Consistently-Detecting Monitors*

Adrian Francalanza

CS, ICT, University of Malta, Msida, Malta
adrian.francalanza@um.edu.mt

Abstract

We study a contextual definition for deterministic monitoring based on consistent detections. It is defined in terms of the observed behaviour of the monitor when instrumented over arbitrary systems. We give an alternative, coinductive definition based on controllability which does not rely on system quantifications, and show that it is fully-abstract *wrt.* the former definition. We then develop a symbolic counterpart to the controllability definition to facilitate an automated analysis for controllable monitors involving data.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs, F.3.2 Semantics of Programming Languages - Process models, D.2.5 Testing and Debugging

Keywords and phrases Runtime Monitoring, Deterministic Behaviour, Controllability, Compositional Reasoning, Symbolic Analysis

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2017.8

1 Introduction

Monitors are computational entities that observe the executions of other entities (referred to hereafter as *systems*) with the aim of accruing system information [32, 26], comparing system executions against some behavioural specification [23, 7], or reacting to the observed executions via adaptation or enforcement procedures [13, 36]. They are typically considered to be part of the *Trusted Computing Base (TCB)* and, consequently, their descriptions are expected to be *correct*. A correctness requirement often presumed of monitors is that they should exhibit *deterministic behaviour*. Yet, for most monitoring frameworks, such a requirement is seldom specified in unambiguous terms. In fact, there are a number of viable alternatives that one could consider (*e.g.*, [44, 29, 25, 1]) and it is unclear how to choose one over the other in an objective manner. Moreover, these definitions often fail to account for the instrumentation mechanism used to compose a monitor with the system under scrutiny which may, in turn, affect monitoring behaviour. All of this leads to a poor understanding of what should be expected of a monitor, and may give rise to discrepancies between these expectations and what needs to be guaranteed by the monitor implementer in practice.

Non-determinism is intrinsic to a number of computational models used for expressing monitors and monitored systems. In fact, a substantial body of work on monitors is either cast in terms of inherently non-deterministic formalisms such as Büchi automata [45, 17], or formalisms that admit non-determinism such as process calculi and labelled transition systems [12, 46, 30, 22, 10, 23]. Non-deterministic computation arises naturally in concurrent and distributed programming, used increasingly for runtime monitoring [37, 24, 21, 8, 11]. Furthermore, a growing number of monitoring tools employ automata-based specification languages [16, 5, 41, 18] that offer rudimentary support for ensuring deterministic behaviour:

* Partly supported by UoM and RANNIS projects CPSRP05-04 and TheoFoMon:163406-051.

their respective implementations are either thread-unsafe [41] or admit arbitrary code for transition-triggered actions [14, 16].

This paper sets out to investigate deterministic behaviour for monitors. The study is limited to execution monitors (sequence recognisers) [43, 36], used extensively for Runtime Verification (RV). Our work is developed in terms of an expository formalism (similar to the aforementioned work on transition-based descriptions) expressing monitored systems that can analyse trace events carrying data and admit degrees of non-determinism. We propose a *contextual* definition for deterministic monitor behaviour, founded on the *observational* behaviour that can be discerned when a monitor is instrumented to execute with any arbitrary system under scrutiny. The definition serves two purposes. First, its contextual nature allows us to admit *as many* correct monitors *as possible*, as long as these cannot be *externally* perceived to behave non-deterministically—we contend that the resulting definition is fairly intuitive. Second, it allows us to *justify* design decisions for an alternative definition describing the deterministic behaviour of monitors, based instead on the notion of *controllability* [31]. We show a correspondence between these two definitions. In addition, we demonstrate how the alternative definition (which is arguably less intuitive than its contextual counterpart) is more amenable to automated analyses for assessing the deterministic behaviour exhibited by monitors. In particular, we study how this alternative definition can be reformulated in symbolic terms, to facilitate a tractable handling of infinite-state monitor analysis due to data.

▶ **Example 1.** The monitor description m_1 accepts traces from an authenticator, that challenges (event chl) a supplicant for an *arbitrary* value x followed by the supplicant's authentication (aut) with the (correct) encoding of x, $y = enc(x)$. The authenticator subsequently acknowledges (ack) using the same value y. The guard construct $\mathsf{chl}(x)$ quantifies over *any* value x; guards $\mathsf{aut}\langle v\rangle$ and $\mathsf{ack}\langle v\rangle$ require authentications (*resp.* acknowledgments) for a *specific* value v. $\top$ denotes acceptance.

$$m_1 \triangleq \mathsf{chl}(x).\,\mathsf{let}\ y = enc(x)\ \mathsf{in}\ \big(\mathsf{aut}\langle y\rangle.\mathsf{ack}\langle y\rangle.\top\big)$$

$$m_2 \triangleq \mathsf{chl}(x).\mathsf{let}\ y = enc(x)\ \mathsf{in}\ \big(\mathsf{aut}\langle y\rangle.\mathsf{ack}\langle y\rangle.\top + \mathsf{aut}(z).\mathsf{if}\ z \neq y\ \mathsf{then}\ \mathsf{ack}(z').\bot\big)$$

$$m_3 \triangleq \mathsf{chl}(x).\mathsf{let}\ y = enc(x)\ \mathsf{in}\ \big(\mathsf{aut}\langle y\rangle.\mathsf{ack}\langle y\rangle.\top + \mathsf{aut}(z).\mathsf{if}\ z \neq y\ \mathsf{then}\ \mathsf{ack}(z').\bot\ \mathsf{else}\ \mathsf{ack}\langle z\rangle.\top\big)$$

It is easy to inadvertently introduce non-determinism. Monitor m_2 extends m_1 (using the choice operator, $+$) with the intention of rejecting (note the verdict $\bot$) acknowledgments following authentications whose value is *not* the encoding of the challenge value x. When $v_2{\neq}enc(v_1)$, the violating trace $t{=}\mathsf{chl}\langle v_1\rangle.\mathsf{aut}\langle v_2\rangle.\mathsf{ack}\langle v_2\rangle\ldots$ is always rejected by m_2. More subtly, however, when $v_2{=}enc(v_1)$ the trace t may cause the monitor to non-deterministically choose either branch, whereby the unintended branch does *not* reach a $\top$ verdict. But non-determinism is not necessarily conducive to inconsistent verdicts and, in cases where verdicts are considered as the only monitor observable behaviour, such non-determinism may be tolerated. Monitor m_3 *deterministically accepts* trace t when $v_2{=}enc(v_1)$, albeit along different execution paths, and *deterministically rejects* it whenever $v_2{\neq}enc(v_1)$. ◀

The main contributions of the paper are:

1. A contextual definition for observationally deterministic monitoring behaviour, Definition 6.
2. An alternative definition based on controllability, Definition 11, that coincides with it, Theorem 13.
3. Symbolic variants of the latter definition enabling tractable automations, Theorems 21 and 25.

Monitors

$$
\begin{array}{rlllll}
w, o \in \textsc{Verd} ::= & \top & \text{(accept)} & \mid \bot & & \text{(reject)} \\
 \mid & \mathbf{0} & \text{(inconclusive)} & & & \\
m, n \in \textsc{Mon} \; ::= & w & \text{(verdict)} & \mid \mathsf{let}\ x = e\ \mathsf{in}\ m & & \text{(evaluate)} \\
 \mid & l\langle e\rangle.m & \text{(expression guard)} & \mid l(x).m & & \text{(quantified guard)} \\
 \mid & m + n & \text{(choice)} & \mid \mathsf{if}\ b\ \mathsf{then}\ m\ \mathsf{else}\ n & & \text{(conditional)} \\
 \mid & \mathsf{rec}\, X.m & \text{(recursion)} & \mid X & & \text{(monitor variable)}
\end{array}
$$

$$
\textsc{Ver}\ \frac{}{w \xrightarrow{\eta} w}
\qquad
\textsc{IfT}\ \frac{[\![b]\!] = \mathsf{true}}{\mathsf{if}\ b\ \mathsf{then}\ m\ \mathsf{else}\ n \xrightarrow{\tau} m}
\qquad
\textsc{IfF}\ \frac{[\![b]\!] = \mathsf{false}}{\mathsf{if}\ b\ \mathsf{then}\ m\ \mathsf{else}\ n \xrightarrow{\tau} n}
\qquad
\textsc{Let}\ \frac{[\![e]\!] = v}{\mathsf{let}\ x = e\ \mathsf{in}\ m \xrightarrow{\tau} m[^{v}\!/x]}
$$

$$
\textsc{GrE}\ \frac{[\![e]\!] = v}{l\langle e\rangle.m \xrightarrow{l\langle v\rangle} m}
\qquad
\textsc{GrQ}\ \frac{}{l(x).m \xrightarrow{l\langle v\rangle} m[^{v}\!/x]}
\qquad
\textsc{Rec}\ \frac{}{\mathsf{rec}\, X.m \xrightarrow{\tau} m[^{\mathsf{rec}\, X.m}\!/X]}
\qquad
\textsc{Ch1}\ \frac{m \xrightarrow{\alpha} m'}{m + n \xrightarrow{\alpha} m'}
$$

Instrumentation

$$
\textsc{Mon}\ \frac{s \xrightarrow{\eta} r \quad m \xrightarrow{\eta} n}{s \triangleleft m \xrightarrow{\eta} r \triangleleft n}
\qquad
\textsc{Ter}\ \frac{s \xrightarrow{\eta} r \quad m \not\xrightarrow{\eta} \quad m \not\xrightarrow{\tau}}{s \triangleleft m \xrightarrow{\eta} r \triangleleft \mathbf{0}}
\qquad
\textsc{AsS}\ \frac{s \xrightarrow{\tau} r}{s \triangleleft m \xrightarrow{\tau} r \triangleleft m}
\qquad
\textsc{AsM}\ \frac{m \xrightarrow{\tau} n}{s \triangleleft m \xrightarrow{\tau} s \triangleleft n}
$$

Figure 1 A Model for describing Instrumented Systems.

The paper is structured as follows. Section 2 presents the monitor framework used for this study, allowing us motivate our touchstone definition for deterministic monitor behaviour in Section 3. Section 4 presents a fully-abstract alternative definition that is amenable to compositional reasoning. In Sections 5 and 6 symbolic variants are developed for automation purposes. Section 7 concludes.

2 Systems, Monitors, Instrumentation and Monitored Systems

We perceive *systems* as entities that generate *events* while executing. *Observable* events, $\eta \in \textsc{Evt}$, are those visible to a monitor and have the form $l\langle v\rangle$ where l is an event *label* taken from a set $l, k \in \textsc{Lab}$, and v is an event *payload* taken from some unspecified value domain $v, u \in \textsc{Val}$. Events capture a number of computational notions such as inputs/outputs in message-passing programs [22], or method/function calls and returns [13, 38]. To simplify our technical development, we consider monadic (*i.e.,* single-valued) events but the formalism can be extended to accommodate polyadicity.

Systems may be described as Labelled Transition Systems (LTSs) [2, 34], triples $\langle \textsc{Sys}, \textsc{Act}, \longrightarrow\rangle$ consisting of a set of states, $s, r \in \textsc{Sys}$, a set of actions, $\alpha, \beta \in \textsc{Act} = \textsc{Evt} \cup \{\tau\}$ that include all observable events $\textsc{Evt}$ and a distinguished silent action $\tau \notin \textsc{Evt}$ for *unobservable* events, and a transition relation, $\longrightarrow \subseteq (\textsc{Sys} \times \textsc{Act} \times \textsc{Sys})$. The suggestive notation $s \xrightarrow{\alpha} s'$ denotes $(s, \alpha, s') \in \longrightarrow$, whereas $s \not\xrightarrow{\alpha}$ denotes $\neg(\exists s' \cdot\ s \xrightarrow{\alpha} s')$. As usual, we write $s \Longrightarrow s'$ in lieu of $s(\xrightarrow{\tau})^* s'$ and $s \stackrel{\eta}{\Longrightarrow} s'$ for $s \Longrightarrow \cdot \xrightarrow{\eta} \cdot \Longrightarrow s'$, referring to s' as a *η-derivative* of s; $s \stackrel{t}{\Longrightarrow}$ stands for $\exists s' \cdot s \stackrel{t}{\Longrightarrow} s'$. We let *traces*, $t, u \in \textsc{Evt}^*$, range over (finite) sequences of *observable* events and write $s \stackrel{\eta_1}{\Longrightarrow} \ldots \stackrel{\eta_n}{\Longrightarrow} s_n$ as $s \stackrel{t}{\Longrightarrow} s_n$, where $t = \eta_1, \ldots, \eta_n$. The notation $u \ldots$ is occasionally used to denote the existence of some trace t with a prefix u.

We presuppose an expression language $e, d \in \textsc{Exp}$ that ranges over the (event) value domain $\textsc{Val}$ and a denumerable set of *expression variables* $x, y, z \in \textsc{Vars}$; $\vec{e}$ and $\vec{x}$ *resp.*

denote lists of expressions and variables. We also assume a boolean expression language $b, c \in \text{BExp}$ defined over Exp that includes standard constructs for conjunctions, $b \wedge c$, and negations, $\neg b$, but also equality predicates over expressions $e = d$. The meta-functions $\mathbf{fv}(e)$ and $\mathbf{fv}(b)$ return the *free variables* in the *resp.* expressions; expressions are *closed* whenever $\mathbf{fv}(e) = \emptyset$ and *open* otherwise, and similarly for boolean expressions. *Valuations* are total maps from variables to values, $\rho \in (\text{Vars} \to \text{Val})$ whereas *substitutions* are partial maps from variables to expressions $\sigma \in (\text{Vars} \rightharpoonup \text{Exp})$; substitutions are denoted as $[\vec{e}/\vec{x}]$, where $d[\vec{e}/\vec{x}]$ represents the (simultaneous) substitution of all occurrences of $x_i \in \vec{x}$ in d by the corresponding $e_i \in \vec{e}$. We assume an *evaluation function* that takes an expression and a valuation and returns a value, $[\![e\rho]\!] = v$. Similarly, boolean expressions have a semantic function mapping them to the boolean domain via a valuation, $[\![b\rho]\!] \in \{\mathsf{true}, \mathsf{false}\}$, where we presume the expected properties, *e.g.,* $[\![(b \wedge c)\rho]\!] = \mathsf{true}$ iff $[\![b\rho]\!] = \mathsf{true}$ and $[\![c\rho]\!] = \mathsf{true}$. We also assume a classical interpretation of boolean expressions, *i.e.,* $[\![(\neg b)\rho]\!] = \mathsf{true}$ iff $[\![b\rho]\!] = \mathsf{false}$. To alleviate the presentation, we often work up to associativity and commutativity for conjunctions, treating $\langle \text{BExp}, \wedge, \mathsf{true} \rangle$ as an abelian monoid. For closed expressions, we elide the valuation and write $[\![e]\!]$ and $[\![b]\!]$ for $[\![e\rho]\!]$ and $[\![b\rho]\!]$ *resp.* The *satisfiability* judgement for boolean expressions, $\mathbf{sat}(b) \stackrel{\mathbf{def}}{=} \exists \rho \cdot [\![b\rho]\!] = \mathsf{true}$, plays a central role in subsequent development.

Monitors, here defined by the syntax in Figure 1, constitute the focus of our study. They may reach two kinds of verdicts, Verd. *Conclusive verdicts* consist of acceptances, $\top$, and rejections, $\bot$. In addition, a monitor may also reach the *inconclusive verdict*, $\mathbf{0}$, a form of premature termination used when the generated system events of the monitor specification itself does not yield sufficient information so as to reach a definite conclusion. The monitor *expression* guard $l\langle e \rangle.m$ expects events with label l and a payload value matching the evaluation of e, whereas the *quantified* guard $l(x).m$ allows the monitor to *dynamically learn* the payload of an event with label l. Monitors may branch (externally) depending on the events observed, $m+n$, or branch (internally) based on data predicates, $\mathsf{if}\, b\, \mathsf{then}\, m \;\mathsf{else}\; n$. They may also perform internal computation themselves by evaluating expressions, $\mathsf{let}\; x = e \;\mathsf{in}\; m$, or recurse, $\mathsf{rec}\, X.p$, via *term variables* $X, Y, Z \in \text{TVars}$. The constructs $l(x).m$ and $\mathsf{rec}\, X.m$ act as *binders* for x and X *resp.* in m, inducing the usual notions of open/closed (monitor) terms. We work up to alpha-conversion of bound expression/term variables and use the shorthand $\mathsf{if}\, b\, \mathsf{then}\, m$ for $\mathsf{if}\, b\, \mathsf{then}\, m \;\mathsf{else}\; \mathbf{0}$ and $\tau.m$ for $\mathsf{let}\; x = v \;\mathsf{in}\; m$ where $x \notin \mathbf{fv}(m)$.

The semantics of *closed* monitors is also defined in terms of an LTS, via the transition rules in Figure 1: for each $m \in \text{Mon}$ we have a dedicated LTS $\langle M, \text{Act}, \longrightarrow \rangle$ where $M \subseteq \text{Mon}$ are the monitors reachable from m via transitions. The rules model the monitor analysis of observable events. Rule Ver describes how verdicts are *irrevocable*, meaning that a verdict can analyse *any* observable event but always transition to itself. In rule GrE, an expression guard $l\langle e \rangle.m$ only transitions to the continuation m when observing an event matching the label l with the payload equal to $[\![e]\!]$. By contrast, a quantified guard $l(x).m$ transitions by analysing any event with label l, binding x to the event payload v in the continuation, $m[v/x]$; see rule GrQ. The remaining rules are as expected where the term $m[\mathsf{rec}\, X.m/X]$ denotes the term substitution of $\mathsf{rec}\, X.m$ for free occurrences of X in m.

A system s *instrumented* with a monitor m is referred to as a *monitored system* and denoted as $s \triangleleft m$. The semantics of monitored systems is defined by the instrumentation rules in Figure 1. We here adopt the composition relation studied in [22, 23], even though other instrumentation relations could have been used. Note that the chosen composition relation is still quite general: it is parametric *wrt.* the system and monitor abstract LTSs and it is *largely independent* of their specific language specifications, since it only requires the monitor LTS to contain an inconclusive (persistent) verdict state, $\mathbf{0}$. The instrumentation relation of Figure 1 is *asymmetric*: a monitored system can transition with an observable event *only when* the system can produce that event *i.e.,* monitors are *passive* and cannot instigate transitions. When the system generates an (observable) event that can be analysed by the monitor, the two transition in lockstep according to their respective LTSs (rule MON). When the monitor *cannot* analyse the event generated [1] and cannot internally transition to a state that enables it to do so (*i.e.,* it is already stable, $m \not\xrightarrow{\tau}$), the instrumentation does *not* block the monitored system: instead, it allows the system to transition *but* aborts monitoring to the inconclusive verdict (rule TER). System-monitor synchronisations are limited to observable events, and the specific entities can transition independently *wrt.* their respective internal moves (rules ASS and ASM).

▶ **Example 2.** Monitor m_4 below listens for input and output events $\mathsf{in}\langle v\rangle$ and $\mathsf{out}\langle v\rangle$ where the (integer) payload $v \in \mathbb{N}$ reports the port number over which the communication operation is performed.

$$m_4 \triangleq \mathsf{rec}\, X.\Big(\big(\mathsf{out}\langle 80\rangle.\bot\big) + \big(\mathsf{in}(x).\,\mathsf{if}\ x{=}80\ \ \mathsf{then}\ \ \mathsf{out}\langle 81\rangle.\top\ \ \mathsf{else}\ \ \mathsf{out}\langle x\rangle.X\ \big)\Big) \tag{1}$$

The monitor rejects system executions starting with an output on port 80 but accepts traces containing an input on port 80 followed by an output on port 81, preceded by an arbitrary number of input-output operations on any matching port other than 80. The execution below shows an *accepted* monitored computation for a system s generating the trace $\mathsf{in}\langle 85\rangle \cdot \mathsf{out}\langle 85\rangle \cdot \mathsf{in}\langle 80\rangle \cdot \mathsf{out}\langle 81\rangle$. In monitor m_4, the binding on $\mathsf{in}(x)$ acts as a *freeze-variable* [19] for the subsequent $\mathsf{out}\langle x\rangle$ guard in the else branch.

$$\begin{array}{llr}
s \triangleleft m_4 \xrightarrow{\tau} s \triangleleft \big(\mathsf{out}\langle 80\rangle.\bot\big) + \big(\mathsf{in}(x).\,\mathsf{if}\ x{=}80\,\mathsf{then}\,\mathsf{out}\langle 81\rangle.\top\ \mathsf{else}\ \mathsf{out}\langle x\rangle.m_4\big) & & \textsc{Rec}\\
\quad\xrightarrow{\mathsf{in}\langle 85\rangle} s' \triangleleft\ \mathsf{if}\ 85{=}80\ \ \mathsf{then}\ \mathsf{out}\langle 81\rangle.\top\ \ \mathsf{else}\ \ \mathsf{out}\langle 85\rangle.m_4 & & \textsc{ChR}{+}\textsc{GrQ}\\
\qquad\xrightarrow{\tau} s' \triangleleft \mathsf{out}\langle 85\rangle.m_4 \xrightarrow{\mathsf{out}\langle 85\rangle} s'' \triangleleft m_4 \xRightarrow{\mathsf{in}\langle 80\rangle\cdot\mathsf{out}\langle 81\rangle} s''' \triangleleft \top & & \textsc{IfF},\ldots
\end{array}$$

The instrumentation of Figure 1 delays system transitions to allow the monitor to internally transition to a state that can process the event. *E.g.,* if a system r can generate event $\mathsf{out}\langle 80\rangle$, $r \triangleleft m_4$ postpones this transition (MON and TER cannot be applied) until m_4 unfolds.

$$r \triangleleft m_4 \xrightarrow{\tau} r \triangleleft \big(\mathsf{out}\langle 80\rangle.\bot\big) + \big(\mathsf{in}(x).\,\mathsf{if}\ \ x{=}80\ \mathsf{then}\ \ldots\big) \xrightarrow{\mathsf{out}\langle 80\rangle} r' \triangleleft \bot \qquad \textsc{AsM},\textsc{Mon}$$

Rule TER is crucial both for allowing monitored computations to proceed when the monitor cannot analyse an event, but also to avoid *unintended detections*. *E.g.,* if system r can generate the trace $\mathsf{out}\langle 90\rangle \cdot \mathsf{in}\langle 80\rangle \cdot \mathsf{out}\langle 81\rangle$, this behaviour should still be permitted when instrumented with the monitor m_4, *but* the behaviour should not be detected according to the description in Equation 1. After the initial unfolding of m_4, TER allows r to transition with $\mathsf{out}\langle 90\rangle$ but transitions m_4 to the inconclusive state, $\mathbf{0}$, since neither guard $\mathsf{out}\langle 80\rangle$ nor guard $\mathsf{in}(x)$ can process the event.

$$\begin{array}{llr}
r \triangleleft m_4 \xrightarrow{\tau} r \triangleleft \big(\mathsf{out}\langle 80\rangle.\bot\big) + \big(\mathsf{in}(x).\,\mathsf{if}\ \ x{=}80\ \mathsf{then}\ \ldots\big) & & \\
\qquad\xrightarrow{\mathsf{out}\langle 90\rangle} r'' \triangleleft \mathbf{0} \xRightarrow{\mathsf{in}\langle 80\rangle\cdot\mathsf{out}\langle 81\rangle} r''' \triangleleft \mathbf{0} & & \textsc{AsM},\textsc{Ter},\ldots
\end{array}$$

Had rule TER been designed otherwise (leaving the monitor state unaltered when transiting with $\mathsf{out}\langle 90\rangle$) the ensuing events $\mathsf{in}\langle 80\rangle \cdot \mathsf{out}\langle 81\rangle$ would lead to the unintended acceptance of the trace. ◀

[1] This may be due to a number of reasons, such as event knowledge gaps or knowledge disagreements [6].

3 Deterministic Monitoring Behaviour

In a monitored system, non-deterministic behaviour can be caused by either the system or the monitor. We focus on identifying *non-determinism attributed to monitors*, teasing it apart from non-determinism caused by system behaviour. This is motivated by the fact that, generally, one has limited control over the behaviour of a system under scrutiny. We target a definition that admits monitor non-determinism that is not externally observable. Concretely, we consider detections (*i.e.*, conclusive verdicts) as the only externally visible aspect of a monitor and base our definition on the notion of *deterministic detections*—in applications such as RV, detections are associated with property satisfactions and violations [23, 7]. This immediately rules out a number of candidate definitions for deterministic monitor behaviour. For instance, a definition that considers a monitor m to be deterministic whenever, for *all* systems s and traces t, $s \triangleleft m \overset{t}{\Longrightarrow} s' \triangleleft m'$ and $s \triangleleft m \overset{t}{\Longrightarrow} s' \triangleleft m''$ implies $m' = m''$ is too stringent: it precludes the monitor description below (a slight modification on m_1 of Example 1)

$$m_5 \triangleq \mathsf{chl}(x).\Big(\big(\mathsf{let}\ y = enc(x)\ \mathsf{in}\ \mathsf{aut}\langle y\rangle.\mathsf{ack}\langle y\rangle.\top\big) + \big(\mathsf{aut}\langle enc(x)\rangle.\mathsf{ack}\langle enc(x)\rangle.\top\big)\Big) \tag{2}$$

even though m_5 deterministically accepts traces of the form $\mathsf{chl}\langle v_1\rangle \cdot \mathsf{aut}\langle v_2\rangle \cdot \mathsf{ack}\langle v_2\rangle$ where $v_2 = enc(v_1)$. In fact, after an event $\mathsf{chl}\langle v\rangle$ (for some value v), monitor m_5 can reach two possible internal states, namely $\big(\mathsf{let}\ y = enc(v)\ \mathsf{in}\ \mathsf{aut}\langle y\rangle.\mathsf{ack}\langle y\rangle.\top\big) + \big(\mathsf{aut}\langle enc(v)\rangle.\mathsf{ack}\langle enc(v)\rangle.\top\big)$ or $\mathsf{aut}\langle v'\rangle.\mathsf{ack}\langle v'\rangle.\top$ where $v' = enc(v)$. Other candidates (*e.g.*, confluence defined over transitions [25, 40]) are either inadequate or not immediately applicable because they do not account for executions that do *not* lead to detections. *E.g.*, m_6 (below) would *not* be confluent (consider event $\mathsf{in}\langle 81\rangle$), even though it consistently rejects any trace with the prefix u=$\mathsf{in}\langle 80\rangle$ (and consistently does *not* detect all the other traces).

$$m_6 \triangleq \ \big(\mathsf{in}\langle 80\rangle.\bot\big) + \Big(\big(\mathsf{in}\langle 81\rangle.\mathsf{out}\langle 81\rangle.\mathbf{0}\big) + \big(\mathsf{in}\langle 81\rangle.\mathsf{out}\langle 81\rangle.\mathsf{in}\langle 82\rangle.\mathbf{0}\big)\Big) \tag{3}$$

▶ **Definition 3** (Detected Computations)**.** The transition sequence

$$s \triangleleft m \ \overset{t}{\Longrightarrow}\ s_0 \triangleleft m_0 \xrightarrow{\tau} s_1 \triangleleft m_1 \xrightarrow{\tau} s_2 \triangleleft m_2 \xrightarrow{\tau} \ldots$$

is called a ***t*-computation** if it is **maximal** *i.e.*, either it is *infinite* or it is finite and *cannot be extended further* using τ-transitions. The t-computation above is called *accepted* whenever $\exists i \in \mathbb{N} \cdot m_i = \top$ and *rejected* when $\exists i \in \mathbb{N} \cdot m_i = \bot$. A *detected* t-computation is either an accepted or a rejected one. ◀

Detected computations are indexed by their trace to allow us to *partition* computations according to the system behaviour exhibited at runtime, thus accounting for *system non-determinism*. Definition 3 also permits monitors to stabilise and reach verdicts in the trailing τ-sequence following a t-trace.

▶ **Definition 4** (Deterministic Detection and Withholding)**.** Monitor m *deterministically accepts* (*resp. deterministically rejects*) for system s along trace $t \in \textsc{Evt}^*$, denoted as $\mathsf{da}(m, s, t)$ and $\mathsf{dr}(m, s, t)$ *resp.*, iff *all* t-computation from $s \triangleleft m$ are accepting (*resp.* rejecting). Monitor m *deterministically detects* for s along t, $\mathsf{dd}(m, s, t)$, whenever $\mathsf{da}(m, s, t)$ or $\mathsf{dr}(m, s, t)$. Monitor m *deterministically withholds* for s along trace t, $\mathsf{dw}(m, s, t)$, iff *no* t-computation from $s \triangleleft m$ is accepting or rejecting. ◀

▶ **Example 5.** For arbitrary system s, monitors m_1 of Example 1 and m_5 of Equation 2 deterministically accept traces with the prefix $t = \mathsf{chl}\langle v_1\rangle.\mathsf{aut}\langle v_2\rangle.\mathsf{ack}\langle v_2\rangle$ where $v_2{=}enc(v_1)$ and deterministically withhold on all the other traces. Monitor m_2 deterministically rejects traces with the prefix t above when $v_2{\neq}enc(v_1)$ but does *not* deterministically detect traces with prefix t when $v_2{=}enc(v_1)$. For arbitrary s, monitor m_3 deterministically detects *any* trace with the above prefix t (accepting or rejecting the trace depending on whether $v_2 = enc(v_1)$ or not) and deterministically withholds otherwise. For any system s, monitor m_4 of Example 2 satisfies $\mathsf{dr}(m_4, s, t)$ when the trace t is of the form $t = \mathsf{out}\langle 80\rangle \ldots$, $\mathsf{da}(m_4, s, t)$ when $t = \big(\mathsf{in}\langle v_i\rangle \cdot \mathsf{out}\langle v_i\rangle\big)^i \cdot \mathsf{in}\langle 80\rangle \cdot \mathsf{out}\langle 81\rangle \ldots$ for some $i \in \mathbb{N}$, and $\mathsf{dw}(m_4, s, t)$ otherwise. Similarly, for all systems s, m_6 satisfies $\mathsf{dr}(m, s, t)$ when $t = \mathsf{in}\langle 80\rangle \ldots$ and $\mathsf{dw}(m, s, t)$ otherwise. ◀

For the rest of our study, monitors with deterministic behaviour are defined as *consistently-detecting*.

▶ **Definition 6** (Consistent Detection). Monitor m *consistently detects* for system s, denoted as $\mathsf{cd}(m, s)$ iff for *all* traces t we have $\mathsf{dd}(m, s, t)$ or $\mathsf{dw}(m, s, t)$. A monitor m is *consistently-detecting*, denoted as $\mathsf{cd}(m)$, whenever $\mathsf{cd}(m, s)$ holds for *any* system s. ◀

▶ **Example 7.** Monitors m_1, m_3, m_4, m_5 and m_6 are consistently-detecting, but m_2 is *not*. Definition 6 does *not* require monitors to perform any detections. The monitor $m_7 \triangleq \mathsf{rec}\, X.\big(\mathsf{in}\langle 81\rangle.X\big) + \big(\mathsf{in}\langle 81\rangle.\mathsf{out}\langle 81\rangle.X\big)$ can consistently analyse an infinite number of traces for any s, $\mathsf{cd}(m_7)$, even though it never flags. ◀

A few comments are in order. First, Definition 6 abstracts away from the particular instances of the systems considered, the specifics of the monitor language and instrumentation mechanism used; this makes is applicable to *arbitrary* monitoring setups. Second, $\mathsf{cd}(m, s)$ may be seen as requiring an *ambiguity of d*=1 from automata theory [29, 4], for the observable behaviour specified in Definition 4. Our setting is however more general, allowing for *infinite* states and alphabets (actions). Moreover, $\mathsf{cd}(m)$ quantifies over *all* possible system compositions. Third, since Definition 6 is defined over *monitored system* behaviour, it allows us to assess the *actual* monitor behaviour at runtime. Particularly, the system quantification in $\mathsf{cd}(m)$ accounts for any (indirect) effects of a system on the execution of a monitor.

▶ **Example 8.** Whereas monitor $m_8 \triangleq \mathsf{in}\langle 81\rangle.\bot$ is (trivially) consistently-detecting in the framework of Figure 1, the monitor $m_9 \triangleq \mathsf{in}(x).\mathsf{if}\ x = 81\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathbf{0}$ is, perhaps surprisingly, *not*. Consider a (diverging) system s with behaviour $s \xrightarrow{\mathsf{in}\langle 81\rangle} s' \xrightarrow{\tau} s'$. Although $s \triangleleft m_9$ can reject the t-computation for $t{=}\mathsf{in}\langle 81\rangle$, another possible t-computation of $s \triangleleft m_9$ is

$$s \triangleleft m_9 \xrightarrow{\mathsf{in}\langle 81\rangle} (s' \triangleleft \mathsf{if}\ 81{=}81\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathbf{0}) \xrightarrow{\tau} (s' \triangleleft \mathsf{if}\ 81{=}81\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathbf{0}) \xrightarrow{\tau} \ldots$$

which *never* reaches a verdict. Therefore, we have $\neg\mathsf{cd}(m_9)$ according to Definition 6. ◀

Fourth, consistently-detecting monitors are *not* compositional, affecting the subsequent machinery.

▶ **Example 9.** Although m_8 (from Example 8) and monitor $m_{10} = \mathsf{in}\langle 81\rangle.\top$ are both consistently-detecting according to Definition 6, their composition, *i.e.*, $m_8 + m_{10}$, is clearly *not*. ◀

4 Controllability

In spite of its generality and intuitive nature, Definition 6 it is hard to automate directly as a correctness analysis. One major obstacle is the inherent universal quantification over systems and traces defining $\mathsf{cd}(m)$. In this section, we set out to give an alternative definition for describing consistently-detecting monitors that does not suffer from these shortcomings. It is based on the notion of *controllability* [20, 31] which, in discrete event settings, roughly refers to the ability to steer a (passive) entity to designated terminal states via a series of admissible controls. In our case, the monitors will constitute the passive entities to be steered, whereas the monitored systems assume the controller's role: the admissible controls are effectively the observable events in a monitoring setup that cause the monitor to transition, whereas the terminal states of interest are the conclusive verdicts. The proposed definition thus *inverts* the focus from how a system is monitored to how a monitor can be driven.

Before giving the actual definition, we first need to lift the technical machinery of Figure 1 to *sets* of monitors, $M, N \subseteq \textsc{Mon}$: this allows us to express the status whereby a monitor that can be in a *number of potential states* after being driven by a sequence of steering controls, which facilitates the analysis of non-compositional properties such as ours (see Example 9).

▶ **Definition 10.** A monitor-set M *potentially reaches* a verdict w, $\mathbf{pr}(M, w)$, when $\exists m \in M \cdot m \Longrightarrow w$, and *potentially analyses* an event η, $\mathbf{pa}(M, \eta)$, when $\exists m \in M \cdot m \stackrel{\eta}{\Longrightarrow}$. Function $\mathbf{aft}(M, \eta)$ is defined as:

$$\mathbf{aft}(M, \eta) \stackrel{\textbf{def}}{=} \textstyle\bigcup_{m \in M} \mathbf{aft}(m, \eta)$$
$$\mathbf{aft}(m, \eta) \stackrel{\textbf{def}}{=} \{\, n \mid m \Longrightarrow \cdot \xrightarrow{\eta} n \,\} \cup \{\, \mathbf{0} \mid \exists n \cdot m \Longrightarrow n \stackrel{\tau}{\not\rightarrow} \text{ and } n \stackrel{\eta}{\not\rightarrow} \,\}$$

◀

Intuitively, $\mathbf{aft}(M, \eta)$ computes the set of reachable states from every $m \in M$ when it is asked by the instrumentation of Figure 1 to analyse an event η. The two conditions defining $\mathbf{aft}(m, \eta)$ correspond to the monitored system transitions dictated by the respective rules Mon and Ter in Figure 1.

▶ **Definition 11** (Controllability). A relation $\mathcal{R} \subseteq \mathcal{P}(\textsc{Mon})$ is *controllable* iff for all $M \in \mathcal{R}$:

1. $\mathbf{pr}(M, w)$ and $w \in \{\top, \bot\}$ implies $M = \{w\}$;
2. $\mathbf{pa}(M, \eta)$ implies $\mathbf{aft}(M, \eta) \in \mathcal{R}$.

Controllability, denoted as the relation $\mathcal{C}$, is the *largest* controllable relation. A monitor m (*resp.* monitor-set M) is said to be controllable iff $\{m\} \in \mathcal{C}$ (*resp.* $M \in \mathcal{C}$). ◀

Controllability is *coinductive*: to show that a monitor m is controllable, *i.e.*, $\{m\} \in \mathcal{C}$, it suffices to provide a witness controllable relation $\mathcal{R}$ such that $\{m\} \in \mathcal{R}$. Condition (i) in Definition 11 requires that if some $m \in M$ can reach a conclusive verdict, then *every* $m' \in M$ must be able to do so *immediately*, without requiring any preceding τ-moves (hence $M = \{w\}$); this rules out the possibility of inconsistent detections and, at the same time, prohibits diverging systems from interfering with the reaching of such verdicts (see Example 8). Condition (ii) in Definition 11 intuitively requires that this condition is satisfied for *any* event η observed, by *all* the states that any $m \in M$ may transition to when analysing η.

▶ **Example 12.** We can show that m_6 (Equation 3) is controllable via the controllable relation $\mathcal{R}_1$ below:

$$\mathcal{R}_1 = \left\{ \begin{array}{l} \{m_6\}, \{\top\}, \{\mathbf{0}, \mathsf{in}\langle 82\rangle.\mathbf{0}\}, \{\mathbf{0}\}, \\ \{\mathsf{out}\langle 81\rangle.\mathbf{0}, \mathsf{out}\langle 81\rangle.\mathsf{in}\langle 82\rangle.\mathbf{0}\} \end{array} \right\}$$

$$\mathcal{R}_2 = \left\{ \begin{array}{l} \{m_7\}, \{m_7, \mathsf{out}\langle 81\rangle.m_7\}, \{\mathbf{0}, m_7\}, \\ \{\mathbf{0}, m_7, \mathsf{out}\langle 81\rangle.m_7\}, \{\mathbf{0}\} \end{array} \right\}$$

Note that $\{m_6\} \in \mathcal{R}_1$. We can also *finitely* determine that the recursive monitor m_7 (Example 7) is controllable via the relation $\mathcal{R}_2$. The reader may want to check that $\mathcal{R}_2$ is controllable. For instance,
$\mathbf{aft}(\{m_7, \mathsf{out}\langle 81\rangle.m_7\}, \mathsf{in}\langle 81\rangle) = \{\mathbf{0}, m_7, \mathsf{out}\langle 81\rangle.m_7\}$, $\mathbf{aft}(\{m_7, \mathsf{out}\langle 81\rangle.m_7\},$ $\mathsf{out}\langle 81\rangle) = \{\mathbf{0}, m_7\}$ and, importantly, $\mathbf{aft}(\{\mathbf{0}, m_7, \mathsf{out}\langle 81\rangle.m_7\}, \mathsf{in}\langle 81\rangle) = \{\mathbf{0}, m_7, \mathsf{out}\langle 81\rangle.m_7\}$ itself. ◀

Controllability coincides with Definition 6: we can use Definition 11 as a *sound* and *complete* proof technique to determine whether a monitor m satisfies $\mathsf{cd}(m)$, side-stepping universal quantifications over systems.

▶ **Theorem 13** (Consistent Detection Full Abstraction)**.** *$cd(m)$ iff $\{m\} \in \mathcal{C}$* ◀

▶ **Example 14.** As a result of Theorem 13, we can show that m_6 and m_7 are *consistently-detecting* via the controllable relations $\mathcal{R}_1$ and $\mathcal{R}_2$ of Example 12. We can also *indirectly* show that $\neg\mathsf{cd}(m_8 + m_{10})$ from Example 9, by arguing that there *cannot* be a controllable relation $\mathcal{R}$ with $\{m_8 + m_{10}\} \in \mathcal{R}$. For suppose that such an $\mathcal{R}$ exists. By Definition 11(ii) the monitor-set $\mathbf{aft}(\{m_8 + m_{10}\}, \mathsf{in}\langle 81\rangle) = \{\top, \bot\}$ must also be in $\mathcal{R}$; this, in turn, would necessarily mean that $\mathcal{R}$ is *not* controllable since $\{\top, \bot\}$ violates Definition 11(i). ◀

5 Symbolic Controllability

Controllability, Definition 11, is still not adequate for a fully automated analysis of consistently-detecting monitors. Particularly, whenever the *resp.* event value domain is infinite, quantified guards induce an infinite number of transitions, *e.g.*, $l(x).m$ generates a transition with the label $l\langle v\rangle$ for every $v \in \textsc{Val}$ (see rule GrQ). As a result, condition Definition 11(ii) may require the monitor analysis to consider a potentially infinite number of monitor-set states whenever monitor descriptions use quantified guards.

To this end, we define a *symbolic* semantics over *both* open[2] and closed monitor terms using *symbolic* events, $\theta \in \textsc{SEvt}$. These are similar to the events of Section 2 except that they carry *variables* instead of values as payloads, $l\langle x\rangle$. Symbolic transitions, $m \xrightarrow[b]{\mu} n$ are defined by the rules in Figure 2, where $\mu \in \textsc{SEvt} \cup \{\tau\}$ ranges over both symbolic and τ events, and the boolean expression b records the condition under which the LTS action may take place. For instance, the term $\mathsf{rec}\, X.m$ may unfold in all circumstances (*i.e.*, $b = \mathsf{true}$ in rule sRec) whereas the term $\mathsf{if}\, b\, \mathsf{then}\, m\, \mathsf{else}\, n$ can either τ-transition to m when b holds, or to n when the converse, $\neg b$, holds (rules sIfT and sIfF). The other key rules in Figure 2 are sGrE and sGrQ: the former transitions with a symbolic event $l\langle x\rangle$ under the condition $x{=}e$, whereas the latter transitions with a similar symbolic event under *any* circumstance. Figure 2 also defines rules for *weak symbolic transitions*, $m \overset{\theta}{\underset{b}{\Longrightarrow}} n$, and *reductions*, $m \underset{b}{\Longrightarrow} n$,

[2] Open *wrt.* expression variables $x, y, \ldots \in \textsc{Vars}$ *not* term variables $X, Y \ldots \in \textsc{TVars}$.

where both relations *aggregate* boolean constraints via conjunctions. Note that weak symbolic transitions describe transition sequences where τ-transitions *must precede* the (final) symbolic event transition. The predicate $m \overset{\mu}{\not\to}$ denotes $\nexists\, b, n \cdot m \xrightarrow[b]{\mu} n$ whereas $m \overset{\theta}{\underset{b}{\Longrightarrow}}\!\!\!\rightarrowtail$ stands for $\exists n \cdot m \overset{\theta}{\underset{b}{\Longrightarrow}}\!\!\!\rightarrowtail n$.

A *constrained monitor-set* $\langle b, M\rangle$ is a tuple where every $m \in M$ may be open, and b is a condition constraining free variables in M. Every $\langle b, M\rangle$ abstractly represents a (potentially infinite) set of *closed* monitor-sets for every valuation ρ satisfying b,

$$\{\, m\rho \mid [\![b\rho]\!] = \mathsf{true} \text{ and } m \in M \,\} \tag{4}$$

In this sense, the monitor-sets in Section 4 are special cases of constrained monitor-sets where $b = \mathsf{true}$ and M is closed. Note that whenever $\neg\mathbf{sat}(b)$, the constrained monitor-set $\langle b, M\rangle$ denotes the *empty* set of monitor-sets, $\emptyset$, which is trivially controllable by Definition 11. We lift functions such as that for free variables $\mathbf{fv}(-)$ to constrained monitor-sets in the obvious manner, *e.g.*, $\mathbf{fv}(\langle b, M\rangle) \overset{\mathsf{def}}{=} \mathbf{fv}(b) \cup \mathbf{fv}(M)$.

▶ **Example 15.** The constrained monitor-set $\langle x \geq 3, \{\mathsf{if}\ x = 2\ \mathsf{then}\ \top\ \mathsf{else}\ \bot\}\rangle$ abstractly describes all monitor-sets $\{\mathsf{if}\ x = 2\ \mathsf{then}\ \top\ \mathsf{else}\ \bot\}\,\rho$ where $\rho(x) \geq 3$. For any such ρ, *no* monitor of the form $(\mathsf{if}\ x = 2\ \mathsf{then}\ \top\ \mathsf{else}\ \bot)\rho$ can transition to a $\top$ verdict according to the concrete semantics of Figure 1. Symbolically, this may be expressed as $\neg\mathbf{sat}(x \geq 3 \wedge x = 2)$. ◀

We abstractly model controllability, Definition 11, in terms of constrained monitor-sets, the symbolic semantics of Figure 2 and the satisfiability judgement $\mathbf{sat}(b)$ defined earlier in Section 2.

▶ **Definition 16.** A constrained monitor-set $\langle b, M\rangle$ *potentially reaches* a verdict w, denoted as $\mathbf{spr}\big(\langle b, M\rangle, w\big)$, whenever $\exists m \in M, c \cdot m \underset{c}{\Longrightarrow} w$ and $\mathbf{sat}(b\wedge c)$. Moreover, $\langle b, M\rangle$ *potentially analyses* a symbolic event θ along c, denoted as $\mathbf{spa}\big(\langle b, M\rangle, \theta, c\big)$, whenever $\exists m \in M \cdot m \overset{\theta}{\underset{c}{\Longrightarrow}}\!\!\!\rightarrowtail$ and $\mathbf{sat}(b\wedge c)$. ◀

Defining the symbolic counterpart to $\mathbf{aft}(M, \eta)$ of Definition 10 is less straightforward. Intuitively, from all the valuations ρ satisfying (and represented by) b in $\langle b, M\rangle$, only a *subset* of them may satisfy the condition c in a *potentially-analyses* judgement $\mathbf{spa}(\langle b, M\rangle, \theta, c)$ from Definition 16. A correct modelling of Definition 11 therefore requires us to take this fact into account.

▶ **Example 17.** Consider the constrained monitor-set $\langle \mathsf{true}, M\rangle$ where $M = \{m_{10}, m_{11}\}$ and

$$m_{10} \triangleq \mathsf{if}\ x{=}2\ \mathsf{then}\ k\langle 1\rangle.\top\ \mathsf{else}\ k\langle 1\rangle.\bot \qquad m_{11} \triangleq \mathsf{if}\ x{\leq}1 \vee x{\geq}3\ \mathsf{then}\ k\langle 1\rangle.\bot\ \mathsf{else}\ k\langle 1\rangle.\top$$

It turns out that for any ρ satisfying true, $M\rho$ is controllable. Therefore, for the judgement $\mathbf{spa}(\langle \mathsf{true}, M\rangle, k\langle y\rangle, (x{=}2 \wedge y{=}1))$ of Definition 16, which holds since $m_{10} \overset{k\langle y\rangle}{\underset{x=2\wedge y=1}{\Longrightarrow}}\!\!\!\rightarrowtail$ and $\mathbf{sat}(\mathsf{true}\wedge x{=}2 \wedge y{=}1)$, the reachable states to be considered by a corresponding symbolic analysis (modelling *Definition* 11(*ii*)) should *not* include the residual state $\bot$, even though it may be reached after the event $k\langle y\rangle$ with $y = 1$ (see rule sGrE). The reason for this is that the conditions required to symbolically reach this state, *i.e.*, $(\neg(x{=}2) \wedge y{=}1)$ or $((x{\leq}1 \vee x{\geq}3) \wedge y{=}1)$, *cannot* be satisfied by *any* ρ that also satisfies the $\mathbf{spa}(-)$ condition $(x{=}2 \wedge y{=}1)$. Symbolically, this may be expressed as $\neg\mathbf{sat}\big((x{=}2\wedge y{=}1) \wedge \big((\neg(x{=}2) \wedge y{=}1)\big)\big)$ and $\neg\mathbf{sat}\big((x{=}2\wedge y{=}1) \wedge \big((x{\leq}1 \vee x{\geq}3) \wedge y{=}1\big)\big)$. ◀

Symbolic Transitions

$$\textsc{sVer}\ \frac{}{w \xrightarrow[\mathsf{true}]{\eta} w} \qquad \textsc{sIfT}\ \frac{}{\mathsf{if}\ b\ \mathsf{then}\ m\ \mathsf{else}\ n \xrightarrow[b]{\tau} m} \qquad \textsc{sIfF}\ \frac{}{\mathsf{if}\ b\ \mathsf{then}\ m\ \mathsf{else}\ n \xrightarrow[\neg b]{\tau} n}$$

$$\textsc{sRec}\ \frac{}{\mathsf{rec}\ X.m \xrightarrow[\mathsf{true}]{\tau} m[\mathsf{rec}\ X.m/X]} \qquad \textsc{sCh1}\ \frac{m \xrightarrow[b]{\mu} m'}{m+n \xrightarrow[b]{\mu} m'} \qquad \textsc{sCh2}\ \frac{n \xrightarrow[b]{\mu} n'}{m+n \xrightarrow[b]{\mu} n'} \qquad \textsc{sGrE}\ \frac{}{l\langle e\rangle.m \xrightarrow[e=x]{l\langle x\rangle} m}$$

$$\textsc{sGrQ}\ \frac{}{l(y).m \xrightarrow[\mathsf{true}]{l\langle x\rangle} m[x/y]} \qquad \textsc{sLet}\ \frac{}{\mathsf{let}\ x=e\ \mathsf{in}\ m \xrightarrow[\mathsf{true}]{\tau} m[e/x]}$$

Weak Symbolic Transitions and Reductions

$$\textsc{sWTr1}\ \frac{m \xrightarrow[b]{\theta} m'}{m \overset{\theta}{\underset{b}{\Rightarrow}} m'} \qquad \textsc{sWTr2}\ \frac{m \xrightarrow[b]{\tau} m' \qquad m' \overset{\theta}{\underset{c}{\Rightarrow}} m''}{m \overset{\theta}{\underset{b\wedge c}{\Rightarrow}} m''} \qquad \textsc{sWRd1}\ \frac{}{m \underset{\mathsf{true}}{\Longrightarrow} m} \qquad \textsc{sWTr2}\ \frac{m \xrightarrow[b]{\tau} m' \qquad m' \underset{c}{\Longrightarrow} m''}{m \underset{b\wedge c}{\Longrightarrow} m''}$$

Figure 2 A Symbolic Semantics for Monitors.

The complications elicited in Example 17 are even more intricate. For instance, for a particular judgement $\mathbf{spa}(\langle b, M\rangle, \theta, c)$, one could have some $m_1, m_2 \in M$ whereby $m_i \overset{\theta}{\underset{c_i}{\Rightarrow}} m_i'$ and $\mathbf{sat}(b\wedge c\wedge c_i)$ for $i \in \{1, 2\}$, but at the same time having c_1 and c_2 being incompatible with one another, *i.e.*, $\neg\mathbf{sat}(c_1\wedge c_2)$. In such cases, the respective residual states m_1' and m_2' should be analysed separately.

▶ **Definition 18.** The *relevant conditions* for a monitor-set M *wrt.* a symbolic event θ are:

$$\mathbf{rc}(M,\theta) \overset{\text{def}}{=} \{\, c \mid \exists m \in M \cdot \big(m \overset{\theta}{\underset{c}{\Rightarrow}} \ \text{ or } \exists n \cdot (m \underset{c}{\Longrightarrow} n \text{ and } n \overset{\tau}{\not\rightarrow} \text{ and } n \overset{\theta}{\not\rightarrow})\big) \,\}$$

The *satisfiability combinations* for a condition-set $\{c_1, \dots, c_n\}$ *wrt.* a condition b are:

$$\mathbf{sc}(b, \{c_1, \dots, c_n\}) \overset{\text{def}}{=} \{\, \{b, c_1', \dots, c_n'\} \mid \forall i \in 1..n \cdot (c_i' = c_i \text{ or } c_i' = \neg c_i) \,\}$$

The *reachable constrained monitor-sets* from $\langle b, M\rangle$ *after* θ with condition c are:

$$\mathbf{saft}(\langle b, M\rangle, \theta, c) \overset{\text{def}}{=} \{\, \langle \wedge B, \mathbf{saft}(M, B, \theta)\rangle \mid B \in \mathbf{sc}(b\wedge c, \mathbf{rc}(M, \theta)) \text{ and } \mathbf{sat}(\wedge B) \,\}$$

$$\mathbf{saft}(M, B, \theta) \overset{\text{def}}{=} \left\{ n \;\middle|\; \begin{array}{l} \exists m \in M, c \cdot\ \mathbf{sat}((\wedge B) \wedge c) \text{ and} \\ \big(m \overset{\theta}{\underset{c}{\Rightarrow}} n \text{ or } (\exists n' \cdot m \underset{c}{\Longrightarrow} n' \overset{\tau}{\not\rightarrow} \text{ and } n' \overset{\theta}{\not\rightarrow} \text{ and } n = \mathbf{0})\big) \end{array} \right\}$$

◀

In Definition 18, the *relevant conditions* for M *wrt.* θ, denoted as $\mathbf{rc}(M,\theta)$, are all the symbolic conditions that need to be considered to assess the reachable states from M for the symbolic event θ — they are the symbolic counterpart to the transition sequences defining $\mathbf{aft}(m, \eta)$ in Definition 10. The *satisfiability combinations* of a condition-set B *wrt.* a condition b, denoted as $\mathbf{sc}(b, B)$, capture the maximal condition subsets in B that any valuation ρ satisfying condition b also satisfies. Every condition set B' returned by $\mathbf{sc}(b, B)$

contains b itself and one condition c' for *every* boolean condition $c \in B$ (either c itself or its negation); these combination sets *partition* all the valuations ρ satisfying b. *Symbolic reachability* for $\langle b, M\rangle$ *after* θ with condition c, $\mathbf{saft}(\langle b, M\rangle, \theta, c)$ in Definition 18, is defined *wrt.* all the satisfiability combinations B of $\mathbf{rc}(M, \theta)$ for the (fixed) condition $b\wedge c$. Although $\mathbf{sc}(b\wedge c, \mathbf{rc}(M, \theta))$ partitions all the ρ satisfying $b\wedge c$, some of these partitions are *empty*. Accordingly, $\mathbf{saft}(\langle b, M\rangle, \theta, c)$ only considers the non-empty partitions via the satisfiability condition $\mathbf{sat}(\wedge B)$, where $\wedge B$ returns the syntactic conjunction formula $c_1 \wedge \ldots \wedge c_n$ for a boolean set $B = \{c_1, \ldots, c_n\}$.

It is worth remarking that the symbolic LTS of Figure 2, is image-finite [42], and thus finitely branching when considering the τ-transition graph of a term m. By König's Infinity Lemma [33] the set of constraints $\{\, c \mid m \overset{\theta}{\underset{c}{\Rightarrow}} \text{ or } \exists n \cdot (m \underset{c}{\Longrightarrow} n \text{ and } n \overset{\tau}{\not\to} \text{ and } n \overset{\theta}{\not\to})\,\}$ must be finite and, as a result, $\mathbf{rc}(M, \theta)$ is finite too for a *finite* monitor-set M. This ensures that $\mathbf{saft}(\langle b, M\rangle, \theta, c)$ is well-defined.

▶ **Definition 19** (Symbolic Controllability)**.** The relation $\mathcal{S} \subseteq \big(\mathrm{BExp} \times \mathcal{P}(\mathrm{Mon})\big)$ is called a *symbolically-controllable* relation iff for all constrained monitor-sets $\langle b, M\rangle \in \mathcal{S}$:

1. $\mathbf{spr}(\langle b,M\rangle, w)$ and $w \in \{\top, \bot\}$ implies $M = \{w\}$;
2. $\mathbf{spa}(\langle b,M\rangle, l\langle x\rangle, c)$ where $\mathbf{frsh}(\mathbf{fv}(\langle b, M\rangle)) = x$ implies $\mathbf{saft}(\langle b, M\rangle, l\langle x\rangle, c) \subseteq \mathcal{S}$.

Symbolic Controllability, denoted as $\mathcal{C}_{\mathrm{sym}}$, is the largest symbolically-controllable relation. A (closed) monitor m is symbolically-controllable iff $\langle \mathsf{true}, \{m\}\rangle \in \mathcal{C}_{\mathrm{sym}}$. ◀

The clause Definition 19(ii) assumes a function $\mathbf{frsh}(V)$ that (deterministically) returns the *next* fresh variable x that is *not* in the variable set V. When compared to Definition 11(ii), this allows us to just consider *one* (symbolic) event, $l\langle x\rangle$, for a *finite* set of constraints, as opposed to a potentially infinite set of events, *i.e.*, $l\langle v\rangle$ for every $v \in \mathrm{Val}$.

▶ **Example 20.** Recall m_{10} and m_{11} from Example 17. The monitor $m_{12} \triangleq l(x).m_{10} + l(x).m_{11}$ can be shown to be symbolically-controllable via the relation $\mathcal{S}_1$ defined below, where $b_1 = (x = 2 \wedge y = 1)$, $b_2 = (\neg(x = 2) \wedge y = 1)$, $b_3 = ((x \leq 1 \vee x \geq 3) \wedge y = 1)$ and $b_4 = (\neg(x \leq 1 \vee x \geq 3) \wedge y = 1)$; these are obtained from the relevant conditions $\mathbf{rc}(\{m_{10}, m_{11}\}, k\langle y\rangle) = \{b_1, b_2, b_3, b_4\}$.

$$\mathcal{S}_1 = \left\{ \begin{array}{ll} \langle \mathsf{true}, \{m_{12}\}\rangle, & \langle \mathsf{true}, \{m_{10}, m_{11}\}\rangle, \\ \langle (\mathsf{true}\wedge b_1)\wedge b_1 \neg b_2 \wedge \neg b_3 \wedge b_4, \{\top\}\rangle, & \langle (\mathsf{true}\wedge b_4)\wedge b_1 \wedge \neg b_2 \wedge \neg b_3 \wedge b_4, \{\top\}\rangle, \\ \langle (\mathsf{true}\wedge b_2)\wedge \neg b_1 \wedge b_2 \wedge b_3 \wedge \neg b_4, \{\bot\}\rangle, & \langle (\mathsf{true}\wedge b_3)\wedge \neg b_1 \wedge b_2 \wedge b_3 \wedge \neg b_4, \{\bot\}\rangle \end{array} \right\}$$

For illustrative purposes, we do not simplify the constraints in the constrained monitor-sets of $\mathcal{S}$ to show how these are derived. *E.g.*, $\langle (\mathsf{true}\wedge b_1)\wedge b_1 \neg b_2 \wedge \neg b_3 \wedge b_4, \{\top\}\rangle$ is obtained as a result of $\mathbf{saft}(\langle \mathsf{true}, \{m_{10}, m_{11}\}\rangle, l\langle x\rangle, \mathsf{true}\wedge b_1)$. In fact, the combination $\{(\mathsf{true}\wedge b_1), b_1, \neg b_2, \neg b_3, b_4\}$ is the only satisfiable condition-set and all the others are filtered out by $\mathbf{saft}(\langle \mathsf{true}, \{m_{10}, m_{11}\}\rangle, l\langle x\rangle, \mathsf{true}\wedge b_1)$. ◀

Symbolic Controllability, Definition 19, is sound and complete *wrt.* Controllability, Definition 11.

▶ **Theorem 21** (Controllability Full Abstraction)**.** $\{m\} \in \mathcal{C}$ *iff* $\langle \mathit{true}, \{m\}\rangle \in \mathcal{C}_{sym}$ ◀

▶ **Example 22.** Recall $m_3 \triangleq \mathsf{chl}(x).m_3'$ from Example 1, recast in terms of m_3' defined below as:

$$m_3' \triangleq \mathsf{let}\ y = enc(x)\ \mathsf{in}\ \big(\mathsf{aut}\langle y\rangle.\mathsf{ack}\langle y\rangle.\top + \mathsf{aut}(z).\mathsf{if}\ z \neq y\ \mathsf{then}\ \mathsf{ack}(z').\bot\ \mathsf{else}\ \mathsf{ack}\langle z\rangle.\top\big)$$

Example 7 stated that m_3 is consistently-detecting. This fact is hard to determine using Definition 6, whereas analyses using Definition 11 are complicated by quantifications over the values of events. By Theorems 13 and 21, we can show that m_3 is consistently-detecting via the *symbolic* controllability relation:

$$\mathcal{S}_2 = \left\{ \begin{array}{l} \langle \mathsf{true}, \{m_3\}\rangle, \quad \langle \mathsf{true}, m_3'\rangle\ , \\ \langle z = enc(x), \{\mathsf{ack}\langle enc(x)\rangle.\top, \mathsf{if}\ z \neq enc(x)\ \mathsf{then}\ \mathsf{ack}(z').\bot\ \mathsf{else}\ \mathsf{ack}\langle z\rangle.\top\}\rangle\ , \\ \langle (z = enc(x)) \wedge (w = z) \wedge (w = enc(x)), \{\top\}\rangle, \\ \langle \neg(z = enc(x)), \{\mathsf{if}\ z \neq enc(x)\ \mathsf{then}\ \mathsf{ack}(z').\bot\ \mathsf{else}\ \mathsf{ack}\langle z\rangle.\top\}\rangle\ , \quad \langle \neg(z = enc(x)), \{\bot\}\rangle \end{array} \right\}$$

In $\mathcal{S}_2$ and the ensuing discussion, we alleviate our presentation by simplifying the boolean conditions used, *e.g.*, we simply write $(z = enc(x))$ in lieu of $\big(\mathsf{true} \wedge (z = enc(x))\big)$. We highlight a few points.

First, consider the second constrained monitor-set in $\mathcal{S}_2$, namely $\langle \mathsf{true}, m_3'\rangle$. Since the semantics of Figure 2 allows expression guards and quantified guards to transition with the *same* symbolic event (albeit with different conditions) we are able to consider the *resp.* continuations in unison for the event $\mathsf{aut}\langle z\rangle$. Concretely, according to Definition 19(ii), for $\mathbf{spa}(\langle \mathsf{true}, m_3'\rangle, \mathsf{aut}\langle z\rangle, (z{=}enc(x)))$ generated by the *expression* guard weak transition of m_3', we need to ensure that the resulting monitor-set $\langle z = enc(x), \mathbf{saft}(\{m_3'\}\ , \{(z = enc(x)), \mathsf{true}\}\ , \mathsf{aut}\langle z\rangle)\rangle$ (which evaluates to the third constrained monitor-set in $\mathcal{S}_2$) is also in the symbolic relation. At the same time, for $\mathbf{spa}(\langle \mathsf{true}, m_3'\rangle, \mathsf{aut}\langle z\rangle, \mathsf{true})$ generated by the *quantified* guard weak transition of m_3', we need to ensure that two monitor-sets are in $\mathcal{S}_2$, namely $\langle z = enc(x), \mathbf{saft}(\{m_3'\}\ , \{(z = enc(x)), \mathsf{true}\}\ , \mathsf{aut}\langle z\rangle)\rangle$ (as before) but also the constrained monitor-set $\langle \neg(z = enc(x)), \mathbf{saft}(\{m_3'\}\ , \{\neg(z = enc(x)), \mathsf{true}\}\ , \mathsf{aut}\langle z\rangle)\rangle$ (which evaluates to the fifth constrained monitor-set in $\mathcal{S}_2$).

The second point we highlight about $\mathcal{S}_2$ concerns its third constrained monitor-set. In particular, the left branch of the conditional term in this set, namely $\mathsf{ack}(z').\bot$ in the term $\mathsf{if}\ z \neq enc(x)\ \mathsf{then}\ \mathsf{ack}(z').\bot\ \mathsf{else}\ \mathsf{ack}\langle z\rangle.\top$, is not considered by our analysis since its condition, $z \neq enc(x)$, is *incompatible* with the constraining condition of the monitor-set, *i.e.*, $\neg\mathbf{sat}((z = enc(x)) \wedge (z \neq enc(x)))$.

Third, we also note how the condition aggregation mechanism for the consecutive symbolic events $\mathsf{aut}\langle z\rangle$ and $\mathsf{ack}\langle w\rangle$ — transferring us from the second constrained monitor-set, $\langle \mathsf{true}, m_3'\rangle$, to the fourth, $\langle (z = enc(x)) \wedge (w = z) \wedge (w = enc(x)), \{\top\}\rangle$, via the third constrained monitor-set in $\mathcal{S}_2$ — enables us to symbolically relate the *expression* guards in m_3, which impose a condition such as $(z = enc(x))$ *upon* transition, with the *quantified* guard that imposes the same condition *after* the transition (by means of a conditional branch in its continuation). We leave it up to the interested reader to check that the remaining monitor-sets in $\mathcal{S}_2$ satisfy the conditions required by Definition 19. ◀

6 On Automating Symbolic Controllability

Despite its merits, a direct implementation of the symbolic controllability from Definition 19 still would *not* perform well for certain recursive monitor descriptions, as shown in the following example.

▶ **Example 23.** Recall monitor m_4 from Example 2. To show that it is controllable, we need to exhibit a symbolic relation that includes $\langle \mathsf{true}, \{m_4\}\rangle$. For some fresh variable x where $\mathbf{frsh}(\mathbf{fv}(\langle \mathsf{true}, \{m_4\}\rangle)) = x$, since the judgement $\mathbf{spa}(\langle \mathsf{true}, \{m_4\}\rangle, \mathsf{in}\langle x\rangle, \mathsf{true})$ holds, this relation needs to include the ensuing monitor-set $\langle \mathsf{true}, \{m_4'\}\rangle$ as well, where $m_4' \triangleq \mathsf{if}\ x{=}80\ \mathsf{then}\ \mathsf{out}\langle 81\rangle.\top\ \mathsf{else}\ \mathsf{out}\langle x\rangle.m_4$. In turn, since $\mathbf{spa}\big(\langle \mathsf{true}, \{m_4'\}\rangle, \mathsf{out}\langle y\rangle, (\neg(x = 80) \wedge y = x)\big)$ (where $\mathbf{frsh}(\mathbf{fv}(\langle b, \{m_4'\}\rangle)) = y$), the symbolic relation must also contain $\langle \neg(x = 80) \wedge y = x, \{m_4\}\rangle$. We thus reach the original monitor set $\{m_4\}$ but with a stronger condition, namely

$\neg(x=80)\wedge y=x$. By extension of this reasoning, it is not hard to see that the symbolic relation required by Definition 19 needs to be infinitely large. ◀

The problem exhibited by Example 23 is that the condition aggregating mechanism of Definition 19 does not specify any means for consolidating the boolean condition b constraining a monitor set M in $\langle b, M\rangle$, *i.e.,* a form of garbage collection of redundant conditions. For instance, in the constrained monitor-set $\langle \neg(x=80)\wedge y=x, \{m_4\}\rangle$ of Example 23, the condition $(\neg(x=80)\wedge y=x)$ plays *no effective role* in constraining the free variables in $\{m_4\}$, of which there are none. We therefore optimise Definition 19 in a sound (and complete) manner by taking into consideration boolean sub-conditions that can be isolated and discarded. This leads to an improved automated analysis for consistently-detecting monitors.

▶ **Definition 24** (Optimised Symbolic Controllability)**.** The consolidation of a boolean expression b *wrt.* a variable set V, denoted as $\mathbf{cns}(b, V)$, is defined as:

$$\mathbf{cns}(b, V) \stackrel{\text{def}}{=} b_1 \text{ whenever } \mathbf{prt}(b, V) = \langle b_1, b_2\rangle \text{ for some } b_2$$

where the boolean expression partitioning operation $\mathbf{prt}(b, V)$ is defined as:

$$\mathbf{prt}(b, V) \stackrel{\text{def}}{=} \begin{cases} \langle b_1, b_2\rangle & \text{if } \mathbf{sat}(b) \text{ and } b = b_1\wedge b_2 \text{ and } \big(\mathbf{fv}(b_1)\subseteq V\big) \text{ and } \big(V\cap\mathbf{fv}(b_2)=\emptyset\big) \\ \langle b, \mathsf{true}\rangle & \text{otherwise} \end{cases}$$

Let the *optimised symbolic reachability* from $\langle b, M\rangle$ for θ and c, $\mathbf{osaft}(\langle b, M\rangle, \theta, c)$, be defined as:

$$\mathbf{osaft}(\langle b, M\rangle, \theta, c) \stackrel{\text{def}}{=} \left\{ \langle \mathbf{cns}(\wedge B, V), \mathbf{saft}(M, B, \theta)\rangle \;\middle|\; \begin{array}{c} B \in \mathbf{sc}(b\wedge c, \mathbf{rc}(M,\theta)) \text{ and } \mathbf{sat}(\wedge B) \\ \text{and } V = \mathbf{fv}(\mathbf{saft}(M, B, \theta)) \end{array} \right\}$$

A relation $\mathcal{S} \subseteq \big(\mathrm{BExp} \times \mathcal{P}(\mathrm{Mon})\big)$ is called *optimised symbolically-controllable* iff for all $\langle b, M\rangle\in\mathcal{S}$:

1. $\mathbf{spr}(\langle b,M\rangle, w)$ and $w \in \{\top, \bot\}$ implies $M = \{w\}$;
2. $\mathbf{spa}(\langle b,M\rangle, l\langle x\rangle, c)$ where $\mathbf{frsh}(\mathbf{fv}(\langle b, M\rangle))=x$ implies $\mathbf{osaft}(\langle b, M\rangle, l\langle x\rangle, c) \subseteq \mathcal{S}$.

The *largest* optimised symbolically-controllable relation is denoted by $\mathcal{C}^{\text{opt}}_{\text{sym}}$. A (closed) monitor m is said to be optimised symbolically-controllable iff $\langle \mathsf{true}, \{m\}\rangle \in \mathcal{C}^{\text{opt}}_{\text{sym}}$. ◀

We highlight the salient points from Definition 24. First, boolean consolidation in a constrained monitor-set, $\langle\mathbf{cns}(b), M\rangle$, should *not* change the set of concrete monitor sets represented by $\langle b, M\rangle$ and, for this reason, we cannot consolidate *unsatisfiable* boolean conditions. For instance, even if $x\notin \mathbf{fv}(M)$, it is still unsound to optimise $\langle\mathsf{true}\wedge(x\neq x), M\rangle$ to $\langle\mathsf{true}, M\rangle$ based on the fact that $\neg\mathbf{sat}(x\neq x)$. Concretely, from Equation 4 of Section 5 we know that $\langle\mathsf{true}\wedge x\neq x, M\rangle$ denotes the empty set of monitor-sets, $\emptyset$, whereas $\langle\mathsf{true}, M\rangle$ represents the set $\{\, m\rho \mid [\![\mathsf{true}\rho]\!] = \mathsf{true} \text{ and } m \in M \,\}$. Second, consolidation should ideally filter out as much redundant constraints as possible, *e.g.,* in $\langle b_1\wedge b_2, M\rangle$ we should remove b_2 whenever $\mathbf{fv}(b_2)\cap\mathbf{fv}(M) = \emptyset$. In Definition 24 we require the strongest possible condition for the residual condition b_1 in $\langle b_1\wedge b_2, M\rangle$, *i.e.,* $\mathbf{fv}(b_1) \subseteq \mathbf{fv}(M)$, which indirectly implies that the *resp.* condition variables are partitioned $\mathbf{fv}(b_1)\cap\mathbf{fv}(b_1) = \emptyset$. This partitioning is crucial for a sound consolidation, *e.g.,* in $\langle(\neg(x=80)\wedge y=x), M\rangle$, it is unsound to just remove the subcondition $\neg(x=80)$ when $x \notin \mathbf{fv}(M)$ and $y \in \mathbf{fv}(M)$. Although $\mathbf{prt}(b, V)$ can be refined further (while still observing core requirements for soundness such as variable condition partitioning), in Definition 24 we opted for a less elaborate condition that suffices our exposition. Third, we highlight the fact that the conditions specifying $\mathbf{cns}(b)$ in Definition 24 yield a unique consolidated condition up to semantic equivalence meaning that, in an implementation of the framework, this can be defined as a function.

▶ **Theorem 25** (Optimised Controllability). $\langle \mathit{true}, \{m\}\rangle \in \mathcal{C}^{opt}_{sym}$ *iff* $\langle \mathit{true}, \{m\}\rangle \in \mathcal{C}_{sym}$ ◀

▶ **Example 26.** As a result of Theorems 13, 21 and 25, we can show that m_4 of Example 2 is consistently-detecting by exhibiting the optimised symbolic relation $\mathcal{S}_3$ below (m_4' is the monitor defined earlier in Example 23). For expository purposes, we show how the consolidated boolean expressions are calculated. In particular, the first constrained monitor-set in $\mathcal{S}_3$ denotes both the starting pair $\langle \mathsf{true}, \{m_4\}\rangle$, but also the pair $\langle \mathbf{cns}(x = 80 \wedge y = 81, \mathbf{fv}(\{m_4\})), \{m_4\}\rangle$.

$$\mathcal{S}_3 = \left\{ \langle \underbrace{\mathsf{true}}_{\mathbf{cns}(\neg(x=80)\wedge y=x,\emptyset)}, \{m_4\}\rangle,\ \langle \underbrace{\mathsf{true}}_{\mathbf{cns}(x=80,\emptyset)}, \{\bot\}\rangle,\ \langle \mathsf{true}, \{m_4'\}\rangle,\ \langle \underbrace{\mathsf{true}}_{\mathbf{cns}(x=80\wedge y=81,\emptyset)}, \{\top\}\rangle \right\}$$

The interested reader may check that the conditions of Definition 24 are satisfied by $\mathcal{S}_3$. ◀

Using standard techniques [42, 3], an algorithm constructing symbolically-controllable relations from Definition 24 can be extracted more easily. Moreover, the completeness aspect in Theorems 13, 21, and 25 should enable such an automation to infer a *counter-example* (system and trace) from failed attempts, thereby explaining why a monitor is *not* consistently-detecting.

▶ **Example 27.** Recall monitor m_2 from Example 1. Assuming the shorthand abbreviations $m_2' \triangleq \mathsf{aut}\langle y\rangle.\mathsf{ack}\langle y\rangle.\top + \mathsf{aut}(z).\mathsf{if}\ z \neq y\ \mathsf{then}\ \mathsf{ack}(z').\bot$ and $M = \{\mathsf{if}\ z \neq enc(x)\ \mathsf{then}\ \mathsf{ack}(z').\bot, \mathsf{ack}\langle enc(x)\rangle.\top\}$, compiling a relation satisfying Definition 24 fails because it needs to include:

$$\begin{array}{lll} \langle \mathsf{true}, \{\mathsf{let}\ y = enc(x)\ \mathsf{in}\ m_2'\}\rangle & \text{since} & \mathbf{spa}(\langle \mathsf{true}, \{m_2\}\rangle, \mathsf{chl}\langle x\rangle, \mathsf{true}) \\ \langle z = enc(x), M\rangle & \text{since} & \mathbf{spa}(\langle \mathsf{true}, \{\mathsf{let}\ y = enc(x)\ \mathsf{in}\ m_2'\}\rangle, \mathsf{aut}\langle z\rangle, z = enc(x)) \\ \langle \mathsf{true}, \{\top, \mathbf{0}\}\rangle & \text{since} & \mathbf{spa}(\langle z = enc(x), M\rangle, \mathsf{ack}\langle w\rangle, (w = enc(x))) \end{array}$$

The final pair $\langle \mathsf{true}, \{\top, \mathbf{0}\}\rangle$ violates Definition 24(i) and, via the symbolic events on right-hand column conatining the $\mathbf{spa}(-)$ assertions that lead to it, one can construct the counter-example inducing the inconsistent detection, *i.e.*, a system s producing the trace $\mathsf{chl}\langle v\rangle \cdot \mathsf{aut}\langle u\rangle \cdot \mathsf{ack}\langle u\rangle$ where $u = enc(v)$. ◀

7 Conclusion

Monitors should provide guarantees that they will operate correctly when instrumented with *any* system [35]. At the same time, for this requirement to be scalable, the corresponding correctness analysis that determines it must also be *compositional*: monitors should be verified separately, independent of the systems they may be instrumented with. The fact that monitors tend to be considerably smaller (in size and complexity) than the systems they observe further justifies this point. This paper provides two definitions that formalise deterministic monitoring behaviour, Definition 6 (consistently-detecting monitors) and Definition 11 (controllable monitors), that address these seemingly conflicting concerns; it also shows that the two definitions coincide, Theorem 13. In addition, the paper also studies alternative definitions for controllability, Definitions 19 and 24, that enable the implementation of sound and complete symbolic analyses, Theorems 21 and 25.

Our methods provide a systematic way for factoring out auxiliary reasoning on data from the analysis relating to the branching structure of the monitors; the former kind of reasoning can be determined by calling on an independent satisfiability solver. In fact, for the specific case of our expository monitor language in Figure 1, one can show that our methods yield *finite* symbolic transition graphs, making the latter reasoning decidable modulo the expression and boolean language used. The results obtained in this work should also be general enough to be applicable to other monitoring systems. For instance, Definitions 6, 11, 19, and 24 are independent of the kind of systems monitored, the syntax of the monitor language, and the event value domains and expression languages used. Instead, they are defined in terms of generic characteristics such as their LTS semantics. As a result, extending the monitor language with constructs such as parallel composition would not affect the existing framework. The instrumentation relation one adopts for composing monitors with systems necessarily affects the compositional properties and the correspondence between the respective definitions. However, these changes do not impinge on the general structure of our definitions and should be local to the detection condition in the *resp.* controllability definitions, namely Definition 11(i), 19(i) and 24(i), and the reachability-set definitions $\mathbf{aft}(-), \mathbf{saft}(-)$ and $\mathbf{osaft}(-)$ of Definitions 10, 18, and 24.

Future work. We will further investigate the implementability aspects of our analysis, possibly as an extension to existing model-checking tools. This may raise further issues and adjustments to our definitions *e.g.,* it may be more efficient to batch the satisfiability checks in Definition 24. We plan to apply this to existing transition-based monitor specifications such as [5, 46] and validate its feasibility as an automated specification assistant.

Related Work. The need for determinising monitor syntheses from logical specifications is frequently discussed in the literature [7, 23]. In [1], the authors employ a trace-based definition of deterministic monitors that takes into consideration verdicts (similar to our definition for consistent detections of Definition 6 but without considering universal quantification over system instrumentations) and establish complexity bounds for determinising monitors *wrt.* this definition. Set-simulations, which are related to our monitor-sets, have been used as a proof technique for testing preorders in [15] but do not consider symbolic analyses. Acceptor ambiguity [29] is closely related to our notion of consistent detection with respect to the three outcomes of acceptance, rejection and withholding, as specified in Definition 6. Crucially, however, our definition universally quantifies over all possible system compositions. Subsequently, the main endeavour of our work was that of developing sound and complete compositional techniques to alleviate the analysis for consistent detection; we are unaware of any compositional or coinductive techniques used for determining acceptor ambiguity. Symbolic LTSs were studied extensively for value-passing CCS in [27, 28], but their use in controllability for reasoning about consistent monitor detections is, to our knowledge, novel. The particular setting where it is used, namely the instrumentation composition relation and the use of monitor-sets, also require new technical machinery, such as that of Definition 18. Our definition of controllability, Definition 11, is related to viability (usability) for clients in contract compliance [39] and must-testing [9]. Particularly, in the case of compliance, viability is defined coinductively and is satisfied whenever there *exists* a server that can engage with the client so as to lead it to success whenever interaction terminates. Apart from the universal quantifications over systems (viability existentially quantifies over servers), our work differs from [39, 9] *wrt.* the treatment of verdicts considered, the composition relation used (*i.e.,* instrumentation), and the development of a symbolic analysis for handling of action/event data.

Acknowledgements. The author acknowledges the Dagstuhl seminar 17051 and would like to thank Luca Aceto, Antonis Achilleos, Duncan Attard, Giovanni Bernardi, Ian Cassar, Anna Ingólfsdóttir, Giles Reger and anonymous reviewers for their help and suggestions.

References

1. Luca Aceto, Antonis Achilleos, Adrian Francalanza, Anna Ingólfsdóttir, and Sævar Örn Kjartansson. On the Complexity of Determinizing Monitors. In *CIAA*, pages 1–13, 2017. doi:10.1007/978-3-319-60134-2_1.
2. Luca Aceto, Anna Ingólfsdóttir, Kim Guldstrand Larsen, and Jiri Srba. *Reactive Systems: Modelling, Specification and Verification*. Cambridge Univ. Press, 2007.
3. Luca Aceto, Anna Ingolfsdottir, and Jiri Srba. *Advanced Topics in Bisimulation and Coinduction*, chapter The Algorithmics of Bisimilarity. Cambridge Univ. Press, 2011.
4. Cyril Allauzen, Mehryar Mohri, and Ashish Rastogi. General algorithms for testing the ambiguity of finite automata. In *DTL*, pages 108–120, 2008. doi:10.1007/978-3-540-85780-8_8.
5. Howard Barringer, Yliès Falcone, Klaus Havelund, Giles Reger, and David E. Rydeheard. Quantified Event Automata: Towards Expressive and Efficient Runtime Monitors. In *FM*, pages 68–84, 2012. doi:10.1007/978-3-642-32759-9_9.
6. David Basin, Felix Klaedtke, Srdjan Marinovic, and Eugen Zălinescu. Monitoring compliance policies over incomplete and disagreeing logs. In *RV*, pages 151–167, 2013. doi:10.1007/978-3-642-35632-2_17.
7. Andreas Bauer, Martin Leucker, and Christian Schallhart. Runtime verification for LTL and TLTL. *TOSEM*, 20(4):14, 2011.
8. Shay Berkovich, Borzoo Bonakdarpour, and Sebastian Fischmeister. Runtime verification with minimal intrusion through parallelism. *Formal Methods in System Design*, 46(3):317–348, 2015. doi:10.1007/s10703-015-0226-3.
9. Giovanni Bernardi and Adrian Francalanza. Full-abstraction for must testing preorders. In *COORDINATION*, pages 237–255, 2017. doi:10.1007/978-3-319-59746-1_13.
10. Laura Bocchi, Tzu-Chun Chen, Romain Demangeon, Kohei Honda, and Nobuko Yoshida. Monitoring networks through multiparty session types. *TCS*, 669:33–58, 2017. doi:10.1016/j.tcs.2017.02.009.
11. Borzoo Bonakdarpour, Pierre Fraigniaud, Sergio Rajsbaum, David A. Rosenblueth, and Corentin Travers. Decentralized asynchronous crash-resilient runtime verification. In *CONCUR*, pages 16:1–16:15, 2016. doi:10.4230/LIPIcs.CONCUR.2016.16.
12. Ian Cassar and Adrian Francalanza. Runtime Adaptation for Actor Systems. In *RV*, volume 9333, pages 38–54. Springer, 2015.
13. Ian Cassar and Adrian Francalanza. On Implementing a Monitor-Oriented Programming Framework for Actor Systems. In *iFM*, pages 176–192, 2016.
14. Feng Chen and Grigore Roşu. MOP: An Efficient and Generic Runtime Verification Framework. In *OOPSLA*, pages 569–588, 2007. doi:10.1145/1297027.1297069.
15. Rance Cleaveland and Matthew Hennessy. Testing equivalence as a bisimulation equivalence. *FACS*, 5(1):1–20, 1993.
16. Christian Colombo, Adrian Francalanza, Ruth Mizzi, and Gordon J. Pace. polylarva: Runtime verification with configurable resource-aware monitoring boundaries. In *SEFM*, pages 218–232, 2012.
17. Marcelo d'Amorim and Grigore Roşu. Efficient monitoring of ω-languages. In *CAV*, pages 364 – 378, 2005.
18. Søren Debois, Thomas Hildebrandt, and Tijs Slaats. Safety, liveness and run-time refinement for modular process-aware systems with dynamic sub processes. In *FM*, pages 143–160, 2015. doi:10.1007/978-3-319-19249-9_10.
19. Normann Decker and Daniel Thoma. On freeze LTL with ordered attributes. In *FOSSACS*, pages 269–284, 2016. doi:10.1007/978-3-662-49630-5_16.
20. John Dorsey. *Continuous and Discrete Control Systems: Modeling, Identification, Design, and Implementation*. McGraw-Hill, 2001.

21 Pierre Fraigniaud, Sergio Rajsbaum, and Corentin Travers. On the number of opinions needed for fault-tolerant run-time monitoring in distributed systems. In *RV*, pages 92–107, 2014. doi:10.1007/978-3-319-11164-3_9.

22 Adrian Francalanza. A Theory of Monitors. In *FoSSaCS*, pages 145–161, 2016.

23 Adrian Francalanza, Luca Aceto, and Anna Ingolfsdottir. Monitorability for the Hennessy–Milner logic with recursion. *FMSD*, pages 1–30, 2017. doi:10.1007/s10703-017-0273-z.

24 Adrian Francalanza and Aldrin Seychell. Synthesising Correct concurrent Runtime Monitors. *FMSD*, 46(3):226–261, 2015. doi:10.1007/s10703-014-0217-9.

25 Jan Friso Groote and M.P.A. Sellink. Confluence for process verification. *TCS*, 170(1):47 – 81, 1996. doi:10.1016/S0304-3975(96)80702-X.

26 Yuqin He, Xiangping Chen, and Ge Lin. Composition of monitoring components for on-demand construction of runtime model based on model synthesis. In *Internetware*, pages 1–5, 2013. doi:10.1145/2532443.2532472.

27 Matthew Hennessy and Anna Ingolfsdottir. A Theory of Communicating Processes with Value Passing. *Information and Computation*, 107(2):202 – 236, 1993. doi:10.1006/inco.1993.1067.

28 Matthew Hennessy and Huimin Lin. Symbolic bisimulations. *TCS*, 138(2):353 – 389, 1995. doi:10.1016/0304-3975(94)00172-F.

29 Oscar H. Ibarra and Bala Ravikumar. On Sparseness, Ambiguity and other decision problems for Acceptors and Transducers. In *STACS*, pages 171–179, 1986. doi:10.1007/3-540-16078-7_74.

30 Limin Jia, Hannah Gommerstadt, and Frank Pfenning. Monitors and blame assignment for higher-order session types. In *POPL*, pages 582–594, 2016. doi:10.1145/2837614.2837662.

31 Jerzy Klamka. *Control System, Robotics and Automation*, volume 7, chapter System Characteristics: Stability, Controllability, Observability. EOLLS, 2009.

32 John Klein and Ian Gorton. Runtime Performance Challenges in Bigdata Systems. In *WOSP*, pages 17–22, 2015. doi:10.1145/2693561.2693563.

33 D. König. Über eine Schlussweise aus dem Endlichen ins Unendliche. *Acta Litt. ac. sci. Szeged*, 3, 1927.

34 Dexter Kozen. Results on the propositional μ-calculus. *TCS*, 27:333–354, 1983.

35 Jonathan Laurent, Alwyn Goodloe, and Lee Pike. Assuring the Guardians. In *RV*, pages 87–101, 2015.

36 Jay Ligatti, Lujo Bauer, and David Walker. Edit automata: enforcement mechanisms for run-time security policies. *Int. J. Inf. Secur.*, 4(1-2):2–16, 2005. doi:10.1007/s10207-004-0046-8.

37 Qingzhou Luo and Grigore Roşu. EnforceMOP: A Runtime Property Enforcement System for Multithreaded Programs. In *ISSTA*, New York, NY, USA, 2013. ACM. doi:10.1145/2483760.2483766.

38 Patrick O'Neil Meredith, Dongyun Jin, Dennis Griffith, Feng Chen, and Grigore Roşu. An overview of the MOP runtime verification framework. *STTT*, 14(3):249–289, 2012. doi:10.1007/s10009-011-0198-6.

39 Luca Padovani. Contract-based discovery of web services modulo simple orchestrators. *TCS*, 411(37), 2010.

40 Anna Philippou and David Walker. On confluence in the π-calculus. In *ICALP*, pages 314–324, 1997. doi:10.1007/3-540-63165-8_188.

41 Giles Reger, Helena Cuenca Cruz, and David E. Rydeheard. MarQ: Monitoring at Runtime with QEA. In *TACAS*, pages 596–610, 2015. doi:10.1007/978-3-662-46681-0_55.

42 Davide Sangiorgi. *An introduction to Bisimulation and Coinduction*. Cambridge University Press, 2012.

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

44 Michael Sipser. *Introduction to the Theory of Computation.* Cengage Learning, 3rd edition, 2012.

45 Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Inf.& Comp.*, 115(1):1–37, 1994. doi:10.1006/inco.1994.1092.

46 Yoriyuki Yamagata, Cyrille Artho, Masami Hagiya, Jun Inoue, Lei Ma, Yoshinori Tanabe, and Mitsuharu Yamamoto. Runtime monitoring for concurrent systems. In *RV*, pages 386–403, 2016. doi:10.1007/978-3-319-46982-9_24.