Monitoring Data Usage in Distributed Systems

David Basin, Matúš Harvan, Felix Klaedtke, and Eugen Zălinescu

Abstract—IT systems manage increasing amounts of sensitive data and there is a growing concern that they comply with policies that regulate data usage. In this article, we use temporal logic to express policies, and runtime monitoring to check system compliance. While well-established methods for monitoring linearly-ordered system behavior exist, a major challenge is monitoring distributed and concurrent systems, where actions are locally observed in the different system parts. These observations can only be partially ordered while policy compliance may depend on the actions' actual order of appearance. Technically speaking, it is in general intractable to check compliance of partially ordered traces. We identify fragments of our policy specification language for which compliance can be checked efficiently, namely, by monitoring a single representative trace in which the observed actions are totally ordered. Through a case study we show that the fragments are capable of expressing non-trivial policies and that monitoring representative traces is feasible on real-world data.

Index Terms—Monitors, Temporal logic, Verification, Distributed Systems, Regulation

1 Introduction

It is a growing concern for companies, administrations, and end users alike whether IT systems comply with policies regulating the usage of sensitive data. Checking their compliance is particularly acute as many of our modern infrastructures (communication, entertainment, finance and banking, social networks, etc.) are based on IT systems that collect, process, and share data. Moreover, increasingly many legal regulations mandate compliance, such as the US Health Insurance Portability and Accountability Act (HIPAA) [2], the Sarbanes-Oxley Act (SOX) [3], and the EU Directive 95/46/EC [4].

A prominent approach to compliance checking is runtime monitoring. Here, system actions are observed and automatically checked for compliance against a policy. Efficient monitoring algorithms have been given for this task for various policy specification languages, see, for example, [5]–[10]. The underlying semantic model of these languages is that the observed system actions are totally ordered. However, a total ordering is often not available. Even simple IT systems are composed of multiple interacting subsystems, which typically are distributed and act concurrently. Hence system actions can only be observed locally and independently in each subsystem. Although we have a total ordering on the actions observed in each individual subsystem, it is unclear how to combine them with actions observed in other subsystems. And policy compliance may depend on how all observed actions are totally ordered.

Synchronization of all subsystems for each observed system action leads to a total ordering, but this is usually prohibitively expensive. Not requiring it leads to a partial order on the observed actions. Determining whether at least one or whether all possible extensions of such a partial order into a total order violate a policy is in general an intractable problem. Intuitively, this is because a partial ordering on a finite set has, in the worst case, exponentially many different extensions to a total ordering.

In this article, we identify policies for which compliance can be checked efficiently by inspecting a single representative sequence in which the observed system actions are totally ordered. Furthermore, we deploy and evaluate our solution in a real-world concurrent and distributed IT system. To explain our approach in more detail, we continue with an abstract description of the systems that we handle and we describe our monitoring setup.

System Model. The types of entities in the systems that we consider are *data*, *(data) stores*, *agents*, and *actions*. Data is stored in distributed data stores such as databases and repositories and created, read, modified, combined, propagated, and deleted by actions initiated by agents. Agents are either humans or applications, including database triggers, and they do not necessarily comply with policies.

In our system model, we assume that agents always access data directly from a store and never indirectly from another agent. Whenever an agent wants to use some data, it accesses the appropriate store, uses the data, and discards it afterwards. For subsequent usage, it must access the store again. Before discarding the data, the agent may write it, possibly after processing it in some way, into the same or a different store. In this way, data can propagate between stores. A consequence of this restriction on the interaction between system entities is that the use of data is always observable at the data stores.

Monitoring Setup. Given an instance of the above system model, we extend it to observe system actions. We log them locally at the data stores, annotating each action with a timestamp. We assume that the clocks are synchronized [11] and of limited precision (timestamps come from a non-dense set). Hence even with clock synchronization, the timestamps lead only to a partial ordering since actions can be logged in

• *David Basin, Matúš Harvan, Felix Klaedtke, and Eugen Zălinescu are with the Institute of Information Security, ETH Zurich, Universitätstr. 6, 8092 Zurich, Switzerland.*
Email: firstname.lastname@inf.ethz.ch

Manuscript received xxx; revised xxx; accepted xxx; published online xxx.
This article is an extended version of the conference paper [1].
Recommended for acceptance by xxx.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference xxx.

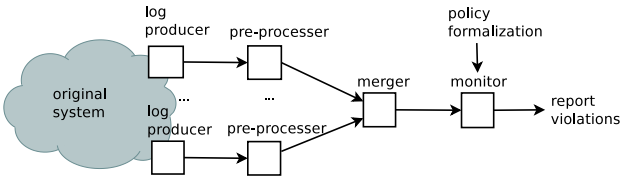

Fig. 1. System Extensions

different subsystems with equal timestamps. We pre-process the local logs, merge them, and monitor this merged stream of logged actions. These system extensions are depicted in Figure 1.

To express policies, we use a metric first-order temporal logic (MFOTL). In general, temporal logics [12] are well suited to formalize system properties and to algorithmically reason about system behavior. In particular, the standard temporal operators allow us to naturally express temporal aspects of data usage policies, such as whenever a user requests the deletion of his data then the data must eventually be deleted. Metric temporal logics [13] associate timing constraints with temporal operators. We can thereby straightforwardly express requirements that commonly occur in data-usage policies, for example that data deletion must happen within 30 days. A first-order logic allows us to formulate dependencies between the finite but unbounded number of agents and data elements in IT systems.

In [10] we presented a monitoring algorithm for an expressive fragment of MFOTL for a totally ordered sequence of timestamped actions and in [14] we described an implementation of this algorithm. We also showed that many policies can naturally be expressed in a fragment of this logic, which we can effectively monitor [15].

Summary. We identify two fragments of MFOTL that describe policies insensitive to the ordering of actions labeled with equal timestamps. For policies expressed in these fragments, it suffices to monitor a single stream of logged actions. For the first fragment, an arbitrary interleaving can be monitored. For the second fragment, it suffices to monitor the collapse of an interleaving, which is where actions with equal timestamps are merged. Both an interleaving and a collapse can be easily obtained by merging the logs produced by the subsystems.

The first fragment subsumes the second one in terms of expressiveness. However, system monitoring with respect to formulas in the second fragment is more efficient. Both fragments are defined by labeling a formula's atomic formulas and using rules to propagate the labels to the formula's root. The labels describe semantic properties about the insensitivity of the labeled subformula to the ordering of actions with equal timestamps. Furthermore, we provide means to approximate policies to fall within these fragments.

We evaluate our approach in a real-world case study, Nokia's Data-collection Campaign [16]. In this campaign, sensitive data is collected by mobile phones and propagated between databases. The underlying IT system is an instance of our system model. For the evaluation, we extended it to support logging and monitoring as indicated in Figure 1. We used MFOTL to express policies and our monitoring tool [14] for compliance checking.

Contributions. We provide a solution for efficiently monitoring partially ordered logs, which is a central problem in monitoring real-time concurrent distributed systems. Moreover, we demonstrate the effectiveness of our approach on a real-world application. In particular, the two identified MFOTL fragments are sufficiently expressive to capture real-world policies and our monitor can efficiently check such policies on real-world logs. Although we focus here on MFOTL as the policy specification language and our monitor [10], [14], the underlying principle of monitoring a single representative to check compliance of an IT system is a general one that applies to other policy specification languages and monitoring algorithms.

Organization. The remainder of this article is structured as follows. In Section 2, we provide background on MFOTL and our monitor. In Section 3, we prove that monitoring a partially ordered set of actions is in general intractable. In the Sections 4 and 5, we define fragments of formulas for which this problem can be solved efficiently. In Section 6, we compare these fragments and explain how a policy can be approximated by one that can be monitored efficiently. In Section 7, we report on our case study. In Section 8, we discuss related work and in Section 9, we draw conclusions. Additional proof details are given in the appendix.

2 Preliminaries

We briefly review metric first-order temporal logic (MFOTL) and our monitoring algorithm.

2.1 Metric First-order Temporal Logic

Syntax and Semantics. Let $\mathbb{I}$ be the set of nonempty intervals over $\mathbb{N}$. We write an interval $I \in \mathbb{I}$ 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 predicate symbols disjoint from C, and the function $\iota : R \to \mathbb{N}$ associates each predicate symbol $r \in R$ 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 $V \cap (C \cup R) = \emptyset$.

Formulas over the signature S are given by the grammar

$$\phi ::= t_1 \approx t_2 \mid t_1 \prec t_2 \mid r(t_1, \ldots, t_{\iota(r)}) \mid (\neg\phi) \mid (\phi \vee \phi) \mid (\exists x.\, \phi) \mid (\ominus_I \phi) \mid (\bigcirc_I \phi) \mid (\phi \,\mathsf{S}_I\, \phi) \mid (\phi \,\mathsf{U}_I\, \phi),$$

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 *structure* $\mathcal{D}$ over the signature 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 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, where the following conditions hold:

(1) The sequence $\bar{\tau}$ is monotonically increasing (that is, $\tau_i \leq \tau_{i+1}$, for all $i \geq 0$) and makes progress (that is, for every $i \geq 0$, there is some $j > i$ such that $\tau_j > \tau_i$).

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

Fig. 2. Semantics of MFOTL

(2) $\bar{\mathcal{D}}$ has constant domains, that is, $|\mathcal{D}_i| = |\mathcal{D}_{i+1}|$, for all $i \ge 0$. We denote the domain by $|\bar{\mathcal{D}}|$ and require that its elements are strictly linearly ordered by the relation $<$.
(3) Each constant symbol $c \in C$ has a rigid interpretation, that is, $c^{\mathcal{D}_i} = c^{\mathcal{D}_{i+1}}$, for all $i \ge 0$. We denote c's interpretation by $c^{\bar{\mathcal{D}}}$.

We call the indexes of the τ_is and $\mathcal{D}_i$s *time points* and the τ_is *timestamps*. In particular, τ_i is the timestamp at time point $i \in \mathbb{N}$. Note that there can be successive time points with equal timestamps. Furthermore, note that the relations $r^{\mathcal{D}_0}, r^{\mathcal{D}_1}, \ldots$ in a temporal structure $(\bar{\mathcal{D}}, \bar{\tau})$ corresponding to a predicate symbol $r \in R$ may change over time. In contrast, the interpretation of the constant symbols $c \in C$ and the domain of the $\mathcal{D}_i$s do not change over time.

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, a variable x, and $d \in |\bar{\mathcal{D}}|$, $v[x/d]$ is the valuation mapping x to d and leaving other variables' valuation unchanged.

The semantics of MFOTL, $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$, is given in Figure 2, 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. The temporal operators $\ominus_I$ ("previous"), $\bigcirc_I$ ("next"), S_I ("since"), and U_I ("until") allow us to express both quantitative and qualitative properties with respect to the ordering of elements in the relations of the $\mathcal{D}_i$s in the temporal structure $(\bar{\mathcal{D}}, \bar{\tau})$. Note that they are labeled with intervals I and a formula of the form $(\ominus_I \phi)$, $(\bigcirc_I \phi)$, $(\phi \mathsf{S}_I \psi)$, 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 timestamp τ_i.

Terminology and Notation. We omit parentheses where possible by using the standard conventions about the binding strengths of the logical connectives. For instance, Boolean operators bind stronger than temporal ones and unary operators bind stronger than binary ones. We use standard syntactic sugar such as $\diamondsuit_I \phi := \mathit{true}\ \mathsf{S}_I\ \phi$, $\Diamond_I \phi := \mathit{true}\ \mathsf{U}_I\ \phi$, $\boxminus_I \phi := \neg \diamondsuit_I \neg \phi$, and $\Box_I \phi := \neg \Diamond_I \neg \phi$, where $\mathit{true} := \exists x.\, x \approx x$. Intuitively, the formula $\diamondsuit_I \phi$ states that ϕ holds at some time point in the past within the time window I and the formula $\boxminus_I \phi$ states that ϕ holds at all time points in the past within the time window I. If the interval I includes zero, then the current time point is also considered. The corresponding future operators are $\Diamond_I$ and $\Box_I$. We also use non-metric operators like $\Box \phi := \Box_{[0,\infty)} \phi$. A formula ϕ is *bounded* if the interval I of every temporal operator U_I occurring in ϕ is finite. We use standard terminology like *atomic formula* and *subformula*.

2.2 Monitoring

We now illustrate our use of MFOTL and our monitoring algorithm for compliance checking [10]. Consider the simple policy stating that reports must have been approved within at most 10 time units before they are published:

$$\Box\, \forall x.\, \mathit{publish}(x) \rightarrow \diamondsuit_{[0,11)}\, \mathit{approve}(x)\,.$$

We assume that the actions for publishing and approving reports are logged in relations. Specifically, for each time point $i \in \mathbb{N}$, we have the unary relations $\mathit{PUBLISH}_i$ and $\mathit{APPROVE}_i$ such that (1) $f \in \mathit{PUBLISH}_i$ iff the report f is published at time point i and (2) $f \in \mathit{APPROVE}_i$ iff the report f is approved at time point i. Observe that there can be multiple approvals at the same time point for different reports. Furthermore, every time point i has a timestamp $\tau_i \in \mathbb{N}$.

Given a sequence of logged publishing and approval actions, the corresponding temporal structure $(\bar{\mathcal{D}}, \bar{\tau})$ with $\bar{\mathcal{D}} = (\mathcal{D}_0, \mathcal{D}_1, \ldots)$ and $\bar{\tau} = (\tau_0, \tau_1, \ldots)$ is as follows. The only predicate symbols in $\bar{\mathcal{D}}$'s signature are *publish* and *approve*, both of arity 1. We assume that every report is uniquely identified by a natural number. The domain of $\bar{\mathcal{D}}$ contains all these numbers, that is, $|\bar{\mathcal{D}}| = \mathbb{N}$. The ith structure in $\bar{\mathcal{D}}$ contains the relations $\mathit{PUBLISH}_i$ and $\mathit{APPROVE}_i$. The ith timestamp is simply τ_i, the time when these actions occurred.

To detect policy violations, our monitoring algorithm iteratively processes the temporal structure $(\bar{\mathcal{D}}, \bar{\tau})$ representing the stream of logged actions. This can be done offline or online. At each time point i, it outputs the valuations satisfying the negation of the formula $\psi = \mathit{publish}(x) \rightarrow \diamondsuit_{[0,11)}\, \mathit{approve}(x)$, which is $\neg\psi$ and equivalent to $\mathit{publish}(x) \wedge \boxminus_{[0,11)} \neg \mathit{approve}(x)$. Note that we drop the outermost quantifier since we are not only interested in whether the policy is violated but we also want to provide additional information about the reported violations, namely, the reports that were published and not approved within the specified time window.

In a nutshell, the monitoring algorithm works as follows. It iterates over the structures $\mathcal{D}_i$ and their associated timestamps τ_i, where i is initially 0 and is incremented with each iteration. At each iteration, the algorithm incrementally maintains a collection of finite auxiliary relations for previous time points. Roughly speaking, for each time point $j \le i$, these relations store the elements that satisfy the temporal subformulas of $\neg\psi$ at the time point j. If the temporal subformula of $\neg\psi$ refers to future time points, the algorithm might need to postpone the construction of such an auxiliary relation to a later iteration, until the processed prefix of $(\bar{\mathcal{D}}, \bar{\tau})$ is long enough to evaluate the subformula at time point j. The algorithm discards auxiliary relations whenever they become irrelevant for detecting further violations. The monitoring algorithm has been implemented in our tool MONPOLY [14].

In general, we assume that policies formalized in MFOTL are of the form $\Box\, \psi$, where ψ is bounded. Since ψ is bounded,

the monitor only needs to take into account a finite prefix of $(\bar{\mathcal{D}}, \bar{\tau})$ when determining the satisfying valuations of $\neg\psi$ at any given time point. To effectively determine all these valuations, we also assume here that predicate symbols have finite interpretations in $(\bar{\mathcal{D}}, \bar{\tau})$, that is, the relation $r^{\mathcal{D}_j}$ is finite, for every predicate symbol r and every $j \in \mathbb{N}$. Furthermore, we require that $\neg\psi$ can be rewritten to a temporal-subformula-domain-independent formula [10], a generalization of the standard notion of domain-independent database queries [17]. We refer to [10] for a detailed description of the monitoring algorithm. Additional algorithmic details are also presented in Appendix A, for the sake of completeness.

Note that the monitoring algorithm assumes a total ordering on the logged actions. However, a total ordering is not necessarily available in a distributed and concurrent system. Moreover, policy compliance may depend on such a total ordering. For example, consider the policy for publishing and approving reports and a system in which the publish and approval actions are performed and logged by different system parts. If two such corresponding actions are equally timestamped and when assuming an interleaving semantics of the system parts, then two orderings of the actions are possible: (i) the report is first approved and then published and (ii) the report is published before being approved. For the ordering (i) the policy is satisfied, while for (ii) it is violated in case there is no other approval within the specified time window.

3 Monitoring Concurrently Logged Actions

In this section, we first prove the intractability of monitoring when multiple log files are produced in a concurrent setting. We then motivate two solutions where only a single representative log is monitored. In Sections 4 and 5, we show when monitoring a single such log is sufficient. A comparison of the two solutions is given in Section 6.

Log Interleavings. Intuitively, an interleaving of logs preserves the ordering of the logged actions with respect to their timestamps, but allows for any possible ordering of actions with equal timestamps that are recorded by different log producers. To define an interleaving, for a function $f : X \to Y$, let $\mathrm{img}(f)$ denote the set $\{y \in Y \mid f(x) = y, \text{ for some } x \in X\}$. Furthermore, we assume in this section that all temporal structures have the same signature (C, R, ι), equal domains, and that constant symbols are equally interpreted. Note that any two temporal structures whose common constant symbols are equally interpreted can easily be extended so that their extensions fulfill this requirement.

Definition 3.1. *Let $(\bar{\mathcal{D}}^1, \bar{\tau}^1)$, $(\bar{\mathcal{D}}^2, \bar{\tau}^2)$, and $(\bar{\mathcal{D}}, \bar{\tau})$ be temporal structures. $(\bar{\mathcal{D}}, \bar{\tau})$ is an* interleaving *of $(\bar{\mathcal{D}}^1, \bar{\tau}^1)$ and $(\bar{\mathcal{D}}^2, \bar{\tau}^2)$ if there are strictly monotonic functions $f_1, f_2 : \mathbb{N} \to \mathbb{N}$ with*
(1) $\mathrm{img}(f_1) \cup \mathrm{img}(f_2) = \mathbb{N}$,
(2) $\mathrm{img}(f_1) \cap \mathrm{img}(f_2) = \emptyset$, *and*
(3) $\tau_i^k = \tau_{f_k(i)}$ *and* $r^{\mathcal{D}_i^k} = r^{\mathcal{D}_{f_k(i)}}$, *for all* $k \in \{1, 2\}$, $i \in \mathbb{N}$, $r \in R$.
We denote by $(\bar{\mathcal{D}}^1, \bar{\tau}^1) \between (\bar{\mathcal{D}}^2, \bar{\tau}^2)$ the set of interleavings *of the temporal structures $(\bar{\mathcal{D}}^1, \bar{\tau}^1)$ and $(\bar{\mathcal{D}}^2, \bar{\tau}^2)$.*

Since there are usually multiple interleavings of two temporal structures, we formulate policy violations with respect to a set of temporal structures.

Definition 3.2. *Let* **T** *be a set of temporal structures.*
(1) **T** weakly violates *the formula ϕ at time point $i \in \mathbb{N}$ if for some $(\bar{\mathcal{D}}, \bar{\tau}) \in \mathbf{T}$ and some valuation v, it holds that $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \not\models \phi$.*
(2) **T** strongly violates *the formula ϕ at time point $i \in \mathbb{N}$ if for all $(\bar{\mathcal{D}}, \bar{\tau}) \in \mathbf{T}$, there is some valuation v such that $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \not\models \phi$.*

Unfortunately, even in a propositional setting, determining whether the set of interleavings weakly or strongly violates a formula is intractable.

Theorem 3.3. *Let $(\bar{\mathcal{D}}^1, \bar{\tau}^1)$ and $(\bar{\mathcal{D}}^2, \bar{\tau}^2)$ be temporal structures, $i \in \mathbb{N}$, and ϕ a quantifier-free sentence with only Boolean and non-metric past operators that neither contains the equality symbol $\approx$ nor the ordering symbol $\prec$.*
1. Determining whether the set of interleavings $(\bar{\mathcal{D}}^1, \bar{\tau}^1) \between (\bar{\mathcal{D}}^2, \bar{\tau}^2)$ weakly violates ϕ at i is NP-complete.
2. Determining whether the set of interleavings $(\bar{\mathcal{D}}^1, \bar{\tau}^1) \between (\bar{\mathcal{D}}^2, \bar{\tau}^2)$ strongly violates ϕ at i is coNP-complete.

Note that both decision problems are well defined as ϕ does not contain future operators. We therefore only need to examine the finite prefixes with length $i+1$ of the interleavings to determine whether ϕ is weakly or strongly violated at the time point i.

We remark that related intractability results for LTL on so-called partially ordered traces are given in [18] and [19]. The setting in [18] is different from ours. In particular, it is unclear how to describe the set of interleavings of two timestamped traces using partially ordered traces as defined in [18]. Moreover, we reduce the checking of the satisfiability and validity of formulas in propositional logic, respectively, to the respective decision problem for proving its hardness. In [18], the global-predicate-detection decision problem [20] is used. The setting in [19] allows for arbitrary partial orders and hence could be used to describe the set of interleavings of two timestamped traces. The authors reduce the decision problem 3-SAT to the problem of determining whether all possible interleavings satisfy a formula.

Sufficient Logs. We first give conditions with respect to an arbitrary set of temporal structures for when it suffices to monitor a single temporal structure.

Definition 3.4. *The temporal structure $(\bar{\mathcal{C}}, \bar{\kappa})$ is* sufficient *for the formula ϕ on the set* **T** *of temporal structures if for all valuations v, the following conditions are fulfilled:*
(S1) If $(\bar{\mathcal{C}}, \bar{\kappa}, v, 0) \models \phi$ then $(\bar{\mathcal{D}}, \bar{\tau}, v, 0) \models \phi$, for all $(\bar{\mathcal{D}}, \bar{\tau}) \in \mathbf{T}$.
(S2) If $(\bar{\mathcal{C}}, \bar{\kappa}, v, 0) \not\models \phi$ then $(\bar{\mathcal{D}}, \bar{\tau}, v, 0) \not\models \phi$, for all $(\bar{\mathcal{D}}, \bar{\tau}) \in \mathbf{T}$.

Note that the actual ordering of actions logged with equal timestamps in a concurrent system cannot be known unless there is an additional mechanism to order these events. Instead of adding such mechanisms, we identify two classes of policies, which are indifferent to the ordering of equally timestamped actions, the *interleaving-sufficient* and *collapse-sufficient* policies. Formulas in both classes can be efficiently monitored by inspecting just a single temporal structure instead of all the possible interleavings. For both classes, the set **T** in Definition 3.4 is the set of all interleavings of

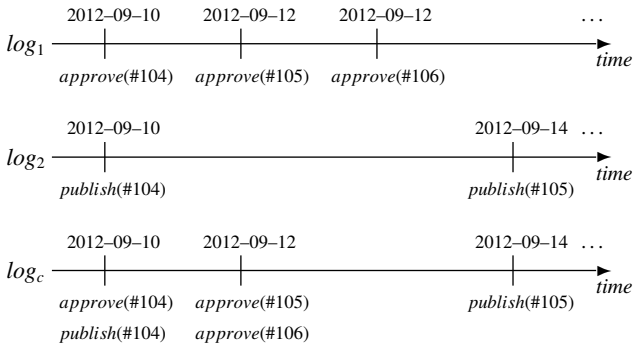

Fig. 3. Example of a Collapsed Interleaving, log_c, of the Temporal Structures log_1 and log_2

two temporal structures. For an interleaving-sufficient policy, inspecting an arbitrary interleaving is sufficient to determine whether the policy is strongly violated. With collapse-sufficient policies we exploit the inability to distinguish the ordering of events logged with equal timestamps to make monitoring more efficient and inspect the so-called collapse of an interleaving:

Definition 3.5. *Let* $(\bar{\mathcal{D}}, \bar{\tau})$ *and* $(\bar{\mathcal{C}}, \bar{\kappa})$ *be temporal structures.* $(\bar{\mathcal{C}}, \bar{\kappa})$ *is a* collapse *of* $(\bar{\mathcal{D}}, \bar{\tau})$ *if there is a monotonic surjective function* $f : \mathbb{N} \to \mathbb{N}$ *such that*

(1) if $\tau_i = \tau_j$ *then* $f(i) = f(j)$*, for all* $i, j \in \mathbb{N}$*,*
(2) $\kappa_{f(i)} = \tau_i$*, for all* $i \in \mathbb{N}$*, and*
(3) $r^{\mathcal{C}_j} = \bigcup_{i \in f^{-1}(j)} r^{\mathcal{D}_i}$*, for all* $j \in \mathbb{N}$ *and* $r \in R$*.*

Intuitively, the structures of the temporal structure $(\bar{\mathcal{D}}, \bar{\tau})$ with equal timestamps are collapsed into a single structure. Figure 3 depicts an example of collapsing. The collapse is uniquely defined and we denote it by $col(\bar{\mathcal{D}}, \bar{\tau})$. Furthermore, the collapses of temporal structures in the set of interleavings of two given temporal structures are all isomorphic. Note that the set of interleavings is strictly included in the set of collapse pre-images, that is, $(\bar{\mathcal{D}}, \bar{\tau}) \between (\bar{\mathcal{D}}', \bar{\tau}') \subsetneq col^{-1}(\bar{\mathcal{C}}, \bar{\kappa})$, where $(\bar{\mathcal{C}}, \bar{\kappa})$ is the collapse of an interleaving of the temporal structures $(\bar{\mathcal{D}}, \bar{\tau})$ and $(\bar{\mathcal{D}}', \bar{\tau}')$.

4 Monitoring an Interleaving

In this section we describe an interleaving-sufficient fragment. Intuitively, interleaving-sufficient formulas are those formulas that yield neither false positives nor false negatives when monitoring an interleaving. This is because they either satisfy all possible interleavings of two temporal structures or they violate all possible interleavings.

Definition 4.1. *Let* ϕ *be a formula. For* $k \in \{1, 2\}$*, we say that* ϕ has the property (Ik) *if* $(\bar{\mathcal{C}}, \bar{\kappa})$ *fulfills the condition (S*k*) in Definition 3.4 with respect to* ϕ *and* $(\bar{\mathcal{D}}, \bar{\tau}) \between (\bar{\mathcal{D}}', \bar{\tau}')$*, for every* $(\bar{\mathcal{D}}, \bar{\tau})$*,* $(\bar{\mathcal{D}}', \bar{\tau}')$*, and* $(\bar{\mathcal{C}}, \bar{\kappa})$*, where* $(\bar{\mathcal{C}}, \bar{\kappa})$ *is an interleaving of* $(\bar{\mathcal{D}}, \bar{\tau})$ *and* $(\bar{\mathcal{D}}', \bar{\tau}')$*. Moreover,* ϕ *is* interleaving-sufficient *if it has the properties (I1) and (I2).*

Note that we define interleaving-sufficiency only as a property of the formula. We could alternatively consider a refined notion that limits the interleavings on which the formula must hold. For example, if the relations of certain predicate symbols are logged by a single logging mechanism then we can impose conditions that the temporal structures $(\bar{\mathcal{D}}, \bar{\tau})$ and $(\bar{\mathcal{D}}', \bar{\tau}')$ in the above definition must fulfill. This would enlarge the set of interleaving-sufficient formulas. However, for the ease of exposition, we restrict ourselves here to the property as defined in Definition 4.1.

Monitoring an arbitrary interleaving with respect to an interleaving-sufficient formula is correct for strong violations. Since the formula has property (I2), violations found in $(\bar{\mathcal{C}}, \bar{\kappa})$ imply that the set of interleavings strongly violates the formula. The converse is ensured by the property (I1): if no violation is found in $(\bar{\mathcal{C}}, \bar{\kappa})$, then all interleavings are policy compliant. Furthermore, by monitoring $(\bar{\mathcal{C}}, \bar{\kappa})$ we also detect when the set of interleavings both weakly and strongly violates the given formula. The reason is that if a formula is strongly violated by a set of interleavings then it is also weakly violated, since the set of interleavings is always nonempty.

Example 4.2. *Recall the formula* $\Box \forall x.\, publish(x) \rightarrow \diamondsuit_{[0,11)}\, approve(x)$ *from the example in Section 2.2. It is not interleaving-sufficient. Suppose that a report* x *is published in* $(\bar{\mathcal{D}}^1, \bar{\tau}^1)$ *at time point* i*, that is,* $x \in publish^{\mathcal{D}^1_i}$ *and only approved in* $(\bar{\mathcal{D}}^2, \bar{\tau}^2)$ *at the equally timestamped time point* j*, that is,* $x \in approve^{\mathcal{D}^2_j}$ *with* $\tau^2_j = \tau^1_i$*. Then there is an interleaving* $(\bar{\mathcal{D}}, \bar{\tau}) \in (\bar{\mathcal{D}}^1, \bar{\tau}^1) \between (\bar{\mathcal{D}}^2, \bar{\tau}^2)$ *where the approval action comes (pointwise) strictly after the publish action. We cannot handle this formula correctly by monitoring just a single interleaving of the given temporal structures* $(\bar{\mathcal{D}}^1, \bar{\tau}^1)$ *and* $(\bar{\mathcal{D}}^2, \bar{\tau}^2)$*.*

A slightly stronger policy however can be efficiently monitored. Namely, the policy that requires that an approval action must happen timewise strictly before the publish action, that is, $\Box \forall x.\, publish(x) \rightarrow \diamondsuit_{[1,11)}\, approve(x)$*. This formula is interleaving-sufficient. Similarly,* $\Box \forall x.\, publish(x) \rightarrow \diamondsuit_{[0,1)} \diamondsuit_{[0,11)}\, approve(x)$ *is also interleaving-sufficient. It formalizes the slightly weaker policy where every publish action must be approved at a time point with a timestamp that is less than or equal to the timestamp of the time point when the publish action happens.*

Theorem 4.3. *Given an MFOTL formula* ϕ*, it is undecidable whether* ϕ *is interleaving-sufficient.*

Given undecidability, we proceed by providing sufficient conditions for ϕ being interleaving-sufficient. We do this by identifying a subset of formulas using a labeling algorithm.

Our algorithm labels the atomic subformulas of the given formula and propagates these labels bottom-up to the formula's root using a fixed set of labeling rules. We use two labels: ONE and ALL. They represent properties that capture the relationship between violations found in one interleaving and violations found in other interleavings. If a formula with the label ONE is satisfied at a time point in one interleaving, then the formula is also satisfied in all other interleavings at the corresponding time point. If a formula with the label ALL is satisfied at a time point with timestamp τ in one interleaving, then the formula is also satisfied in all other interleavings at all time points with the timestamp τ. We formally state these

$$\frac{\phi : \mathsf{ALL}}{\phi : \mathsf{ONE}}$$

$$\frac{}{t \approx t' : \mathsf{ALL}} \qquad \frac{}{t \prec t' : \mathsf{ALL}} \qquad \frac{}{r(t_1, \ldots, t_{\iota(r)}) : \mathsf{ONE}}$$

$$\frac{\phi : \mathsf{ONE}}{\neg\phi : \mathsf{ONE}} \qquad \frac{\phi : \mathsf{ALL}}{\neg\phi : \mathsf{ALL}} \qquad \frac{\phi : \mathsf{ONE} \quad \psi : \mathsf{ONE}}{\phi \vee \psi : \mathsf{ONE}} \qquad \frac{\phi : \mathsf{ALL} \quad \psi : \mathsf{ALL}}{\phi \vee \psi : \mathsf{ALL}}$$

$$\frac{\exists x.\, \phi : \mathsf{ONE}}{\exists x.\, \phi : \mathsf{ONE}} \qquad \frac{\exists x.\, \phi : \mathsf{ALL}}{\exists x.\, \phi : \mathsf{ALL}}$$

$$\frac{\phi : \mathsf{ALL} \quad \psi : \mathsf{ALL}}{\phi\, \mathsf{S}_I\, \psi : \mathsf{ALL}} \qquad \frac{\phi : \mathsf{ALL} \quad \psi : \mathsf{ALL}}{\phi\, \mathsf{U}_I\, \psi : \mathsf{ALL}}$$

$$\frac{\phi : \mathsf{ONE}}{\diamondsuit_I \phi : \mathsf{ALL}}\ 0 \notin I \qquad \frac{\phi : \mathsf{ONE}}{\Diamond_I \phi : \mathsf{ALL}}\ 0 \notin I \qquad \frac{\phi : \mathsf{ONE}}{\diamondsuit_I \Diamond_J \phi : \mathsf{ALL}} \qquad \frac{\phi : \mathsf{ONE}}{\Diamond_I \diamondsuit_J \phi : \mathsf{ALL}}$$

Fig. 4. Labeling Rules (Interleaving)

properties in the following definition.

Definition 4.4. *Let* $(\bar{\mathcal{D}}^1, \bar{\tau}^1)$ *and* $(\bar{\mathcal{D}}^2, \bar{\tau}^2)$ *be two temporal structures and* $(\bar{\mathcal{D}}, \bar{\tau})$ *and* $(\bar{\mathcal{D}}', \bar{\tau}')$ *be two arbitrary interleavings from the set* $(\bar{\mathcal{D}}^1, \bar{\tau}^1) \between (\bar{\mathcal{D}}^2, \bar{\tau}^2)$.

- *We say that the formula* ϕ has the property ONE *when the following holds: If* $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$*, for some valuation* v *and time point* $i \in \mathbb{N}$ *then* $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \models \phi$ *for the time point* $i' \in \mathbb{N}$*, where* i' *is the time point corresponding to* i*. That is, there are* $k \in \{1, 2\}$ *and* $j \in \mathbb{N}$ *with* $i = f_k(j)$ *and* $i' = f'_k(j)$ *with* f_1*,* f_2 *being the functions used in the interleaving* $(\bar{\mathcal{D}}, \bar{\tau})$*, and* f'_1*,* f'_2 *the functions used in the interleaving* $(\bar{\mathcal{D}}', \bar{\tau}')$.
- *We say that the formula* ϕ has the property ALL *when the following holds: If* $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$*, for some valuation* v *and time point* $i \in \mathbb{N}$ *then for all time points* $i' \in \mathbb{N}$ *with* $\tau_i = \tau'_{i'}$*, it holds that* $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \models \phi$.

We overload notation and identify each label with its corresponding property. The labeling is done using the rules in Figure 4. To improve readability, we use syntactic sugar in the rules. When applying the rules, we assume that syntactic sugar is unfolded in both the rules and the formula. Note that multiple rules may be applicable to a subformula. In this case, multiple labels may be assigned to the subformula. We use the notation $\phi : \ell$ as shorthand for "ϕ's label includes ℓ." By labeling the subformula bottom-up and by attempting to apply all rules before proceeding up to the next subformula, we ensure that a formula is assigned with all possible labels.

Lemma 4.5. *Let* ϕ *be a formula. If* ϕ *can be labeled with* ℓ*, then* ϕ *has the property* ℓ*, where* $\ell \in \{\mathsf{ONE}, \mathsf{ALL}\}$.

Lemma 4.5 shows the soundness of our labeling rules. Here, we explain just the most representative rules. The first line in Figure 4 shows the weakening rule. The property corresponding to the label ALL implies the property corresponding to ONE.

The next line shows rules for atomic formulas. An atomic formula $t \approx t'$ or $t \prec t'$ depends only on the valuation and therefore can be labeled ALL. An atomic formula of the form $r(t_1, \ldots, t_{\iota(r)})$ can be labeled ONE. If a predicate symbol is satisfied at some time point in an interleaving, then it is also satisfied at the corresponding time point in another interleaving. Hence, we label it with ONE. However, we cannot label it with ALL because we do not know whether it is satisfied at other time points with equal timestamps.

Next, we consider labeling rules for the temporal operator $\diamondsuit_I$. If the formula $\diamondsuit_I \phi$ is satisfied at time point i, then the subformula ϕ is satisfied at some time point $j \leq i$. If ϕ is labeled ONE, then in any other interleaving the time point corresponding to j also satisfies ϕ. However, if the time points i and j have equal timestamps, then their relative ordering can be exchanged in another interleaving. In this case, the formula $\diamondsuit_I \phi$ would not be satisfied at the time point corresponding to i.

If $0 \notin I$ then the time points i and j must have different timestamps. Therefore, their relative ordering cannot be changed in any interleaving. In this case, not only the time point corresponding to i satisfies the formula $\diamondsuit_I \phi$, but all time points with an equal timestamp as i satisfy this formula. We can therefore propagate the label ONE as ALL if $0 \notin I$, but cannot propagate it if $0 \in I$.

We also consider the case when the subformula ϕ is labeled ALL. In this case, all time points with an equal timestamp as j satisfy ϕ. But then, independent of the relative ordering of these time points, all time points with an equal timestamp as i satisfy $\diamondsuit_I \phi$. Hence, we can propagate ϕ's label ALL to $\diamondsuit_I \phi$ without any restrictions on I. The rule for ALL is not shown in Figure 4, but can be derived from the rule for the operator S_I after unfolding the syntactic sugar of $\diamondsuit_I \phi$ into $(\exists x.\, x \approx x)\mathsf{S}_I \phi$.

We can try to label a formula solely based on labeling rules that involve only a single Boolean or temporal operator. However, by using more specialized labeling rules like the one for $\diamondsuit_I \Diamond_J \psi$, we are more likely to succeed in propagating a label to the formula's root. Intuitively, with the nesting of the operators $\diamondsuit_I$ and $\Diamond_J$, the ordering of equally timestamped time points becomes irrelevant since, from a given time point, we can freely choose any of these time points that satisfy the metric constraints given by the intervals I and J. Hence, a labeling ONE for ψ allows us to label $\diamondsuit_I \Diamond_J \psi$ with ALL.

Finally, there are no labeling rules for the temporal operators $\ominus_I$ and $\bigcirc_I$ because these operators inherently rely on the relative ordering of time points.

Theorem 4.6. *Let* ϕ *be a formula.*
1. If ϕ *is labeled* ALL*, then* ϕ *is interleaving-sufficient.*
2. If ϕ *is labeled* ONE*, then* $\Box\phi$ *is interleaving-sufficient.*
Moreover, we can determine in linear time in the formula's length whether ϕ *can be labeled by* ONE *or* ALL.

Example 4.7. *We illustrate our algorithm by applying it to the formula* $\Box \forall x.\, publish(x) \rightarrow \diamondsuit_{[0,11)}\, approve(x)$*. We first remove some syntactic sugar and obtain the formula* $\Box \forall x.\, \neg publish(x) \vee \diamondsuit_{[0,11)}\, approve(x)$*. We start by labeling the atomic subformulas. Both* $publish(x)$ *and* $approve(x)$ *are labeled with* ONE*. Hence, the subformula* $\neg publish(x)$ *is also labeled with* ONE*. However, we cannot propagate the label* ONE *to the subformula* $\diamondsuit_{[0,11)}\, approve(x)$ *because the interval of the operator* $\diamondsuit$ *includes* 0*. We therefore cannot propagate any labels to the subformulas* $\neg publish(x) \vee \diamondsuit_{[0,11)}\, approve(x)$ *and* $\forall x.\, \neg publish(x) \vee \diamondsuit_{[0,11)}\, approve(x)$*. We conclude that the formula* $\Box \forall x.\, \neg publish(x) \vee \diamondsuit_{[0,11)}\, approve(x)$ *is not interleaving-sufficient, as explained in Example 4.2.*

The formula $\Box\forall x.\, publish(x) \rightarrow \Diamond_{[1,11)}\, approve(x)$ *is interleaving-sufficient. The labeling starts similarly but* $\Diamond_{[1,11)}\, approve(x)$ *can be labeled with* ALL *since the interval of the temporal operator does not contain* 0*. This label is weakened to* ONE *and propagates to the formula* $\forall x.\, \neg publish(x) \vee \Diamond_{[1,11)}\, approve(x)$*. We conclude that* $\Box\forall x.\, \neg publish(x) \vee \Diamond_{[1,11)}\, approve(x)$ *is interleaving-sufficient.*

Note that the defined fragment is sound, but incomplete. In particular, the converse of statements 1 and 2 in Theorem 4.6 is false. For example, the formula $\Box\forall x.\, publish(x) \rightarrow (\Diamond_{[0,1)}\, approve(x)) \vee \Diamond_{[0,11)}\, approve(x)$ is interleaving-sufficient, but cannot be labeled as required by Theorem 4.6. However, the semantically equivalent formula $\Box\forall x.\, publish(x) \rightarrow \Diamond_{[0,1)} \Diamond_{[0,11)}\, approve(x)$ is recognized as interleaving-sufficient by the rules in Figure 4. We could enlarge the fragment by adding rules that handle this particular formula. However, since it is undecidable whether a formula is interleaving-sufficient (Theorem 4.3) we cannot make the syntactically-defined fragment decidable, sound, and complete.

5 Monitoring the Collapse

In this section we describe a collapse-sufficient fragment. Intuitively, collapse-sufficient formulas are those formulas that do not yield false positives and false negatives when monitoring the collapse of an interleaving:

Definition 5.1. *Let* ϕ *be a formula. For* $k \in \{1, 2\}$*, we say that* ϕ has the property (Ck) *if* $(\bar{\mathcal{C}}, \bar{\kappa})$ *fulfills the condition (Sk) in Definition 3.4 with respect to* ϕ *and* $(\bar{\mathcal{D}}, \bar{\tau}) \between (\bar{\mathcal{D}}', \bar{\tau}')$*, for every* $(\bar{\mathcal{D}}, \bar{\tau})$*,* $(\bar{\mathcal{D}}', \bar{\tau}')$*, and* $(\bar{\mathcal{C}}, \bar{\kappa})$*, where* $(\bar{\mathcal{C}}, \bar{\kappa})$ *is the collapse of an interleaving of* $(\bar{\mathcal{D}}, \bar{\tau})$ *and* $(\bar{\mathcal{D}}', \bar{\tau}')$*. Moreover,* ϕ *is* collapse-sufficient *if it has the properties (C1) and (C2).*

Monitoring the collapse with respect to a collapse-sufficient formula is correct for strong violations. Since strong violations are trivially also weak violations, we detect some weak violations as well. However, we may miss violations that are weak, but not strong.

Note that the formula from the example in Section 2.2 is not collapse-sufficient, but the weaker and stronger formulas from Example 4.2 are collapse-sufficient. Also note that stutter-invariance [21] is a necessary condition for collapse-sufficiency. However, it is not a sufficient condition. For example, the formula $\Box\forall x.\, p(x) \wedge q(x)$ is stuttering-invariant but not collapse-sufficient.

As with interleaving-sufficient formulas, it is undecidable whether a formula is collapse-sufficient, as stated in Theorem 5.2.

Theorem 5.2. *Given an MFOTL formula* ϕ*, it is undecidable whether* ϕ *is collapse-sufficient.*

Our collapse-sufficient fragment is, similar to the interleaving-sufficient fragment in Section 4, defined by a labeling algorithm. The labels represent properties, which capture the relation between violations found in a collapsed temporal structure at some time point and violations found in pre-images of the collapsing at a time point with an equal timestamp. We formally state these properties in the following definition.

Definition 5.3. *Let* $(\bar{\mathcal{C}}, \bar{\kappa})$ *be a collapsed temporal structure and let* $col^{-1}(\bar{\mathcal{C}}, \bar{\kappa})$ *denote the pre-images of collapsing, that is, the set of temporal structures* $(\bar{\mathcal{D}}, \bar{\tau})$ *with* $col(\bar{\mathcal{D}}, \bar{\tau}) = (\bar{\mathcal{C}}, \bar{\kappa})$*.*

- *The formula* ϕ has the property ($\models\forall$) *when the following holds: For all valuations* v *and all* $i \in \mathbb{N}$*, if* $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \phi$ *then for every* $(\bar{\mathcal{D}}, \bar{\tau}) \in col^{-1}(\bar{\mathcal{C}}, \bar{\kappa})$ *and every* $j \in \mathbb{N}$ *with* $\kappa_i = \tau_j$*, it holds that* $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \phi$*.*
- *The formula* ϕ has the property ($\models\exists$) *when the following holds: For all valuations* v *and all* $i \in \mathbb{N}$*, if* $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \phi$ *then for every* $(\bar{\mathcal{D}}, \bar{\tau}) \in col^{-1}(\bar{\mathcal{C}}, \bar{\kappa})$*, there is some* $j \in \mathbb{N}$ *with* $\kappa_i = \tau_j$ *such that* $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \phi$*.*
- *The formula* ϕ has the property ($\not\models\forall$) *when the following holds: For all valuations* v *and all* $i \in \mathbb{N}$*, if* $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \not\models \phi$ *then for every* $(\bar{\mathcal{D}}, \bar{\tau}) \in col^{-1}(\bar{\mathcal{C}}, \bar{\kappa})$ *and every* $j \in \mathbb{N}$ *with* $\kappa_i = \tau_j$*, it holds that* $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \not\models \phi$*.*
- *The formula* ϕ has the property ($\not\models\exists$) *when the following holds: For all valuations* v *and all* $i \in \mathbb{N}$*, if* $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \not\models \phi$ *then for every* $(\bar{\mathcal{D}}, \bar{\tau}) \in col^{-1}(\bar{\mathcal{C}}, \bar{\kappa})$*, there is some* $j \in \mathbb{N}$ *with* $\kappa_i = \tau_j$ *such that* $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \not\models \phi$*.*

The first symbol ($\models$ or $\not\models$) in a property indicates whether the formula is satisfied in the collapsed temporal structure $(\bar{\mathcal{C}}, \bar{\kappa})$. The second symbol ($\exists$ or $\forall$) states whether the formula is satisfied at all equally timestamped time points or at some equally timestamped time point in all temporal structures $(\bar{\mathcal{D}}, \bar{\tau}) \in col^{-1}(\bar{\mathcal{C}}, \bar{\kappa})$.

Again, we overload notation and identify each label with its corresponding property. Figure 5 lists the labeling rules. In addition, Figure 6 lists rules for the Boolean operator $\wedge$, the quantifier $\forall$, and the temporal operators trigger T_I and release R_I. These rules are used for formulas in positive normal form, which we require in Section 6. Recall that formulas in this normal form are obtained by pushing negation inside until it appears only in front of atomic formulas. When considering these formulas, the operators $\wedge$, $\forall$, T_I, and R_I are seen as primitives, instead of being defined as syntactic sugar. We recall that $\psi \mathsf{T}_I \chi$ abbreviates $\neg(\neg\psi \mathsf{S}_I \neg\chi)$ and $\psi \mathsf{R}_I \chi$ abbreviates $\neg(\neg\psi \,\mathsf{U}_I\, \neg\chi)$.

Lemma 5.4. *Let* ϕ *be a formula. If* ϕ *can be labeled with* ℓ*, then* ϕ *has the property* ℓ*, where* $\ell \in \{(\models\forall), (\not\models\forall), (\not\models\exists), (\models\exists)\}$*.*

Lemma 5.4 shows the soundness of our labeling rules. Here, we explain just the most representative rules. The first two rules in Figure 5 express that the properties corresponding to the labels ($\models\forall$) and ($\not\models\forall$) imply the properties corresponding to ($\models\exists$) and ($\not\models\exists$), respectively.

The next two lines in Figure 5 are rules for atomic formulas. An atomic formula $t \approx t'$ or $t \prec t'$ depends only on the valuation and therefore can be labeled ($\models\forall$) and ($\not\models\forall$). An atomic formula of the form $r(t_1, \ldots, t_{\iota(r)})$ can be labeled ($\models\exists$) and ($\not\models\forall$). We only explain the labeling ($\models\exists$). The explanation for the label ($\not\models\forall$) is analogous. The interpretation of a predicate symbol in a collapsed temporal structure $(\bar{\mathcal{C}}, \bar{\kappa})$ at a time point i is the union of the predicate symbol's interpretations at all time points j in a temporal structure $(\bar{\mathcal{D}}, \bar{\tau}) \in col^{-1}(\bar{\mathcal{C}}, \bar{\kappa})$ for which τ_j equals κ_i. Therefore, if $\bar{a} \in r^{\mathcal{C}_i}$ then $\bar{a} \in r^{\mathcal{D}_j}$, for some $j \in \mathbb{N}$

$$\frac{\phi : (\models \forall)}{\phi : (\models \exists)} \qquad \frac{\phi : (\not\models \forall)}{\phi : (\not\models \exists)}$$

$$\frac{}{t \approx t' : (\models \forall)} \qquad \frac{}{t \approx t' : (\not\models \forall)} \qquad \frac{}{t \prec t' : (\models \forall)} \qquad \frac{}{t \prec t' : (\not\models \forall)}$$

$$\frac{}{r(t_1, \ldots, t_{\iota(r)}) : (\models \exists)} \qquad \frac{}{r(t_1, \ldots, t_{\iota(r)}) : (\not\models \forall)}$$

$$\frac{\psi : (\models \exists)}{\neg\psi : (\not\models \exists)} \qquad \frac{\psi : (\models \forall)}{\neg\psi : (\not\models \forall)} \qquad \frac{\psi : (\not\models \exists)}{\neg\psi : (\models \exists)} \qquad \frac{\psi : (\not\models \forall)}{\neg\psi : (\models \forall)}$$

$$\frac{\psi : (\not\models \forall) \quad \chi : (\not\models \forall)}{\psi \vee \chi : (\not\models \forall)} \qquad \frac{\psi : (\not\models \forall) \quad \chi : (\not\models \exists)}{\psi \vee \chi : (\not\models \exists)}$$

$$\frac{\psi : (\models \forall) \quad \chi : (\models \forall)}{\psi \vee \chi : (\models \forall)} \qquad \frac{\psi : (\models \exists) \quad \chi : (\models \exists)}{\psi \vee \chi : (\models \exists)}$$

$$\frac{\psi : (\models \forall)}{\exists x.\, \psi : (\models \forall)} \qquad \frac{\psi : (\models \exists)}{\exists x.\, \psi : (\models \exists)} \qquad \frac{\psi : (\not\models \forall)}{\exists x.\, \psi : (\not\models \forall)} \qquad \frac{\psi : (\not\models \exists)}{\exists x.\, \psi : (\not\models \exists)}$$

$$\frac{\psi : (\models \forall) \quad \chi : (\models \forall)}{\psi\ \mathsf{S}_I\ \chi : (\models \forall)} \qquad \frac{\psi : (\not\models \forall) \quad \chi : (\not\models \forall)}{\psi\ \mathsf{S}_I\ \chi : (\not\models \forall)} \qquad \frac{\psi : (\not\models \exists) \quad \chi : (\not\models \forall)}{\psi\ \mathsf{S}_I\ \chi : (\not\models \exists)}$$

$$\frac{\psi : (\models \forall) \quad \chi : (\models \forall)}{\psi\ \mathsf{U}_I\ \chi : (\models \forall)} \qquad \frac{\psi : (\not\models \forall) \quad \chi : (\not\models \forall)}{\psi\ \mathsf{U}_I\ \chi : (\not\models \forall)} \qquad \frac{\psi : (\not\models \exists) \quad \chi : (\not\models \forall)}{\psi\ \mathsf{U}_I\ \chi : (\not\models \exists)}$$

$$\frac{\psi : (\not\models \exists) \quad \chi : (\not\models \forall)}{(\psi\ \mathsf{S}_I\ \chi) \wedge (\Box_J\, \psi) : (\not\models \forall)}\ 0 \notin I, 0 \in J \qquad \frac{\psi : (\not\models \exists) \quad \chi : (\not\models \forall)}{(\psi\ \mathsf{U}_I\ \chi) \wedge (\Box_J\, \psi) : (\not\models \forall)}\ 0 \notin I, 0 \in J$$

$$\frac{\psi : (\models \exists)}{\diamondsuit_I\, \psi : (\models \exists)} \qquad \frac{\psi : (\models \exists)}{\diamondsuit_I\, \psi : (\models \forall)}\ 0 \notin I \qquad \frac{\psi : (\models \exists)}{\Diamond_I\, \psi : (\models \exists)} \qquad \frac{\psi : (\models \exists)}{\Diamond_I\, \psi : (\models \forall)}\ 0 \notin I$$

$$\frac{\psi : (\models \exists)}{\diamondsuit_I \Diamond_J\, \psi : (\models \forall)}\ 0 \in I \cap J \qquad \frac{\psi : (\models \exists)}{\Diamond_I \diamondsuit_J\, \psi : (\models \forall)}\ 0 \in I \cap J$$

Fig. 5. Labeling Rules (Collapse)

$$\frac{\psi : (\models \forall) \quad \chi : (\models \forall)}{\psi \wedge \chi : (\models \forall)} \qquad \frac{\psi : (\models \forall) \quad \chi : (\models \exists)}{\psi \wedge \chi : (\models \exists)}$$

$$\frac{\psi : (\not\models \forall) \quad \chi : (\not\models \forall)}{\psi \wedge \chi : (\not\models \forall)} \qquad \frac{\psi : (\not\models \exists) \quad \chi : (\not\models \exists)}{\psi \wedge \chi : (\not\models \exists)}$$

$$\frac{\psi : (\models \forall)}{\forall x.\, \psi : (\models \forall)} \qquad \frac{\psi : (\models \exists)}{\forall x.\, \psi : (\models \exists)} \qquad \frac{\psi : (\not\models \forall)}{\forall x.\, \psi : (\not\models \forall)} \qquad \frac{\psi : (\not\models \exists)}{\forall x.\, \psi : (\not\models \exists)}$$

$$\frac{\psi : (\not\models \forall) \quad \chi : (\not\models \forall)}{\psi\ \mathsf{T}_I\ \chi : (\not\models \forall)} \qquad \frac{\psi : (\models \forall) \quad \chi : (\models \forall)}{\psi\ \mathsf{T}_I\ \chi : (\models \forall)}$$

$$\frac{\psi : (\models \exists) \quad \chi : (\models \forall)}{\psi\ \mathsf{T}_I\ \chi : (\models \exists)} \qquad \frac{\psi : (\models \exists) \quad \chi : (\models \forall)}{(\psi\ \mathsf{T}_I\ \chi) \vee (\Diamond_J\, \psi) : (\models \forall)}\ 0 \notin I, 0 \in J$$

$$\frac{\psi : (\not\models \forall) \quad \chi : (\not\models \forall)}{\psi\ \mathsf{R}_I\ \chi : (\not\models \forall)} \qquad \frac{\psi : (\models \forall) \quad \chi : (\models \forall)}{\psi\ \mathsf{R}_I\ \chi : (\models \forall)}$$

$$\frac{\psi : (\models \exists) \quad \chi : (\models \forall)}{\psi\ \mathsf{R}_I\ \chi : (\models \exists)} \qquad \frac{\psi : (\models \exists) \quad \chi : (\models \forall)}{(\psi\ \mathsf{R}_I\ \chi) \vee (\diamondsuit_J\, \psi) : (\models \forall)}\ 0 \notin I, 0 \in J$$

Fig. 6. Labeling Rules for Formulas in Positive Normal Form

with $\tau_j = \kappa_i$. Note that $\bar{a} \in r^{\bar{\mathcal{D}}_j}$ does not necessarily hold for all of these js; hence, we cannot label $r(t_1, \ldots, t_{\iota(r)})$ with $(\models \forall)$.

We next consider the labeling rules for the temporal operator S_I. To ease our explanation, we just consider the special case $\diamondsuit_I\, \psi = \mathit{true}\ \mathsf{S}_I\, \psi$. We first justify the rule that propagates the label $(\models \forall)$ from ψ to $\diamondsuit_I\, \psi$. It is not shown in Figure 5, but can be derived from the rule for the operator S_I after unfolding the syntactic sugar $\diamondsuit_I\, \phi$, by observing that *true* can be labeled with $(\models \forall)$. If $\diamondsuit_I\, \psi$ is satisfied in the collapsed temporal structure $(\bar{\mathcal{C}}, \bar{\kappa})$ at time point i then ψ is satisfied at some previous time point $j \leq i$ in $(\bar{\mathcal{C}}, \bar{\kappa})$ with $\kappa_i - \kappa_j \in I$. Because ψ is labeled with $(\models \forall)$, all time points with timestamp κ_j in the temporal structure $(\bar{\mathcal{D}}, \bar{\tau}) \in \mathit{col}^{-1}(\bar{\mathcal{C}}, \bar{\kappa})$ also satisfy ψ, and hence, all time points with timestamp κ_i satisfy $\diamondsuit_I\, \psi$ in $(\bar{\mathcal{D}}, \bar{\tau})$. When ψ is labeled with $(\models \exists)$, possibly only a single time point k in $(\bar{\mathcal{D}}, \bar{\tau})$ with $\tau_k = \kappa_j$ satisfies ψ. If $0 \in I$ then $\diamondsuit_I\, \psi$ might not be satisfied at time points before k, even if these time points have the timestamp κ_i. So, we can label $\diamondsuit_I$ with $(\models \exists)$ but not with $(\models \forall)$. However, if $0 \notin I$ then ψ is satisfied in $(\bar{\mathcal{C}}, \bar{\kappa})$ at a time point j with the timestamp $\kappa_j < \kappa_i$. Hence $\diamondsuit_I\, \psi$ is satisfied in $(\bar{\mathcal{D}}, \bar{\tau})$ at all time points with the timestamp κ_i. This allows us to label $\diamondsuit_I\, \psi$ with $(\models \forall)$. Finally, when ψ is labeled $(\not\models \forall)$, then $\diamondsuit_I\, \psi$ can also be labeled with $(\not\models \forall)$. This rule is not shown in Figure 5, but it can be derived from the rule for the operator S_I, like the rule for the label $(\models \forall)$. If $\diamondsuit_I\, \psi$ is violated in the collapsed temporal structure $(\bar{\mathcal{C}}, \bar{\kappa})$ at timestamp κ_i, then ψ is violated at all previous points in the temporal structure $(\bar{\mathcal{D}}, \bar{\tau}) \in \mathit{col}^{-1}(\bar{\mathcal{C}}, \bar{\kappa})$ that satisfy the metric constraints given by I. But then $\diamondsuit_I\, \psi$ is also violated in $(\bar{\mathcal{D}}, \bar{\tau})$ at all time points with the timestamp κ_i. Hence we can label $\diamondsuit_I\, \psi$ with $(\not\models \forall)$.

We can try to label a formula solely based on labeling rules that involve only a single Boolean or temporal operator. However, with additional specialized labeling rules like the one for $\diamondsuit_I \Diamond_J\, \psi$, we are more likely to succeed in propagating labels to the root of the formula. Intuitively, with the nesting of the operators $\diamondsuit_I$ and $\Diamond_J$, and when $0 \in I \cap J$, the ordering of equally timestamped time points becomes irrelevant since from a given time point, we can freely choose any of the time points that satisfy the metric constraints given by the intervals I and J. Hence, a labeling $(\models \exists)$ for ψ allows us to label $\diamondsuit_I \Diamond_J\, \psi$ with $(\models \forall)$.

Based on the labels at a formula's root, we can determine if the formula has the property (C1) or (C2). The conclusions we can draw are stated in the following lemma, which follows from the soundness of the labeling rules.

Lemma 5.5. *Let ϕ be a formula.*

1. *If ϕ can be labeled by $(\models \forall)$ then ϕ has the property (C1).*
2. *If ϕ can be labeled by $(\not\models \forall)$ then ϕ has the property (C2).*
3. *If ϕ can be labeled by $(\models \exists)$ then $\Diamond\, \phi$ has the property (C1).*
4. *If ϕ can be labeled by $(\not\models \exists)$ then $\Box\, \phi$ has the property (C2).*

Based on this lemma, we obtain the following theorem.

Theorem 5.6. *If the formula ϕ can be labeled by $(\models \forall)$ and $(\not\models \forall)$, then it is collapse-sufficient. Moreover, we can determine in linear time in the formula's length whether ϕ can be labeled by $(\models \forall)$, $(\models \exists)$, $(\not\models \forall)$, or $(\not\models \exists)$.*

Note that formulas of the form $\Box\,\psi$ are already collapse-sufficient if ψ can be labeled by $(\not\models\exists)$ and $\Box\,\psi$ can be labeled by $(\models\forall)$. Even if only one of these labellings can be derived, monitoring $\Box\,\psi$ on the collapsed temporal structure of an interleaving is still useful. For example, if ψ is labeled by $(\not\models\exists)$ then violations that are found on the collapsed temporal structure relate to strong violations on the set of interleavings. However, we might miss some violations.

Example 5.7. *We illustrate our algorithm by applying it to the formulas from Example 4.2. Unlike in Example 4.2, we use the labels* $(\models\forall)$, $(\models\exists)$, $(\not\models\forall)$, *and* $(\not\models\exists)$ *and the rules shown in Figure 5. First, we consider the formula* $\Box\,\forall x.\,\neg publish(x) \vee \diamondsuit_{[0,11)}\, approve(x)$. *Both of its atomic subformulas* $publish(x)$ *and* $approve(x)$ *are labeled with* $(\models\exists)$ *and* $(\not\models\forall)$. *We label the subformula* $\diamondsuit_{[0,11)}\, approve(x)$ *with* $(\models\exists)$ *and* $(\not\models\forall)$. *We cannot label it with* $(\models\forall)$ *since the interval contains* 0. *The subformula* $\neg publish(x)$ *is labeled with* $(\not\models\exists)$ *and* $(\models\forall)$. *The subformulas* $\neg publish(x) \vee \diamondsuit_{[0,11)}\, approve(x)$ *and* $\forall x.\,\neg publish(x) \vee \diamondsuit_{[0,11)}\, approve(x)$ *are labeled* $(\models\exists)$ *and* $(\not\models\exists)$. *We conclude that the formula* $\Box\,\forall x.\,\neg publish(x) \vee \diamondsuit_{[0,11)}\, approve(x)$ *has the property (C2). It does not have the property (C1), as explained in Example 4.2.*

The formula $\Box\,\forall x.\, publish(x) \rightarrow \diamondsuit_{[1,11)}\, approve(x)$ *has both properties (C1) and (C2). The labeling starts similarly but* $\diamondsuit_{[1,11)}\, approve(x)$ *is additionally labeled with* $(\models\forall)$ *since the interval of the temporal operator does not contain* 0. *This label propagates to the formula's root. We conclude that* $\Box\,\forall x.\,\neg publish(x) \vee \diamondsuit_{[1,11)}\, approve(x)$ *also has property (C1).*

As in the interleaving-sufficient case, the defined fragment is incomplete, which is again witnessed by the formula $\Box\,\forall x.\, publish(x) \rightarrow (\diamondsuit_{[0,1)}\, approve(x)) \vee \diamondsuit_{[0,11)}\, approve(x)$. It is collapse-sufficient, but cannot be labeled as required by Theorem 5.6. Note that the semantically equivalent formula $\Box\,\forall x.\, publish(x) \rightarrow \diamondsuit_{[0,1)}\, \diamondsuit_{[0,11)}\, approve(x)$ is recognized as collapse-sufficient using the rules shown in Figure 5.

6 Sufficient Fragments

In this section, we compare the interleaving-sufficient and collapse-sufficient fragments and present a generic recipe that approximates policies to obtain formulas in these fragments.

6.1 Comparison

The interleaving-sufficient fragment is larger than the collapse-sufficient fragment. In contrast, the collapse-sufficient fragment is more efficient to monitor. We explain these two aspects in more detail.

Intuitively, a collapse-sufficient formula is satisfied either on all pre-images of a collapse or on none. An interleaving-sufficient formula is satisfied either on all interleavings or on none. As the set of all interleavings is a strict subset of all collapse pre-images, the interleaving-sufficient property is a weaker requirement than the collapse-sufficient property. Theorem 6.1 shows that a collapse-sufficient formula is indeed always also interleaving-sufficient.

Theorem 6.1. *If a formula* ϕ *is collapse-sufficient then* ϕ *is also interleaving-sufficient.*

The converse does not hold. There are interleaving-sufficient formulas that are not collapse-sufficient. Intuitively, the interleaving-sufficient formulas allow us to check individual time points from the original traces, but collapse-sufficient formulas are restricted to checking the collapsed time points. For example, the policy that if p happens, then q must happen at the same time point, formalized as $\Box\, p \rightarrow q$, is interleaving-sufficient, but not collapse-sufficient. Inspecting only collapsed time points still allows us to check that an event never occurs. That is, $\Box\,\neg p$ is collapse-sufficient. However, $\Box\, p$ is interleaving-sufficient but not collapse-sufficient.

A practical advantage of the collapse-sufficient fragment is that monitoring a collapsed temporal structure is more efficient than monitoring an interleaving, as our case study described in Section 7 demonstrates. The main reason is that time points with equal timestamps are merged to a single time point in a collapsed temporal structure. Hence, the monitor processes the logged actions with equal timestamps in a single invocation.

The structure of the collapsed log can be further exploited to increase monitor performance by rewriting the monitored formulas. In particular, if we are monitoring the collapsed log, rather than an interleaving, then we can rewrite formulas of the form $\diamondsuit_{[0,1)}\,\phi$, $\diamondsuit_{[0,1)}\,\phi$, $\Box_{[0,1)}\,\phi$, and $\boxminus_{[0,1)}\,\phi$ to ϕ. The reason is that in a collapsed trace, there is at most one time point for each timestamp. We call the rewritten formulas *collapse-optimized*.

6.2 Policy Approximation

In Example 4.2, we have seen that we can obtain an interleaving-sufficient policy by strengthening or weakening the original policy. We now generalize this observation.

Let ϕ be a formula in positive normal form. That is, negations in ϕ are pushed inside and occur only in front of atomic formulas. We obtain a *weakened* formula ϕ^w by replacing each atomic subformula $r(t_1, \ldots, t_{\iota(r)})$ that occurs positively in ϕ by $\diamondsuit_I\, \diamondsuit_{I'}\, r(t_1, \ldots, t_{\iota(r)})$, for some intervals I and I' with $0 \in I \cap I'$. Analogously, in a *strengthened* formula ϕ^s, we replace each negative occurrence of an atomic subformula $r(t_1, \ldots, t_{\iota(r)})$ by $\diamondsuit_I\, \diamondsuit_{I'}\, r(t_1, \ldots, t_{\iota(r)})$ for some intervals I, I'.

Theorem 6.2. *Let* ϕ^w *and* ϕ^s *be weakened and strengthened formulas of the formula* ϕ *in positive normal form. The formulas* $\phi \rightarrow \phi^w$ *and* $\phi^s \rightarrow \phi$ *are valid. Moreover,*

1. *if* ϕ^s *is collapse-sufficient then* ϕ *has property (C1), and*
2. *if* ϕ^w *is collapse-sufficient then* ϕ *has property (C2).*

Weakened and strengthened formulas are more likely to be collapse-sufficient, since their subformulas of the form $\diamondsuit_I\, \diamondsuit_{I'}\, r(t_1, \ldots, t_{\iota(r)})$ can be labeled with $(\models\forall)$, while $r(t_1, \ldots, t_{\iota(r)})$ can only be labeled with the weaker label $(\models\exists)$. Simultaneously weakening and strengthening always results in a collapse-sufficient formula. However, the resulting formula does not necessarily relate to the original formula.

Since a collapse-sufficient formula is also interleaving-sufficient (Theorem 6.1), the above rewriting can also be used when an interleaving-sufficient formula is desired.

Note that by inserting the temporal operators $\diamondsuit_{[0,1)}$ and $\diamondsuit_{[0,1)}$ around positively occurring atomic subformulas, the ordering of equally timestamped actions becomes irrelevant. This

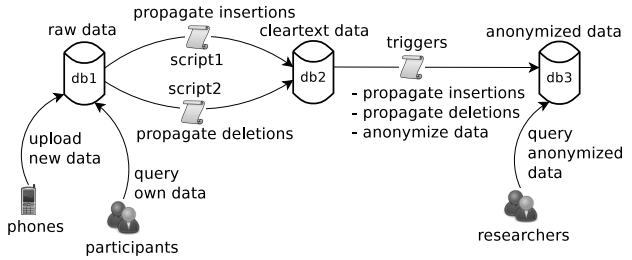

Fig. 7. Nokia's Data-collection Campaign

is desirable in systems where the clocks used to timestamp the actions are synchronized but too coarse-grained to capture the relative ordering of events occurring almost concurrently. Taking this idea further, by putting temporal operators $\diamondsuit_{[0,b)}$ and $\diamondsuit_{[0,b)}$ around these subformulas with $b \geq 1$, we take into account that the timestamps in a temporal structure are inaccurate and might differ from their actual value by the threshold b—a situation that occurs in practice.

7 Practical Experience

In this section, we describe the deployment of our monitoring approach within Nokia's Data-collection Campaign [16], which is a real-world application with realistic data-usage policies. Furthermore, we report on the monitor's performance and our findings.

7.1 Nokia's Data-collection Campaign

Scenario. The campaign collects contextual information from cell phones of about 180 participants. This sensitive data includes phone locations, call and SMS information, and the like. The data collected by a participant's phone is propagated into the databases db1, db2, and db3. The phones use WLAN to periodically upload their data to database db1. Every night, the synchronization script script1 copies the data from db1 to db2. Furthermore, triggers running on db2 anonymize and copy the data to db3, where researchers can access and analyze the anonymized data. The participants can access and delete their own data using a web interface to db1. Deletions are propagated to all databases: from db1 to db2 by the synchronization script script2, which also runs every night, and from db2 to db3 by database triggers. Figure 7 summarizes the various data usages.

Within the campaign, data is organized by records and can easily be identified. When uploading data from a phone into db1, a unique identifier is generated for each record. This identifier, together with an identifier of the participant who contributed the data, is attached to the record.

Policies. The collected data is subject to various policies that protect the participants' privacy. For example, there are access-control policies and policies governing the process of propagating the data between databases. In particular, insertions and deletions of data must be propagated within a given time limit. Furthermore, the latest version of the synchronization scripts must be used and their running times are restricted. Finally, access to the databases is restricted to selected user accounts and the account used by the script script1 may be used only while the script is running.

TABLE 1. Policy Formalizations in MFOTL

policy	MFOTL formalization
delete	$\square\, \forall user.\, \forall data.\, delete(user, \mathsf{db2}, data) \rightarrow user \approx \mathsf{script2}$
insert	$\square\, \forall user.\, \forall data.\, insert(user, \mathsf{db2}, data) \rightarrow user \approx \mathsf{script1}$
select	$\square\, \forall user.\, \forall data.\, select(user, \mathsf{db2}, data) \rightarrow$ $user \approx \mathsf{script1} \vee user \approx \mathsf{script2} \vee user \approx \mathsf{triggers}$
update	$\square\, \forall user.\, \forall data.\, \neg update(user, \mathsf{db2}, data)$
script1	$\square\, \forall db.\, \forall data.\, select(\mathsf{script1}, db, data) \vee insert(\mathsf{script1}, db, data) \vee$ $delete(\mathsf{script1}, db, data) \vee update(\mathsf{script1}, db, data) \rightarrow$ $((\neg \diamondsuit_{[0,1s)} \diamondsuit_{[0,1s)} end(\mathsf{script1}))\ \mathsf{S}\ (\diamondsuit_{[0,1s)} \diamondsuit_{[0,1s)} start(\mathsf{script1}))) \vee$ $\diamondsuit_{[0,1s)} \diamondsuit_{[0,1s)} end(\mathsf{script1})$
runtime	$\square\, \forall script.\, start(script) \rightarrow$ $(\neg \diamondsuit_{[0,1s)} \diamondsuit_{[0,1s)} end(script)) \wedge \diamondsuit_{[1s,6h)} end(script)$
svn	$\square\, \forall script.\, start(script) \rightarrow$ $\diamondsuit_{[0,1s)} \diamondsuit_{[0,10s)} \exists url.\, \exists rev.\, svn(script, \mathsf{latest}, url, rev)$
svn2	$\square\, \forall script.\, \forall status.\, \forall url.\, \forall rev.\, svn(script, status, url, rev) \rightarrow$ $\boxminus_{[1s,\infty)} (commit(url, rev') \rightarrow rev' \leq rev)$
ins-1-2	$\square\, \forall user.\, \forall data.\, insert(user, \mathsf{db1}, data) \wedge data \not\approx \mathsf{unknown} \rightarrow$ $\diamondsuit_{[0,1s)} \diamondsuit_{[0,30h]} \exists user'.\, insert(user', \mathsf{db2}, data) \vee$ $delete(user', \mathsf{db1}, data)$
ins-2-3	$\square\, \forall user.\, \forall data.\, insert(user, \mathsf{db2}, data) \wedge data \not\approx \mathsf{unknown} \rightarrow$ $\diamondsuit_{[0,1s)} \diamondsuit_{[0,60s)} \exists user'.\, insert(user', \mathsf{db3}, data)$
ins-3-2	$\square\, \forall user.\, \forall data.\, insert(user, \mathsf{db3}, data) \wedge data \not\approx \mathsf{unknown} \rightarrow$ $\diamondsuit_{[0,60s)} \diamondsuit_{[0,1s)} \exists user'.\, insert(user', \mathsf{db2}, data)$
del-1-2	$\square\, \forall user.\, \forall data.\, delete(user, \mathsf{db1}, data) \wedge data \not\approx \mathsf{unknown} \rightarrow$ $(\diamondsuit_{[0,1s)} \diamondsuit_{[0,30h)} \exists user'.\, delete(user', \mathsf{db2}, data)) \vee$ $((\diamondsuit_{[0,1s)} \diamondsuit_{[0,30h)} \exists user'.\, insert(user', \mathsf{db1}, data)) \wedge$ $(\boxminus_{[0,30h)} \square_{[0,30h)} \neg \exists user'.\, insert(user', \mathsf{db2}, data)))$
del-2-3	$\square\, \forall user.\, \forall data.\, delete(user, \mathsf{db2}, data) \wedge data \not\approx \mathsf{unknown} \rightarrow$ $\diamondsuit_{[0,1s)} \diamondsuit_{[0,60s)} \exists user'.\, delete(user', \mathsf{db3}, data)$
del-3-2	$\square\, \forall user.\, \forall data.\, delete(user, \mathsf{db3}, data) \wedge data \not\approx \mathsf{unknown} \rightarrow$ $\diamondsuit_{[0,60s)} \diamondsuit_{[0,1s)} \exists user'.\, delete(user', \mathsf{db2}, data)$

We now describe these policies in more detail and present their formalization in MFOTL. We start with the predicate symbols used for formalizing the policies. We represent system actions as elements in relations interpreting the predicate symbols at the time points. The elements of the relations for the predicate symbols *select*, *insert*, *delete*, and *update* correspond to database operations with equally-named SQL commands. The parameters are the user executing the operation, the name of the database, and an identifier of the involved data. The predicate symbols *start* and *stop* indicate the starting and finishing of a synchronization script and include the script's name. After the script script1 starts, it logs additional details about its SVN status using the predicate symbol *svn*. The parameters are the script's name, its SVN status determined by the command `svn status -u -v`, the SVN URL, and the SVN revision number. When the script is the latest version, we use the value latest for the SVN status. The predicate symbol *commit* represents committing a new script version into the subversion repository. The parameters are the SVN URL and revision number.

The MFOTL formalization of the policies uses the predicate symbols just described. The formulas are shown in Table 1. In the following, we informally state the detailed policies in natural language and for the more involved policies, we provide additional explanations:

– *delete:* Only user script2, representing the synchronization script script2, may delete data in db2 by executing the SQL delete command.

– *insert:* Only user script1, representing the synchronization script script1, may insert data in db2 by executing the SQL insert command.
– *select:* Only a limited set of users, namely script1, script2, and triggers, may read data from db2 by executing the SQL select command.
– *update:* No SQL update commands are allowed in db2.
– *script1:* Database operations may be executed under the user account script1 only while the script script1 is running. The motivation for this policy is that the account script1 should only be used by the script, so if the account is used while the script is not running, the account may have been compromised. The database operation can happen while the script is running, including when the script starts or finishes. That is, the time points when an operation happens and when the script starts or ends may have equal timestamps. The semantics of the S operator includes the script start, but excludes the script end. Therefore, the script end is allowed with the additional disjunct at the end of the formula.
– *runtime:* The synchronization scripts must run for at least 1 second and for no longer than 6 hours.
– *svn, svn2:* The synchronization scripts are maintained in an SVN repository. We require that when started, the synchronization scripts are the latest version available in the repository (largest SVN revision number). We use two different formalizations, *svn* and *svn2*. The policy *svn* uses the status parameter of the predicate symbol *svn*. The policy *svn2* compares the revision number parameter of the predicate symbol *svn* with the committed revision numbers obtained from the subversion log via the predicate symbol *commit*. For the policy *svn* we let the logging mechanism compute the latest revision number, while for the policy *svn2* we compute it using the monitor. Monitoring both policies allows us to compare how efficiently the monitor copes with these different formalizations and to observe the impact of offloading the monitor by doing pre-computations in the logging mechanisms.
– *ins-1-2, ins-2-3, ins-3-2:* Data uploaded by the phone into db1 must be propagated to all databases. In particular, *ins-1-2* requires that data uploaded into db1 must be inserted into db2 within 30 hours after the upload, unless it has been deleted from db1 in the meantime. Furthermore, *ins-2-3* and *ins-3-2* require that data may be inserted into db2 iff it is inserted into db3 within 1 minute. The time limit from db1 to db2 is 30 hours because the synchronization scripts run once every 24 hours and can run for up to 6 hours. The time limit from db2 to db3 is only 60 seconds as this synchronization is implemented by database triggers that start immediately upon a change in db2. Note that these policies require propagating new data between db2 and db3 in both directions. However, between db1 and db2 only one direction is required. The reason is the incomplete logging for db1.
– *del-1-2, del-2-3, del-3-2:* Data deleted from db1 must be consistently deleted from all databases. The policies *del-2-3* and *del-3-2* are analogous to the policies *ins-2-3* and *ins-3-2*, respectively. The formalization of the policy *del-1-2* is more involved: If data is deleted from db1, then this data must also be deleted from db2 within 30 hours. However, if the data has just been uploaded to db1 and not yet propagated to db2, then it should not be propagated to db2 in the future either. Since the propagation would happen within at most 30 hours, we can simply consider the past and the future 30 hours to determine whether data has been or will be propagated to db2.

Note that all formulas in Table 1 are collapse-sufficient. However, some policies have slightly weaker or stronger variants that are not collapse-sufficient. For example, we obtained *ins-2-3* from the policy "all data inserted into db2 must also be inserted into db3 within 60 seconds" by weakening the formula $\Box \forall users.\, \forall data.\, insert(user, \mathsf{db2}, data) \wedge data \not\approx \mathsf{unknown} \rightarrow \Diamond_{[0,60s)} \exists user'.\, insert(user', \mathsf{db3}, data)$. Intuitively, *ins-2-3* is the policy formalization that does not distinguish the relative ordering of the insertions into db2 and db3 when they are logged with equal timestamps. This is because the 1 second timestamp granularity that is used may not be fine enough: the database triggers may be activated within milliseconds.

Logging Mechanisms. We extended the data-collection setup with mechanisms to log policy-relevant actions. We installed logging mechanisms for the three databases, the script script1, and the SVN repository, assuming synchronized clocks for timestamping. The databases' logging mechanisms were not straightforward, so we discuss them in more detail.

As logs for the database db1 were not available, we implemented a proxy to inspect interactions of participants and phones with db1. The proxy logs what data is inserted and deleted. To observe the insertion of new data, we monitor the network traffic when the phone uploads data. For deletions, we use a custom front-end that logs the requests for deleting data. For practical reasons, we could deploy these mechanisms only for 2 out of the 180 participants. Hence, we have only partial logging for db1. However, the partial logging affects only 2 out of the 14 policies.

The databases db2 and db3 reside physically on a single PostgreSQL server, which logs the SQL queries. We extract relevant actions from these PostgreSQL logs. The main challenge is to determine what data is processed in a query since only the query itself is logged. Fortunately, most relevant queries are made by automated scripts or database triggers and contain enough information to determine what data is used. For example, an insert or delete query initiated by a synchronization script includes the identifier of the used data record. Hence, a simple syntactic analysis of these queries suffices to log the relevant actions in sufficient detail. When the analysis failed to extract the data, we identified the data with the constant unknown.

7.2 Evaluation

Performance. We evaluated the performance of our monitor on logs from the data-collection campaign. We now describe the logs, the optimizations we made when monitoring the logs, and the monitor's performance.

We monitored all formulas shown in Table 1 on a log file covering approximately one year of the data-collection

campaign. We obtained this log file by interleaving logs from the different log producers to produce one interleaved log that we subsequently collapsed. Note that all monitored formulas are collapse-sufficient, so the monitor correctly reports all violations after inspecting the collapsed log.

We now describe the collapsed log. It contains approximately 5 million time points and 218 million actions (the total number of tuples in all relations). The major part consists of insertions into the three databases: more than 107 million insertions into db2 and db3 each, and 360,000 insertions into db1. The smaller number of logged insertions for db1 is due to the incomplete logging. There are about 3 million select actions and 700,000 update actions on db2 and db3. All other types of actions occurred less than 1,000 times in the whole log.

For comparison, before collapsing, the interleaved log was larger. It contained approximately 400 million time points and a similar number of log actions. The collapsing reduced the number of time points because not all time points in the interleaving had a unique timestamp. The main reason why the total number of log actions was decreased by collapsing is that for most SQL select queries on database db3 we could not determine what data was used. These were logged as using unknown data and therefore could not be distinguished from each other. As there were multiple such indistinguishable actions logged per time unit (second), they were always preserved as only one logged action per timestamp.

For the evaluation, we used the MONPOLY tool [14] running on a desktop computer with an Intel Core i5 2.67 GHz CPU and 8 GB of RAM. To monitor all policies we made two optimizations.

The first optimization was collapsing the interleaving. Note that all monitored formulas are interleaving-sufficient, so the monitor would correctly report all violations after inspecting an arbitrary interleaving. However, it turned out that monitoring an interleaving was computationally infeasible for four policies: *del-1-2*, *ins-1-2*, *ins-2-3*, and *ins-3-2*. Monitoring the policy *del-1-2* exceeded the memory available on our computer and monitoring the policies *ins-1-2*, *ins-2-3*, and *ins-3-2* took too long. For example, monitoring the policy *ins-2-3* only on the first two months of the one year log file took 17 days. Monitoring the collapsed interleaving was computationally feasible for all policies. For policies, which we could monitor already on the interleaving, the monitor was up to three times faster. However, monitoring *ins-1-2* still took a long time, namely 2 weeks.

The second optimization was rewriting formulas into collapse-optimized formulas. The time needed to monitor the policy *ins-1-2* improved from 2 weeks to 34 minutes and for *del-1-2* it improved from 69 minutes to 57 minutes. For other policies, the difference was negligible.

Table 2 shows the monitor's running times and memory usage for each policy with the different optimizations. A missing value in the table signifies that we could not monitor the policy.

We now report in detail on the performance of the monitor after the two optimizations: monitoring performance-optimized formulas on the collapsed log. Monitoring invariants like the policy *delete* is fast: the monitor needed around 20 minutes. The more complex formulas with temporal operators were similarly fast when the formulas matched only a small number of events from the log file. For example, monitoring the policy *svn2* also took less than 20 minutes. Finally, formulas involving temporal operators with large time windows and matching a large part of the events in the log were the most expensive ones to monitor. This was the case for the policies *ins-1-2*, *ins-2-3*, and *ins-3-2* because the log consists mainly of insert events. The policies *ins-2-3* and *ins-3-2* took 49 minutes each. The policy *ins-1-2* took only 34 minutes because there are significantly fewer inserts into db1 than into db2 or db3. The most expensive policy for monitoring was *del-1-2*. It took 57 minutes because its formalization includes multiple temporal operators with large time windows and the subformulas of these temporal operators match the abundant insert events. *ins-1-2* has only one such operator.

We monitored the logs offline. That is, we first collected the complete logs and then monitored them. However, an online monitoring approach, where the logs are monitored as they are generated, seems possible because the running times are orders of magnitude smaller than the time period covered by the logs.

For comparison, we have also monitored the simple access control policies on the collapsed log with the Unix tool grep. The policies *update* and *delete* each took only 5 minutes, which is 3 times faster than the monitor. However, for *insert* grep needed 4 hours, which is 10 times slower than the monitor. We suspect that the automaton used by grep to represent a regular expression was inefficient in dealing with the many insert actions.

The monitor's memory requirements are also modest. For most policies, the monitor does not require more than 30 MB of RAM. The only exceptions are *ins-1-2* with 1 GB and *del-1-2* with 3.3 GB. Again, the reason is temporal operators with large time windows matching a large number of log events.

We now describe in more detail how the monitor coped with the most difficult policy, *del-1-2*. The dashed line in Figure 8 shows the monitor's accumulated running time. The solid line shows the number of log actions since the beginning of the log. The steepness of the solid line indicates the amount of data logged at the corresponding time on the x-axis. We see several flat parts in this curve. During these flat parts, no logs were produced due to server migration and upgrades of the logging infrastructure. We can also see a steep part at the end of August 2010. A new version of the synchronization scripts was copying additional data into the databases for several days. This resulted in an increased number of logged actions. The running time (dashed line) closely followed the number of logged actions (solid line), except for a noticeable slow-down of the monitor during the steep part. The dashed line in Figure 9 shows the amount of memory used by the monitor. Most of the time the monitor needed less than 0.5 GB of memory, but peaked at 3.3 GB at the point where the log curve was steep. Due to the increased log density, more log actions fell into the time windows of the temporal operators, causing the monitor to use larger auxiliary relations. Also note

TABLE 2. Monitor Performance

	collapse, rewritten formulas		collapse		interleaving	
policy	running time	memory used	running time	memory used	running time	memory used
delete	17 min	14 MB	17 min	14 MB	31 min	12 MB
insert	21 min	14 MB	21 min	14 MB	32 min	12 MB
select	17 min	15 MB	17 min	15 MB	34 min	12 MB
update	17 min	14 MB	17 min	14 MB	33 min	12 MB
script1	21 min	14 MB	22 min	14 MB	57 min	13 MB
runtime	18 min	30 MB	19 min	30 MB	52 min	2866 MB
svn	17 min	14 MB	17 min	14 MB	38 min	22 MB
svn2	17 min	14 MB	17 min	14 MB	33 min	12 MB
ins-1-2	34 min	1014 MB	14 days	1387 MB	-	-
ins-2-3	49 min	20 MB	52 min	20 MB	-	-
ins-3-2	49 min	15 MB	49 min	15 MB	-	-
del-1-2	57 min	3313 MB	69 min	3248 MB	-	-
del-2-3	18 min	14 MB	17 min	14 MB	38 min	26 MB
del-3-2	17 min	14 MB	17 min	14 MB	37 min	12 MB

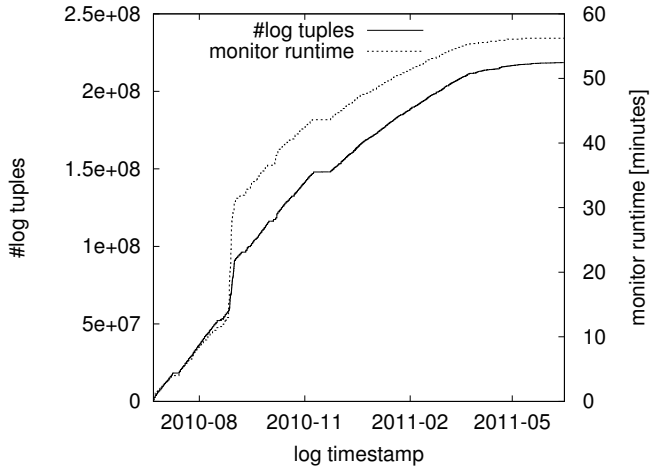

Fig. 8. Monitor Running Time on Policy *del-1-2*

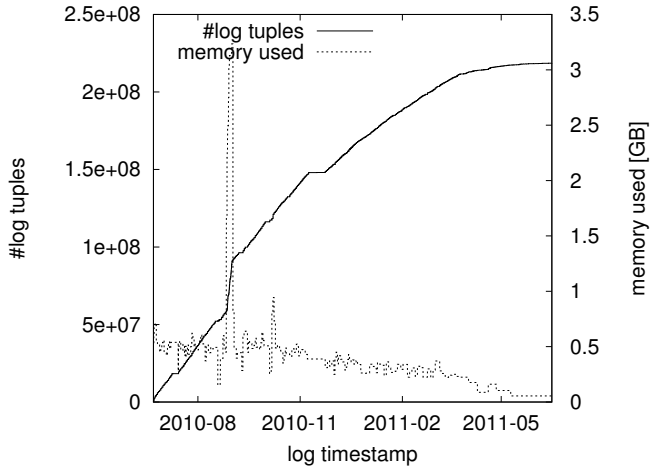

Fig. 9. Monitor Memory Usage on Policy *del-1-2*

that both the amount of new actions logged and the memory used by the monitor decreased towards the log's end. The campaign ended in 2011 and the participants gradually stopped contributing data towards the campaign's end.

Findings. To our surprise, the monitor reported a number of policy violations. First, some access control policies like *delete* were violated. These violations were due to testing, debugging, and other improvement activities going on while the system was running. Second, the policy *runtime* was violated several times, such as when synchronizing the databases after the server migration. Third, an earlier version of one of the synchronization scripts contained a bug, which was not detected in previous tests. Only a subset of the insertions were propagated between the databases. Fourth, while the campaign was running, the infrastructure was migrated to another server. After the migration, the deployment of the scripts was delayed, which caused policy violations.

Overall, the main reason for these violations is that we monitored an experimental system still under development. It is worth pointing out that the privacy of the participants was guaranteed at all times during the campaign and no data elements were unintendedly lost. However, as this case study shows, the monitor can be a powerful debugging tool. For commercial systems, it can detect policy violations thereby protecting the users' privacy and increasing users' trust in the systems. Our findings also show that policy monitoring makes sense even in systems where users and system administrators are honest and interested in honoring the policies.

8 Related Work

Various algorithms have been presented for efficiently monitoring system behavior by inspecting totally ordered logs [6]–[10]. The monitor [10], [14] used in this work extends Chomicki's monitor [22] and can be directly used to monitor a single system component or a log file. A broader overview on the state of the art of monitoring distributed systems can be found in the survey by Goodloe and Pike [23].

Sen et al. [24] present a distributed monitoring approach, where multiple monitors which communicate with each other are implemented locally. The authors use a propositional past linear-time distributed temporal logic with epistemic operators [25] that reflect the local knowledge of a process. The semantics of temporal operators in this logic is defined with respect to a partial ordering, the causal ordering [26] commonly used in distributed systems. Their logic therefore does not allow one to express temporal constraints on events that are not causally related. Policies are defined with respect to the local view point of a single process and checked with respect

to these view points, using the last known states of other processes. Thus two processes can reach different verdicts as to whether a property is satisfied or violated. This is in contrast to our approach where the semantics of the temporal operators is defined with respect to a total ordering and a single central monitor determines whether a global property holds. In addition, note that distributed monitoring entails communication overhead between the monitors whereas we must merge distributed logs.

Genon et al. [19] present a monitoring algorithm for propositional LTL, where events are partially ordered. Whereas we restrict ourselves to formulas for which monitoring a single interleaving is sufficient, their approach checks a formula on all interleavings using symbolic exploration methods. These methods can decrease the number of interleavings considered but, in the worst case, exponentially many must still be examined. Furthermore, it is unclear how their algorithm for the propositional setting extends to a timed and first-order setting.

Wang et al. [27] consider a problem similar to that of Genon et al. [19]. Their monitoring algorithm for past-only propositional LTL with a three-valued semantics explicitly explores the possible interleavings of a partially ordered trace. Matching our notion of strong violations (Definition 3.2(2)), their algorithm returns the truth value false only if the formula is violated on all interleavings. However, their algorithm is not complete in the sense that it might return an inconclusive answer, represented as the third truth value, although all possible interleavings violate the given formula. The third truth value is also returned if some interleavings violate the formula and others satisfy it. Note that in our approach, the formulas in the syntactically-defined fragments either satisfy all interleavings or violate all interleavings.

Several monitoring approaches [28]–[30] have been proposed where actions logged with equal timestamps are considered to happen simultaneously. This corresponds to defining their semantics with respect to the collapsed log in our setting and thereby restricting the expressiveness of the policy specification language. Therefore different possible interleavings need not be considered and the monitoring can be more efficient. We discuss examples below.

Bauer et al. [29] assume a setting where system actions are totally ordered, thereby abstracting away distributivity and concurrency. In their setting, system requirements are given in a propositional linear-time temporal logic. Their monitoring architecture additionally includes a component that analyzes the cause of a failure, which is fed back into the system.

Bauer and Falcone [28] assume synchronized clocks and that observations of the system are done simultaneously in lock-step. This leads to a totally ordered trace of system actions, which corresponds to the collapsed log in our setting. They present a distributed monitoring algorithm for propositional future linear-time temporal logic where monitors are distributed throughout the system and exchange partially evaluated formulas between each other. Each monitor evaluates the subformulas for which it can observe the relevant system actions. Note that their monitoring algorithm is based on rewriting of formulas so, technically speaking, partially rewritten formulas are exchanged.

Zhou et al. [30] check policies against a totally ordered log with exactly one time point per timestamp. Again, this corresponds to the collapsed log in our setting. The authors present a distributed monitoring framework aimed at monitoring properties of network protocols specified in a declarative language based on Datalog. The monitored properties are translated into this language and the monitoring algorithm is executed together with the network protocols.

Mazurkiewicz traces [31] provide an abstract view on partially ordered logs. With this view, the problem of checking whether a policy is strongly violated on a partially ordered log can be stated as checking whether all linearizations of a Mazurkiewicz trace satisfy a temporal property. We are not aware of any work that solves this problem by inspecting a single sequence representing all linearizations. In Mazurkiewicz traces, an independence relation on actions specifies which actions can be reordered. This is independent of the timestamps of actions, whereas in our setting the possibility of reordering depends on the timestamp and not the action.

Also related to our work is partial-order reduction [32]. Partial-order techniques aim at reducing the number of interleavings that are sufficient for checking whether a temporal property is satisfied on all possible interleavings. Partial-order reduction techniques have successfully been used in finite-state model checking where one checks all possible system executions. In contrast, we check compliance of all linearizations of a single observed system execution. Nevertheless, our approach for the interleaving-sufficient fragment can be seen as a special case of partial-order reduction. Namely, we restrict the logical formulas so that it is sufficient to inspect a single interleaving to determine compliance of all possible interleavings. For the collapse-sufficient fragment, we additionally compress the inspected interleaving.

9 Conclusion

We offer a solution to monitor the usage of data in concurrent distributed systems. In particular, we show the intractability of monitoring an arbitrary linear-time temporal logic formula on partially ordered logs and we identify two fragments, which can be monitored efficiently by inspecting representative traces. We also show that membership of a formula in the semantically-defined fragments is undecidable (Theorems 4.3 and 5.2) and we approximate them with sound, but incomplete, syntactically-defined fragments (Theorems 4.6 and 5.6). Finally, we deploy and evaluate our monitoring architecture in a real-world application. Our case study demonstrates the feasibility and benefits of monitoring the usage of sensitive data.

As future work, we plan to develop monitoring techniques for more complex systems with more agents, actions, and databases. The challenges will be to handle less accurate and less complete logging, and to provide monitoring algorithms that scale up from millions to billions of log entries per day. Our future work also includes developing monitoring techniques that can be used for policy enforcement, that is, preventing policy violations.

1 Rewrite $\neg\psi$; proceed only if rewritten formula ϕ is in the monitorable fragment of MFOTL.
2 $\ell \leftarrow 0$
3 $i \leftarrow 0$
4 $Q \leftarrow \{(\alpha, 0, wait(\alpha)) \mid \alpha \text{ is a temporal subformula of } \phi\}$
5 **loop**
6 **forall** $(\alpha, j, \emptyset) \in Q$ **do**
7 Build auxiliary relation for α and time point j using the auxiliary relations for the temporal subformulas of α at time point $j-1$, if $j > 0$.
8 **while** all auxiliary relations for time point i are built **do**
9 Evaluate ϕ at time point i by using the auxiliary relations for time point i; output violations together with timestamp τ_i.
10 If $i > 0$, discard all relations for time point $i-1$.
11 $i \leftarrow i + 1$
12 $Q \leftarrow \{(\alpha, \ell+1, wait(\alpha)) \mid \alpha \text{ is a temporal subformula of } \phi\} \cup$
$\{(\alpha, j, \bigcup_{\alpha' \in up(S, \tau_{\ell+1} - \tau_\ell)} wait(\alpha')) \mid (\alpha, j, S) \in Q \text{ and } S \neq \emptyset\}$
13 $\ell \leftarrow \ell + 1$

Fig. 10. The monitoring algorithm M_ψ

Appendix A
Monitoring Algorithm

Figure 10 presents our monitoring algorithm M_ψ. In the following, we briefly describe M_ψ's operation. A detailed description is given in [10].

M_ψ first rewrites the negation of ψ. Heuristics are used that try to rewrite $\neg\psi$ into a formula ϕ that satisfies certain syntactic criteria that guarantee temporal-subformula-domain-independence. If these criteria are not satisfied, then M_ψ stops. For monitoring, M_ψ 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 sequentially. Initially, ℓ is 0 and it is incremented with each loop iteration (lines 5–13). The counter i is the index of the next time point i (possibly in the past, from ℓ's point of view) that is checked for violations, i.e., the next time point for which the monitor outputs the assignments satisfying ϕ.

The evaluation is delayed until all auxiliary relations for the temporal subformulas of ϕ are built (lines 8–11), i.e., subformulas of ϕ where the outermost connective is one of the temporal operators $\ominus_I$, $\bigcirc_I$, S_I, or U_I. Furthermore, M_ψ uses the list[1] Q to ensure that the auxiliary relations for the time points 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 auxiliary relations for the temporal subformula α and time point j. Without loss of generality, we assume that each temporal subformula α occurs only once in ϕ. M_ψ initializes Q in line 4. The function *wait* identifies the subformulas that delay the formula evaluation:

$$wait(\alpha) := \begin{cases} wait(\beta) & \text{if } \alpha = \neg\beta,\ \alpha = \exists x.\beta, \text{ or} \\ & \quad \alpha = \ominus_I \beta, \\ wait(\beta) \cup wait(\gamma) & \text{if } \alpha = \beta \vee \gamma \text{ or } \alpha = \beta\, \mathsf{S}_I\, \gamma, \\ \{\alpha\} & \text{if } \alpha = \bigcirc_I \beta \text{ or } \alpha = \beta\, \mathsf{U}_I\, \gamma, \\ \emptyset & \text{otherwise.} \end{cases}$$

1. 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 $j < j'$.

The list Q is updated in line 12 before we increment ℓ in line 13 and start a new loop iteration. The update adds a new tuple $(\alpha, \ell+1, wait(\alpha))$ to Q, for each temporal subformula α of ϕ, and it removes 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 *wait* and *up*, accounting for the elapsed time to the next time point, that is, $\tau_{\ell+1} - \tau_\ell$. For a set of formulas U and $t \in \mathbb{N}$, $up(U, t)$ is the set

$$\begin{aligned} &\{\beta \mid \bigcirc_I \beta \in U\} \cup \\ &\{\beta\, \mathsf{U}_{[\max\{0, b-t\}, b'-t)}\, \gamma \mid \beta\, \mathsf{U}_{[b,b')}\, \gamma \in U, \text{ with } b' - t > 0\} \cup \\ &\{\beta \mid \beta\, \mathsf{U}_{[b,b')}\, \gamma \in U \text{ or } \gamma\, \mathsf{U}_{[b,b')}\, \beta \in U, \text{ with } b' - t \leq 0\}. \end{aligned}$$

In line 7, we build the auxiliary relations for which enough time has elapsed, that is, the relations for α at time point j with $(\alpha, j, \emptyset) \in Q$. To efficiently build these relations, we use incremental constructions that reuse relations from the previous time point. In lines 8–11, if all the relations for time point i have been built, then M_ψ outputs the valuations violating ϕ at time point i together with the timestamp τ_i. Furthermore, after each output, the relations at time point $i-1$ are discarded (if $i > 0$) and i is incremented.

Appendix B
Additional Proof Details

In this appendix we provide additional proof details for the assertions made in this paper.

B.1 Intractability

In this section, we show the correctness of Theorem 3.3 about the intractability of checking weak and strong violations.

Membership. The decision problem in Theorem 3.3(1) is in NP as a nondeterministic Turing machine can first guess the violating interleaving up to the given time point and then verify its guess in polynomial time [33]. Note that the Turing machine does not need to guess a valuation, as the input formula is a quantifier-free sentence and thus contains no variables. The decision problem in Theorem 3.3(2) is in coNP since its complement is in NP.

Hardness. Hardness of the decision problem in Theorem 3.3(1) is established by polynomially reducing SAT to it. Analogously, the coNP-hardness of the decision problem in Theorem 3.3(2) is shown by polynomially reducing TAUT to it.

Reduction from SAT. We show NP-hardness of the decision problem in Theorem 3.3(1) by a reduction from SAT. The SAT problem asks whether a given propositional formula is satisfiable. SAT is NP-hard.

To fix notation, we recall that a propositional formula α over a set of atomic propositions P is satisfiable if there is an assignment θ of propositions to truth values $\bot$ (denoting false) and $\top$ (denoting true), that is, $\theta : P \to \{\bot, \top\}$, such that $\theta(\alpha) = \top$, where θ is homomorphically extended from atomic propositions to formulas.

Suppose $P = \{p_0, \ldots, p_{n-1}\}$, with $n \geq 0$, is a set of atomic propositions. Let S be the signature (C, R, ι) with $C = \{c\}$, $R = \{q_0, r_0, \ldots, q_{n-1}, r_{n-1}\}$, and $\iota(q_i) = \iota(r_i) = 1$, for any $0 \leq i < n$.

The two temporal structures $(\bar{\mathcal{D}}^1, \bar{\tau}^1)$ and $(\bar{\mathcal{D}}^2, \bar{\tau}^2)$ over S are given by $|\bar{\mathcal{D}}| = \{c\}$, $c^{\bar{\mathcal{D}}} = c$, $\tau_i^1 = \tau_i^2 = i$, and

$$q_i^{\mathcal{D}_j^k} = \begin{cases} \{c\} & \text{if } k = 1 \text{ and } i = j, \\ \emptyset & \text{otherwise,} \end{cases}$$
$$r_i^{\mathcal{D}_j^k} = \begin{cases} \{c\} & \text{if } k = 2 \text{ and } i = j, \\ \emptyset & \text{otherwise,} \end{cases}$$

for any $i \in \mathbb{N}$, $k \in \{1, 2\}$, and $i, j \in \mathbb{N}$ with $0 \le i < n$,

Given a propositional formula α over P, the MFOTL formula $\ulcorner\alpha\urcorner$ is obtained by replacing each occurrence of a proposition p_i in α with $\diamondsuit(r_i(c) \wedge \diamondsuit q_i(c))$. Thus, given a propositional formula α, the reduction constructs the two prefixes of length n of $(\bar{\mathcal{D}}^1, \bar{\tau}^1)$ and $(\bar{\mathcal{D}}^2, \bar{\tau}^2)$ and the MFOTL formula $\ulcorner\alpha\urcorner$. This reduction is linear in the length of α. Its correctness is shown by Lemma B.2. The following remarks and lemma will be needed.

Remark 1. For any interleaving $(\bar{\mathcal{D}}, \bar{\tau}) \in (\bar{\mathcal{D}}^1, \bar{\tau}^1) \between (\bar{\mathcal{D}}^2, \bar{\tau}^2)$, the functions f_1 and f_2 in Definition 3.1 satisfy $f_k(i) \in \{2i, 2i+1\}$ where $k \in \{1, 2\}$. Moreover, these functions are unique, that is, if $g_1, g_2 : \mathbb{N} \to \mathbb{N}$ are strictly monotonic functions satisfying conditions (1)–(3) in Definition 3.1, then either $g_1 = f_1$ and $g_2 = f_2$, or $g_1 = f_2$ and $g_2 = f_1$. Furthermore, for any strictly monotonic functions f_1 and f_2 satisfying conditions (1) and (2) in Definition 3.1 and with $f_1(i), f_2(i) \in \{2i, 2i+1\}$ for $0 \le i < n$, there is a unique temporal structure $(\bar{\mathcal{D}}, \bar{\tau})$ such that f_1 and f_2 also satisfy condition (3). In other words, the functions f_1 and f_2 determine an interleaving of $(\bar{\mathcal{D}}^1, \bar{\tau}^1)$ and $(\bar{\mathcal{D}}^2, \bar{\tau}^2)$.

Lemma B.1. *Let α be a propositional formula, θ a truth value assignment, v a valuation, and $(\bar{\mathcal{D}}, \bar{\tau})$ an interleaving of $(\bar{\mathcal{D}}^1, \bar{\tau}^1) \between (\bar{\mathcal{D}}^2, \bar{\tau}^2)$ given by the functions f_1 and f_2 such that $\theta(p_i) = \top$ iff $f_1(i) = 2i$, for any i with $0 \le i < n$. Then $\theta(\alpha) = \top$ iff $(\bar{\mathcal{D}}, \bar{\tau}, v, 2n) \models \ulcorner\alpha\urcorner$.*

Proof: We use structural induction on the form of α. The only interesting case is the base case; the other cases follow directly from the induction hypotheses. Thus let $\alpha = p_i \in P$.

Suppose that $(\bar{\mathcal{D}}, \bar{\tau}, v, 2n) \models \diamondsuit(r_i(c) \wedge \diamondsuit q_i(c))$. That is, there is a time point $j \le 2n$ such that $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \models r_i(c)$ and such that there is a time point $j' \le j$ for which $(\bar{\mathcal{D}}, \bar{\tau}, v, j') \models q_i(c)$. Then $c \in r_i^{\mathcal{D}_j}$ and $c \in q_i^{\mathcal{D}_{j'}}$. From the definition of an interleaving and the definitions of the interpretations of the predicate symbols q_i and r_i, it follows that $j = f_2(i)$ and $j' = f_1(i)$. Then, as $f_1(i), f_2(i) \in \{2i, 2i+1\}$, $f_1(i) \ne f_2(i)$, and $j' \le j$, we have that $f_1(i) = 2i$ and $f_2(i) = 2i + 1$. Thus $\theta(p_i) = \top$.

Suppose that $\theta(\alpha) = \top$. Then $f_1(i) = 2i$ and $f_2(i) = 2i + 1$. We have $(\bar{\mathcal{D}}, \bar{\tau}, v, 2i) \models q_i(c)$ and $(\bar{\mathcal{D}}, \bar{\tau}, v, 2i + 1) \models r_i(c)$. Thus $(\bar{\mathcal{D}}, \bar{\tau}, v, 2i + 1) \models r_i(c) \wedge \diamondsuit q_i(c)$ and clearly $(\bar{\mathcal{D}}, \bar{\tau}, v, 2n) \models \diamondsuit(r_i(c) \wedge \diamondsuit q_i(c))$. □

Lemma B.2. *Let α be a propositional formula. It holds that α is satisfiable iff $(\bar{\mathcal{D}}^1, \bar{\tau}^1) \between (\bar{\mathcal{D}}^2, \bar{\tau}^2)$ weakly violates $\neg\ulcorner\alpha\urcorner$ at time point $2n$.*

Proof: Suppose first that α is satisfiable. Then there is a truth value assignment θ such that $\theta(\alpha) = \top$. Let $(\bar{\mathcal{D}}, \bar{\tau})$ be the interleaving determined by the functions f_1 and f_2 given by

$$f_1(i) = \begin{cases} 2i & \text{if } \theta(p_i) = \top, \\ 2i + 1 & \text{otherwise,} \end{cases}$$

and

$$f_2(i) = \begin{cases} 2i & \text{if } \theta(p_i) = \bot, \\ 2i + 1 & \text{otherwise.} \end{cases}$$

Let v be an arbitrary valuation. From Lemma B.1, we obtain that $(\bar{\mathcal{D}}, \bar{\tau}, v, 2n) \models \ulcorner\alpha\urcorner$, that is, $(\bar{\mathcal{D}}, \bar{\tau}, v, 2n) \not\models \neg\ulcorner\alpha\urcorner$.

Suppose now that $(\bar{\mathcal{D}}^1, \bar{\tau}^1) \between (\bar{\mathcal{D}}^2, \bar{\tau}^2)$ weakly violates $\neg\ulcorner\alpha\urcorner$ at time point $2n$. Then there is an interleaving $(\bar{\mathcal{D}}, \bar{\tau})$ and a valuation v such $(\bar{\mathcal{D}}, \bar{\tau}, v, 2n) \not\models \neg\ulcorner\alpha\urcorner$. Let f_1 and f_2 be the functions determined by $(\bar{\mathcal{D}}, \bar{\tau})$ as in Definition 3.1. Let θ be a truth value assignment such that $\theta(p_i) = \top$ if $f_1(i) = 2i$. Using again Lemma B.1, we get that θ is a satisfying assignment for α. □

Reduction from TAUT. We show coNP-hardness of the decision problem in Theorem 3.3(2) by reduction from TAUT. The TAUT problem asks whether a given propositional formula is a tautology. TAUT is coNP-hard.

We recall that a propositional formula α over a set of atomic propositions P is a tautology if $\theta(\alpha) = \top$ for any assignment θ of propositions to truth values.

We use the same reduction that was used in the decision problem in Theorem 3.3(1). The correctness of the reduction follows from the following lemma.

Lemma B.3. *Let α be a propositional formula. Then α is a tautology iff $(\bar{\mathcal{D}}^1, \bar{\tau}^1) \between (\bar{\mathcal{D}}^2, \bar{\tau}^2)$ strongly violates $\neg\ulcorner\alpha\urcorner$ at time point $2n$.*

Proof: Suppose first that α is a tautology. Let $(\bar{\mathcal{D}}, \bar{\tau})$ be an arbitrary interleaving in $(\bar{\mathcal{D}}^1, \bar{\tau}^1) \between (\bar{\mathcal{D}}^2, \bar{\tau}^2)$ and f_1 and f_2 be functions as in Definition 3.1. Let θ be a truth value assignment such that $\theta(p_i) = \top$ iff $f_1(i) = 2i$. Let v be an arbitrary valuation. Using Lemma B.1, we obtain that $(\bar{\mathcal{D}}, \bar{\tau}, v, 2n) \not\models \neg\ulcorner\alpha\urcorner$. Hence $(\bar{\mathcal{D}}^1, \bar{\tau}^1) \between (\bar{\mathcal{D}}^2, \bar{\tau}^2)$ strongly violates $\neg\ulcorner\alpha\urcorner$ at time point $2n$.

Suppose now that $(\bar{\mathcal{D}}^1, \bar{\tau}^1) \between (\bar{\mathcal{D}}^2, \bar{\tau}^2)$ strongly violates $\neg\ulcorner\alpha\urcorner$ at time point $2n$. Let θ be an arbitrary truth value assignment. Let $(\bar{\mathcal{D}}, \bar{\tau})$ be the interleaving determined by the functions f_1 and f_2 given by

$$f_1(i) = \begin{cases} 2i & \text{if } \theta(p_i) = \top, \\ 2i + 1 & \text{otherwise,} \end{cases}$$

and

$$f_2(i) = \begin{cases} 2i & \text{if } \theta(p_i) = \bot, \\ 2i + 1 & \text{otherwise.} \end{cases}$$

There is a valuation v such $(\bar{\mathcal{D}}, \bar{\tau}, v, 2n) \not\models \neg\ulcorner\alpha\urcorner$. Using again Lemma B.1, we have that θ is a satisfying assignment for α. Hence α is a tautology. □

B.2 Undecidability

In this section, we prove Theorem 4.3 and Theorem 5.2 claiming that the checking whether a formula is interleaving-sufficient and collapse-sufficient is undecidable. In the proofs, we restrict ourselves without loss of generality to FOTL formulas, that is, MFOTL formulas where the temporal operators do not have any metric constraints. We first show the undecidability of the tautology problem for FOTL. We use this in the proofs afterwards.

Lemma B.4. *Given a FOTL formula ϕ, it is undecidable whether ϕ is a tautology.*

Proof: We reduce the halting problem of a deterministic Turing machine (DTM) with the empty word as input to the FOTL tautology problem. We first introduce notation for a DTM and then proceed with the reduction.

Different types of Turing machines are used in the literature, so we briefly describe the one we use. For a more detailed introduction to Turing machines, see [34]. Our DTM has a tape and a head to read and write the tape. The tape consists of cells and is infinite in both directions. The cells of the tape are indexed with the indexes coming from $\mathbb{Z}$. A single tape symbol is written in each cell. Initially, the input to the DTM is written on the tape starting at cell 0. The rest of the tape is filled with the blank symbol. We consider only the empty word as input, so the whole tape is filled with the blank symbol. The head of the DTM is initially positioned at cell 0.

The DTM is always in one of finitely many states and executes in steps. In each step, the DTM reads the symbol on the tape at the cell where the head is positioned, writes a new symbol into the cell, moves the head to the left, right, or not at all, and finally, the DTM may make a transition into a new state. When the DTM reaches a final state, it continues to loop forever in this state, always writes the same symbol onto the tape, and does not move the head. We say that it halts.

Formally, the DTM is described by a tuple $(Q, \Gamma, \Sigma, \delta, q_0, B, F)$, where

- Q is the finite set of states in which the DTM can be.
- Γ is the finite set of tape symbols.
- $\Sigma \subseteq \Gamma$ is the finite set of input symbols.
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\text{left, right, none}\}$ is the transition function. The arguments of $\delta(q, x)$ are a state q in which the DTM is and a symbol x read from the tape where the head is positioned. The value of $\delta(q, x)$ is a triple (p, y, d), where:
  - p is the next state into which the DTM transitions.
  - y is the symbol to be written in the cell where the head is positioned.
  - d is left, right, or none indicating whether the head should move to the left, to the right, or not move at all, respectively.
- $q_0 \in Q$ is the initial state of the DTM.
- $B \in \Gamma \setminus \Sigma$ is the blank symbol.
- $F \subseteq Q$ is the set of final states.

To ensure that the DTM loops in final states, $\delta(q, x) = (q, x, \text{none})$ for all $q \in F$ and all $x \in \Gamma$.

We now reduce the undecidable problem of deciding whether a DTM halts on the empty word to the problem of deciding whether a FOTL formula is unsatisfiable. To describe a run of the DTM we use the following predicate symbols:

- $H(i)$ iff the head is positioned at cell i.
- $T_x(i)$ iff the tape contains symbol x, other than the blank symbol, at position i.
- I_q iff the DTM is in state q.

To check for an empty symbol at position i on the tape, we use the syntactic sugar $T_B(i) := \bigwedge_{x \in \Gamma \setminus \{B\}} \neg T_x(i)$.

We represent positions on the tape with an index $i \in \mathbb{Z}$. The left-most symbol of the input is at position 0, where also the head is initially positioned.

We also need a successor function on $\mathbb{Z}$. We define it as

$$S(i, j) := i \prec j \wedge \forall k. (k \prec i \vee k \approx i \vee k \approx j \vee j \prec k) .$$

We describe a non-halting run of the DTM M with the FOTL formula

$$\rho_M := \Box \mathit{WELLFORMED} \wedge (\diamondsuit \mathit{INIT}) \wedge \mathit{STEP} \wedge \neg \mathit{FINAL} ,$$

where its subformulas are as follows:

- *WELLFORMED* ensures that the DTM is in a proper configuration. We define it as

$$\begin{aligned} \mathit{WELLFORMED} := (\forall i. \forall j. H(i) \wedge H(j) \to i \approx j) \wedge \\ (\forall i. \bigwedge_{x,y \in \Gamma} (T_x(i) \wedge T_y(i) \to x \approx y)) \wedge \\ \bigwedge_{p,q \in Q} (I_p \wedge I_q \to p \approx q) . \end{aligned}$$

  It ensures that the head is positioned at exactly one tape cell, that there is exactly one symbol written on each tape cell, and that the DTM is in exactly one state.
- *INIT* describes the initial configuration of the DTM. We define it as

$$\mathit{INIT} := H(0) \wedge I_{q_0} \wedge \forall i. T_B(i) .$$

  Note that we consider only the empty word as the input, so the whole tape is initially filled with the blank symbol.
- *STEP* describes one step of the DTM. Let the tuples (q, x, p, y, d) represent the transition function δ where $(p, y, d) = \delta(q, x)$. *STEP* is a conjunction of formulas representing all those tuples. For tuples with d = left, the formula is

$$\forall i. \forall j. I_q \wedge H(j) \wedge T_x(j) \wedge S(i, j) \to \bigcirc I_p \wedge H(i) \wedge T_y(j) .$$

  For tuples with d = right, the formula is

$$\forall i. \forall j. I_q \wedge H(i) \wedge T_x(i) \wedge S(i, j) \to \bigcirc I_p \wedge H(j) \wedge T_y(i) .$$

  For tuples with d = none, the formula is

$$\forall i. I_q \wedge H(i) \wedge T_x(i) \to \bigcirc I_p \wedge H(i) \wedge T_y(i) .$$

  In addition, the conjunction also contains the formula

$$\forall i. \forall j. \bigwedge_{z \in \Gamma} (H(i) \wedge i \not\approx j \wedge T_z(j) \to \bigcirc T_z(j))$$

  expressing the fact that the tape can change only at the position where the head is positioned.
- *FINAL* describes the DTM entering a final state. We define

$$\mathit{FINAL} := \bigvee_{q \in F} I_q .$$

Every configuration of the DTM M in a run is represented by a time point in the model of the FOTL formula ρ_M. Note that this is a valid MFOTL model. At any step it holds that M has written at most a finite number of cells on the tape, so the relations of the predicate symbols T_x for $x \in \Gamma \setminus \{B\}$ are always

finite. There is no predicate symbol to directly represent the blank symbol.

The DTM M does not halt if there is a model where ρ_M is satisfied. Since the formula ρ_M can be effectively constructed from the description of M, the undecidability of the halting problem implies the undecidability of the unsatisfiability problem for FOTL formulas. By considering the negation of a FOTL formula, it follows that the tautology problem is also undecidable. □

Next, we prove Theorem 4.3, claiming that the interleaving-sufficient property is undecidable.

Proof: From Lemma B.4 we know that the problem whether a FOTL formula ϕ is a tautology is undecidable. Hence, also determining whether ϕ is unsatisfiable is an undecidable problem. We proceed by reducing the problem of deciding whether ϕ is unsatisfiable to deciding whether ϕ is interleaving-sufficient. To this end, we show the following equivalence: ϕ is unsatisfiable iff the formula $\phi \wedge \Box p \rightarrow \Diamond q$ is interleaving-sufficient, where the predicate symbols p and q do not occur in ϕ.

Note that if ϕ falls into the fragment of MFOTL that we can monitor, then it is of the form $\Box \psi$. The formula $\phi \wedge \Box p \rightarrow \Diamond q$ can then be rewritten to $\Box \psi \wedge (p \rightarrow \Diamond q)$ and hence also falls into the fragment of MFOTL that we can monitor.

We first show the direction from left to right. As ϕ is unsatisfiable, $\phi \wedge \Box p \rightarrow \Diamond q$ is unsatisfiable. Hence, it is interleaving-sufficient.

Next, we show the direction from right to left and prove that if ϕ is satisfiable then $\phi \wedge \Box p \rightarrow \Diamond q$ is not interleaving-sufficient. If ϕ is satisfiable, there is a temporal structure $(\bar{\mathcal{D}}, \bar{\tau})$ on which ϕ is satisfied. As ϕ's temporal operators do not have any temporal constraints, we can pick $\bar{\tau}$ so that the first two time points have an equal timestamp. That is, $\tau_0 = \tau_1$. Furthermore, because the predicate symbols p and q do not occur in the formula ϕ, we can pick $\bar{\mathcal{D}}$ in so that the predicate symbol q is satisfied only at the first time point and the predicate symbol p is satisfied only at the second time point. That is, $(\bar{\mathcal{D}}, \bar{\tau}, v, 0) \models q$, but $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \not\models q$, for all $i \neq 0$ and any valuation v. Similarly, $(\bar{\mathcal{D}}, \bar{\tau}, v, 1) \models p$, but $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \not\models p$, for all $i \neq 1$. Clearly, this temporal structure satisfies our formula, that is, $(\bar{\mathcal{D}}, \bar{\tau}, v, 0) \models \phi \wedge \Box p \rightarrow \Diamond q$.

Let $(\bar{\mathcal{D}}_1, \bar{\tau}_1)$ and $(\bar{\mathcal{D}}_2, \bar{\tau}_2)$ be two temporal structures such that $(\bar{\mathcal{D}}, \bar{\tau}) \in (\bar{\mathcal{D}}_1, \bar{\tau}_1) \between (\bar{\mathcal{D}}_2, \bar{\tau}_2)$ and for all $i \in \mathbb{N}$ we have that $(\bar{\mathcal{D}}_1, \bar{\tau}_1, v, i) \not\models q$ and $(\bar{\mathcal{D}}_2, \bar{\tau}_2, v, i) \not\models p$. Clearly, there is another interleaving $(\bar{\mathcal{D}}', \bar{\tau}') \in (\bar{\mathcal{D}}_1, \bar{\tau}_1) \between (\bar{\mathcal{D}}_2, \bar{\tau}_2)$, where the first two time points are in the opposite order as in $(\bar{\mathcal{D}}, \bar{\tau})$. That is, p is satisfied only on the first time point and q is satisfied only on the second time point. Then $(\bar{\mathcal{D}}', \bar{\tau}', v, 0) \not\models \Box p \rightarrow \Diamond q$ and $(\bar{\mathcal{D}}', \bar{\tau}', v, 0) \not\models \phi \wedge \Box p \rightarrow \Diamond q$. As the formula $\phi \wedge \Box p \rightarrow \Diamond q$ is satisfied on one interleaving, but not on another one, the formula is not interleaving-sufficient. □

The proof of Theorem 5.2, claiming that the collapse-sufficient property is undecidable, is analogous to the interleaving-sufficient case. We omit it.

B.3 Labeling Rules (Interleaving)

In this section, we show the correctness of Lemma 4.5, that is, the soundness of the labeling rules shown in Figure 4.

Proof: We proceed by induction on the size of the derivation tree assigning label ℓ to ϕ. We make a case distinction based on the rule applied to label the formula, that is, the rule at the tree's root. However, for clarity, we generally group cases by the formula's form.

For readability, and without loss of generality, we already fix two arbitrary interleavings $(\bar{\mathcal{D}}, \bar{\tau})$ and $(\bar{\mathcal{D}}', \bar{\tau}')$ of two given temporal structures. We also fix an arbitrary valuation v, an arbitrary time point i in $(\bar{\mathcal{D}}, \bar{\tau})$, and the time point i' in $(\bar{\mathcal{D}}', \bar{\tau}')$ corresponding to the time point i.

We first consider the weakening rule. Suppose that $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$. By the induction hypothesis, ϕ has the property ALL, so for all time points $j \in \mathbb{N}$ with $\tau_i = \tau'_j$ we have that $(\bar{\mathcal{D}}', \bar{\tau}', v, j) \models \phi$. But then the time point i' is among those j's, so $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \models \phi$ and ϕ has the property ONE.

Next, we make a case distinction on the form of the formula. Consider formulas of the form:

- $t \approx t'$, where t and t' are variables or constants. Suppose that $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models t \approx t'$. It follows that $v(t) = v(t')$. As this only depends on the valuation v, we have $(\bar{\mathcal{D}}', \bar{\tau}', v, j) \models t \approx t'$ for all $j \in \mathbb{N}$ and hence we can add the label ALL to the formula $t \approx t'$.
- $t \prec t'$, where t and t' are variables or constants. This case is similar to the previous one.
- $r(t_1, \ldots, t_{\iota(r)})$, where $t_1, \ldots, t_{\iota(r)}$ are variables or constants. Suppose that $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models r(t_1, \ldots, t_{\iota(r)})$. As i' is the time point corresponding to i, it follows that $r^{\mathcal{D}_i} = r^{\mathcal{D}'_{i'}}$. Hence, $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \models r(t_1, \ldots, t_{\iota(r)})$ and $r(t_1, \ldots, t_{\iota(r)})$ has the property ONE.
- $\neg\phi$
  - We first show why the label ONE can be propagated. Suppose that ϕ : ONE and $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \neg\phi$, from which it follows that $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \not\models \phi$. We claim that from ϕ : ONE it follows that $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \not\models \phi$. To achieve a contradiction, suppose that $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \models \phi$. By the induction hypothesis, ϕ has the property ONE, so it follows that $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$, which is a contradiction. Hence, $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \not\models \phi$, so that $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \models \neg\phi$ and $\neg\phi$ has the property ONE.
  - Next we show why the label ALL can be propagated. Suppose that ϕ : ALL and $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \neg\phi$, from which it follows that $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \not\models \phi$. We claim that from ϕ : ALL it follows that $(\bar{\mathcal{D}}', \bar{\tau}', v, j) \not\models \phi$ for all $j \in \mathbb{N}$ with $\tau_i = \tau'_j$. To achieve a contradiction, suppose that $(\bar{\mathcal{D}}', \bar{\tau}', v, k) \models \phi$ for some k with $\tau_i = \tau'_k$. By the induction hypothesis, ϕ has the property ALL, so it follows that $(\bar{\mathcal{D}}, \bar{\tau}, v, \ell) \models \phi$ for all $\ell \in \mathbb{N}$ with $\tau_\ell = \tau'_k = \tau_i$ and hence $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$, which is a contradiction. Therefore, $(\bar{\mathcal{D}}', \bar{\tau}', v, j) \not\models \phi$, so that $(\bar{\mathcal{D}}', \bar{\tau}', v, j) \models \neg\phi$ and $\neg\phi$ has the property ALL.
- $\phi \vee \psi$
  - We first show why the label ONE can be propagated. Suppose that ϕ : ONE, ψ : ONE, and $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi \vee \psi$. It follows that 1) $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$ or 2) $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \psi$. If 1), then by the induction hypothesis ϕ has the property ONE and it follows that $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \models \phi$. If 2), then by the induction hypothesis ψ has the property

ONE and it follows that $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \models \psi$. Therefore, in both cases we have that $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \models \phi \vee \psi$, so ϕ has the property ONE.
  - The argument why the label ALL can be propagated is analogous.
- $\exists x.\, \phi$
  - We first show why the label ONE can be propagated. Suppose that ϕ : ONE and $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \exists x.\, \phi$. It follows that $(\bar{\mathcal{D}}, \bar{\tau}, v[x/d], i) \models \phi$, for some $d \in |\bar{\mathcal{D}}|$. Since $|\bar{\mathcal{D}}| = |\bar{\mathcal{D}}'|$, d is also in $|\bar{\mathcal{D}}'|$. By the induction hypothesis, ϕ has the property ONE, and hence $(\bar{\mathcal{D}}', \bar{\tau}', v[x/d], i') \models \phi$ and $(\bar{\mathcal{D}}', \bar{\tau}', v, i) \models \exists x.\, \phi$. Therefore, $\exists x.\, \phi$ has the property ONE.
  - The argument why the label ALL can be propagated is analogous.
- $\phi\, \mathsf{S}_I\, \psi$. Suppose that ϕ : ALL, ψ : ALL, and $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi\, \mathsf{S}_I\, \psi$. It follows that there is a $j \leq i$ such that $\tau_i - \tau_j \in I$, $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \psi$, and $(\bar{\mathcal{D}}, \bar{\tau}, v, k) \models \phi$ for all k with $j < k \leq i$. By the induction hypothesis, ψ has the property ALL. It follows that $(\bar{\mathcal{D}}', \bar{\tau}', v, j) \models \psi$ for all time points j' with $\tau_j = \tau'_{j'}$. Let $j'_{\max}$ be the largest of such j'. By the induction hypothesis, ϕ also has the property ALL. It follows that $(\bar{\mathcal{D}}', \bar{\tau}', v, k') \models \phi$ for all k' with $\tau'_{j'_{\max}} < \tau'_{k'} \leq \tau'_{i'}$. Hence, for every time point i'' with $\tau'_{i''} = \tau'_{i'}$ there is a $j' \leq i''$ such that $\tau'_{i''} - \tau'_{j'} \in I$, $(\bar{\mathcal{D}}', \bar{\tau}', v, j') \models \psi$, and $(\bar{\mathcal{D}}', \bar{\tau}', v, k'') \models \phi$ for all k'' with $j'' < k'' \leq i''$. Therefore, $(\bar{\mathcal{D}}', \bar{\tau}', v, k') \models \phi\, \mathsf{S}_I\, \psi$ and $\phi\, \mathsf{S}_I\, \psi$ has the property ALL.
- $\phi\, \mathsf{U}_I\, \psi$. This case is similar to the previous one.
- $\Diamond\!\!\!-_I\, \phi$ with $0 \notin I$.
  Suppose that ϕ : ONE and $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \Diamond\!\!\!-_I\, \phi$. There is then a time point j with $j \leq i$, $\tau_i - \tau_j \in I$ such that $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \phi$. By the induction hypothesis, ϕ has the property ONE. It follows that there is a time point j' in $(\bar{\mathcal{D}}', \bar{\tau}')$ corresponding to j with $(\bar{\mathcal{D}}', \bar{\tau}', v, j') \models \phi$. From $0 \notin I$ it follows that $\tau_j < \tau_i$, so that $\tau'_{j'} < \tau'_{i'}$, and hence $j' < i''$ for all $i'' \in \mathbb{N}$ with $\tau_i = \tau'_{i''}$. Therefore, $(\bar{\mathcal{D}}', \bar{\tau}', v, i'') \models \Diamond\!\!\!-_I\, \phi$ for all such i'' and $\Diamond\!\!\!-_I\, \phi$ has the property ALL.
- $\Diamond_I\, \phi$ with $0 \notin I$. This case is similar to the previous one.
- $\Diamond_I\, \Diamond\!\!\!-_J\, \phi$.
  Suppose that ϕ : ONE and $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \Diamond_I\, \Diamond\!\!\!-_J\, \phi$, so there are time points j and k with $j \geq i$, $\tau_j - \tau_i \in I$, $k \leq j$, $\tau_j - \tau_k \in J$, and $(\bar{\mathcal{D}}, \bar{\tau}, v, k) \models \phi$. By the induction hypothesis, ϕ has the property ONE, so there is a time point k' in $(\bar{\mathcal{D}}', \bar{\tau}')$ corresponding to k with $(\bar{\mathcal{D}}', \bar{\tau}', v, k') \models \phi$. To satisfy the formula $\Diamond_I\, \Diamond\!\!\!-_J\, \phi$ on $(\bar{\mathcal{D}}', \bar{\tau}')$ we pick the maximal j' with $\tau'_{j'} = \tau_j$. To see that this satisfies the formula we need to show that 1) $j' \geq i'$, 2) $\tau'_{j'} - \tau'_{i'} \in I$, 3) $k' \leq j'$, and 4) $\tau'_{j'} - \tau'_{k'} \in J$.
  1. From $\tau_j \geq \tau_i$, $\tau'_{j'} = \tau_j$, $\tau'_{i'} = \tau_i$ we see that $\tau'_{j'} \geq \tau'_{i'}$. From j' being the maximal time point with the timestamp $\tau'_{j'}$ it follows that $j' \geq i'$.
  2. From $\tau_j - \tau_i \in I$, $\tau'_{i'} = \tau_i$, and $\tau'_{j'} = \tau_j$ it follows that $\tau'_{j'} - \tau'_{i'} \in I$.
  3. From $\tau_k \leq \tau_j$, $\tau'_{k'} = \tau_k$, $\tau'_{j'} = \tau_j$ we see that $\tau'_{k'} \leq \tau'_{j'}$. From j' being the maximal time point with the timestamp $\tau'_{j'}$, it follows that $k' \leq j'$.
  4. From $\tau_j - \tau_k \in J$, $\tau'_{j'} = \tau_j$, and $\tau'_{k'} = \tau_k$ it follows that $\tau'_{j'} - \tau'_{k'} \in J$.

  Therefore, $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \models \Diamond_I\, \Diamond\!\!\!-_J\, \phi$ and $\Diamond_I\, \Diamond\!\!\!-_J\, \phi$ has the property ALL.
- $\Diamond\!\!\!-_I\, \Diamond_J\, \phi$. This case is similar to the previous one, but we pick the minimal time point for the temporal operator $\Diamond\!\!\!-_I$.

□

B.4 Interleaving-sufficient Formulas

In this section, we show the correctness of Theorem 4.6. The implications in Theorem 4.6 follow directly from the correctness of the labeling rules (Lemma 4.5) and from the following lemma.

Lemma B.5. *Let ϕ be a formula.*

1. *If ϕ has the property* ALL*, then ϕ has the properties (I1) and (I2).*
2. *If ϕ has the property* ONE*, then $\Box\, \phi$ has the properties (I1) and (I2).*

Proof: We fix two arbitrary interleavings $(\bar{\mathcal{D}}, \bar{\tau})$ and $(\bar{\mathcal{D}}', \bar{\tau}')$ of two given temporal structures.

1. We first show that ϕ has (I1). Suppose that ϕ has the property ALL and that $(\bar{\mathcal{D}}, \bar{\tau}, v, 0) \models \phi$ for some valuation v. Since ϕ has the property ALL, it follows that $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \models \phi$ for all time points i' with $\tau_0 = \tau'_{i'}$. Since $\tau_0 = \tau'_0$, it follows that $(\bar{\mathcal{D}}', \bar{\tau}', v, 0) \models \phi$ and hence ϕ has (I1).
   Next, we show that ϕ has (I2). Suppose that ϕ has the property ALL and that $(\bar{\mathcal{D}}, \bar{\tau}, v, 0) \not\models \phi$ for some valuation v. To achieve a contradiction, suppose that $(\bar{\mathcal{D}}', \bar{\tau}', v, 0) \models \phi$. From this and from ϕ : ALL, it follows that $(\bar{\mathcal{D}}, \bar{\tau}, v, 0) \models \phi$, which is a contradiction. Hence, $(\bar{\mathcal{D}}', \bar{\tau}', v, 0) \not\models \phi$ and ϕ has (I2).
2. We first show that $\Box\, \phi$ has (I1). Suppose that ϕ has the property ONE and that $(\bar{\mathcal{D}}, \bar{\tau}, v, 0) \models \Box\, \phi$ for some valuation v. Then $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$ for all time points $i \in \mathbb{N}$. Since ϕ has the property ONE, it follows that $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \models \phi$ for all corresponding time points $i' \in \mathbb{N}$. Hence $(\bar{\mathcal{D}}', \bar{\tau}', v, 0) \models \Box\, \phi$ and $\Box\, \phi$ has (I1).
   We continue to show that $\Box\, \phi$ has also (I2). Suppose that ϕ has the property ONE and that $(\bar{\mathcal{D}}, \bar{\tau}, v, 0) \not\models \Box\, \phi$ for some valuation v. Then $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \not\models \phi$ for some time point $i \in \mathbb{N}$. To achieve a contradiction, suppose that $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \models \phi$, where i' is the time point corresponding to i. But from this and ϕ : ONE it follows that $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$, which is a contradiction. Hence, $(\bar{\mathcal{D}}', \bar{\tau}', v, i') \not\models \phi$, so that $(\bar{\mathcal{D}}', \bar{\tau}', v, 0) \not\models \Box\, \phi$ and $\Box\, \phi$ has (I2). □

We now prove the other part of Theorem 4.6, which states that a formula ϕ can be labeled in time linear in its length. We start with some definitions and then present a simple labeling algorithm and analyze its complexity.

For a formula ϕ, we define its immediate subformulas $isub(\phi)$ to be: (i) $\{\psi\}$ if $\phi = \neg\psi$, $\phi = \exists x.\, \psi$, $\phi = \ominus_I\, \psi$, or $\phi = \bigcirc_I\, \psi$; (ii) $\{\psi, \chi\}$ if $\phi = \psi \wedge \chi$, $\phi = \psi\, \mathsf{S}_I\, \chi$, or $\phi = \psi\, \mathsf{U}_I\, \chi$; and (iii) $\emptyset$ otherwise. For a rule r, we denote $\ell(r)$ the label of the rule's conclusion.

We assume that the data structure used to represent formulas is a tree corresponding to the formula's syntax tree and that

each node in the tree also stores two bits to represent the two different labels. Initially these bits are set to 0, meaning that no label is associated with the corresponding subformula.

```
1 add_labels(φ)
2   foreach ψ ∈ isub(φ)
3     add_labels(ψ)
4   foreach rule r
5     if matches(φ, r) then
6       add_label(φ, ℓ(r))
```

The function $matches(\phi, r)$ checks if the formula ϕ pattern matches a rule r. The order of rules is arbitrary, with the exception that the weakening rules are checked last. So, for instance if ϕ received label ALL, then ϕ will match the appropriate weakening rule and it will also be labeled with ONE. As rules have constant size, and only at most the first two levels of the tree representing the formula ϕ need to be inspected, we conclude that the function executes in constant time.

The function $add\_label(\phi, \ell)$ simply adds the label ℓ to ϕ. Clearly, this operation can be performed in constant time.

Note that the execution of the lines 2 and 4–6 takes constant time: $|isub(\phi)| \leq 2$ for any ϕ, there is a fixed, constant number of rules, and the functions *matches* and *add_label* execute in constant time. Furthermore, the function *add_labels* is executed once for each subformula of ϕ. Hence the whole labeling procedure of ϕ takes time linear in the length of ϕ.

B.5 Labeling Rules (Collapse)

In this section, we show the correctness of Lemma 5.4, that is, the soundness of the labeling rules shown in Figure 5, and of the rules shown in Figure 6.

Proof: We first show the correctness of the labeling rules from Figure 5. We proceed by induction on the size of the derivation tree assigning label ℓ to ϕ. We make a case distinction based on the rule applied to label the formula, that is, the rule at the tree's root. However, for clarity, we generally group cases by the formula's form.

Let $(\bar{\mathcal{C}}, \bar{\kappa})$ be the collapse of an interleaving of two given temporal structures. For readability, and without loss of generality, we already fix an arbitrary valuation v, an arbitrary time point i, and an arbitrary temporal structure $(\bar{\mathcal{D}}, \bar{\tau}) \in col^{-1}(\bar{\mathcal{C}}, \bar{\kappa})$.

We first consider the weakening rules:

- ϕ is labeled with $(\models \forall)$ and $(\models \exists)$. Suppose that $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \phi$. By the induction hypothesis, ϕ has the property $(\models \forall)$, thus $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \phi$ for any j with $\tau_j = \kappa_i$. By the definition of $(\bar{\mathcal{C}}, \bar{\kappa})$, there is at least one j with $\tau_j = \kappa_i$. Hence ϕ has the property $(\models \exists)$.
- ϕ is labeled with $(\not\models \forall)$ and with $(\not\models \exists)$. This case is analogous to the previous one.

Next, we make a case distinction on the form of the formula. Consider formulas of the form:

- $\phi = t \approx t'$, where t and t' are variables or constants. In this case ϕ is labeled with $(\models \forall)$ and $(\not\models \forall)$.
  - ϕ is labeled with $(\models \forall)$. Suppose that $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \phi$. Then $v(t) = v(t')$. Clearly, $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \phi$ for any time point j, as ϕ only depends on the valuation. The property $(\models \forall)$ is hence satisfied.
  - ϕ is labeled with $(\not\models \forall)$. This case is analogous to the previous one.
- $\phi = t \prec t'$, where t and t' are variables or constants. This case is analogous to the previous one.
- $\phi = r(t_1, \ldots, t_{\iota(r)})$, where $t_1, \ldots, t_{\iota(r)}$ are variables or constants. In this case ϕ is labeled with $(\models \exists)$ and $(\not\models \forall)$.
  - ϕ is labeled with $(\models \exists)$. Suppose that $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \phi$. Then $(v(t_1), \ldots, v(t_{\iota(r)})) \in r^{\mathcal{C}_i}$. As $r^{\mathcal{C}_i} = \bigcup_{\{j \mid \tau_j = \kappa_i\}} r^{\mathcal{D}_j}$, there is a j with $\tau_j = \kappa_i$ such that $(v(t_1), \ldots, v(t_{\iota(r)})) \in r^{\mathcal{D}_j}$. Therefore $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \phi$. Thus ϕ has the property $(\models \exists)$.
  - ϕ is labeled with $(\not\models \forall)$. Suppose that $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \not\models \phi$. Then for any j with $\tau_j = \kappa_i$ we have that $(v(t_1), \ldots, v(t_{\iota(r)})) \notin r^{\mathcal{D}_j}$, that is, $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \not\models \phi$. Thus ϕ has the property $(\not\models \forall)$.
- $\phi = \neg\psi$. If ψ is labeled with ℓ, then ϕ is labeled with $\neg\ell$, where $\neg\ell$ is $(\models \forall)$, $(\not\models \forall)$, $(\not\models \exists)$, or $(\models \exists)$ when ℓ is $(\not\models \forall)$, $(\models \forall)$, $(\models \exists)$, or $(\not\models \exists)$, respectively.
  - ϕ is labeled with $(\models \forall)$. Suppose that $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \neg\psi$. By the induction hypothesis, ψ has the property $(\not\models \forall)$. As $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \not\models \psi$, we have that $(\bar{\mathcal{D}}, \bar{\tau}, v, k) \not\models \psi$, that is, $(\bar{\mathcal{D}}, \bar{\tau}, v, k) \models \phi$, for all k with $\tau_k = \kappa_i$. Thus ϕ has the property $(\models \forall)$.
  - The other cases are similar.
- $\phi = \psi \vee \chi$. There are four rules to be analyzed.
  - ϕ, ψ, and χ are labeled with $(\not\models \forall)$. Suppose that $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \not\models \psi \vee \chi$. Then $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \not\models \psi$ and $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \not\models \chi$. By the induction hypothesis, ψ and χ have the property $(\not\models \forall)$. Hence, for all j with $\tau_j = \kappa_i$, we have $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \not\models \psi$ and $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \not\models \chi$. Thus $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \not\models \phi$ for all j with $\tau_j = \kappa_i$. Hence, ϕ has the property $(\not\models \forall)$.
  - The other cases are similar.
- $\phi = \exists x.\psi$. There are four rules, one for each label: if ψ is labeled with ℓ, then ϕ is labeled with ℓ.
  - ℓ is $(\models \forall)$. Suppose that $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \exists x.\psi$. Then there is a $d \in |\bar{\mathcal{D}}|$ such that $(\bar{\mathcal{C}}, \bar{\kappa}, v[x/d], i) \models \psi$. As ψ has the property $(\models \forall)$, we have $(\bar{\mathcal{D}}, \bar{\tau}, v[x/d], j) \models \psi$ for all j with $\tau_j = \kappa_i$. That is, $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \exists x.\psi$ for all j with $\tau_j = \kappa_i$. Hence ϕ has the property $(\models \forall)$.
  - The other cases are similar.
- $\phi = \psi \,\mathsf{S}_I\, \chi$. We have three rules to analyze.
  - ϕ, ψ, and χ are each labeled with $(\models \forall)$. By the induction hypothesis, ψ and χ have the property $(\models \forall)$. Suppose that $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \phi$. Then, for some $j \leq i$ with $\kappa_i - \kappa_j \in I$, we have $(\bar{\mathcal{C}}, \bar{\kappa}, v, j) \models \chi$ and $(\bar{\mathcal{C}}, \bar{\kappa}, v, k) \models \psi$ for all $k \in [j+1, i+1)$. Let i' be an arbitrary time point such that $\tau_{i'} = \kappa_i$. As χ has the property $(\models \forall)$, for the largest j' with $\tau_{j'} = \kappa_j$ we have $(\bar{\mathcal{D}}, \bar{\tau}, v, j') \models \chi$. Clearly, $\tau_{i'} - \tau_{j'} \in I$. From the definition of $(\bar{\mathcal{C}}, \bar{\kappa})$, for any $k' \in [j'+1, i'+1)$, there is a $k \in [j+1, i+1)$ such that $\tau_{k'} = \kappa_k$. Then, as ψ has the property $(\models \forall)$, for any $k' \in [j'+1, i'+1)$, we have $(\bar{\mathcal{D}}, \bar{\tau}, v, k') \models \psi$. As ψ has the property $(\models \forall)$, for all $k \in [j+1, i+1)$ and all k' with $\tau_{k'} = \kappa_k$, we have $(\bar{\mathcal{D}}, \bar{\tau}, v, k') \models \psi$. Hence $(\bar{\mathcal{D}}, \bar{\tau}, v, i') \models \psi \,\mathsf{S}_I\, \chi$, and thus ϕ has the property $(\models \forall)$.
  - ϕ, ψ, and χ are each labeled with $(\not\models \forall)$. By the

induction hypothesis, ψ and χ have the property ($\not\models\forall$). Suppose that $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \not\models \phi$. Furthermore, to achieve a contradiction, suppose that ϕ does not have the property ($\not\models\forall$). That is, there is an i' with $\tau_{i'} = \kappa_i$ such that $(\bar{\mathcal{D}}, \bar{\tau}, v, i') \models \phi$. Then there is a $j' \leq i'$ with $\tau_{i'} - \tau_{j'} \in I$ such that $(\bar{\mathcal{D}}, \bar{\tau}, v, j') \models \chi$ and for all $k' \in [j'+1, i'+1)$ we have $(\bar{\mathcal{D}}, \bar{\tau}, v, k) \models \psi$. By the definition of $(\bar{\mathcal{C}}, \bar{\kappa})$, there is a j with $\kappa_j = \tau_{j'}$. As χ has the property ($\not\models\forall$), we have that $(\bar{\mathcal{C}}, \bar{\kappa}, v, j) \models \chi$. Similarly, we have that $(\bar{\mathcal{C}}, \bar{\kappa}, v, k) \models \psi$ for all $k \in [j+1, i+1)$. That is, $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \phi$, which is a contradiction.

- ϕ and ψ are labeled with ($\not\models\exists$), and χ is labeled by ($\not\models\forall$). By the induction hypothesis, ψ and χ have the properties ($\not\models\exists$) and ($\not\models\forall$), respectively. As before, suppose that $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \not\models \phi$. Furthermore, to achieve a contradiction, suppose that ϕ does not have the property ($\not\models\exists$). That is, for all i' with $\tau_{i'} = \kappa_i$ we have $(\bar{\mathcal{D}}, \bar{\tau}, v, i') \models \phi$. Consider the largest such i'. Then there is a $j' \leq i'$ with $\tau_{i'} - \tau_{j'} \in I$ such that $(\bar{\mathcal{D}}, \bar{\tau}, v, j') \models \chi$ and for all $k' \in [j'+1, i'+1)$ we have $(\bar{\mathcal{D}}, \bar{\tau}, v, k') \models \psi$. By the definition of $(\bar{\mathcal{C}}, \bar{\kappa})$, there is a j with $\kappa_j = \tau_{j'}$. As χ has the property ($\not\models\forall$), we have that $(\bar{\mathcal{C}}, \bar{\kappa}, v, j) \models \chi$. Take $k \in [j+1, i+1)$ arbitrarily. If $(\bar{\mathcal{C}}, \bar{\kappa}, v, k) \not\models \psi$, as ψ has the property ($\not\models\exists$), then there is a k' with $\tau_{k'} = \kappa_k$ such that $(\bar{\mathcal{D}}, \bar{\tau}, v, k') \not\models \psi$. This contradicts our assumption that $(\bar{\mathcal{D}}, \bar{\tau}, v, i') \models \phi$, since k' must be in the interval $[j'+1, i'+1)$. We thus have that $(\bar{\mathcal{C}}, \bar{\kappa}, v, k) \models \psi$ for all $k \in [j+1, i+1)$. Hence $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \phi$, which is a contradiction.

- $\phi = \psi \,\mathsf{U}_I\, \chi$. This case is analogous to the previous one.
- $\phi = (\psi \,\mathsf{S}_I\, \chi) \wedge (\Box_J \psi)$ with $0 \notin I$ and $0 \in J$. ϕ and χ are labeled with ($\not\models\forall$), and ψ is labeled by ($\not\models\exists$). By the induction hypothesis, ψ and χ have the properties ($\not\models\exists$) and ($\not\models\forall$), respectively. Suppose that $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \not\models \phi$. Furthermore, to achieve a contradiction, suppose that ϕ does not have the property ($\not\models\forall$). That is, there is an i' with $\tau_{i'} = \kappa_i$ such that $(\bar{\mathcal{D}}, \bar{\tau}, v, i') \models \phi$. Then there is a $j' \leq i'$ with $\tau_{i'} - \tau_{j'} \in I$ such that $(\bar{\mathcal{D}}, \bar{\tau}, v, j') \models \chi$ and for all $k' \in [j'+1, i'+1)$ we have $(\bar{\mathcal{D}}, \bar{\tau}, v, k') \models \psi$; and for all $j'' \geq i'$ with $\tau_{j''} - \tau_{i'} \in J$ we have $(\bar{\mathcal{D}}, \bar{\tau}, v, j'') \models \psi$.
  By the definition of $(\bar{\mathcal{C}}, \bar{\kappa})$, there is a j with $\kappa_j = \tau_{j'}$. As χ has the property ($\not\models\forall$), we have that $(\bar{\mathcal{C}}, \bar{\kappa}, v, j) \models \chi$. Take $k \in [j+1, i)$ arbitrarily. If $(\bar{\mathcal{C}}, \bar{\kappa}, v, k) \not\models \psi$, as ψ has the property ($\not\models\exists$), then there is a k' with $\tau_{k'} = \kappa_k$ such that $(\bar{\mathcal{D}}, \bar{\tau}, v, k') \not\models \psi$. This contradicts our assumption that $(\bar{\mathcal{D}}, \bar{\tau}, v, i') \models \phi$. Indeed, k' must be in the interval $[j'+1, i''+1)$, where i'' is the largest time point such that $\tau_{i''} = \kappa_i$. If $k' \leq i'$ then $(\bar{\mathcal{D}}, \bar{\tau}, v, i') \not\models \psi \,\mathsf{S}_I\, \chi$. If $k' > i'$ then $(\bar{\mathcal{D}}, \bar{\tau}, v, i') \not\models \Box_J \psi$, as $0 \in J$. We thus have that $(\bar{\mathcal{C}}, \bar{\kappa}, v, k) \models \psi$ for all $k \in [j+1, i+1)$. Hence $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \psi \,\mathsf{S}_I\, \chi$.
  As $(\bar{\mathcal{D}}, \bar{\tau}, v, i') \models \Box_J \psi$ and $0 \in J$, it follows that for all $k' \geq i'$ with $\tau_{k'} = \tau_{i'}$ we have $(\bar{\mathcal{D}}, \bar{\tau}, v, k') \models \psi$. We have seen that $(\bar{\mathcal{D}}, \bar{\tau}, v, k') \models \psi$ for all $k' \in [j'+1, i'+1)$. Because $\tau_{j'} < \tau_{i'}$ (as $0 \notin I$), it also follows that for all $k' \leq i'$ with $\tau_{k'} = \tau_{i'}$ we have $(\bar{\mathcal{D}}, \bar{\tau}, v, k') \models \psi$. Hence $(\bar{\mathcal{D}}, \bar{\tau}, v, k') \models \psi$ for all k' with $\tau_{k'} = \tau_{i'}$. As ψ has the property ($\not\models\exists$), we obtain that $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \psi$. Similarly, we obtain that $(\bar{\mathcal{C}}, \bar{\kappa}, v, k) \models \psi$ for all $k > i$ such that $\kappa_k - \kappa_i \in J$. Hence $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \Box_J \psi$.
  We showed that $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \phi$, which is a contradiction. Thus ϕ has the property ($\not\models\forall$).
- $\phi = (\psi \,\mathsf{U}_I\, \chi) \wedge (\Box_J \psi)$ with $0 \notin I$ and $0 \in J$. This case is analogous to the previous one.
- $\phi = \Diamond_I \psi$. There are two rules to analyze. For both rules, ψ is labeled with ($\models\exists$). Suppose that $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \phi$. Then there is a $j \leq i$ with $\kappa_i - \kappa_j \in I$ such that $(\bar{\mathcal{C}}, \bar{\kappa}, v, j) \models \psi$. As, by the induction hypothesis, ψ has the property ($\models\exists$), there is a j' with $\tau_{j'} = \kappa_j$ such that $(\bar{\mathcal{D}}, \bar{\tau}, v, j') \models \psi$.
  - ϕ is labeled with ($\models\exists$). Take i' to be the largest k such that $\tau_k = \kappa_i$. Clearly, $\tau_{i'} - \tau_{j'} \in I$ and $j' \leq i'$. Hence $(\bar{\mathcal{D}}, \bar{\tau}, v, i') \models \Diamond_I \psi$ and ϕ has the property ($\models\exists$).
  - $0 \notin I$ and ϕ is labeled with ($\models\forall$). Take i' arbitrarily such that $\tau_{i'} = \kappa_i$. Clearly, $\tau_{i'} - \tau_{j'} \in I$ and, as $0 \notin I$, $\tau_{i'} - \tau_{j'} > 0$, thus $j' < i'$. Hence $(\bar{\mathcal{D}}, \bar{\tau}, v, i') \models \Diamond_I \psi$. Thus ϕ has the property ($\models\forall$).
- $\phi = \Diamond_I \psi$. This case is analogous to the previous one.
- $\phi = \Diamond_I \Diamond_J \psi$ with $0 \in I \cap J$. There is only one rule to consider: ψ is labeled with ($\models\exists$) and ϕ is labeled by ($\models\forall$). Suppose that $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \phi$. Then there is a $j \leq i$ with $\kappa_i - \kappa_j \in I$ and there is a $k \geq j$ with $\kappa_k - \kappa_j \in J$ such that $(\bar{\mathcal{C}}, \bar{\kappa}, v, k) \models \psi$. As, by the induction hypothesis, ψ has the property ($\models\exists$), there is a k' with $\tau_{k'} = \kappa_k$ such that $(\bar{\mathcal{D}}, \bar{\tau}, v, k') \models \psi$. Take i' arbitrarily such that $\tau_{i'} = \kappa_i$. If $k' \geq i'$ then $0 \leq \tau_{k'} - \tau_{i'} = \kappa_k - \kappa_i \leq \kappa_k - \kappa_j \in J$. As $0 \in J$, we have $\tau_{k'} - \tau_{i'} \in J$. Thus $(\bar{\mathcal{D}}, \bar{\tau}, v, i') \models \Diamond_J \psi$ and, as $0 \in I$, $(\bar{\mathcal{D}}, \bar{\tau}, v, i') \models \Diamond_I \Diamond_J \psi$. The case when $k' < i'$ is similar. Hence ϕ has the property ($\models\forall$).

We continue to show the soundness of the rules for the Boolean operator $\wedge$, the quantifier $\forall$, and the temporal operators trigger T_I and release R_I, shown in Figure 6. The soundness of these rules follows from the soundness of the rules in Figure 5 and the mentioned equivalences. For instance, the correctness of the rule

$$\frac{\psi : (\models\exists) \quad \chi : (\models\forall)}{(\psi \,\mathsf{T}_I\, \chi) \vee (\Diamond_J \psi) : (\models\forall)}\ 0 \notin I, 0 \in J$$

follows from unfolding the abbreviation $(\psi \mathsf{T}_I \chi) \vee (\Diamond_J \psi)$, which is $\neg((\neg\psi \,\mathsf{S}_I\, \neg\chi) \wedge (\Box_J \neg\psi))$, and the following derivation:

$$\frac{\dfrac{\dfrac{\psi : (\models\exists)}{\neg\psi : (\not\models\exists)} \quad \dfrac{\chi : (\models\forall)}{\neg\chi : (\not\models\forall)}}{(\neg\psi \,\mathsf{S}_I\, \neg\chi) \wedge (\Box_J \neg\psi) : (\not\models\forall)}\ 0 \notin I, 0 \in J}{\neg((\neg\psi \,\mathsf{S}_I\, \neg\chi) \wedge (\Box_J \neg\psi)) : (\models\forall)}$$

□

B.6 Collapse-sufficient Formulas

In this section, we show the correctness of Lemma 5.5 and Theorem 5.6.

The implication in Theorem 5.6 follows directly from Lemma 5.5, which in turn follows from the correctness of the derivation rules (Lemma 5.4) and from the following lemma.

Lemma B.6. *Let ϕ be a formula.*

1. If ϕ has the property ($\models\forall$), then ϕ has property (C1).

2. *If ϕ has the property ($\not\models\forall$), then ϕ has property (C2).*
3. *If ϕ has the property ($\models\exists$), then $\Diamond\phi$ has property (C1).*
4. *If ϕ has the property ($\not\models\exists$), then $\Box\phi$ has property (C2).*

Proof: We fix a temporal structure $(\bar{\mathcal{C}}, \bar{\kappa})$.

1. Suppose ϕ has the property ($\models\forall$) and that $(\bar{\mathcal{C}}, \bar{\kappa}, v, 0) \models \phi$ for some valuation v. Then, for any $(\bar{\mathcal{D}}, \bar{\tau}) \in col^{-1}(\bar{\mathcal{C}}, \bar{\kappa})$ and every $j \in \mathbb{N}$ with $\kappa_0 = \tau_j$, it holds that $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \phi$. By the definition of collapsed temporal structure, we have $\kappa_0 = \tau_0$. Hence ϕ has (C1).
2. This case is analogous to the previous one.
3. Suppose ϕ has the property ($\models\exists$) and that $(\bar{\mathcal{C}}, \bar{\kappa}, v, 0) \models \Diamond\phi$ for some arbitrary valuation v. Then $(\bar{\mathcal{C}}, \bar{\kappa}, v, i) \models \phi$ for some $i \in \mathbb{N}$. Because ϕ has the property ($\models\exists$), for every $(\bar{\mathcal{D}}, \bar{\tau}) \in col^{-1}(\bar{\mathcal{C}}, \bar{\kappa})$, there is some $j \in \mathbb{N}$ with $\kappa_i = \tau_j$ such that $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \phi$. It follows that $(\bar{\mathcal{D}}, \bar{\tau}, v, 0) \models \Diamond\phi$. Hence $\Diamond\phi$ has (C1).
4. This case is analogous to the previous one. □

The proof for the complexity of the labeling procedure is analogous to the proof for Theorem 4.6. The only difference is in using four bits for the four different labels instead of using two bits for two labels.

B.7 Policy Approximation

In this section, we show the correctness of Theorem 6.2 concerned with weakening and strengthening formulas.

Proof: We first show that ϕ^w is weaker than ϕ, or more precisely, that the formula $\phi \rightarrow \phi^w$ is valid. We proceed by structural induction on ϕ.

- $\phi = t \approx t'$, $\phi = t \prec t'$, $\phi = \neg(t \approx t')$, $\phi = \neg(t \prec t')$, or $\neg r(t_1, \ldots, t_{\iota(r)})$, where t, t', and t_i with $1 \leq i \leq \iota(r)$ are variables or constants. Then $\phi^w = \phi$, and the statement clearly holds.
- $\phi = r(t_1, \ldots, t_{\iota(r)})$. Then $\phi^w = \Diamond_J \Diamond_{J'} r(t_1, \ldots, t_{\iota(r)})$, for some intervals J and J' with $0 \in J \cap J'$. Let $(\bar{\mathcal{D}}, \bar{\tau})$ be a temporal structure, v a valuation, and i a time point. Suppose that $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$. As $0 \in I \cap J$, we clearly have $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \Diamond_J \Diamond_{J'} \phi$, that is, $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi'$.
- $\phi = \psi \wedge \chi$, $\phi = \exists x.\, \psi$, $\phi = \ominus_I \psi$, $\phi = \bigcirc_I \psi$, $\phi = \psi\ \mathsf{S}_I\ \chi$, or $\phi = \psi\ \mathsf{U}_I\ \chi$. These cases follow directly from the induction hypotheses. We only present the case $\phi = \psi\ \mathsf{S}_I\ \chi$. We have $\phi^w = \psi^w\ \mathsf{S}_I\ \chi^w$. Let $(\bar{\mathcal{D}}, \bar{\tau})$ be a temporal structure, v a valuation, and i a time point. Suppose that $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$. Then there is a $j \leq i$ with $\tau_i - \tau_j \in I$ such that $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \chi$ and $(\bar{\mathcal{D}}, \bar{\tau}, v, k) \models \psi$ for any $k \in [i+1, j+1)$. Using the induction hypotheses for ψ and χ, we obtain that $(\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \chi^w$ and $(\bar{\mathcal{D}}, \bar{\tau}, v, k) \models \psi^w$ for any $k \in [i+1, j+1)$. Hence $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi^w$.

The proof of the dual case, that is, that the formula $\phi^s \rightarrow \phi$ is valid, is similar. It is based on the remark that the formula $(\neg \Diamond_J \Diamond_{J'} r(t_1, \ldots, t_{\iota(r)})) \rightarrow \neg r(t_1, \ldots, t_{\iota(r)})$ is valid.

Finally, we prove Statement (1). Statement (2) is similar. Let $(\bar{\mathcal{C}}, \bar{\kappa})$ be the collapse of two temporal structures $(\bar{\mathcal{D}}^1, \bar{\tau}^1)$ and $(\bar{\mathcal{D}}^2, \bar{\tau}^2)$. Suppose that ϕ^s is collapse-sufficient and that $(\bar{\mathcal{C}}, \bar{\kappa}, v, 0) \models \phi^s$, for some arbitrary valuation v. It follows that $(\bar{\mathcal{D}}, \bar{\tau}, v, 0) \models \phi^s$ for any $(\bar{\mathcal{D}}, \bar{\tau}) \in (\bar{\mathcal{D}}^1, \bar{\tau}^1) \between (\bar{\mathcal{D}}^2, \bar{\tau}^2)$. As $\phi^s \rightarrow \phi$ is valid, we have that $(\bar{\mathcal{D}}, \bar{\tau}, v, 0) \models \phi$, for any $(\bar{\mathcal{D}}, \bar{\tau}) \in (\bar{\mathcal{D}}^1, \bar{\tau}^1) \between (\bar{\mathcal{D}}^2, \bar{\tau}^2)$. □

B.8 Relationship between Interleaving-sufficient and Collapse-sufficient Formulas

In this section, we show the correctness of Theorem 6.1 claiming that if an MFOTL formula is collapse-sufficient, then it is also interleaving-sufficient.

Proof: Suppose that ϕ is a collapse-sufficient MFOTL formula. Moreover, let $(\bar{\mathcal{D}}_1, \bar{\tau}_1)$, $(\bar{\mathcal{D}}_2, \bar{\tau}_2)$, $(\bar{\mathcal{D}}, \bar{\tau})$, and $(\bar{\mathcal{C}}, \bar{\kappa})$ be temporal structures where $(\bar{\mathcal{D}}, \bar{\tau}) \in (\bar{\mathcal{D}}_1, \bar{\tau}_1) \between (\bar{\mathcal{D}}_2, \bar{\tau}_2)$ and $(\bar{\mathcal{C}}, \bar{\kappa}) = col((\bar{\mathcal{D}}, \bar{\tau}))$. We fix an arbitrary valuation v. There are two cases to consider: either ϕ is satisfied on $(\bar{\mathcal{D}}, \bar{\tau})$ or it is not.

First, if $(\bar{\mathcal{D}}, \bar{\tau}, v, 0) \models \phi$, then also $(\bar{\mathcal{C}}, \bar{\kappa}, v, 0) \models \phi$. To see this, suppose that $(\bar{\mathcal{C}}, \bar{\kappa}, v, 0) \not\models \phi$. As ϕ has the property (C2), it would follow that $(\bar{\mathcal{D}}, \bar{\tau}, v, 0) \not\models \phi$, which is a contradiction. From $(\bar{\mathcal{C}}, \bar{\kappa}, v, 0) \models \phi$ and ϕ having the property (C1), it follows that $(\bar{\mathcal{D}}', \bar{\tau}', v, 0) \models \phi$ for all $(\bar{\mathcal{D}}', \bar{\tau}') \in (\bar{\mathcal{D}}_1, \bar{\tau}_1) \between (\bar{\mathcal{D}}_2, \bar{\tau}_2)$. Hence, ϕ has the property (I1).

Second, if $(\bar{\mathcal{D}}, \bar{\tau}, v, 0) \not\models \phi$, then since ϕ has the property (C1), it follows that $(\bar{\mathcal{C}}, \bar{\kappa}, v, 0) \not\models \phi$. Since ϕ has the property (C2), it follows that $(\bar{\mathcal{D}}', \bar{\tau}', v, 0) \not\models \phi$, for all $(\bar{\mathcal{D}}', \bar{\tau}') \in (\bar{\mathcal{D}}_1, \bar{\tau}_1) \between (\bar{\mathcal{D}}_2, \bar{\tau}_2)$. Hence, ϕ has the property (I2).

Since ϕ has the properties (I1) and (I2), it is interleaving-sufficient. □

Acknowledgments

This work was supported by the Nokia Research Center, Switzerland. The authors thank Imad Aad, Debmalya Biswas, Olivier Bornet, Olivier Dousse, Juha Laurila, and Valtteri Niemi for valuable input.

References

[1] D. Basin, M. Harvan, F. Klaedtke, and E. Zălinescu, "Monitoring usage-control policies in distributed systems," in *Proceedings of the 18th International Symposium on Temporal Representation and Reasoning (TIME)*. IEEE Computer Society, 2011, pp. 88–95.

[2] "The Health Insurance Portability and Accountability Act of 1996 (HIPAA)," 104th Congress, 1996, Public Law 104-191.

[3] "Sarbanes-Oxley Act of 2002," 107th Congress, 2002, Public Law 107-204.

[4] "Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data," 1995.

[5] M. Roger and J. Goubault-Larrecq, "Log auditing through model-checking," in *Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW)*. IEEE Computer Society, 2001, pp. 220–234.

[6] N. Dinesh, A. K. Joshi, I. Lee, and O. Sokolsky, "Checking traces for regulatory conformance," in *Proceedings of the 8th International Workshop on Runtime Verification (RV)*, ser. Lect. Notes Comput. Sci., vol. 5289, 2008, pp. 86–103.

[7] A. Groce, K. Havelund, and M. Smith, "From scripts to specification: The evaluation of a flight testing effort," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE)*, vol. 2. ACM Press, 2010, pp. 129–138.

[8] H. Barringer, A. Groce, K. Havelund, and M. Smith, "Formal analysis of log files," *J. Aero. Comput. Inform. Comm.*, vol. 7, pp. 365–390, 2010.

[9] S. Hallé and R. Villemaire, "Runtime enforcement of web service message contracts with data," *IEEE Trans. Serv. Comput.*, vol. 5, no. 2, pp. 192–206, 2012.

[10] D. Basin, F. Klaedtke, S. Müller, and B. Pfitzmann, "Runtime monitoring of metric first-order temporal properties," in *Proceedings of the 28th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, ser. Leibiz International Proceedings in Informatics (LIPIcs), vol. 2. Schloss Dagstuhl - Leibniz Center for Informatics, 2008, pp. 49–60.

[11] A. S. Tanenbaum and M. van Steen, *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.
[12] A. Pnueli, "The temporal logic of programs," *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 46–57, 1977.
[13] 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*, ser. Lect. Notes Comput. Sci., vol. 600. Springer, 1991, pp. 74–106.
[14] 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)*, ser. Lect. Notes Comput. Sci., vol. 7186. Springer, 2012, pp. 360–364.
[15] D. Basin, F. Klaedtke, and S. Müller, "Monitoring security policies with metric first-order temporal logic," in *Proceeding of the 15th ACM Symposium on Access Control Models and Technologies (SACMAT)*. ACM Press, 2010, pp. 23–34.
[16] I. Aad and V. Niemi, "NRC data collection campaign and the privacy by design principles," in *Proceedings of the International Workshop on Sensing for App Phones (PhoneSense)*, 2010.
[17] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases: The Logical Level*. Addison Wesley, 1994.
[18] T. Massart, C. Meuter, and L. Van Begin, "On the complexity of partial order trace model checking," *Inform. Process. Lett.*, vol. 106, no. 3, pp. 120–126, 2008.
[19] A. Genon, T. Massart, and C. Meuter, "Monitoring distributed controllers: When an efficient LTL algorithm on sequences is needed to model-check traces," in *Proceedings of the 14th International Symposium on Formal Methods (FM)*, ser. Lect. Notes Comput. Sci., vol. 4085. Springer, 2006, pp. 557–572.
[20] C. M. Chase and V. K. Garg, "Detection of global predicates: Techniques and their limitations," *Distributed Computing*, vol. 11, pp. 191–201, 1998.
[21] L. Lamport, "What good is temporal logic?" in *Proceedings of the IFIP 9th World Computer Congress*, ser. Information Processing, vol. 83. North-Holland, 1983, pp. 657–668.
[22] J. Chomicki, "Efficient checking of temporal integrity constraints using bounded history encoding," *ACM Trans. Database Syst.*, vol. 20, no. 2, pp. 149–186, 1995.
[23] A. Goodloe and L. Pike, "Monitoring distributed real-time systems: A survey and future directions," NASA Langley Research Center, Tech. Rep. NASA/CR-2010-216724, July 2010.
[24] 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)*. IEEE Computer Society, 2004, pp. 418–427.
[25] R. Ramanujam, "Local knowledge assertions in a changing world," in *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*. Morgan Kaufmann, 1996, pp. 1–14.
[26] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, vol. 21, no. 7, pp. 558–565, 1978.
[27] S. Wang, A. Ayoub, O. Sokolsky, and I. Lee, "Runtime verification of traces under recording uncertainty," in *Proceedings of the 2nd International Conference on Runtime Verification (RV)*, ser. Lect. Notes Comput. Sci., vol. 7186. Springer, 2012, pp. 442–456.
[28] A. Bauer and Y. Falcone, "Decentralised LTL monitoring," in *Proceedings of the 18th International Symposium on Formal Methods (FM)*, ser. Lect. Notes Comput. Sci., vol. 7436. Springer, 2012, pp. 85–100.
[29] A. Bauer, M. Leucker, and C. Schallhart, "Model-based runtime analysis of distributed reactive systems," in *Proceedings of the 2006 Australian Software Engineering Conference (ASWEC)*. IEEE Computer Society, 2006.
[30] W. Zhou, O. Sokolsky, B. T. Loo, and I. Lee, "*DMaC*: Distributed monitoring and checking," in *Proceedings of the 9th International Workshop on Runtime Verification (RV)*, ser. Lecture Notes in Computer Science, vol. 5779. Springer, 2009, pp. 184–201.
[31] V. Diekert and G. Rozenberg, Eds., *The Book of Traces*. World Scientific Publishing Co., Inc., 1995.
[32] D. Peled, "Ten years of partial order reduction," in *Proceedings of the 10th International Conference on Computer Aided Verification*, ser. Lect. Notes Comput. Sci., vol. 1427. Springer, 1998, pp. 17–28.
[33] N. Markey and P. Schnoebelen, "Model checking a path," in *Proceedings of the 14th International Conference on Concurrency Theory (CONCUR)*, ser. Lect. Notes Comput. Sci., vol. 2761. Springer, 2003, pp. 248–262.
[34] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to automata theory, languages, and computation*, 2nd ed. Addison-Wesley, 2000.

David Basin is a full professor and has the chair for Information Security at the Department of Computer Science, ETH Zurich since 2003. From 2003–2011 he was founding director of the ZISC, the Zurich Information Security Center. He received his Ph.D. from Cornell University in 1989, and his Habilitation from the University of Saarbrücken in 1996. His research focuses on information security, in particular methods and tools for modeling, building, and validating secure and reliable systems.

Matúš Harvan is a Ph.D. student at the Department of Computer Science, ETH Zurich. He received his M.Sc. from the Jacobs University Bremen in 2007. His research focuses on building reliable and secure IT systems, in particular on monitoring data usage.

Felix Klaedtke is a senior researcher and lecturer at the Department of Computer Science, ETH Zurich. He received his Ph.D. from the Albert-Ludwigs-University Freiburg in 2004. His research focuses on building reliable and secure IT systems. His research interests are in monitoring, algorithmic verification, mathematical logic, and automata theory.

Eugen Zălinescu is a researcher in the Information Security group at the Department of Computer Science, ETH Zurich since 2009. He received his Ph.D. in 2007 from the Henry Poincaré University in Nancy, France for his research on the verification of security protocols. He is currently working on monitoring and enforcement of temporal properties. He is the main developer of the MONPOLY tool.