc \beta) \, \mathsf{S} \, \gamma, \bullet \beta'\}$.

For a formula α with the free variables $\bar{x} = (x_1, \ldots, x_n)$, we define the set of satisfying elements at time point $i \in \mathbb{N}$ in the temporal structure $(\bar{\mathcal{D}}, \bar{\tau})$ as

$$\alpha^{(\bar{\mathcal{D}}, \bar{\tau}, i)} := \big\{ \bar{d} \in |\bar{\mathcal{D}}|^n \mid (\bar{\mathcal{D}}, \bar{\tau}, v[\bar{x}/\bar{d}], i) \models \alpha, \text{ for some valuation } v \big\}.$$

If α is first-order, $\alpha^{(\bar{\mathcal{D}}, \bar{\tau}, i)}$ only depends on the structure $\mathcal{D}_i$ and we just write $\alpha^{\mathcal{D}_i}$ in this case.

2.2   Monitoring

In the following, let Ψ be an MFOTL formula over the signature $S = (C, R, \iota)$. To effectively monitor Ψ, we restrict both the formula Ψ and the temporal structure $(\bar{\mathcal{D}}, \bar{\tau})$ over S, where $\bar{\mathcal{D}} = (\mathcal{D}_0, \mathcal{D}_1, \dots)$ and $\bar{\tau} = (\tau_0, \tau_1, \dots)$. To begin with, we require Ψ to be of the form $\square \Psi'$, where Ψ' is bounded.[1] To detect violations, prior to monitoring, we try to rewrite $\neg\Psi'$ to a logically equivalent formula Φ, belonging to a syntactically-defined fragment. The monitoring algorithm then iteratively processes the temporal structure $(\bar{\mathcal{D}}, \bar{\tau})$, evaluating Φ at each time point. Note that to identify violations, Ψ usually contains free variables and the violations are the satisfying assignments of Φ, which the monitor outputs.

The reason for rewriting $\neg\Psi'$ to Φ, rather than using $\neg\Psi'$ directly, is that the monitoring algorithm stores intermediate results when processing $(\bar{\mathcal{D}}, \bar{\tau})$ and therefore these results must be finite relations.[2] In particular, every relation $r^{\mathcal{D}_i}$ must be finite, for $i \in \mathbb{N}$ and $r \in R$. With the restriction to finite relations, we inherit a standard problem from database theory [3]. Namely, when $|\bar{\mathcal{D}}|$ is infinite, a query with negation can have an infinite answer set that itself cannot be represented by a finite relation. The restriction to so-called domain-independent queries, i.e., queries for which the answer set only depends on elements that occur in the database, only partially solves the problem: This guarantees finiteness but checking domain independence is undecidable [22]. A standard approach taken in database theory is therefore to try to rewrite a query into a form that falls into a syntactically-defined fragment that guarantees both the domain independence and the finiteness of the intermediate results. We take this approach and further details on rewrite rules and such a syntactically-defined fragment for MFOTL can be found in [10]. In the remainder of this section, we assume that Φ is from this monitorable fragment.

Overview. Our monitoring algorithm incrementally builds a sequence of structures $\hat{\mathcal{D}}_0, \hat{\mathcal{D}}_1, \dots$ over an extended signature $\hat{S}$. The extension depends on the temporal subformulae of Φ. For each time point i, we determine the elements that satisfy Φ by evaluating a first-order formula $\hat{\Phi}$ over $\hat{\mathcal{D}}_i$. Observe that for a temporal subformula with a future operator as its main connective, we usually cannot yet carry out this evaluation at time point i. The monitoring algorithm therefore maintains a queue of unevaluated formulae and evaluates them when enough time has elapsed.

We describe first how we extend S and transform Φ. Afterwards, we explain how we incrementally build $\hat{\mathcal{D}}_i$. Finally, we present our monitoring algorithm. For

---

[1] It follows that Ψ describes a safety property. Note, however, there are safety properties expressible in MFOTL that do not have such a syntactic form [15]. This is in contrast to propositional linear temporal logic, where every ω-regular safety property can be expressed as a formula $\square\beta$, where β contains only past operators [36].

[2] In fact, a weaker requirement suffices, namely, each $\mathcal{D}_i$ is an automatic structure [12, 32] and the $\mathcal{D}_i$s are uniformly represented. When using automatic structures, no further requirements on Ψ' are necessary and our monitoring algorithm can work with any Φ that is logically equivalent to $\neg\Psi'$. The intermediate results are also "automatic" and effectively computable [10].

the ease of exposition, we assume in the following that the temporal subformulae of Φ are of the form $\beta \, \mathsf{S}_I \, \gamma$ and $\square_{[0,b)} \, \beta$. The more general case for the temporal operator U_I is along the same lines as $\square_{[0,b)}$ but is technically more involved. The cases for $\bullet_I$ and $\bigcirc_I$ are straightforward and omitted here.

Signature Extension and Structure Construction. The extended signature $\hat{S}$ contains all constants and predicates in S, with the same arities. Moreover, for each temporal subformula α of Φ, $\hat{S}$ includes the new auxiliary predicates p_α and r_α, of arities n and $n+1$ respectively, where n is the number of free variables in α. For θ, a subformula of Φ over the signature S, $\hat{\theta}$ denotes the transformed formula over $\hat{S}$, where each $\alpha \in tsub(\theta)$ with the free variables $\bar{x}$ is replaced by $p_\alpha(\bar{x})$.

For $i \in \mathbb{N}$, $c \in C$, and $r \in R$, we define $|\hat{\mathcal{D}}_i| := |\bar{\mathcal{D}}| \cup \mathbb{N}$, $c^{\hat{\mathcal{D}}_i} := c^{\mathcal{D}_i}$, and $r^{\hat{\mathcal{D}}_i} := r^{\mathcal{D}_i}$. The auxiliary relations in the $\hat{\mathcal{D}}_i$s are defined inductively over both time and the formula structure. Furthermore, their construction is incremental in the sense that it reuses the auxiliary relations from the previous time points.

We start with the auxiliary relations for a subformula α of the form $\beta \, \mathsf{S}_{[b,b')} \, \gamma$. The non-metric variant of the construction reflects that $\beta \, \mathsf{S} \, \gamma$ is logically equivalent to $\gamma \vee \beta \wedge \bullet(\beta \, \mathsf{S} \, \gamma)$: For $i \geq 0$ and assuming without loss of generality that β and γ have the same vector of free variables, we define

$$p_{\beta \mathsf{S} \gamma}^{\hat{\mathcal{D}}_i} := \hat{\gamma}^{\hat{\mathcal{D}}_i} \cup \begin{cases} \emptyset & \text{if } i = 0, \\ \hat{\beta}^{\hat{\mathcal{D}}_i} \cap p_{\beta \mathsf{S} \gamma}^{\hat{\mathcal{D}}_{i-1}} & \text{if } i > 0. \end{cases}$$

Observe that this definition only depends on the relations in $\hat{\mathcal{D}}_i$ for which the corresponding predicates occur in the subformulae of $\hat{\beta}$ or $\hat{\gamma}$, and on the auxiliary relation $p_{\beta \mathsf{S} \gamma}^{\hat{\mathcal{D}}_{i-1}}$, when $i > 0$.

To incorporate the timing constraint for the interval $[b, b')$ of the S operator, we first incrementally construct the auxiliary relations for r_α similar to the definition above: For $i \geq 0$, we define $r_\alpha^{\hat{\mathcal{D}}_i} := N \cup U$, where $N := \hat{\gamma}^{\hat{\mathcal{D}}_i} \times \{0\}$ and

$$U := \begin{cases} \emptyset & \text{if } i = 0, \\ \left\{ (\bar{a}, y) \mid \bar{a} \in \hat{\beta}^{\hat{\mathcal{D}}_i}, \ y < b', \text{ and } (\bar{a}, y + \tau_{i-1} - \tau_i) \in r_\alpha^{\hat{\mathcal{D}}_{i-1}} \right\} & \text{if } i > 0. \end{cases}$$

Intuitively, a pair $(\bar{a}, y)$ is in $r_\alpha^{\hat{\mathcal{D}}_i}$ if $\bar{a}$ satisfies α at the time point i independent of the lower bound b, where the "age" y indicates how long ago the formula α was satisfied by $\bar{a}$. If $\bar{a}$ satisfies γ at the time point i, it is added to $r_\alpha^{\hat{\mathcal{D}}_i}$ with the age 0. For $i > 0$, we also update the tuples $(\bar{a}, y) \in r_\alpha^{\hat{\mathcal{D}}_{i-1}}$ when $\bar{a}$ satisfies β at time point i, i.e., the age is adjusted by the difference of the time stamps τ_{i-1} and τ_i in case the new age is less than b'. Otherwise it is too old to satisfy α and the updated tuple is not included in $r_\alpha^{\hat{\mathcal{D}}_i}$.

Finally, we obtain the auxiliary relation $p_\alpha^{\hat{\mathcal{D}}_i}$ from $r_\alpha^{\hat{\mathcal{D}}_i}$ by checking whether the age of a tuple in $r_\alpha^{\hat{\mathcal{D}}_i}$ is old enough:

$$p_\alpha^{\hat{\mathcal{D}}_i} := \left\{ \bar{a} \mid (\bar{a}, y) \in r_\alpha^{\hat{\mathcal{D}}_i}, \text{ for some } y \geq b \right\}.$$

We now address the bounded future operator $\square_{[0,b)}$, with $b \in \mathbb{N} \setminus \{0\}$. Assume that $\alpha = \square_{[0,b)} \, \beta$. For $i \in \mathbb{N}$, let $\ell_i := \max\{k \in \mathbb{N} \mid \tau_{i+k} - \tau_i < b\}$ denote the

lookahead offset at time point i. Note that only $\hat{\beta}^{\hat{\mathcal{D}}_i}, \ldots, \hat{\beta}^{\hat{\mathcal{D}}_{i+\ell_i}}$ are relevant for determining $\alpha^{(\bar{\mathcal{D}}, \bar{\tau}, i)}$. For $i \in \mathbb{N}$, we could directly define $p_\alpha^{\hat{\mathcal{D}}_i}$ as $\bigcap_{j \in \{0, \ldots, \ell_i\}} \hat{\beta}^{\hat{\mathcal{D}}_{i+j}}$. However, this construction has the drawback that for i and $i + 1$, we must recompute the intersections of the $\hat{\beta}^{\hat{\mathcal{D}}_{i+j}}$s for $j \in \{1, \ldots, \ell_i\}$.

We instead define $p_\alpha^{\hat{\mathcal{D}}_i}$ in terms of the incrementally-built auxiliary relation $r_\alpha^{\hat{\mathcal{D}}_i}$, where $(\bar{a}, k) \in r_\alpha^{\hat{\mathcal{D}}_i}$ iff $\bar{a} \in \hat{\beta}^{\hat{\mathcal{D}}_{i+j}}$, for all $j \in \{k, \ldots, \ell_i\}$. As before, we construct $r_\alpha^{\hat{\mathcal{D}}_i}$ from two sets N and U. N contains the elements from the new time points $i + \ell_{i-1}, \ldots, i + \ell_i$, where $\ell_{-1} := 0$ for convenience. U contains the updated elements from $r_\alpha^{\hat{\mathcal{D}}_{i-1}}$, if $i > 0$. To update an element $(\bar{a}, k) \in r_\alpha^{\hat{\mathcal{D}}_{i-1}}$, we check that $\bar{a}$ also satisfies β at the new time points. Furthermore, we decrease its index k, if $k > 0$. Formally, for $i \geq 0$, we define $r_\alpha^{\hat{\mathcal{D}}_i} := N \cup U$, where

$$N := \left\{ (\bar{a}, k) \mid \ell_{i-1} \leq k \leq \ell_i \text{ and } \bar{a} \in \hat{\beta}^{\hat{\mathcal{D}}_{i+k+j}}, \text{ for all } j \in \mathbb{N} \text{ with } k + j \leq \ell_i \right\}$$

and $U := \emptyset$ when $i = 0$ and if $i > 0$, then

$$U := \left\{ (\bar{a}, \max\{0, k-1\}) \mid (\bar{a}, k) \in r_\alpha^{\hat{\mathcal{D}}_{i-1}} \text{ and if } \ell_i - \ell_{i-1} \geq 0 \text{ then } (\bar{a}, \ell_{i-1}) \in N \right\}.$$

Finally, we define $p_\alpha^{\hat{\mathcal{D}}_i} := \{\bar{a} \mid (\bar{a}, 0) \in r_\alpha^{\hat{\mathcal{D}}_i}\}$.

We remark that both constructions of the auxiliary relations for the subformulae for the forms $\beta \, \mathsf{S}_I \, \gamma$ and $\square_{[0,b)} \, \beta$ can be optimized. For example, we can delete a tuple $(\bar{a}, k)$ in $r_{\square_{[0,b)} \, \beta}^{\hat{\mathcal{D}}_i}$ if it also contains a tuple $(\bar{a}, k')$ with $k' < k$.

Example. Before presenting our monitoring algorithm, we illustrate the formula transformation and the constructions of the auxiliary relations with the formula

$$\square \, \forall x. \, in(x) \to \lozenge_{[0,6)} \, out(x) \, .$$

To detect violations, we negate this formula and push negation inwards. To determine which elements violate the property, we also drop the quantifier, obtaining the formula $\lozenge \left(in(x) \wedge \square_{[0,6)} \, \neg out(x) \right)$. Since relations for out are finite, $\neg out(x)$ describes an infinite set and therefore the auxiliary relations for the subformula $\square_{[0,6)} \, \neg out(x)$ are infinite. Hence, we further rewrite the formula into the logically equivalent formula $\lozenge \, \Phi$, with $\Phi := in(x) \wedge \square_{[0,6)} \left(\neg out(x) \wedge \blacklozenge_{[0,6)} \, in(x) \right)$. The formula Φ is in our monitorable MFOTL fragment.

Observe that $\alpha := \square_{[0,6)} \left(\neg out(x) \wedge \blacklozenge_{[0,6)} \, in(x) \right)$ and $\alpha' := \blacklozenge_{[0,6)} \, in(x)$ are the only temporal subformulae of Φ. The transformed formula $\hat{\Phi} = in(x) \wedge p_\alpha(x)$ is over the signature $\hat{S}$ that extends Φ's signature with the auxiliary unary predicates p_α and $p_{\alpha'}$ and the binary predicates r_α and $r_{\alpha'}$.

We only illustrate the incremental constructions of the auxiliary relations for α by considering the temporal structure in Figure 2, which also depicts the relations for $p_{\alpha'}$. Observe that to build the relations $r_\alpha^{\hat{\mathcal{D}}_i}$, for $i \geq 0$, we require the relations $out^{\mathcal{D}_{i+k}}$ and the auxiliary relations $p_{\alpha'}^{\hat{\mathcal{D}}_{i+k}}$ with $k \in \{0, \ldots, \ell_i\}$, for the lookahead offset ℓ_i at time point i.

For the time point $i = 0$, we have $\ell_0 = 3$ because $\tau_3 - \tau_0 < 6$ and $\tau_4 - \tau_0 = 6$. Furthermore, the auxiliary relation $r_\alpha^{\hat{\mathcal{D}}_0}$ is $\{(\mathsf{c}, k) | 0 \leq k \leq 3\} \cup \{(\mathsf{d}, k) | 1 \leq k \leq 3\}$.

index i:	0	1	2	3	4	5	$\cdots$
time stamp τ_i:	1	1	3	6	7	9	$\cdots$ time
$in^{\mathcal{D}_i}$:	$\{a,c\}$	$\{b,d\}$	$\emptyset$	$\{c\}$	$\emptyset$	$\{d\}$	$\cdots$
$out^{\mathcal{D}_i}$:	$\emptyset$	$\emptyset$	$\{b\}$	$\{a\}$	$\{d\}$	$\emptyset$	$\cdots$
$p^{\hat{\mathcal{D}}_i}_{\alpha'}$:	$\{a,c\}$	$\{a,b,c,d\}$	$\{a,b,c,d\}$	$\{a,b,c,d\}$	$\{c\}$	$\{c,d\}$	$\cdots$

Fig. 2. A temporal structure.

1 $\ell \leftarrow 0$ *% current index in input sequence* $(\mathcal{D}_0, \tau_0), (\mathcal{D}_1, \tau_1), \ldots$

2 $i \leftarrow 0$ *% index of next query evaluation in input sequence* $(\mathcal{D}_0, \tau_0), (\mathcal{D}_1, \tau_1), \ldots$

3 $Q \leftarrow \big\{\big(\alpha, 0, \mathit{waitfor}(\alpha)\big) \,\big|\, \alpha$ *is a temporal subformula of* $\Phi\big\}$

4 loop

5   Carry over constants and relations of $\mathcal{D}_\ell$ to $\hat{\mathcal{D}}_\ell$.

6   **forall** $(\alpha, j, \emptyset) \in Q$ **do**   *% respect ordering of subformulae*

7   *Build auxiliary relations* $p^{\hat{\mathcal{D}}_j}_\alpha$ *and* $r^{\hat{\mathcal{D}}_j}_\alpha$.

8   *Discard auxiliary relations that were involved in the construction of* $r^{\hat{\mathcal{D}}_j}_\alpha$.

9   **while** *all auxiliary relations* $p^{\hat{\mathcal{D}}_i}_\alpha$ *are built for* $\alpha \in \mathit{tsub}(\Phi)$ **do**   *% eval query*

10   *Output* $(\hat{\Phi})^{\hat{\mathcal{D}}_i}$ *and* τ_i.

11   *Discard structure* $\hat{\mathcal{D}}_{i-1}$, *if* $i > 0$.

12 $i \leftarrow i + 1$

13 $Q \leftarrow \big\{\big(\alpha, \ell+1, \mathit{waitfor}(\alpha)\big) \,\big|\, \alpha$ *is a temporal subformula of* $\Phi\big\} \cup$
       $\big\{\big(\alpha, j, \bigcup_{\alpha' \in \mathit{update}(S, \tau_{\ell+1} - \tau_\ell)} \mathit{waitfor}(\alpha')\big) \,\big|\, (\alpha, j, S) \in Q$ *and* $S \neq \emptyset\big\}$

14 $\ell \leftarrow \ell + 1$ *% process next element* $(\mathcal{D}_{\ell+1}, \tau_{\ell+1})$ *in input sequence*

Fig. 3. Monitoring algorithm M_Φ.

In general, a pair $(\bar{a}, k)$ is in $r^{\hat{\mathcal{D}}_i}_\alpha$ iff $\bar{a}$ did not occur in one of the relations $out^{\mathcal{D}_{i+j}}$, with $j \in \{k, \ldots, \ell_i\}$ and $\bar{a}$ previously appeared in $in^{\mathcal{D}_{j'}}$, for some $j' \leq i+j$ with $\tau_{i+j} - \tau_{j'} < 6$. For example, the pair $(\mathsf{c}, 2)$ is in $r^{\hat{\mathcal{D}}_0}_\alpha$, since c is not in $out^{\mathcal{D}_2} \cup out^{\mathcal{D}_3}$ and c is in $in^{\mathcal{D}_0}$ and hence in $p^{\hat{\mathcal{D}}_2}_{\alpha'}$ and $p^{\hat{\mathcal{D}}_3}_{\alpha'}$. Recall that the lookahead offset ℓ_0 is 3 and therefore we only look at the time points 0 through 3. We obtain $p^{\hat{\mathcal{D}}_0}_\alpha$ as $\{\bar{a} \mid (\bar{a}, 0) \in r^{\hat{\mathcal{D}}_0}_\alpha\} = \{\mathsf{c}\}$, which contains also the satisfying elements for Φ at time point 0, since $p^{\hat{\mathcal{D}}_0}_\alpha \cap in^{\mathcal{D}_0} = \{\mathsf{c}\}$.

For the time point $i = 1$, the lookahead offset ℓ_1 is 2. Since $\ell_1 = \ell_0 - 1$, we need not consider any new time points, i.e., we obtain $r^{\hat{\mathcal{D}}_1}_\alpha$ from $r^{\hat{\mathcal{D}}_0}_\alpha$ by updating the tuples contained in $r^{\hat{\mathcal{D}}_0}_\alpha$, yielding $r^{\hat{\mathcal{D}}_1}_\alpha = \{(\mathsf{c}, 0), (\mathsf{c}, 1), (\mathsf{c}, 2), (\mathsf{d}, 0), (\mathsf{d}, 1), (\mathsf{d}, 2)\}$ and $p^{\hat{\mathcal{D}}_1}_\alpha = \{\mathsf{c}, \mathsf{d}\}$. The corresponding set of violating elements is $p^{\hat{\mathcal{D}}_1}_\alpha \cap in^{\mathcal{D}_1} = \{\mathsf{d}\}$. For the time point $i = 2$, we must also account for the new time point 4, since $\ell_2 = 2$. The only new element in $r^{\hat{\mathcal{D}}_2}_\alpha$ is $(\mathsf{c}, 2)$. The updated elements are $(\mathsf{c}, 0)$ and $(\mathsf{c}, 1)$. The pairs in $r^{\hat{\mathcal{D}}_1}_\alpha$ with the first component d are not updated since $\mathsf{d} \in out^{\mathcal{D}_4}$. We therefore obtain $p^{\hat{\mathcal{D}}_2}_\alpha = \{\mathsf{c}\}$ and $p^{\hat{\mathcal{D}}_2}_\alpha \cap in^{\mathcal{D}_2} = \emptyset$.

The Monitoring Algorithm. Figure 3 presents our monitoring algorithm M_Φ. Without loss of generality, we assume that each temporal subformula occurs only once in Φ. In the following, we describe M_Φ's operation.

M_Φ uses two counters ℓ and i. The counter ℓ is the index of the current element $(\mathcal{D}_\ell, \tau_\ell)$ in the input sequence $(\mathcal{D}_0, \tau_0), (\mathcal{D}_1, \tau_1), \ldots$, which is processed sequen-

tially. Initially, ℓ is 0 and it is incremented with each loop iteration (lines 4–14). The counter $i \leq \ell$ is the index of the next time point i (possibly in the past, from ℓ's point of view) for which we evaluate $\hat{\Phi}$ over the structure $\hat{\mathcal{D}}_i$. The evaluation is delayed until the relations $p_\alpha^{\hat{\mathcal{D}}_i}$ for $\alpha \in tsub(\Phi)$ have all been built (lines 10–12). Furthermore, M_Φ uses the list[3] Q to ensure that the auxiliary relations of $\hat{\mathcal{D}}_0, \hat{\mathcal{D}}_1, \dots$ are built at the right time: if $(\alpha, j, \emptyset)$ is an element of Q at the beginning of a loop iteration, enough time has elapsed to build the relations for the temporal subformula α of the structure $\hat{\mathcal{D}}_j$. M_Φ initializes Q in line 3. The function *waitfor* identifies the subformulae that delay the formula evaluation:

$$waitfor(\alpha) := \begin{cases} waitfor(\beta) & \text{if } \alpha = \neg\beta \text{ or } \alpha = \exists x.\,\beta, \\ waitfor(\beta) \cup waitfor(\gamma) & \text{if } \alpha = \beta \wedge \gamma \text{ or } \alpha = \beta \,\mathsf{S}_I\, \gamma, \\ \{\alpha\} & \text{if } \alpha = \Box_{[0,b)}\,\beta, \\ \emptyset & \text{otherwise.} \end{cases}$$

The list Q is updated in line 13 before we increment ℓ and start a new loop iteration. For an update, we use the set $update(U, t)$ defined as

$$\{\Box_{[0,b-t)}\,\beta \mid \Box_{[0,b)}\,\beta \in U \text{ and } b - t > 0\} \cup \{\beta \mid \Box_{[0,b)}\,\beta \in U \text{ and } b - t \leq 0\},$$

where U is a set of formulae and $t \in \mathbb{N}$. The update adds a new tuple $(\alpha, \ell + 1, waitfor(\alpha))$ to Q, for each temporal subformula α of Φ, and it removes the tuples of the form $(\alpha, j, \emptyset)$ from Q. Moreover, for tuples (α, j, S) with $S \neq \emptyset$, the set S is updated using the functions *waitfor* and *update*, accounting for the elapsed time to the next time point, i.e. $\tau_{\ell+1} - \tau_\ell$.

In lines 6–8, we build the relations for which enough time has elapsed, i.e., the auxiliary relations for α in $\hat{\mathcal{D}}_j$ with $(\alpha, j, \emptyset) \in Q$. Since a tuple $(\alpha', j, \emptyset)$ does not occur before a tuple $(\alpha, j, \emptyset)$ in Q, where α is a subformula of α', the relations in $\hat{\mathcal{D}}_j$ for α are built before those for α'. To build the relations, we use the incremental constructions described earlier in this section. After we have built these relations for α in $\hat{\mathcal{D}}_j$, we discard relations no longer needed to reduce space consumption. For instance, if $j > 0$ and $\alpha = \beta \,\mathsf{S}_I\, \gamma$, then we discard the relations $r_\alpha^{\hat{\mathcal{D}}_{j-1}}$ and $p_{\alpha'}^{\hat{\mathcal{D}}_j}$ with $\alpha' \in tsub(\beta) \cup tsub(\gamma)$.

In lines 9–12, if the auxiliary relations for p_α in $\hat{\mathcal{D}}_i$ of all immediate temporal subformulae α of Φ have been built, then M_Φ outputs the valuations violating Ψ' at time point i together with τ_i. Furthermore, after each output, the remainder of the extended structure $\hat{\mathcal{D}}_{i-1}$ is discarded (if $i > 0$) and i is incremented by 1.

Note that because M_Φ does not terminate, it is not an algorithm in the strict sense. However, it effectively computes the elements violating Ψ', for every time point n. More precisely, whenever M_Φ outputs the set $(\hat{\Phi})^{\hat{\mathcal{D}}_i}$ in line 10, then this set is finite, effectively computable, and $(\hat{\Phi})^{\hat{\mathcal{D}}_i} = (\neg\Psi')^{(\bar{\mathcal{D}}, \bar{\tau}, i)}$. Moreover, for each $n \in \mathbb{N}$, M_Φ eventually sets the counter i to n in some loop iteration.

---

[3] We abuse notation by using set notation for lists. Moreover, we assume that Q is ordered so that (α, j, S) occurs before (α', j', S'), whenever α is a proper subformula of α', or $\alpha = \alpha'$ and $j < j'$.

Since M_Φ iteratively processes the structures and time stamps in the temporal structure $(\bar{\mathcal{D}}, \bar{\tau})$, we measure its memory usage with respect to the processed prefix of $(\bar{\mathcal{D}}, \bar{\tau})$. The counters ℓ and i are at most the length of the processed prefix. Hence, in the nth loop iteration, we need $O(\log n)$ bits for these two counters. We can modify the monitoring algorithm M_Φ so that it is independent of the prefix length by replacing the two counters with a single counter that stores $\ell - i$, i.e., the distance of ℓ from i. Since the list Q stores tuples that contain indices of the processed prefix, we must make them relative to the next query evaluation. Under the additional assumption that there are at most m equal time stamps in $\bar{\tau}$, the number of bits for the new counter is then logarithmically bounded by the maximal lookahead offset, which is at most $m \cdot s$, where s is the sum of the upper bounds of the intervals of the future operators occurring in Φ. Furthermore, the number of elements in the list Q is bounded by $m \cdot s \cdot k$, where k is the number of temporal subformulae in Φ. Most importantly, the number of elements in the auxiliary relations that M_Φ stores in the nth loop iteration can be polynomially bounded by m, s, k, and the cardinality of the *active domain* of the processed prefix, where $adom_\ell(\bar{\mathcal{D}}) := \{c^{\bar{\mathcal{D}}} \mid c \in C\} \cup \bigcup_{0 \le n \le \ell} \bigcup_{r \in R} \{d_j \mid (d_1, \dots, d_{\iota(r)}) \in r^{\mathcal{D}_n} \text{ and } 1 \le j \le \iota(r)\}$. The degree of the polynomial is linear in the maximal number of free variables occurring in a temporal subformula of Φ. To achieve this polynomial bound, we must optimize the incremental construction of the auxiliary relations for $r_{\beta \mathsf{S}_{[b, \infty)} \gamma}$ so that the age of an element is the minimum of its actual age and the interval's lower bound b.

Given the above modifications to M_Φ and the additional assumption on the number of equal time stamps, the monitor's memory usage is polynomially bounded and independent of the length of the processed prefix. Moreover, the bound on the cardinalities of the auxiliary relations is independent of how often an element of $|\bar{\mathcal{D}}|$ appears in the relations of the processed prefix of the given temporal structure $(\bar{\mathcal{D}}, \bar{\tau})$.

3   Case Studies

We have carried out several case studies where we formalized and monitored a wide range of policies from the domain of security and regulatory compliance. In the following, we give two representative examples and report on the monitors' runtime performance for these cases. Other examples are given in [9].

3.1   Approval Requirements

Consider a policy governing the publication of business reports within a company, where all reports must be approved prior to publication. A simplified form of such a policy might be

$$\Box \, \forall f. \, publish(f) \rightarrow \blacklozenge \, approve(f) \, .$$

But this is too simplistic. More realistically, we would also require, for example, that the person publishing the report must be an accountant and the person approving the publication must be the accountant's manager. Moreover, the approval must happen within, say, 10 days prior to publication.

Note that predicates like approving a report and being someone's manager differ in the following respect. The act of approving a report represents an *event*: It happens at a time point and does not have a duration. In contrast, being someone's manager describes a *state* that has a duration. Since MFOTL's semantics is point-based, it naturally captures events. Entities like system states do not have a direct counterpart in MFOTL. However, we can model them using start and finish events. The following formalization of the above policy illustrates these two different kinds of entities and how we deal with them in MFOTL. To distinguish between them, we use the terms *event predicate* and *state predicate*.

Signature. The signature consists of the unary relation symbols acc_S and acc_F, and the binary relation symbols mgr_S, mgr_F, *publish*, and *approve*. Intuitively, $mgr_S(m, a)$ marks the time when m becomes a's manager and $mgr_F(m, a)$ marks the corresponding finishing time. Analogously, $acc_S(a)$ and $acc_F(a)$ mark the starting and finishing times of a being an accountant. We use these predicates to simulate state predicates in MFOTL, e.g., the formula $\underline{acc}(a) := \neg acc_F(a) \mathsf{S}\, acc_S(a)$ holds at the time points where a is an accountant. It states that a starting event for a being an accountant has previously occurred and the corresponding finishing event has not occurred since then. Analogously, $\underline{mgr}(m, a) := \neg mgr_F(m, a) \mathsf{S}\, mgr_S(m, a)$ is the state predicate expressing that m is a's manager.

Formalization. Before formalizing the approval policy, we formalize assumptions about the start and finish events in a temporal structure $(\bar{\mathcal{D}}, \bar{\tau})$. These assumptions reflect the system requirement that these events are generated in a "well-formed" way. First, we assume that start and finish events do not occur at the same time point, since their ordering would then be unclear. For example, for the start and finish events of being an accountant, we assume that $(\bar{\mathcal{D}}, \bar{\tau})$ satisfies the formula

$$\Box\, \forall a.\, \neg\big(acc_S(a) \wedge acc_F(a)\big)\,.$$

Furthermore, we assume that every finish event is preceded by a matching start event and between two start events there is a finish event. Formally, for the start and finish events of being an accountant, we assume that $(\bar{\mathcal{D}}, \bar{\tau})$ satisfies

$$\Box\, \forall a.\, acc_F(a) \rightarrow \bullet\, \underline{acc}(a) \qquad \text{and} \qquad \Box\, \forall a.\, acc_S(a) \rightarrow \neg\, \bullet\, \underline{acc}(a)\,.$$

The assumptions for mgr_S and mgr_F are similar and we omit them.

Our formalization of the policy that whenever a report is published, it must be published by an accountant and the report must be approved by her manager within at most 10 time units prior to publication is now given by the formula

$$\Box\, \forall a.\, \forall f.\, publish(a, f) \rightarrow \underline{acc}(a) \wedge \blacklozenge_{[0,11)} \exists m.\, \underline{mgr}(m, a) \wedge approve(m, f)\,. \quad \text{(P1)}$$

Note that the state predicates $\underline{acc}$ and $\underline{mgr}$ can change over time and that such changes are accounted for in our MFOTL formalization of this security policy. In particular, at the time point where m approves the report f, the formula (P1) requires that m is a's manager. However, m need no longer be a's manager when a publishes f, although a must be an accountant at that time point.

The resulting monitor for (P1) can be used in an offline setting, e.g., to read log files and report policy violations. When the monitor is built into a policy decision point, it can also be used, in this case, for policy enforcement.

3.2 Transaction Requirements

Our next example is a compliance policy for a banking system that processes customer transactions. The requirements stem from anti-money laundering regulations such as the Bank Secrecy Act [1] and the USA Patriot Act [2].

Signature. Let S be the signature (C, R, ι), with $C := \{th\}$, $R := \{trans, auth, report\}$, and $\iota(trans) := 3$, $\iota(auth) := 2$, and $\iota(report) := 1$. The ternary predicate *trans* represents the execution of a transaction of some customer transferring a sum of money. The binary predicate *auth* denotes the authorization of a transaction by some employee. Finally, the unary predicate *report* represents the situation where a transaction is reported as suspicious.

Formalization. We first formalize that executed transactions t of any customers c must be reported within at most 5 days if the transferred money a exceeds a given threshold th.

$$\square\,\forall c.\,\forall t.\,\forall a.\,trans(c,t,a) \wedge th \prec a \rightarrow \Diamond_{[0,6)}\,report(t)\,. \tag{P2}$$

Moreover, transactions that exceed the threshold must be authorized by some employee e prior to execution.

$$\square\,\forall c.\,\forall t.\,\forall a.\,trans(c,t,a) \wedge th \prec a \rightarrow \blacklozenge_{[2,21)}\,\exists e.\,auth(e,t)\,. \tag{P3}$$

Here we require that the authorization takes place at least 2 days and at most 20 days before executing the transaction. Our last requirement concerns the transactions of a customer that has previously made transactions that were classified as suspicious. Namely, every executed transaction t of a customer c, who has within the last 30 days been involved in a suspicious transaction t', must be reported as suspicious within 2 days.

$$\square\,\forall c.\,\forall t.\,\forall a.\,trans(c,t,a) \wedge \big(\blacklozenge_{[0,31)}\,\exists t'.\,\exists a'.\,trans(c,t',a') \wedge \Diamond_{[0,6)}\,report(t') \big) \rightarrow \Diamond_{[0,3)}\,report(t)\,. \tag{P4}$$

3.3 Experimental Evaluation

We implemented a Java prototype of the monitoring algorithm and evaluated its performance on the above policies. As input data, we synthetically generate finite prefixes of temporal structures, as this allows us to study the algorithm's performance under different parameter settings. Namely, for each formula, we synthesize finite prefixes of temporal structures over the formula's signature by drawing the time stamps and the elements of the relations from predefined sample spaces using a discrete uniform distribution. We restrict ourselves to relational structures with singleton relations that also satisfy the given well-formedness assumptions. To assess the monitor's long-run performance, we then

Table 1. Experimental results of the steady-state analysis.

formula	aspect	event frequency					sample space
		110	220	330	440	550	
(P1)	*ipt*	14.1	21.8	26.0	37.7	39.4	$\Omega_{20 \times 20 \times 2000}$
	sc	672±70.5	1,267±135.2	1,857±200.3	2,442±265.4	3,024±331.2	
	omax	1,208	2,155	3,006	3,988	4,884	
	radom	281	477	661	818	950	
(P2)	*ipt*	7.0	13.1	17.9	21.0	29.6	$\Omega_{1000 \times 25000 \times 2}$
	sc	353±4.4	700±8.7	1,044±12.0	1,386±15.2	1,725±20.7	
	omax	2,135	3,959	5,172	7,377	8,714	
	radom	404	762	1,098	1,422	1,726	
(P3)	*ipt*	1.7	2.8	3.7	4.8	10.4	$\Omega_{1000 \times 25000 \times 2 \times 200}$
	sc	119±1.3	235±2.6	350±3.9	465±5.0	579±5.6	
	omax	158	282	412	545	659	
	radom	492	893	1,252	1,583	1,893	
(P4)	*ipt*	2.2	3.5	4.7	6.0	7.6	$\Omega_{1000 \times 25000 \times 2}$
	sc	140±2.8	405±9.0	801±19.1	1,334±32.2	1,994±47.8	
	omax	723	1,270	2,242	3,302	4,360	
	radom	404	762	1,098	1,422	1,726	

conduct a steady-state analysis [35], which is a standard method for estimating the behavior of non-terminating processes in the limit. For more information on our experimental setup, see [9].

Table 1 summarizes our experimental results using a 1.4 GHz dual core computer with 3 GBytes of RAM. The size of the sample space for the m different kinds of data, e.g., managers, accountants, and files, is denoted by $\Omega_{n_1 \times \cdots \times n_m}$. The sample space for time stamps is chosen so that the different lengths of the generated temporal structures simulate scenarios with the (approximate) event frequencies $110, 220, \ldots, 550$, i.e., the number of structures associated with each time point that approximately occur in the time window specified by the metric temporal operators of the given formula. We measure the following aspects. (1) *ipt* denotes the steady-state mean incremental processing times, in milliseconds. The incremental processing time is the time the monitor needs to construct and update the auxiliary relations in one loop iteration. (2) *sc* denotes a point estimate of the steady-state mean space consumption, where the actual average space consumption lies in the specified interval with a probability of 95%. We measured the monitor's space consumption as the sum of the cardinalities of the auxiliary relations. (3) *omax* denotes the maximal space consumption that we observed in our experiments. Finally, (4) *radom* denotes the average of the cardinalities of the relevant active domains[4] after the warm-up phase.

The results of our experiments, depicted in Table 1, predict low space consumption and running times of the monitoring algorithm in the long run. This

---

[4] The *relevant active domain* with respect to a time point is the set of data elements of the temporal structure that appear in the relations in the formula's time window at the time point. Although these cardinalities are only a rough complexity measure for the processed input prefix, they help us judge the monitor's performance better than more simplistic measures like the cardinality of the active domain of the processed prefix or the length of the prefix. In particular, the cardinalities of the relevant active domains relate the incremental update times and the cardinalities of the auxiliary relations to the input prefix of a temporal structure with respect to the formula to be monitored. The elements that do not occur in the relevant active domain for a time point are irrelevant for detecting policy violations at that time point.

suggests that we can monitor realistic policies with manageable overhead. Moreover, the monitoring algorithm scales well with respect to the event frequency: the growth rates of all four aspects measured are approximately linear with respect to the event frequency.

Our results also shed light on the relationship between formula structure and monitoring efficiency. The state predicates used in (P1) result in additional temporal subformulae and hence increased space consumption and processing time. Moreover, the maximal observed space consumption is close to the estimated steady-state mean space consumption for formulae only referring to the past. For formulae containing future operators, i.e. (P2) and (P4), these values differ up to a factor of 6 since the monitoring algorithm delays the policy check at time points when it depends on future events. The information about the current time point must be stored in auxiliary relations until this check is performed.

4   Related Work

Temporal logics are widely applicable in computing since they allow one to formally and naturally express system properties and they can be handled algorithmically. For instance, in system verification, the propositional temporal logics LTL, CTL, and PSL are widely used [16,42,50]. Linear-time temporal logics like first-order extensions of LTL and different real-time extensions [4] have also been used to formalize [8,19,24,28,29,31,51] and to reason about [8,14,19,20] system policies. However, reasoning about policies has been mostly carried out in just the propositional setting [8,20]. For instance, in [8], policy consistency is reduced to checking whether an LTL formula is satisfiable and verification techniques for LTL are proposed for checking runtime compliance. This kind of reasoning is inadequate for systems with unboundedly many users or data elements. Note that although a system has only finitely many users at each time point, the number of users over time is usually unbounded.

In the domain of security and compliance checking, bounds on the number of users or data elements are usually unrealistic. Hence most monitoring algorithms, e.g. [11,17,18,21,23,30,34,40,44], are of limited use in this domain. The rule-based monitoring approach implemented in the closely related EAGLE [6] and RuleR [7] frameworks partially overcomes this limitation. There, properties are given as so-called temporal equations, which can have parameters referring to data that are instantiated during monitoring. EAGLE's rule-based approach has been used in [19] to monitor regulations, where one distinguishes between provisions and obligations and where regulations can refer to other regulations. Analogous to the use of parametric temporal equations in EAGLE and RuleR, the monitoring algorithm from [41,43] for auditing log files instantiates the parameters occurring in the given temporal formula during the monitoring process. Roughly speaking, such instantiations create propositions on demand and the number of propositions can be unbounded. These instantiations can also be seen as a restricted form of existential quantification, where variables are assigned to values that appear at the current position of the input trace.

The linear-time temporal logic used for monitoring in [25, 26] directly supports quantification. However, quantified variables only range over elements that appear at the current position of the input trace. Similar to [6, 7, 43], quantification is handled by instantiations. In contrast, our monitoring algorithm does not create propositions at runtime. Instead it creates auxiliary relations for the temporal subformulae of the given MFOTL formula. Our monitoring algorithm thereby handles more general existential and universal quantification; however, formulae must be domain independent. A simple, albeit artificial, example is the formula $\square\,\exists x.\,(\bullet\, p(x)) \wedge \bigcirc\,\neg q(x)$ whose negation is $\Diamond\,\forall x.\,(\bullet\, p(x)) \rightarrow \bigcirc q(x)$, which is in our monitorable fragment. However, elements $a \in |\bar{\mathcal{D}}|$ for which $a \in p^{\mathcal{D}_{i-1}}$ holds, need not appear at the current time point i, for $i > 0$. The monitoring approach in [48] is similar to the one in [25] but instead of using a tableau construction as in [25], it uses so-called parametric alternating automata, which are instantiated during runtime. Other differences to our monitoring algorithm are that the monitoring algorithms in [25, 48] do not handle past operators and future operators need not be bounded.

Our monitoring algorithm is based on Chomicki's monitor for checking integrity constraints on temporal databases [13]. It extends and improves Chomicki's monitor by supporting bounded future operators and by simplifying and optimizing the incremental update constructions for the metric operators. Moreover, when using automatic structures, no syntactic restrictions on the MFOTL formula to domain-independent queries are necessary. Other monitoring algorithms for temporal databases are given in [38, 46]. Both of these algorithms support only future operators and neither handles arbitrary quantifier alternation. Processing database streams is also related to monitoring and compliance checking. However, query languages like CQL [5] are less expressive temporally. What they usually provide instead are operators for manipulating sequences, for example, transforming streams into relations and vice versa.

In this paper, our focus is on monitoring for compliance checking, rather than policy enforcement [37, 45]. Enforcement is more difficult as it may necessitate changing future actions or predicting when existing actions have consistent extensions. It is also complicated by distribution, as a monitor may be able to observe events, but not necessarily control them.

5   Conclusions

We have given an overview of some of the ideas behind our approach to runtime monitoring using an expressive fragment of a metric first-order temporal logic. We have also given examples illustrating how policies can be formalized and we have analyzed the monitor's resource requirements.

Of course, our approach is not a panacea. Policies outside the scope of MFOTL include those for which no domain-independent formalization exists or those requiring a more expressive logic. An example of the latter is the requirement *a report must be filed within 3 days when all transactions of a trader over the last week sum up to more than $50 million*, involving the aggregation operator for summation. Similarly, our experiments indicate that the monitoring

algorithm does not handle all policies equally well as a policy's syntactic form may influence monitoring efficiency. In general, for monitoring those properties formalizable in MFOTL, there may be more efficient, specialized algorithms than ours. Despite these limitations, MFOTL appears to sit in the sweet spot between expressivity and complexity: it is a large hammer, applicable to many problems, and has acceptable runtime performance.

We have indicated that our monitors can be used in some cases for policy enforcement. We plan to explore how this can best be done and to compare the performance with competing approaches. We would also like to carry out concrete case studies in the application domains presented in this paper.

References

1. Bank Secrecy Act of 1970, 1970. 31 USC 5311-5332 and 31 CFR 103.
2. USA Patriot Act of 2001, 2001. Public Law 107-56, HR 3162 RDS.
3. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.
4. R. Alur and T. A. Henzinger. Logics and models of real time: A survey. In *Proc. of the REX Workshop on Real Time: Theory in Practice*, vol. 600 of *LNCS*, pp. 74–106. Springer, 1992.
5. A. Arasu, S. Babu, and J. Widom. The CQL continuous query language: semantic foundations and query execution. *VLDB Journal*, 15(2):121–142, 2006.
6. H. Barringer, A. Goldberg, K. Havelund, and K. Sen. Rule-based runtime verification. In *Proc. of the 5th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI)*, vol. 2937 of *LNCS*, pp. 44–57. Springer, 2004.
7. H. Barringer, D. E. Rydeheard, and K. Havelund. Rule systems for run-time monitoring: From Eagle to RuleR. *J. Logic Comput.*, to appear.
8. A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum. Privacy and contextual integrity: Framework and applications. In *Proc. of the 2006 IEEE Symposium on Security and Privacy*, pp. 184–198. IEEE Computer Society, 2006.
9. D. Basin, F. Klaedtke, and S. Müller. Monitoring security policies with metric first-order temporal logic. In *15th ACM Symposium on Access Control Models and Technologies (SACMAT)*, 2010. Accepted for publication.
10. D. Basin, F. Klaedtke, S. Müller, and B. Pfitzmann. Runtime monitoring of metric first-order temporal properties. In *Proc. of the 28th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, Dagstuhl Seminar Proc., pp. 49–60, 2008.
11. A. Bauer, M. Leucker, and C. Schallhart. Monitoring of real-time properties. In *Proc. of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, vol. 4337 of *LNCS*, pp. 260–272. Springer, 2006.
12. A. Blumensath and E. Grädel. Finite presentations of infinite structures: Automata and interpretations. *Theory Comput. Syst.*, 37(6):641–674, 2004.
13. J. Chomicki. Efficient checking of temporal integrity constraints using bounded history encoding. *ACM Trans. Database Syst.*, 20(2):149–186, 1995.
14. J. Chomicki and J. Lobo. Monitors for history-based policies. In *Proc. of the International Workshop on Policies for Distributed Systems and Networks (POLICY)*, vol. 1995 of *LNCS*, pp. 57–72. Springer, 2001.

15. J. Chomicki and D. Niwiński. On the feasibility of checking temporal integrity constraints. *J. Comput. Syst. Sci.*, 51(3):523–535, 1995.
16. E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Proc. of the Workshop on Logics of Programs*, vol. 131 of *LNCS*, pp. 52–71. Springer, 1982.
17. M. d'Amorim and G. Roşu. Efficient monitoring of ω-languages. In *Proc. of 17th International Conference on Computer Aided Verification (CAV)*, vol. 3576 of *LNCS*, pp. 364–378. Springer, 2005.
18. B. D'Angelo, S. Sankaranarayanan, C. Sánchez, W. Robinson, B. Finkbeiner, H. B. Sipma, S. Mehrotra, and Z. Manna. LOLA: Runtime monitoring of synchronous systems. In *Proc. of the 12th International Symposium on Temporal Representation and Reasoning (TIME)*, pp. 166–174. IEEE Computer Society, 2005.
19. N. Dinesh, A. Joshi, I. Lee, and O. Sokolsky. Checking traces for regulatory conformance. In *Proc. of the 8th Workshop on Runtime Verification (RV)*, vol. 5289 of *LNCS*, pp. 86–103. Springer, 2008.
20. D. J. Dougherty, K. Fisler, and S. Krishnamurthi. Obligations and their interaction with programs. In *Proc. of the 12th European Symposium on Research in Computer Security (ESORICS)*, vol. 4734 of *LNCS*, pp. 375–289. Springer, 2007.
21. D. Drusinsky. On-line monitoring of metric temporal logic with time-series constraints using alternating finite automata. *Journal of Universal Computer Science*, 12(5):482–498, 2006.
22. R. Fagin. Horn clauses and database dependencies. *J. ACM*, 29(4):952–985, 1982.
23. D. Giannakopoulou and K. Havelund. Automata-based verification of temporal properties on running programs. In *Proc. of the 16th IEEE International Conference on Automated Software Engineering (ASE)*, pp. 412–416. IEEE Computer Society, 2001.
24. C. Giblin, A. Y. Liu, S. Müller, B. Pfitzmann, and X. Zhou. Regulations expressed as logical models (REALM). In *Proc. of the 18th Annual Conference on Legal Knowledge and Information Systems (JURIX)*, vol. 134 of *Frontiers Artificial Intelligence Appl.*, pp. 37–48. IOS Press, 2005.
25. S. Hallé and R. Villemaire. Runtime monitoring of message-based workflows with data. In *Proc. of the 12th International IEEE Enterprise Distributed Object Computing Conference (EDOC)*, pp. 63–72. IEEE Computer Society, 2008.
26. S. Hallé and R. Villemaire. Browser-based enforcement of interface contracts in web applications with BeepBeep. In *Proc. of the 21st International Conference on Computer Aided Verification (CAV)*, vol. 5643 of *LNCS*, pp. 648–653. Springer, 2009.
27. K. Havelund and G. Roşu. Efficient monitoring of safety properties. *Int. J. Softw. Tools Technol. Trans.*, 6(2):158–173, 2004.
28. M. Hilty, D. Basin, and A. Pretschner. On obligations. In *Proc. of the 10th European Symposium on Research in Computer Security (ESORICS)*, vol. 3679 of *LNCS*, pp. 98–117. Springer, 2005.
29. M. Hilty, A. Pretschner, D. Basin, C. Schaefer, and T. Walter. A policy language for distributed usage control. In *Proc. of the 12th European Symposium on Research in Computer Security (ESORICS)*, vol. 4734 of *LNCS*, pp. 531–546. Springer, 2007.
30. J. Håkansson, B. Jonsson, and O. Lundqvist. Generating online test oracles from temporal logic specifications. *Int. J. Softw. Tools Technol. Trans.*, 4(4):456–471, 2003.
31. H. Janicke, A. Cau, and H. Zedan. A note on the formalisation of UCON. In *Proc. of the 12th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pp. 163–168. ACM Press, 2007.

32. B. Khoussainov and A. Nerode. Automatic presentations of structures. In *Proc. of the International Workshop on Logical and Computational Complexity*, vol. 960 of *LNCS*, pp. 367–392. Springer, 1995.
33. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Syst.*, 2(4):255–299, 1990.
34. K. J. Kristoffersen, C. Pedersen, and H. R. Andersen. Runtime verification of timed LTL using disjunctive normalized equation systems. *Elec. Notes Theo. Comput. Sci.*, 89(2):1–16, 2003.
35. A. M. Law. *Simulation, Modeling & Analysis*. McGraw-Hill, New York, NY, USA, 4th edition, 2007.
36. O. Lichtenstein, A. Pnueli, and L. D. Zuck. The glory of the past. In *Proc. of the Conference on Logic of Programs*, vol. 193 of *LNCS*, pp. 196–218. Springer, 1985.
37. J. Ligatti, L. Bauer, and D. Walker. Edit automata: enforcement mechanisms for run-time security policies. *Int. J. Inf. Secur.*, 4(1-2):2–16, 2005.
38. U. W. Lipeck and G. Saake. Monitoring dynamic integrity constraints based on temporal logic. *Information Systems*, 12(3):255–269, 1987.
39. O. Maler, D. Nickovic, and A. Pnueli. From MITL to timed automata. In *Proc. of the 4th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, vol. 4202 of *LNCS*, pp. 274–289. Springer, 2006.
40. D. Nickovic and O. Maler. AMT: A property-based monitoring tool for analog systems. In *Proc. of the 5th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, vol. 4763 of *LNCS*, pp. 304–319. Springer, 2007.
41. J. Olivain and J. Goubault-Larrecq. The Orchids intrusion detection tool. In *Proc. of the 17th International Conference on Computer Aided Verification*, vol. 3576 of *LNCS*, pp. 286–290. Springer, 2005.
42. A. Pnueli. The temporal logic of programs. In *Proc. of the 18th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 46–57. IEEE Computer Society, 1977.
43. M. Roger and J. Goubault-Larrecq. Log auditing through model-checking. In *Proc. of the 14th IEEE Computer Security Foundations Workshop (CSFW)*, pp. 220–234. IEEE Computer Society, 2001.
44. G. Roşu and K. Havelund. Rewriting-based techniques for runtime verification. *Automat. Softw. Eng.*, 12(2):151–197, 2005.
45. F. B. Schneider. Enforceable security policies. *ACM Trans. Inform. Syst. Secur.*, 3(1):30–50, 2000.
46. A. P. Sistla and O. Wolfson. Temporal triggers in active databases. *IEEE Trans. Knowl. Data Eng.*, 7(3):471–486, 1995.
47. O. Sokolsky, U. Sammapun, I. Lee, and J. Kim. Run-time checking of dynamic properties. *Elec. Notes Theo. Comput. Sci.*, 144(4):91–108, 2006.
48. V. Stolz. Temporal assertions with parameterized propositions. *J. Logic Comput.*, to appear.
49. P. Thati and G. Roşu. Monitoring algorithms for metric temporal logic specifications. *Elec. Notes Theo. Comput. Sci.*, 113:145–162, 2005.
50. M. Y. Vardi. From philosophical to industrial logics. In *Proc. of the 3rd Indian Conference on Logic and its Applications (ICLA)*, vol. 5378 of *LNCS*, pp. 89–115. Springer, 2009.
51. X. Zhang, F. Parisi-Presicce, R. Sandhu, and J. Park. Formal model and policy specification of usage control. *ACM Trans. Inform. Syst. Secur.*, 8(4):351–387, 2005.