Monitoring for Silent Actions*

Luca Aceto[1,3], Antonis Achilleos[1], Adrian Francalanza[2], and Anna Ingólfsdóttir[1]

1 School of Computer Science, Reykjavik University, Reykjavik, Iceland
2 Dept. of Computer Science, ICT, University of Malta, Msida, Malta
3 Gran Sasso Science Institute, L'Aquila, Italy

Abstract

Silent actions are an essential mechanism for system modelling and specification. They are used to abstractly report the occurrence of computation steps without divulging their precise details, thereby enabling the description of important aspects such as the branching structure of a system. Yet, their use rarely features in specification logics used in runtime verification. We study monitorability aspects of a branching-time logic that employs silent actions, identifying which formulas are monitorable for a number of instrumentation setups. We also consider defective instrumentation setups that imprecisely report silent events, and establish monitorability results for tolerating these imperfections.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Runtime Verification, Monitorability, Hennessy-Milner Logic with Recursion, Silent Actions

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2017.7

1 Introduction

Runtime verification (RV) [19, 12] is a lightweight verification technique that strives to determine whether a system under scrutiny satisfies or violates a property—typically expressed as a formula from some logic—by incrementally analysing its current execution. In general, the runtime analysis is carried out by a *monitor*, a computational entity that observes the exhibited system execution and reaches a verdict once sufficient evidence is observed; the exhibited execution is characterised by a *trace*, a finite sequence of *events* describing the discrete system computational steps. Although the technique may obtain additional (runtime) information that could be useful for verification purposes, it is generally less expressive than exhaustive approaches such as model checking since the verification analysis is limited to the information inferred from the execution trace under consideration. *Monitorability* thus concerns itself with identifying the properties that are analysable by this runtime analysis.

RV setups typically partition computational steps of systems into two groups. On the one hand, *observable* events are those events that are visible (in full) to external entities such as monitors; they are used in the specifications describing system properties and are reported in the system trace. Observable events usually contain runtime data associated with that event (*e.g.* a method-call event would carry information relating the receiver, the method name and the arguments passed as parameters). On the other hand, *unobservable* events broadly encompass the computational steps that are abstracted away either from system

* This research was supported by the project "TheoFoMon: Theoretical Foundations for Monitorability" (grant number: 163406-051) of the Icelandic Research Fund.

modelling or from the respective property specifications; RV setups may occasionally remove these events from a trace so as to allow for a smoother monitoring process [15, 10].

In this work we investigate events that broadly fall somewhere in between these two groups. Concretely, *silent* events (or actions) are computational steps whose specific nature is not disclosed at the level of abstraction of the system model. Nevertheless the model still provides enough evidence of their manifestation during execution, which may play an important role in capturing vital behavioural aspects of the system: they may describe the branching structure of the modelled system behaviour [20, 16] or provide a measure of computational cost and efficiency [4]. In practice, one comes across various instances of such events. For example, the precise details of reported computational steps may be abstracted away for confidentiality/security reasons. Alternatively, the monitoring setup may be unable to report the details of certain computations due to limitations in the instrumentation technology used. In cyber-physical systems, there are also cases where one could detect the occurrence of certain (internal) computation by way of indirect means, such as via the sound of a running motor or the increase in temperature of an enclosed object. For these reasons, behavioural specifications often include descriptions involving silent actions. However, it is unclear how these silent actions are best handled in an RV setup. It is even less clear to what extent silent actions affect the monitorability of the respective specifications.

Our goals are to develop a foundational framework in which these questions may be addressed, and to logically characterize the properties that are monitorable within this framework. Following our work [13, 10, 1, 14, 2, 12, 11] and that of others [22, 7], we conduct our investigations in a process-calculus setting, where internal actions have long been studied from both behavioural and specification perspectives. Our study considers a standard labelled-transition-system model that represents silent computational steps as τ-transitions [20, 3], and a variant of the modal μ-calculus [17, 18] with *strong* modal operators that also describe τ-transitions. Our main contributions can be found in the middle sections of the paper:

- Section 3 studies the monitorability of this logic *w.r.t.* a number of monitoring setups that handle τ-actions differently, thus generalising the results obtained in [13, 14].
- Sections 4 and 5 investigate the monitorability of the logic for *imperfect* monitoring setups that obscure aspects of the silent system behaviour expressed by the model, and establish results for tolerating such imperfections.

2 Preliminaries

We assume the following disjoint sets: $\textsc{Act}$, a (possibly empty) set containing *external* actions, and $\textsc{Sil}$, a finite set containing *silent* actions. We let α range over $\textsc{Act}$, δ over $\textsc{Sil}$, and μ over $\textsc{Act} \cup \textsc{Sil}$. A *Labelled Transition System* (LTS) on $(\textsc{Act}, \textsc{Sil})$ is a triple

$$L = \langle P, (\textsc{Act}, \textsc{Sil}), \rightarrow_L \rangle,$$

where P is a nonempty set of system states referred to as *processes* $p, q, \ldots$, and $\rightarrow_L \subseteq P \times (\textsc{Act} \cup \textsc{Sil}) \times P$ is a transition relation. We write $p \xrightarrow{\mu}_L q$ instead of $(p, \mu, q) \in \rightarrow_L$ and $p \rightarrow_L q$ if $p \xrightarrow{\delta}_L q$ for some $\delta \in \textsc{Sil}$. We use $p \overset{\mu}{\Rightarrow}_L q$ to mean that, in L, p can derive q using a single μ action and any number of silent actions, that is, $p(\rightarrow_L)^*. \xrightarrow{\mu}_L .(\rightarrow_L)^* q$. We distinguish between (general) traces $s = \mu_1\mu_2 \ldots \mu_r \in (\textsc{Act} \cup \textsc{Sil})^*$ and external traces $t = \alpha_1\alpha_2 \ldots \alpha_r \in \textsc{Act}^*$, and use $p \overset{s}{\Rightarrow}_L q$ to mean $p \xrightarrow{\mu_1}_L . \xrightarrow{\mu_2}_L \ldots \xrightarrow{\mu_r}_L q$ and $p \overset{t}{\Rightarrow}_L q$ to mean $p \overset{\alpha_1}{\Longrightarrow}_L . \overset{\alpha_2}{\Longrightarrow}_L \ldots \overset{\alpha_r}{\Longrightarrow}_L q$. By $p \xrightarrow{\mu}_L$ we mean that there is some q such that $p \xrightarrow{\mu}_L q$. We occasionally omit the subscript L when it is clear from the context.

▶ **Example 1.** The (standard) regular fragment of CCS [20] with grammar:

$$p, q \in \textsc{Proc} ::= \texttt{nil} \quad | \ \mu.p \quad | \ p+q \quad | \ \texttt{rec}x.p \quad | \ x,$$

with x from some countably infinite set of variables, and the transition relation defined as:

$$\textsc{Act}\frac{}{\mu.p \xrightarrow{\mu} p} \qquad \textsc{Rec}\frac{p[\texttt{rec}x.p/x] \xrightarrow{\mu} q}{\texttt{rec}x.p \xrightarrow{\mu} q} \qquad \textsc{SelL}\frac{p \xrightarrow{\mu} p'}{p+q \xrightarrow{\mu} p'} \qquad \textsc{SelR}\frac{q \xrightarrow{\mu} q'}{p+q \xrightarrow{\mu} q'}$$

constitutes the LTS $\langle \textsc{Proc}, (\textsc{Act}, \{\tau\}), \rightarrow\rangle$ where τ is the only silent action. ◀

Properties of processes may be specified via the logic μHML [18], a reformulation of the modal μ-calculus [17].

▶ **Definition 2.** μHML formulae on $(\textsc{Act}, \textsc{Sil})$ are defined by the grammar:

$$\begin{array}{llllll} \varphi, \psi \in \mu\text{HML} ::= & \texttt{tt} & | \ \texttt{ff} & | \ \varphi \wedge \psi & | \ \varphi \vee \psi & \\ & | \ \langle\mu\rangle\varphi & | \ [\mu]\varphi & | \ \texttt{min}\ X.\varphi & | \ \texttt{max}\ X.\varphi & | \ X \end{array}$$

where X comes from a countably infinite set of logical variables LVar. For a given LTS $L = \langle P, (\textsc{Act}, \textsc{Sil}), \rightarrow\rangle$, an environment ρ is a function $\rho : \text{LVar} \rightarrow 2^P$. Given an environment ρ, $X \in \text{LVar}$, and $S \subseteq P$, $\rho[X \mapsto S]$ denotes the environment where $\rho[X \mapsto S](X) = S$ and $\rho[X \mapsto S](Y) = \rho(Y)$, for all $Y \neq X$. The semantics of a μHML formula φ over an LTS L relative to an environment ρ, denoted as $[\![\varphi, \rho]\!]_L$, is defined as follows:

$$[\![\texttt{tt}, \rho]\!]_L = P \qquad [\![\texttt{ff}, \rho]\!]_L = \emptyset \qquad [\![X, \rho]\!]_L = \rho(X)$$

$$[\![\varphi_1 \wedge \varphi_2, \rho]\!]_L = [\![\varphi_1, \rho]\!]_L \cap [\![\varphi_2, \rho]\!]_L \qquad [\![\varphi_1 \vee \varphi_2, \rho]\!]_L = [\![\varphi_1, \rho]\!]_L \cup [\![\varphi_2, \rho]\!]_L$$

$$[\![[\mu]\varphi, \rho]\!]_L = \left\{p \mid \forall q.\ p \xrightarrow{\mu} q \text{ implies } q \in [\![\varphi, \rho]\!]_L\right\}$$

$$[\![\langle\mu\rangle\varphi, \rho]\!]_L = \left\{p \mid \exists q.\ p \xrightarrow{\mu} q \text{ and } q \in [\![\varphi, \rho]\!]_L\right\}$$

$$[\![\texttt{min}\ X.\varphi, \rho]\!]_L = \bigcap \{S \mid S \supseteq [\![\varphi, \rho[X \mapsto S]]\!]_L\}$$

$$[\![\texttt{max}\ X.\varphi, \rho]\!]_L = \bigcup \{S \mid S \subseteq [\![\varphi, \rho[X \mapsto S]]\!]_L\}$$

Two formulae φ and ψ are equivalent, denoted as $\varphi \equiv \psi$, when $[\![\varphi, \rho]\!]_L = [\![\psi, \rho]\!]_L$ for every environment ρ and LTS L. We often consider closed formulae and simply write $[\![\varphi]\!]_L$ for $[\![\varphi, \rho]\!]_L$, as their semantics is independent of ρ. ◀

Let $[\textsc{Sil}]\varphi$ stand for $\bigwedge_{\delta \in \textsc{Sil}}[\delta]\varphi$ and $\langle\textsc{Sil}\rangle\varphi$ for $\bigvee_{\delta \in \textsc{Sil}}\langle\delta\rangle\varphi$. Then, the weak versions of the modalities employed in [13, 1, 14] may be expressed as follows:

$$[[\mu]]\varphi \equiv \texttt{max}\ X.([\mu]wb(\varphi) \wedge [\textsc{Sil}]X) \qquad \langle\langle\mu\rangle\rangle\varphi \equiv \texttt{min}\ X.(\langle\mu\rangle wd(\varphi) \vee \langle\textsc{Sil}\rangle X),$$

where $wb(\varphi) \equiv \texttt{max}\ Y.(\varphi \wedge [\tau]Y)$ and $wd(\varphi) \equiv \texttt{min}\ Y.(\varphi \vee \langle\tau\rangle Y)$. Readers should consult [18, 3], or more recently [14, 1], for more details on μHML.

3 Monitorability

The logic μHML of Section 2 is very expressive. It is also agnostic of the technique to be employed for verification. This level of generality provides an ideal basis for investigating the interplay between silent actions and the RV technique, and permits us to extend our findings to other specification logics (*e.g.* CTL and CTL* [8] can be encoded in μHML [17]). The property of monitorability, however, fundamentally relies on the monitoring setup considered.

Monitor Semantics

$$\textsc{mRec}\,\frac{m[\mathtt{rec}x.m/x] \xrightarrow{\mu} m'}{\mathtt{rec}x.m \xrightarrow{\mu} m'} \qquad \textsc{mSel}\,\frac{m \xrightarrow{\mu} m'}{m+n \xrightarrow{\mu} m'} \qquad \textsc{mAct}\,\frac{}{\mu.m \xrightarrow{\mu} m} \qquad \textsc{mVrd}\,\frac{}{v \xrightarrow{\mu} v}$$

Instrumentation Semantics

$$\textsc{iMon}\,\frac{p \xrightarrow{\mu}_L q \quad m \xrightarrow{\mu}_M n}{m \triangleleft p \xrightarrow{\mu}_{I(M,L)} n \triangleleft q} \qquad \textsc{iTer}\,\frac{p \xrightarrow{\mu}_L q \quad m \not\xrightarrow{\mu}_M}{m \triangleleft p \xrightarrow{\mu}_{I(M,L)} \mathtt{end} \triangleleft q} \qquad \textsc{iAbs}\,\frac{p \xrightarrow{\delta}_L q}{m \triangleleft p \xrightarrow{\delta}_{I(M,L)} m \triangleleft q}$$

where $\mu \in \textsc{Act} \cup \textsc{Sil}$, $v \in \{\mathtt{end}, \mathtt{no}\}$, and $\delta \in \textsc{Sil}$.

Table 1 Behaviour and Instrumentation Rules for Monitored Systems

Monitoring Systems: A *monitoring setup* on $(\textsc{Act}, \textsc{Sil})$ is a triple $S = \langle M, I, L\rangle$, where L is a system LTS on $(\textsc{Act}, \textsc{Sil})$, M is a monitor LTS on $(\textsc{Act}, \textsc{Sil})$, and I is the instrumentation describing how to compose L and M into an LTS, denoted by $I(M, L)$, on $(\textsc{Act}, \textsc{Sil})$. We call the pair (M, I) a *monitoring system* on $(\textsc{Act}, \textsc{Sil})$. For $M = (\textsc{Mon}, (\textsc{Act}, \textsc{Sil}), \rightarrow_M)$, $\textsc{Mon}$ is a set of monitor states (ranged over by m) and $\rightarrow_M$ is the *monitor semantics* described in terms of the behavioural state transitions a monitor takes when it analyses trace events $\mu \in \textsc{Act} \cup \textsc{Sil}$. The states of the composite LTS $I(M, L)$ are written as $m \triangleleft p$, where m is a monitor state and p is a system state; the monitored-system transition relation is here denoted by $\rightarrow_{I(M,L)}$. We focus on *rejection* monitors, *i.e.,* monitors with a designated rejection state no, and hence safety fragments of the logic μHML. However, our arguments apply dually to acceptance monitors and co-safety properties; see [13, 14] for details.

▶ **Definition 3.** Fix a monitoring setup $S = \langle M, I, L\rangle$ on $(\textsc{Act}, \textsc{Sil})$ and let m be a monitor of M and φ a formula of μHML on $(\textsc{Act}, \textsc{Sil})$. We say that m (M, I)*-rejects* (or simply *rejects*, if M, I are evident) a process p in L, written as $\mathbf{rej}_S(m, p)$, when there are a process q in L and a trace $s \in (\textsc{Act} \cup \textsc{Sil})^*$ such that $m \triangleleft p \overset{s}{\Rightarrow}_{I(M,L)} \mathtt{no} \triangleleft q$. We say that m (M, I)*-monitors for* φ on L whenever

$$\text{for each process } p \text{ of } L,\ \mathbf{rej}_S(m, p) \text{ if and only if } p \notin [\![\varphi]\!]_L.$$

Finally, m (M, I)*-monitors for a formula* φ when m (M, I)-monitors for φ on L for every LTS L on $(\textsc{Act}, \textsc{Sil})$. The monitoring system (M, I) is often omitted when evident. ◀

Monitoring for Silent Actions: The first monitoring system we consider does *not* distinguish between silent actions and external actions.

▶ **Definition 4.** A *full monitor* on $(\textsc{Act}, \textsc{Sil})$ is defined by the grammar:

$$m, n \in \textsc{Mon}_\delta ::= \mathtt{end} \quad | \ \mathtt{no} \quad | \ \mu.m \quad | \ m + n \quad | \ \mathtt{rec}\ x.m \quad | \ x,$$

where x comes from a countably infinite set of monitor variables. Constant no denotes the *rejection verdict* state whereas end denotes the *inconclusive verdict* state. The rules in Table 1 describe the behaviour for full monitors (we elide the obvious symmetric rule for $m + n$). ◀

Note that rule $\textsc{mVrd}$ in Table 1 describes how verdicts are irrevocable; monitors can therefore only describe suffix-closed behaviour.

▶ **Definition 5.** For any system LTS L and monitor LTS M agreeing on $(\textsc{Act}, \textsc{Sil})$, a *full instrumentation* LTS, denoted by $\rightarrow_{I(M,L)}$, is defined by rules $\textsc{iMon}$ and $\textsc{iTer}$ in Table 1. ◀

In rule iMon, when the system produces a trace event μ that the monitor is able to analyse by transitioning from m to n, the constituent components of a monitored system $m \triangleleft p$ move in lockstep. Conversely, when the system produces an event μ that the monitor is *unable* to analyse, the monitored system still executes, according to iTer, but the monitor transitions to the inconclusive state, where it remains for the rest of the computation.

We refer to the monitor LTS in Definition 4 as M^δ, the full instrumentation of Definition 5 as I^δ and the pair (M^δ, I^δ) as the *full monitoring system*. For each system LTS L that agrees with the full monitoring system on $(\textsc{Act}, \textsc{Sil})$, we can show a correspondence between the respective monitoring setup $\langle M^\delta, I^\delta, L\rangle$ and the following syntactic subset of μHML.

▶ **Definition 6.** The *strong safety* μHML is defined by the grammar:

$$\theta, \chi \in \text{ssHML} ::= \texttt{tt} \quad | \ \texttt{ff} \quad | \ [\mu]\theta \quad | \ \theta \wedge \chi \quad | \ \texttt{max}\ X.\theta \quad | \ X$$ ◀

As opposed to sHML from [13, 14], ssHML is defined using strong transitions $p \xrightarrow{\mu} q$ (not weak ones, $p \overset{\mu}{\Rightarrow} q$) and the modalities $[\mu]\theta$ employ *any* action μ, not just external ones.

▶ **Definition 7.** Fix a monitoring system (M, I), a fragment Λ of μHML, and an LTS L on $(\textsc{Act}, \textsc{Sil})$. We say that (M, I) monitors for Λ on L whenever:

- For all $\varphi \in \Lambda$, there exists some $m \in M$ that monitors for it on L.
- For all $m \in M$, there exists some $\varphi \in \Lambda$ that is monitored by it on L.

We say that (M, I) monitors for Λ when it monitors for Λ on every LTS L. ◀

▶ **Theorem 8.** *The* full monitoring system (M^δ, I^δ) *monitors for* ssHML*.*

Monitoring for External Actions: The results obtained in [13, 14] can be expressed and recovered within our more general framework.

▶ **Definition 9.** *Safety* μHML, presented in [13, 14], is defined by the grammar:

$$\pi, \kappa \in \text{sHML} ::= \texttt{tt} \quad | \ \texttt{ff} \quad | \ [[\alpha]]\pi \quad | \ \pi \wedge \kappa \quad | \ \texttt{max}\ X.\pi \quad | \ X.$$

Note that $[[\alpha]]\pi$ uses *external actions*. Its semantics is given as in Definition 2. We can also give a direct inductive definition, *i.e.*, $[\![[[\alpha]]\varphi, \rho]\!] = \{p \mid \forall q.\ p \overset{\alpha}{\Rightarrow} q \text{ implies } q \in [\![\varphi, \rho]\!]\}$. ◀

▶ **Definition 10.** An *external monitor* on $(\textsc{Act}, \textsc{Sil})$ is defined by the grammar:

$$m, n \in \textsc{Mon}_\alpha ::= \texttt{end} \quad | \ \texttt{no} \quad | \ \alpha.m \quad | \ m + n \quad | \ \texttt{rec}\ x.m \quad | \ x.$$

Table 1 defines its LTS transition semantics, yielding $M^\alpha = \langle \textsc{Mon}_\alpha, (\textsc{Act}, \textsc{Sil}), \rightarrow\rangle$. *External instrumentation*, denote by I^α, is defined by the *three* rules iMon, iTer, and iAbs in Table 1; in the case of iMon and iTer action μ is substituted by the external action α. We refer to the pair (M^α, I^α) as the *external monitoring system*, amounting to the setup in [13, 14]. ◀

▶ **Theorem 11** (from [14])**.** *The* external monitoring system (M^α, I^α) *monitors for the sublogic* sHML*.* ◀

▶ **Example 12.** Consider a simple server interface that receives requests from a client, represented by action `req`, and then sends a reply, represented by action `ans`. Between `req` and `ans`, a server implementation may upload a copy of the request transcript; this computation is represented as a sequence of silent τ-transitions that do not divulge information relating to the upload. In LTS L_1 of Figure 1, process 1 represents a server implementation

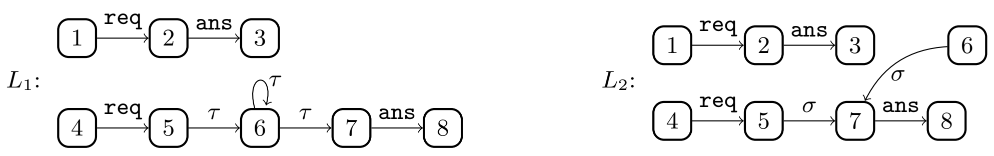

Figure 1 LTS L_1 depicts the two variations of the server from Example 12.

that never uploads anything, whereas process 4 represents an alternative implementation that creates a transcript (the τ-transition from 5 to 6) and repeatedly attempts to upload the copy until it succeeds (the τ-loop on 6 followed by the transition to 7). An external monitor does not see processes 1 and 4 differently, as it does not observe the silent transitions. On the other hand, a full monitor can observe all the silent transitions that occur during an execution. We note that both process 1 and process 4 in L_1 violate the specification $[[\texttt{req}]][[\texttt{ans}]]\texttt{ff}$. Process 1 violates $[\texttt{req}][\texttt{ans}]\texttt{ff}$, while 4 does not. Conversely, process 1 does not violate the sSHML-specification $[\texttt{req}][[\tau]][\texttt{ans}]\texttt{ff}$, but 4 does: this can be observed by the full monitor $\texttt{req}.\texttt{rec}\ x.(\tau.(\texttt{ans}.\texttt{no} + x))$. ◀

We conclude the section by commenting on other potential monitoring systems and their expressive power. In particular, the monitoring system (M^δ, I^α) yields monitoring setups whereby monitor δ-transitions are suppressed by the instrumentation, effectively making full monitors behave like external monitors from Definition 10. In the case of the monitoring system (M^α, I^δ), the instrumentation forces the monitor to transition to the inconclusive state more often since it does not abstract away from δ-transitions.

4 Obscuring the Silent Transitions

The full monitoring system (M^δ, I^δ) presented in Section 3 is straightforward and powerful. One might however argue that, in practice, it is *too* powerful: it is plausible that the visibility of certain silent transitions be somehow more *obscure* than that of external transitions. The external monitoring system (M^α, I^α) sits at the other end of the spectrum because it completely ignores all silent transitions. We consider monitoring systems that fall between these extremes: they can clearly observe certain silent transitions, but may receive imperfect information on others *i.e.,* observing that some number of transitions occurred, but *not* how many. In this case, we say that the transitions were *obscure.*

4.1 A Preorder on Obscure LTSs and Reliable Monitoring

We consider two silent actions: τ is a silent action that can be clearly observed and σ is the *obscure* silent action, representing an undetermined positive number of τ-transitions. In the following, we consider only monitoring setups on $(\textsc{Act}, \{\tau, \sigma\})$ and, whenever we say that L is an LTS, we mean that it is a system LTS on $(\textsc{Act}, \{\tau, \sigma\})$, unless otherwise stated; if L reports perfect information, it is assumed to be an LTS on $(\textsc{Act}, \{\tau\})$.

We consider a preorder $\leq_o$ on LTSs, where $L \leq_o L'$ intuitively means that L and L' have the same processes, but the silent transitions in L' are somehow more obscure than in L. Although we do not identify a specific such preorder, in Subsection 4.2, we introduce properties that we require of it. We say that L' is *an obscuring* of L when $L \leq_o L'$. We also introduce the obscuring preorder $\leq$ on $\textsc{Act} \cup \{\tau, \sigma\}$: $\mu_1 \leq \mu_2$ iff $\mu_1 = \mu_2$ or $\mu_1 = \tau$ and

$\mu_2 = \sigma$. The intuition is that, whenever $\mu_1 \leq \mu_2$ and the system performs a μ_1-transition, the monitor may observe a (more obscure) μ_2-transition.

▶ **Example 13.** Consider a simple LTS L which contains exactly *one* maximal path:

$$p \xrightarrow{\mu_1}_L p_1 \xrightarrow{\mu_2}_L \cdots \xrightarrow{\mu_i}_L p_i \xrightarrow{\tau}_L q_1 \xrightarrow{\tau}_L \cdots \xrightarrow{\tau}_L q_r \xrightarrow{\mu_{i+1}}_L p_{i+1} \xrightarrow{\mu_{i+2}}_L \cdots \xrightarrow{\mu_k}_L p_k$$

of $k+r$ transitions, where $r > 0$; note that states $q_2 \ldots q_{r-1}$ have no outgoing external actions. An obscuring of L may result from replacing $p_i \xrightarrow{\tau}_L q_1$ by a direct transition $p_i \xrightarrow{\sigma}_{L'} q_r$, thus obscuring the path $p_i \xrightarrow{\tau}_L q_1 \xrightarrow{\tau}_L \cdots \xrightarrow{\tau}_L q_r$ and leaving the remaining path unchanged. Thus in L' we have:

$$p \xrightarrow{\mu_1}_{L'} p_1 \xrightarrow{\mu_2}_{L'} \cdots \xrightarrow{\mu_i}_{L'} p_i \xrightarrow{\sigma}_{L'} q_r \xrightarrow{\mu_{i+1}}_{L'} p_{i+1} \xrightarrow{\mu_{i+2}}_{L'} \cdots \xrightarrow{\mu_k}_{L'} p_k$$

This would mean that as the system progresses from p to p_k, μ_1 through μ_k are clearly observed, but when the system performs $p_i \xrightarrow{\tau}_L q_1 \xrightarrow{\tau}_L \cdots \xrightarrow{\tau}_L q_r$, we only observe that at least one silent transition occurred, without discerning the exact number. ◀

▶ **Definition 14.** Let m be a monitor of a monitoring system (M, I); φ a formula of μHML on $(\textsc{Act}, \{\tau\})$; L an LTS on $(\textsc{Act}, \{\tau\})$ — that is, L completely reports silent transitions; and L' an obscuring of L. We say that m (M, I)*-monitors* for φ on L from L' iff

for every process p of L', $p \notin [\![\varphi]\!]_L$ if and only if $\mathbf{rej}_{\langle M,I,L'\rangle}(m, p)$.

We say that m *reliably* (M, I)*-monitors* for φ on L if m (M, I)-monitors for φ on L from each obscuring of L. Monitor m reliably (M, I)-monitors for φ if m reliably (M, I)-monitors for φ on any LTS L. We often omit the monitoring system (M, I) whenever it is evident. ◀

▶ **Definition 15.** Fix a monitoring system (M, I) and a fragment Λ of μHML on $(\textsc{Act}, \{\tau, \sigma\})$. (M, I) *reliably monitors* for Λ on LTS L iff

- For every $\varphi \in \Lambda$, there is a monitor m of M such that m reliably monitors for φ on L.
- For every monitor m of M, there is a $\varphi \in \Lambda$ such that m reliably monitors for φ on L.

(M, I) *reliably monitors* for Λ when (M, I) *reliably monitors* for Λ on every LTS. ◀

4.2 Requirements on Obscuring Preorders

We identify certain properties of the obscuring ordering $\leq_o$ that we consider natural. These properties suffice to prove the results of Section 5. Consequently, the conclusions we draw about reliably monitorable formulas of μHML are proven for every $\leq_o$ that has these properties. Our intuition is that if $L \leq_o L'$, then L' is the same LTS as L, but seen with less precision with respect to the silent transitions. So, every transition we observe in L' is either a transition from L, or an obscure view of a sequence of transitions from L.

Natural Properties of Obscurings. We fix two LTSs $L \leq_o L'$. Since L' should at most provide imperfect information on the *silent* transitions of the system, external transitions should be unaffected:

A. $\xrightarrow{\alpha}_{L'} = \xrightarrow{\alpha}_L$ for every $\alpha \in \textsc{Act}$.

As L' obscures the information on the silent transitions of L, τ-transitions will become fewer: L' should have at most the τ-transitions of L (Property B). Furthermore, every σ-transition in L' represents a non-empty sequence of silent transitions from L (Property C).

B. $\xrightarrow{\tau}_{L'} \subseteq \xrightarrow{\tau}_L$ and

C. $\xrightarrow{\sigma}_{L'} \subseteq (\xrightarrow{\tau}_L \cup \xrightarrow{\sigma}_L)^+$.
The following properties ensure that a certain level of information is retained in L'. In particular, if a state has a silent transition in L, it should still have a silent transition in L' (Property D). Moreover, if a state p has a sequence of silent transitions in L that lead to a state q that can perform an external action, then this observation should be preserved in L'. Following Property A, it suffices to require that q is reachable from p in L' via a sequence of silent actions (Property E).

D. For all p if $p \xrightarrow{\tau}_L$ or $p \xrightarrow{\sigma}_L$, then $p \xrightarrow{\tau}_{L'}$ or $p \xrightarrow{\sigma}_{L'}$.
E. For all p, p', if $p(\xrightarrow{\tau}_L \cup \xrightarrow{\sigma}_L)^+ p' \xrightarrow{\alpha}$ for some $\alpha \in \text{Act}$, then $p(\xrightarrow{\tau}_{L'} \cup \xrightarrow{\sigma}_{L'})^+ p'$.

The Strength of Obscuring. Properties F, G, and H capture the kind of obscuring ordering considered in this paper. We assume that there is a certain level of obscuring, beyond which adequate monitoring is deemed infeasible. In Property F below, obscuring can reach a point, represented as an LTS L^o, where all the silent-action information is hidden. That is, if $p \xrightarrow{\sigma}_{L^o} \xrightarrow{\sigma}_{L^o} p'$, then process p can also perform the more obscure transition $p \xrightarrow{\sigma}_{L^o} p'$ and furthermore, at no point does L^o reveal any clear τ-transition. We call such an obscuring L^o, as described by Property F, a *total* obscuring.

F. Each L has an obscuring L^o, such that $\xrightarrow{\tau}_{L^o} = \emptyset$ and $\xrightarrow{\sigma}_{L^o}$ is transitive.

For an LTS L, let L^τ be the LTS on $(\text{Act}, \{\tau\})$ with the same set of processes, so that for $\alpha \in \text{Act}$, $\xrightarrow{\alpha}_{L^\tau} = \xrightarrow{\alpha}_L$ and $\xrightarrow{\tau}_{L^\tau} = \xrightarrow{\tau}_L \cup \xrightarrow{\sigma}_L$. Property G assures us that we can always obscure any selection of τ-transitions by turning them into σ-transitions, thus "forgetting" how many transitions were taking place at certain points. Property G can also be interpreted to mean that σ-transitions may indeed just represent single τ-transitions.

G. $L^\tau \leq_o L$ for each LTS L.

For the last requirement, we need the following definitions. For a process p in L and a trace $s \in (\text{Act} \cup \{\tau, \sigma\})^*$, we say that p *represents* s *in* L when s is the only maximal trace that p can produce — that is, when $\forall s'.\big(\exists q.\ p \overset{s'}{\Rightarrow}_L q$ iff s' is a prefix of $s\big)$. For a trace $s \in (\text{Act} \cup \{\sigma\})^*$, we define the *total obscuring* of s, denoted as $o(s)$, as follows: $o(\epsilon) = \epsilon$; $o(\sigma^k) = \sigma$ and $o(\sigma^k \alpha s) = \sigma\alpha\, o(s)$ for $k > 0$; and $o(\alpha s) = \alpha\, o(s)$. Property H ensures that any sequence of silent transitions can be obscured into a σ-transition at least for some LTSs:

H. for every trace $s \in (\text{Act} \cup \{\sigma\})^*$, there are LTSs $L \leq_o L'$ and a process p in L, such that p represents s in L and $o(s)$ in L'.
Property H may seem arbitrary, but it is not hard to justify that it is an immediate consequence of our intuition, as depicted in Example 13. Consider a maximal-path LTS L as in Example 13, but with τ-transitions replaced by σ-transitions, such that $s_1 = \mu_1 \cdots \mu_i \in \text{Act}^*$ and $s_2 = \mu_{i+1} \cdots \mu_k \in \text{Act}^*$. Then, p represents $s = s_1 \sigma^k s_2$ in L and $o(s) = s_1 \sigma s_2$ in L'.

4.3 An Ordering of Obscurings

We provide a natural instance of an ordering that has all the properties of Subsection 4.2.

▶ **Definition 16.** Relation $\leq_c$ is the transitive closure of $\leq_1$, where for LTSs L_1 and L_2 on $(\text{Act}, \{\tau, \sigma\})$, $L_1 \leq_1 L_2$ when for every $\alpha \in \text{Act}$, $\xrightarrow{\alpha}_{L_1} = \xrightarrow{\alpha}_{L_2}$ and one of the following holds:

1. $\xrightarrow{\tau}_{L_1} = \xrightarrow{\tau}_{L_2}$ and $\xrightarrow{\sigma}_{L_1} \subseteq \xrightarrow{\sigma}_{L_2} \subseteq \xrightarrow{\sigma}_{L_1} \cup \xrightarrow{\tau}_{L_1}$;
2. $\xrightarrow{\sigma}_{L_1} = \xrightarrow{\sigma}_{L_2}$ and $\xrightarrow{\tau}_{L_2} \subseteq \xrightarrow{\tau}_{L_1} \subseteq \xrightarrow{\sigma}_{L_2} \cup \xrightarrow{\tau}_{L_2}$;

3. $\xrightarrow{\tau}_{L_1} = \xrightarrow{\tau}_{L_2}$ and $\xrightarrow{\sigma}_{L_1} \subseteq \xrightarrow{\sigma}_{L_2} \subseteq (\xrightarrow{\sigma}_{L_1})^+$; or
4. $\xrightarrow{\tau}_{L_1} = \xrightarrow{\tau}_{L_2}$, $\xrightarrow{\sigma}_{L_2} \subseteq \xrightarrow{\sigma}_{L_1}$, and for all $p \xrightarrow{\sigma}_{L_1} p'$, if $p \not\xrightarrow{\sigma}_{L_2} p'$, then all the following hold:
   $p' \not\xrightarrow{\alpha}_{L_1}$ for all $\alpha \in$ ACT,
   $p' \xrightarrow{\sigma}_{L_1} p''$ or $p' \xrightarrow{\tau}_{L_1} p''$ for some $p'' \neq p'$, and
   $p \xrightarrow{\sigma}_{L_2} p''$ for every p'' such that $p' \xrightarrow{\sigma}_{L_1} p''$ or $p' \xrightarrow{\tau}_{L_1} p''$. ◀

The cases presented in Definition 16 give a set of *moves* we can apply to construct a more obscure LTS from a given one. Informally:

1. According to move 1, for any transition $p \xrightarrow{\tau} q$, we can add transition $p \xrightarrow{\sigma} q$.
2. Following move 1, we can remove transition $p \xrightarrow{\tau} q$.
3. For transitions $p \xrightarrow{\sigma} p' \xrightarrow{\sigma} p''$, we can insert a new transition $p \xrightarrow{\sigma} p''$.
4. For transition $p \xrightarrow{\sigma} p'$, if move 3 has already been applied to $p \xrightarrow{\sigma} p' \xrightarrow{\sigma} p''$ for all possible and at least one $p' \xrightarrow{\delta} p''$, where $p' \neq p''$ and $\delta \in \{\tau, \sigma\}$, and $p' \not\xrightarrow{\alpha}$ for all $\alpha \in$ ACT then we can remove $p \xrightarrow{\sigma} p'$.

▶ **Example 17.** We revisit Example 12 of a simple server. The LTS L_2 of Figure 1 presents a maximal obscuring of L_1, according to $\leq_c$. Moves 1 and 2 can replace all τ-transitions by σ-transitions; move 3 can be used to introduce a transition from process 5 directly to process 7; and move 4 can eliminate incoming transitions to process 6, including the self-loop. Thus, the LTS retains the information that the server uploads the transcript to a remote location, but not any information of intermediate steps. We observe that formula $\psi = [\mathtt{req}][[\tau]][\mathtt{ans}]\mathtt{ff}$ is reliably monitorable on L_1 from L_2 by the full monitor $\mathtt{req}.\mathtt{rec}\ x.(\tau.(\mathtt{ans}.\mathtt{no} + x) + \sigma.(\mathtt{ans}.\mathtt{no} + x))$. ◀

▶ **Proposition 18.** *Relation $\leq_c$ has all the properties listed in Subsection 4.2.* ◀

5 Reliable Monitorability

In this section, we identify a maximal reliably monitorable fragment of μHML — up to logical equivalence — and a monitoring system that monitors for it. The results of this section are relative to any fixed preorder $\leq_o$ that satisfies the properties presented in Subsection 4.2.

▶ **Example 19.** Let $\varphi_1 = [\tau][\alpha]\mathtt{ff}$ (*i.e.,* after any τ-action, a process cannot perform an α-action), $\varphi_2 = [\tau]\mathtt{ff}$ (*i.e.,* a process cannot perform a τ-action), and $\varphi_3 = \mathtt{max}\ X.([\tau][\alpha]\mathtt{ff} \wedge [\tau]X)$ (*i.e.,* a process cannot perform an α-action after any non-empty sequence of τ-actions). Notice that $\varphi_3 \equiv [[\tau]][\alpha]\mathtt{ff}$. Let L_1, L_2 be the LTSs described below, where $L_1 \leq_o L_2$:

$$L_1: \quad p_0 \xrightarrow{\tau} p_1 \xrightarrow{\tau} p_2 \xrightarrow{\alpha} p_3 \qquad\qquad L_2: \quad p_0 \xrightarrow{\sigma} p_2 \xrightarrow{\alpha} p_3 \quad \text{and} \quad p_1 \xrightarrow{\sigma} p_2.$$

L_2 is a $\leq_o$-maximal obscuring of L_1: any LTS L' with $L_2 \leq_o L'$ will have to be exactly L_2 according to Properties B through E. LTSs L_1 and L_2 are really instances of LTSs L and L' from Example 13, resp. Consider L_3 described below:

$$L_3: \quad p_0 \xrightarrow{\tau} p_2 \xrightarrow{\alpha} p_3 \quad \text{and} \quad p_1 \xrightarrow{\tau} p_2$$

where L_2 is also an obscuring of L_3, $L_3 \leq_o L_2$. We observe that φ_1 is not reliably monitorable according to Definition 14: $p_0 \in [\![\varphi_1]\!]_{L_1}$ and $p_0 \notin [\![\varphi_1]\!]_{L_3}$, so a monitor that reliably monitors for φ_1 would need to reject and not reject p_0 in L_2. On the other hand, both φ_2 and φ_3 are reliably monitorable *w.r.t.* $\leq_o$. Let

$$m_2 = \sigma.\mathtt{no} + \tau.\mathtt{no} \qquad \text{and} \qquad m_3 = \mathtt{rec}\ x.(\sigma.\alpha.\mathtt{no} + \sigma.x + \tau.\alpha.\mathtt{no} + \tau.x)$$

$$\textsc{iMon}\frac{p \xrightarrow{\mu}_L p' \quad m \xrightarrow{\mu'}_M m' \quad \mu \leq \mu'}{m \triangleleft p \xrightarrow{\mu'}_{I(L,M)} m' \triangleleft p'} \qquad \textsc{iTer}\frac{p \xrightarrow{\mu}_L p' \quad \forall \mu' \geq \mu.\ m \not\xrightarrow{\mu'}_M}{m \triangleleft p \xrightarrow{\mu}_{I(L,M)} \texttt{end} \triangleleft p'}$$

$$\textsc{iTran}\frac{m \triangleleft p \xrightarrow{\sigma}_{I(L,M)} m' \triangleleft p' \quad p' \xrightarrow{\mu}_L p'' \quad \mu \leq \sigma}{m \triangleleft p \xrightarrow{\sigma}_{I(L,M)} m' \triangleleft p''}$$

Table 2 Instrumentation rules for myopic monitors.

be monitors from the full monitoring system (M^δ, I^δ). According to Properties B, C, and D, for all LTSs $L \leq_o L'$, where L is an LTS on $(\textsc{Act}, \{\tau\})$, and every process p in L we have that $p \xrightarrow{\tau}_L$ if and only if $p \xrightarrow{\tau}_{L'}$ or $p \xrightarrow{\sigma}_{L'}$; therefore, m_2 monitors for φ_2 on L from L'. A process p of L violates φ_3 iff $p(\xrightarrow{\tau}_L)^+ q \xrightarrow{\alpha}_L$ for some q; by Properties A, B, C, $p(\xrightarrow{\tau}_L)^+ q \xrightarrow{\alpha}_L$ iff $p(\xrightarrow{\tau}_{L'} \cup \xrightarrow{\sigma}_{L'})^+ q \xrightarrow{\alpha}_{L'}$, and therefore, m_3 monitors for φ_3 on L from L'. ◀

We first introduce myopic monitors, that are equivalent to full monitors on total obscurings.

▶ **Definition 20.** A *myopic monitor* on $(\textsc{Act}, \textsc{Sil})$ is defined by the grammar:

$$m, n \in \textsc{Mon}_\sigma ::= \quad \texttt{end} \quad | \ \texttt{no} \quad | \ \alpha.m \quad | \ \sigma.m \quad | \ m + n \quad | \ \texttt{rec}\ x.m \quad | \ x.$$

A myopic monitor's LTS semantics is defined by the transition rules in Table 1; the resulting monitor LTS is $M^\sigma = \langle \textsc{Mon}_\sigma, (\textsc{Act}, \{\sigma\}), \rightarrow\rangle$. The instrumentation I^σ of myopic monitors is then defined by the rules in Table 2. ◀

Rules iMon and iTer are similar to those for I^δ. The difference is that when the monitor is expecting a more obscure action (*i.e.*, σ), the instrumentation can pass along a possibly less obscure process action. So, the instrumentation may interpret τ-transitions as σ-transitions. Rule iTran is new, but the intuition behind it is similar: the instrumentation may interpret a (possibly mixed) sequence of τ- and σ-transitions as a single σ-transition, if that is what the monitor was expecting. On total obscurings, myopic monitors behave like full monitors.

▶ **Lemma 21.** *If L is a total obscuring on $(\textsc{Act}, \{\tau, \sigma\})$, then for every $m \in \textsc{Mon}_\sigma$ and process p of L, $\mathbf{rej}_{\langle M^\sigma, I^\sigma, L\rangle}(m,p)$ iff $\mathbf{rej}_{\langle M^\delta, I^\delta, L\rangle}(m,p)$.* ◀

As we see in the following, we can further restrict the syntax of myopic monitors while preserving monitorability with respect to reliably monitorable formulas. The *σ-alternating myopic monitors* are the myopic monitors restricted to the following syntax:

$$m, n \in \textsc{Mon}_{alt} ::= \quad \texttt{end} \quad | \ \texttt{no} \quad | \ \sigma.\texttt{no} \quad | \ \alpha.m \quad | \ \sigma.\alpha.m \quad | \ m{+}n \quad | \ \texttt{rec}\ x.m \quad | \ x.$$

The resulting monitor LTS is called M^{alt} and it is a fragment of M^δ.

▶ **Corollary 22.** *If φ is a reliably monitorable formula on $(\textsc{Act}, \{\tau\})$, then there is a σ-alternating myopic monitor that monitors for φ on every LTS L on $(\textsc{Act}, \{\tau\})$ from every total obscuring of L.* ◀

▶ **Example 23.** We revisit the LTSs $L_1 \leq_o L_2$ from Example 19. Let $m_4 = \sigma.\sigma.\alpha.\texttt{no}$ be a myopic monitor but *not* a σ-alternating one. We see that m_4 rejects process p_0 in L_1, but not in L_2 since m_4 flags processes that perform *at least two* silent actions before performing α; this is *not* the case for p_0 in L_2. The constraint of σ-alternation ensures that the monitors are not allowed to count silent actions and thus rely on information that may be hidden in a further obscuring of the LTS: the information that a monitor of the form $\sigma.\mathtt{no}$ or $\sigma.\alpha.m$ can analyse is "at least one" silent transition (and then α in the second case), which is information guaranteed to be preserved by Properties E and D. ◀

The following results describe how monitor rejections are preserved by the obscuring preorder.

▶ **Lemma 24.** *For each $m \in \textsc{Mon}_\sigma$ and each process p of L where $L \leq_o L'$, $\mathbf{rej}_{\langle M^\sigma, I^\sigma, L' \rangle}(m, p)$ implies $\mathbf{rej}_{\langle M^\sigma, I^\sigma, L \rangle}(m, p)$.* ◀

▶ **Lemma 25.** *For every σ-alternating myopic monitor m and p a process of L where $L \leq_o L'$, $\mathbf{rej}_{\langle M^\sigma, I^\sigma, L \rangle}(m, p)$ implies $\mathbf{rej}_{\langle M^\sigma, I^\sigma, L' \rangle}(m, p)$.* ◀

▶ **Corollary 26.** *If a σ-alternating myopic monitor monitors for a formula φ on LTS L on $(\textsc{Act}, \{\tau\})$ from an obscuring of L, then the monitor reliably monitors for φ on L.* ◀

Thus, monitorability implies reliable monitorability for such monitors. We can now identify a maximal reliably monitorable fragment of μHML on $(\textsc{Act}, \{\tau\})$, which we call RsHML:

$$\theta, \chi \in \textsc{RsHML} ::= \ \mathtt{tt} \quad | \ \mathtt{ff} \quad | \ [\tau]\mathtt{ff} \quad | \ [\alpha]\theta \quad | \ [[\tau]][\alpha]\theta \quad | \ \theta \wedge \chi \quad | \ \mathtt{max}\ \ X.\theta \quad | \ X.$$

▶ **Definition 27 (Reliable Monitor Synthesis).** We define a reliable monitor synthesis function $(\!|\cdot|\!)_r$ from RsHML to σ-alternating myopic monitors.

$$(\!|\mathtt{tt}|\!)_r = \mathtt{end} \qquad (\!|\mathtt{ff}|\!)_r = \mathtt{no} \qquad (\!|X|\!)_r = x \qquad (\!|[\tau]\mathtt{ff}|\!)_r = \sigma.\mathtt{no}$$

$$(\!|\psi_1 \wedge \psi_2|\!)_r = \begin{cases} (\!|\psi_1|\!)_r & \text{if } (\!|\psi_2|\!)_r = \mathtt{end} \\ (\!|\psi_2|\!)_r & \text{if } (\!|\psi_1|\!)_r = \mathtt{end} \\ (\!|\psi_1|\!)_r + (\!|\psi_2|\!)_r & \text{otherwise} \end{cases} \qquad (\!|[\alpha]\psi|\!)_r = \begin{cases} \mathtt{end} & \text{if } (\!|\psi|\!)_r = \mathtt{end} \\ \alpha.(\!|\psi|\!)_r & \text{otherwise} \end{cases}$$

$$(\!|\mathtt{max}\ \ X.\psi|\!)_r = \begin{cases} \mathtt{end} & \text{if } (\!|\psi|\!)_r = \mathtt{end} \\ \mathtt{rec}\ x.(\!|\psi|\!)_r & \text{otherwise} \end{cases} \qquad (\!|[[\tau]][\alpha]\psi|\!)_r = \begin{cases} \mathtt{end} & \text{if } (\!|\psi|\!)_r = \mathtt{end} \\ \sigma.\alpha.(\!|\psi|\!)_r & \text{otherwise} \end{cases}$$ ◀

▶ **Lemma 28.** *For every formula $\varphi \in$ RsHML, $(\!|\varphi|\!)_r$ reliably monitors for φ.*

▶ **Definition 29 (Reliable Formula Synthesis).** We define a reliable formula synthesis function $\|\cdot\|_r$ from σ-alternating myopic monitors to RsHML.

$$\|\mathtt{end}\|_r = \mathtt{tt} \qquad \|\mathtt{no}\|_r = \mathtt{ff} \qquad \|x\|_r = X$$
$$\|\sigma.\mathtt{no}\|_r = [\tau]\mathtt{ff} \qquad \|\sigma.\alpha.m\|_r = [[\tau]][\alpha]\|m\|_r \qquad \|\alpha.m\|_r = [\alpha]\|m\|_r$$
$$\|m + n\|_r = \|m\|_r \wedge \|n\|_r \qquad \|\mathtt{rec}\ x.m\|_r = \mathtt{max}\ \ X.\|m\|_r$$ ◀

▶ **Lemma 30.** *For every σ-alternating myopic monitor m on $(\textsc{Act}, \{\tau, \sigma\})$, m reliably monitors for $\|m\|_r$.* ◀

Theorem 31 presents the main result of this section. The first part follows from Lemmata 28 and 30 whereas the second part is a consequence of Lemma 30 and Corollaries 22 and 26.

▶ **Theorem 31.** *The monitoring system (M^{alt}, I^σ) on $(\textsc{Act}, \{\tau, \sigma\})$ reliably monitors for* RsHML *on $(\textsc{Act}, \{\tau\})$. Moreover,* RsHML *is the largest reliably monitorable fragment of μHML up to logical equivalence.* ◀

We note that RsHML is also a fragment of sHML, the maximally monitorable fragment of μHML identified in Section 3. Theorem 31 holds for every preorder $\leq_o$ that has the properties listed in Subsection 4.2. Therefore, it also holds for $\leq_c$ from Definition 16.

▶ **Example 32.** We return to the server from Examples 12 and 17. Notice that formula ψ is a RsHML-formula, and therefore it is reliably monitorable. ◀

We conclude by noting that myopic monitors can also be described as a fragment of full monitors by replacing $\sigma.\texttt{no}$ by $\tau.\texttt{no}+\sigma.no$ and $\sigma.\alpha.m$ with $\texttt{rec}\ x.(\tau.x+\sigma.x+\tau.\alpha.m+\sigma.\alpha.m)$ in any myopic monitor. However, myopic monitors provide a cleaner, more efficient description of the same monitors.

6 Conclusions

We developed a general framework for *reliable monitorability* for monitoring setups that obscure information about the internal behaviour of systems. The framework is described through a family of LTS preorders that satisfy natural properties (A through E of Subsection 4.2). Via further assumptions (properties F, G, H) that guarantee a certain level of information obscuring, we identified RsHML, a maximal *reliably* monitorable fragment of μHML. Then, we provided a monitoring system, (M^{alt}, I^{σ}) that reliably monitors for RsHML.

Related work: In [9], Dwyer *et al.* use the approach of combining several properties to be monitored to produce a composite property and then project this composite property onto a smaller set of observable actions. This sampling technique effectively "silences" some of the observable actions and focuses on the rest to reduce overhead without risking unsound monitoring. Their approach highlights the importance of silent actions for RV and the need for a framework to handle imperfect information about silent events.

In [5] Basin *et al.* consider the problem of monitoring over defective traces (called incomplete/disagreeing logs). They propose an augmented LTL specification language that permits reasoning about incompleteness and handling of inconsistencies. In some sense, this is related to our reliable monitors that are able to provide correct verdicts in the presence of event obscuring; however, the authors in [5] do not tackle issues related to monitorability.

In [21], Shi *et al.* consider the problem of monitoring a wireless network via a wireless sniffer. A wireless sniffer may introduce uncertainty over a monitoring setup, as the trace it detects may not necessarily be the actual trace of the system, due to the intrinsic unreliability of the wireless network. The authors in [21] thus develop a monitoring framework to tolerate such errors. In separate work [6], Basin *et al.* tackle the problem of distributed monitoring over a network which may produce delays and/or failures; they use a monitoring system based on a real-time three-valued logic that can track when an event took place. Their monitors may then need to draw appropriate conclusions under incomplete or scrambled information. Although we do not consider aspects such as event reordering, our work could serve as a basis for a better understanding of the level of obscuring these systems can tolerate.

Variations: Our system can be adjusted to describe diverse situations, by either weakening or strengthening the power of obscuring preorders. For instance, one can relax properties F to H to describe situations where there is a guarantee that the system reveals its internal behaviour at least partly and to a certain degree. For example, we can take $\leq_o$ to be the identity relation on LTSs, as it satisfies properties A to E and therefore is an obscuring preorder; then, the reliably monitorable properties would be all of sHML and the corresponding monitoring system would be that of full monitors. A preorder between $\leq_c$ and $=$ would perhaps be more interesting.

References

1 Luca Aceto, Antonis Achilleos, Adrian Francalanza, Anna Ingólfsdóttir, and Sævar Örn Kjartansson. Determinizing monitors for HML with recursion. *arXiv preprint arXiv:1611.10212*, 2016.

2 Luca Aceto, Antonis Achilleos, Adrian Francalanza, Anna Ingólfsdóttir, and Sævar Örn Kjartansson. On the complexity of determinizing monitors. In Arnaud Carayol and Cyril Nicaud, editors, *Implementation and Application of Automata: 22nd International Conference, CIAA 2017, Marne-la-Vallée, France, June 27-30, 2017, Proceedings*, pages 1–13, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-60134-2_1.

3 Luca Aceto, Anna Ingólfsdóttir, Kim Guldstrand Larsen, and Jiri Srba. *Reactive Systems: Modelling, Specification and Verification.* Cambridge University Press, New York, NY, USA, 2007. doi:10.1017/cbo9780511814105.

4 S. Arun-Kumar and Matthew Hennessy. An efficiency preorder for processes. *Acta Informatica*, 29(8):737–760, 1992. doi:10.1007/BF01191894.

5 David Basin, Felix Klaedtke, Srdjan Marinovic, and Eugen Zălinescu. Monitoring compliance policies over incomplete and disagreeing logs. In Shaz Qadeer and Serdar Tasiran, editors, *Runtime Verification, Third International Conference, RV 2012*, volume 7687 of *Lecture Notes in Computer Science*, pages 151–167. Springer, 2013. doi:10.1007/978-3-642-35632-2_17.

6 David Basin, Felix Klaedtke, and Eugen Zalinescu. Failure-aware runtime verification of distributed systems. In *35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2015)*, pages 590–603, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.590.

7 Laura Bocchi, Tzu-Chun Chen, Romain Demangeon, Kohei Honda, and Nobuko Yoshida. Monitoring networks through multiparty session types. *Theoretical Computer Science*, 669:33–58, 2017. doi:10.1016/j.tcs.2017.02.009.

8 Edmund M. Clarke, Jr., Orna Grumberg, and Doron A. Peled. *Model Checking.* MIT Press, Cambridge, MA, USA, 1999.

9 Matthew B. Dwyer, Madeline Diep, and Sebastian Elbaum. Reducing the cost of path property monitoring through sampling. In *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, pages 228–237, 2008. doi:10.1109/ASE.2008.33.

10 Adrian Francalanza. A theory of monitors. In Bart Jacobs and Christof Löding, editors, *Foundations of Software Science and Computation Structures: 19th International Conference, FOSSACS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2–8, 2016, Proceedings*, pages 145–161, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. doi:10.1007/978-3-662-49630-5_9.

11 Adrian Francalanza. Consistently-detecting monitors. In Roland Meyer and Uwe Nestmann, editors, *28th International Conference on Concurrency Theory (CONCUR 2017)*, volume 85 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:19, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CONCUR.2017.8.

12 Adrian Francalanza, Luca Aceto, Antonis Achilleos, Duncan Paul Attard, Ian Cassar, Dario Della Monica, and Anna Ingólfsdóttir. A foundation for runtime monitoring. In Shuvendu K. Lahiri and Giles Reger, editors, *Runtime Verification - 17th International Conference, RV 2017, Seattle, WA, USA, September 13-16, 2017, Proceedings*, volume 10548 of *Lecture Notes in Computer Science*, pages 8–29. Springer International Publishing, 2017. doi:10.1007/978-3-319-67531-2_2.

13 Adrian Francalanza, Luca Aceto, and Anna Ingólfsdóttir. On verifying Hennessy-Milner logic with recursion at runtime. In Ezio Bartocci and Rupak Majumdar, editors, *Runtime Verification - 6th International Conference, RV 2015 Vienna, Austria, September 22-25, 2015. Proceedings*, volume 9333 of *Lecture Notes in Computer Science*, pages 71–86. Springer International Publishing, 2015. `doi:10.1007/978-3-319-23820-3_5`.

14 Adrian Francalanza, Luca Aceto, and Anna Ingolfsdottir. Monitorability for the Hennessy–Milner logic with recursion. *Formal Methods in System Design (FMSD)*, 51(1):87–116, March 2017. `doi:10.1007/s10703-017-0273-z`.

15 Adrian Francalanza and Aldrin Seychell. Synthesising correct concurrent runtime monitors. *Formal Methods in System Design (FMSD)*, 46(3):226–261, June 2015. `doi:10.1007/s10703-014-0217-9`.

16 Matthew Hennessy. *Algebraic Theory of Processes*. Foundations of Computing. MIT Press, 1988.

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

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

19 Martin Leucker and Christian Schallhart. A brief account of Runtime Verification. *The Journal of Logic and Algebraic Programming*, 78(5):293–303, May 2009. `doi:10.1016/j.jlap.2008.08.004`.

20 Robin Milner. *Communication and Concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

21 Jinghao Shi, Shuvendu K. Lahiri, Ranveer Chandra, and Geoffrey Challen. Wireless protocol validation under uncertainty. In *Runtime Verification - 16th International Conference, RV 2016, Madrid, Spain, September 23-30, 2016, Proceedings*, pages 351–367. Springer International Publishing, 2016. `doi:10.1007/978-3-319-46982-9_22`.

22 Yoriyuki Yamagata, Cyrille Artho, Masami Hagiya, Jun Inoue, Lei Ma, Yoshinori Tanabe, and Mitsuharu Yamamoto. Runtime monitoring for concurrent systems. In *Runtime Verification - 16th International Conference, RV 2016, Madrid, Spain, September 23-30, 2016, Proceedings*, pages 386–403. Springer International Publishing, 2016. `doi:10.1007/978-3-319-46982-9_24`.