








# On Probabilistic Monitorability<sup>\*</sup>

Luca Aceto<sup>1,4</sup> , Antonis Achilleos<sup>1</sup> , Elli Anastasiadi<sup>1</sup> , Adrian Francalanza<sup>2</sup> , Anna Ingólfssdóttir<sup>1</sup> , Karoliina Lehtinen<sup>3</sup> , and Mathias Ruggaard Pedersen<sup>1</sup> 

<sup>1</sup> ICE-TCS, Department of Computer Science, Reykjavík University, Iceland

<sup>2</sup> Department of Computer Science, University of Malta, Msida, Malta

<sup>3</sup> CNRS, Aix-Marseille University and University of Toulon, LIS, Marseille, France

<sup>4</sup> Gran Sasso Science Institute, L’Aquila, Italy

**Abstract.** This paper investigates monitorability in the context of probabilistic systems. We specify how monitor verdicts, reached over finite (partial) traces, can be given a probabilistic interpretation. For monitors that are used to verify properties at runtime, we also relate their probabilistic verdicts to the probability that a trace satisfies the property of interest. This leads us to define probabilistic monitor soundness and completeness, which are then used to formulate probabilistic monitorability. Surprisingly, we show that the resulting notions coincide with standard monitorability definitions from the literature. This allows us to transfer prior results from the standard setting to the probabilistic realm.

## 1 Introduction

Some of Thomas A. Henzinger’s recent work has given seminal contributions to the field of runtime monitoring—see, for instance, the papers [18,21,22,29,30]. Moreover, in light of the new Advanced Grant he received from the European Research Council in April 2021 for the project ‘Vigilant Algorithmic Monitoring Of Software (VAMOS)’, we expect that, in the coming years, Thomas A. Henzinger and his group at IST Austria will contribute substantial new developments to both the theoretical foundations and the practice of runtime monitoring for modern software-based systems that rely on artificial intelligence and cloud computing, amongst other paradigms, and interact with an uncertain cyber-physical environment. To our mind, Thomas A. Henzinger cogently articulated the vision for the VAMOS project, and indeed for the field of runtime monitoring as a whole, in his keynote address at the 2020 edition of the conference on Runtime

---

<sup>\*</sup> This research was supported by the projects ‘TheoFoMon: Theoretical Foundations for Monitorability’ (no. 163406-051), ‘Open Problems in the Equational Logic of Processes (OPEL)’ (no. 196050-051) and ‘MoVeMnt: Mode(l)s of Verification and Monitorability’ (no. 217987-051) of the Icelandic Research Fund, project BehAPI, funded by the EU H2020 RISE programme under the Marie Skłodowska-Curie grant (no. 778233), and the Italian MIUR project PRIN 2017FTXR7S IT MATTERS ‘Methods and Tools for Trustworthy Smart Systems’.

Verification. The key idea is to ensure that the runtime behaviour of critical software components be *always* observed and vetted online by other software devices, the so-called *monitors*, in order to identify possible misbehaviours at execution time in a timely fashion. Ideally, the monitors used for that purpose should be developed independently of the systems whose behaviour they observe and be synthesised automatically from system specifications. According to an IST Austria press release<sup>5</sup>, the aim of the project VAMOS is to increase the robustness, dependability and trustworthiness of critical software systems by harnessing ‘the increasing availability of hardware resources, from multicore processors to data centers.’

As we trust the above prefatory text makes clear, monitors are key components in runtime monitoring. They are passive computational entities that observe the execution of a system, *i.e.*, a finite trace of events, to determine properties about it [11,23,24,25]. When monitoring the behaviour of systems involving randomised choices, such as communication protocols and randomised algorithms, the observed systems are naturally equipped with probabilistic information about their branching behaviour and, due to their passivity, monitors intrinsically inherit this probabilistic behaviour. It is then natural, and fairly straightforward to ascribe the associated probability measure to monitor verdicts. However, when relating monitors to (linear-time) specifications, it is unclear whether the resulting probabilistic verdicts, reached by the monitor over finite trace observations, are in accordance with the probability that the completed trace (which may be infinite) satisfies the specification being monitored at runtime. This constitutes a monitorability problem that, to our mind, still deserves to be studied.

This paper investigates monitorability for probabilistic systems. Our results are modelled on the monitorability definition given in [4,25] which, opportunely, teases apart the monitor behaviour from the semantics of the properties being monitored, and relates them in terms of standard soundness and completeness criteria; it has also been formally related to other variants in the literature [6] and used for branching-time settings [2,3,5,26]. Our contributions in this celebratory article are as follows:

1. We define probabilistic versions of monitor soundness and completeness relating the probability of reaching each verdict after a finite prefix to the probability that a complete trace extending it satisfies the property (Definitions 8 and 9).
2. We show a surprising correspondence between probabilistic monitorability and its classical variant (Theorem 1), which allows us to inherit prior results such as syntactic characterisations of monitorable properties.
3. We show how this framework is general enough to be adapted to probabilistic settings that permit a margin of error (Definition 11 and Theorem 2).
4. Section 4 concludes our contribution with an application of these results to estimate probabilities in settings that allow for repeated monitored runs while still treating the observed system as a black box.

---

<sup>5</sup> See <https://ist.ac.at/en/news/erc-grants-beacon-of-scientific-success/>.

We end this article with some concluding remarks, a discussion of related literature and some avenues for future research (Section 5).

## 2 Preliminaries

We introduce the core concepts of measure and probability theory needed in this study. We refer the interested reader to [7,10,12] for a more in-depth presentation.

**Definition 1 ( $\sigma$ -algebra [10, p. 754]).** For a set  $X$ , a  $\sigma$ -algebra on  $X$  is a set  $\Sigma \subseteq 2^X$  such that

- $X \in \Sigma$ ,
- if  $A \in \Sigma$  then  $\bar{A} = X \setminus A \in \Sigma$  (closure under complement), and
- if  $A_1, A_2, \dots \in \Sigma$  then  $\bigcup_{n \geq 1} A_n \in \Sigma$  (closure under countable unions).

A pair  $(X, \Sigma)$  of a set  $X$  together with a  $\sigma$ -algebra  $\Sigma$  on  $X$  is known as a measurable space. If  $\Sigma$  is a  $\sigma$ -algebra and  $A \in \Sigma$ , we say that  $A$  is measurable for  $\Sigma$ , and if  $\Sigma$  is evident from the context, we simply say that  $A$  is measurable. With a  $\sigma$ -algebra on  $X$  at hand, we can define a probability measure on  $X$ .

**Definition 2 (Probability measure [10, p. 754]).** Given a measurable space  $(X, \Sigma)$ , a probability measure is a function  $\mathbb{P} : \Sigma \rightarrow [0, 1]$  such that  $\mathbb{P}(X) = 1$  and  $\mathbb{P}(\bigcup_{i \in I} A_i) = \sum_{i \in I} \mathbb{P}(A_i)$  for any countable, pairwise disjoint collection  $\{A_i\}_{i \in I} \subseteq \Sigma$ . We denote by  $\mathcal{D}(X)$  the set of all probability measures on  $X$ .

Hence a probability measure assigns a probability to any measurable set in such a way that, for example,  $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B)$ , if  $A$  and  $B$  are disjoint sets, as well as ensuring that  $\mathbb{P}(\emptyset) = 0$  and  $\mathbb{P}(\bar{A}) = 1 - \mathbb{P}(A)$ , for each  $A$ .

A probabilistic system is one in which the evolution of the system is governed by some probability distribution. We use here one of the simplest probabilistic systems, namely (generative) labelled Markov chains. Assume a fixed, finite set of actions  $\text{Act}$ .

**Definition 3 (Labelled Markov chain).** A labelled Markov chain is a tuple  $M = (S, s_*, \Delta)$ , where  $S$  is a countable set of states,  $s_* \in S$  is the start state, and  $\Delta : S \rightarrow \mathcal{D}(\text{Act} \times S)$  is the transition function assigning to each state a distribution over actions and states.

*Remark 1.* The model defined above is essentially an action-based formulation of the hidden Markov chains considered in [34], which are called labelled Markov chains in [17]. Note, however, that the state-observation function in the models studied in those papers associates an observation with each state deterministically.

A labelled Markov chain  $M = (S, s_*, \Delta)$  currently in state  $s \in S$  evolves by choosing action  $a$  and state  $s'$  with probability  $\Delta(s)(a, s')$ , moving to  $s'$  while outputting the action  $a$ . In this paper we consider the trace-based behaviour of labelled Markov chains. A trace is an infinite sequence of actions  $a_1 a_2 \dots \in \text{Act}^\omega$ .

We let  $\pi, \pi'$  range over traces. A finite trace is a sequence of actions  $a_1 a_2 \dots a_n \in \text{Act}^*$ , which we range over by  $w, w'$ , and sets of finite traces are ranged over by  $F$ . We denote the empty trace by  $\varepsilon$ . Given two finite traces  $w$  and  $w'$ , we write  $w \preceq w'$  if  $w$  is a prefix of  $w'$ , meaning that there exists a finite trace  $w''$  such that  $ww'' = w'$ . For a trace  $\pi = a_1 a_2 \dots$ , we let  $\pi\langle i \rangle = a_i$ ,  $\pi|_i = a_1 \dots a_i$  and  $\pi^i = a_{i+1} \dots$ .

For a labelled Markov chain  $M = (S, s_*, \Delta)$ , we obtain a measurable space of traces  $(\text{Act}^\omega, \Sigma)$  using the cylinder construction (see e.g. [10, pp. 757–758]) as follows. Given a finite trace  $a_1 \dots a_n$ , we define the cylinder of that trace as

$$\mathbb{C}(a_1 \dots a_n) = \{\pi \in \text{Act}^\omega \mid \pi|_n = a_1 \dots a_n\}.$$

Thus  $\mathbb{C}(a_1 \dots a_n)$  is the set of infinite traces that all agree on the finite prefix  $a_1 \dots a_n$ . In the following, we fix the  $\sigma$ -algebra  $\Sigma$  on  $\text{Act}^\omega$  defined as the smallest  $\sigma$ -algebra containing all cylinders. For a given state  $s$ , we define a probability measure  $\mathbb{P}_M^s$  on the measurable space  $(\text{Act}^\omega, \Sigma)$  inductively as  $\mathbb{P}_M^s(\mathbb{C}(\varepsilon)) = 1$  and

$$\mathbb{P}_M^s(\mathbb{C}(a_1 a_2 \dots a_n)) = \sum_{s' \in S} \Delta(s)(a_1, s') \cdot \mathbb{P}_M^{s'}(\mathbb{C}(a_2 \dots a_n)).$$

Although we only define  $\mathbb{P}_M^s$  on cylinders, the probability extends uniquely to the whole  $\sigma$ -algebra  $\Sigma$  using the Hahn-Kolmogorov theorem [36, Theorem 1.7.8]. Thus for any measurable set  $A \in \Sigma$ , the probability  $\mathbb{P}_M^s(A)$  is well-defined.

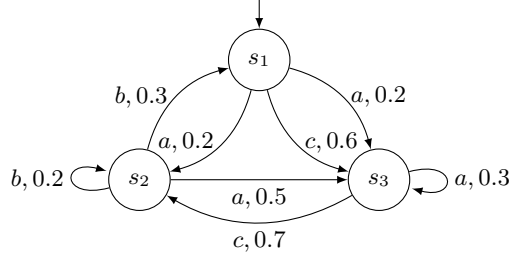
### 3 Monitoring

Runtime verification employs monitors to observe the behaviour of some system, typically as a black box; the system emits sequences of events/actions from some set  $\text{Act}$ . A monitor accepts if the (finite) observations lead it to conclude that the system satisfies a property of interest, and rejects if it observes enough events to conclude that the property is violated. Our objective is to give an account of monitoring in the case where the system being monitored is a probabilistic system. In this case, the monitor itself is still non-probabilistic, and can only observe the actions emitted by the probabilistic system. Thus the monitored system is still a black box, and the monitor has no way of knowing the internal state or the transition probabilities of the system.

**Definition 4 (Monitor).** A monitor  $m = (F_{acc}, F_{rej})$  is a pair of sets of finite traces  $F_{acc}, F_{rej} \subseteq \text{Act}^*$  satisfying: (i)  $F_{acc} \cap F_{rej} = \emptyset$ ; (ii) for  $i \in \{acc, rej\}$ :

if  $w \in F_i$  then for any  $w' \in \text{Act}^*$  where  $w \preceq w'$  we also have  $w' \in F_i$ .

The traces in  $F_{acc}$  denote the finite observations accepted by the monitor whereas those in  $F_{rej}$  are the finite traces the monitor rejects. Condition (ii) in Definition 4 ensures that verdicts (i.e., acceptances and rejections) are irrevocable. For a set  $F \subseteq \text{Act}^*$  we define  $\mathbb{C}(F) = \bigcup_{w \in F} \mathbb{C}(w)$ , so that  $\mathbb{C}(F)$  is the union of the cylinders generated by each finite trace in  $F$ . Since each cylinder  $\mathbb{C}(w)$  is measurable by definition,  $\mathbb{C}(F)$  is also measurable, being a countable union of measurable sets.



**Fig. 1.** A labelled Markov chain with three states, the initial state being  $s_1$ . The symbol and number above each transition indicates which action is taken and with what probability.

*Example 1.* Assume that  $\text{Act} = \{a, b, c\}$ . Consider a monitor whose accepting set is

$$F_{acc} = \{w \in \text{Act}^* \mid \exists w'. w = aaw' \text{ or } w = cw'\},$$

and let  $M = (S, s_1, \Delta)$  be the labelled Markov chain describing the system depicted in Figure 1. In order to calculate the probability of the monitor accepting when monitoring this system, we first note that  $\mathbb{C}(F_{acc}) = \mathbb{C}(aa) \cup \mathbb{C}(c)$ . Since these are disjoint sets, we can calculate the probability as

$$\begin{aligned} \mathbb{P}_M^{s_1}(\mathbb{C}(F_{acc})) &= \mathbb{P}_M^{s_1}(\mathbb{C}(aa)) + \mathbb{P}_M^{s_1}(\mathbb{C}(c)) = (0.2 \cdot \mathbb{P}_M^{s_2}(\mathbb{C}(a)) + 0.2 \cdot \mathbb{P}_M^{s_3}(\mathbb{C}(a))) + 0.6 \\ &= (0.2 \cdot 0.5 + 0.2 \cdot 0.3) + 0.6 = 0.76. \end{aligned}$$

Properties of systems will be described in the linear-time  $\mu$ -calculus [4,38], whose formulas are given by the following grammar

$$\varphi, \psi ::= \mathbf{tt} \mid \mathbf{ff} \mid X \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid [a]\varphi \mid \langle a \rangle \varphi \mid \mu X. \varphi \mid \nu X. \varphi,$$

where  $X$  comes from a countable set  $L_{\text{VAR}}$  of logical variables and  $a \in \text{Act}$ . Formulas are interpreted over infinite traces using an interpretation  $\rho : L_{\text{VAR}} \rightarrow 2^{\text{Act}^\omega}$  for variables. The semantics is standard; we present here only the cases dealing with the modal and the fixed-point operators.

$$\begin{aligned} \llbracket [a]\varphi \rrbracket_\rho &= \{\pi \in \text{Act}^\omega \mid \pi^1 \in \llbracket \varphi \rrbracket_\rho \text{ whenever } \pi\langle 1 \rangle = a\} \\ \llbracket \langle a \rangle \varphi \rrbracket_\rho &= \{\pi \in \text{Act}^\omega \mid \pi\langle 1 \rangle = a \text{ and } \pi^1 \in \llbracket \varphi \rrbracket_\rho\} \\ \llbracket \mu X. \varphi \rrbracket_\rho &= \bigcap \{S \subseteq \text{Act}^\omega \mid \llbracket \varphi \rrbracket_{\rho[X \mapsto S]} \subseteq S\} \\ \llbracket \nu X. \varphi \rrbracket_\rho &= \bigcup \{S \subseteq \text{Act}^\omega \mid \llbracket \varphi \rrbracket_{\rho[X \mapsto S]} \supseteq S\} \end{aligned}$$

Henceforth, we only consider closed formulas, for which we may omit the subscript and simply write  $\llbracket \varphi \rrbracket$ . Since the logic is semantically closed under complement, we define negation as complement, meaning that  $\llbracket \neg \varphi \rrbracket = \text{Act}^\omega \setminus \llbracket \varphi \rrbracket$ . We next prove that each formula denotes a measurable property over infinite traces.

**Lemma 1.** *For each  $\varphi$ , the property  $\llbracket\varphi\rrbracket$  is measurable.*

*Proof.* The linear-time  $\mu$ -calculus and Büchi automata are equivalent [16], and [37, Proposition 2.3] states that the set of traces recognisable by a given Büchi automaton is measurable. Therefore,  $\llbracket\varphi\rrbracket$  is measurable.  $\square$

Lemma 1 means that the probability  $\mathbb{P}_M^s(\llbracket\varphi\rrbracket)$  of a property is well-defined.

*Example 2.* The property  $\varphi = [a]\langle a \rangle \mathbf{tt} \wedge [b]\mathbf{ff}$  states that a trace cannot start with  $b$ , and whenever it starts with  $a$ , it must be followed by another  $a$ . Assume that  $\mathbf{Act} = \{a, b, c\}$ . The probability that  $M = (S, s_1, \Delta)$ , from Figure 1, does *not* satisfy  $\varphi$  is

$$\begin{aligned} \mathbb{P}_M^{s_1}(\llbracket\neg\varphi\rrbracket) &= \mathbb{P}_M^{s_1}(\mathbb{C}(b) \cup \mathbb{C}(ab) \cup \mathbb{C}(ac)) \\ &= \mathbb{P}_M^{s_1}(\mathbb{C}(b)) + \mathbb{P}_M^{s_1}(\mathbb{C}(ab)) + \mathbb{P}_M^{s_1}(\mathbb{C}(ac)) \\ &= 0 + (0.2 \cdot \mathbb{P}_M^{s_2}(\mathbb{C}(b)) + 0.2 \cdot \mathbb{P}_M^{s_3}(\mathbb{C}(b))) + (0.2 \cdot \mathbb{P}_M^{s_2}(\mathbb{C}(c)) + 0.2 \cdot \mathbb{P}_M^{s_3}(\mathbb{C}(c))) \\ &= 0 + (0.2 \cdot 0.5 + 0.2 \cdot 0) + (0.2 \cdot 0 + 0.2 \cdot 0.7) = 0.24. \end{aligned}$$

It follows that  $\mathbb{P}_M^{s_1}(\llbracket\varphi\rrbracket) = 0.76$ , which is the ‘acceptance probability’ of a monitor of the type we considered in Example 1. In the subsequent section, we will explore the precise connections between monitors and properties in the setting we study in this paper.

### 3.1 Soundness, Completeness, and Monitorability

In the non-probabilistic setting [4], a monitor is sound with respect to some property of interest if any trace accepted by the monitor also satisfies the property, and any trace rejected by the monitor does not satisfy the property. In other words, soundness means that the monitor is an *underapproximation* of the property.

**Definition 5 (Soundness).** *A monitor  $m = (F_{acc}, F_{rej})$  is sound for a formula  $\varphi$  if  $\mathbb{C}(F_{acc}) \subseteq \llbracket\varphi\rrbracket$  and  $\mathbb{C}(F_{rej}) \subseteq \llbracket\neg\varphi\rrbracket$ .*

Dually, completeness requires the monitor to *overapproximate* the property being monitored: if a trace satisfies the property, the monitor must accept that trace, and if a trace violates the property, the monitor should reject the trace.

**Definition 6 (Completeness).** *A monitor  $m = (F_{acc}, F_{rej})$  is*

- satisfaction complete for a formula  $\varphi$ , if  $\llbracket\varphi\rrbracket \subseteq \mathbb{C}(F_{acc})$ ,
- violation complete for  $\varphi$  if  $\llbracket\neg\varphi\rrbracket \subseteq \mathbb{C}(F_{rej})$ , and
- complete for  $\varphi$  if it is both satisfaction and violation complete for  $\varphi$ .

Together, soundness and completeness require a monitor to fully agree with the property being monitored, i.e.  $\mathbb{C}(F_{acc}) = \llbracket\varphi\rrbracket$  and  $\mathbb{C}(F_{rej}) = \llbracket\neg\varphi\rrbracket$ . Completeness is, in general, a strong condition for a monitor, which often cannot be met [4,6]. In fact, we observe that completeness implies soundness.

**Lemma 2.** *Every complete monitor for a closed formula  $\varphi$  is also sound for  $\varphi$ .*

*Proof.* Assume that  $m = (F_{acc}, F_{rej})$  is complete for  $\varphi$ . To prove that  $m$  is also sound, let  $\pi \in \mathbb{C}(F_{acc})$ . This means that there is some  $w \in F_{acc}$  such that  $\pi \in \mathbb{C}(w)$ . If  $\pi \in \llbracket \neg\varphi \rrbracket$ , then due to completeness,  $\pi \in \mathbb{C}(F_{rej})$ . Again, this means that there is some  $w' \in F_{rej}$  such that  $\pi \in \mathbb{C}(w')$ . Since  $w$  and  $w'$  are both prefixes of  $\pi$ , either of them is contained in  $F_{acc} \cap F_{rej}$ , which is a contradiction, because  $F_{acc} \cap F_{rej} = \emptyset$ . Therefore,  $\pi \in \llbracket \varphi \rrbracket$ , because  $\llbracket \varphi \rrbracket \cup \llbracket \neg\varphi \rrbracket = \text{Act}^\omega$ . The case for  $\pi \in \mathbb{C}(F_{rej})$  is analogous.  $\square$

Lemma 2 only applies to complete monitors, and, in general, neither to satisfaction-complete nor to violation-complete monitors. For example, the monitor with  $F_{rej} = \text{Act}^*$  is violation complete for  $[a][b]\text{ff}$ , but is not sound. We are interested in the properties that have sound, and complete, satisfaction- or violation-complete monitors.

**Definition 7 (Monitorability).** *A formula  $\varphi$  is*

- satisfaction monitorable *if there exists a monitor that is sound and satisfaction complete for  $\varphi$ ,*
- violation monitorable *if there exists a monitor that is sound and violation complete for  $\varphi$ , and*
- fully monitorable *if there exists a monitor that is (sound and) complete for  $\varphi$ .*

Several properties are not fully monitorable, and therefore we often aim to show that they are either satisfaction or violation monitorable, and to synthesise corresponding monitors [4,26].

*Example 3.* For  $\text{Act} = \{a, b\}$ , the formula  $\varphi = \mu X.(\langle a \rangle \text{tt} \vee [b]X)$  is true exactly for the traces that have some occurrence of  $a$ . It is not hard to see that  $\varphi$  is satisfaction monitorable, but not violation monitorable.

In the probabilistic setting, we change neither the monitors nor the properties, but we interpret them over probabilistic systems. Hence, whereas non-probabilistic soundness and completeness range over satisfaction of the property in *all* models, the probabilistic version will range over the probability of the property in *all probabilistic models*. In order to extend the notions of soundness and completeness to the probabilistic setting, we impose two criteria: (1) the extension should be conservative, so that if  $m$  is sound and complete for  $\varphi$ , it is also probabilistically sound and complete for  $\varphi$ ; (2) the extension should preserve the idea of soundness being an underapproximation and completeness being an overapproximation, but in a probabilistic setting.

**Definition 8 (Probabilistic soundness).** *A monitor  $m = (F_{acc}, F_{rej})$  is probabilistically sound for  $\varphi$  if  $\mathbb{P}_M^{s*}(\mathbb{C}(F_{acc})) \leq \mathbb{P}_M^{s*}(\llbracket \varphi \rrbracket)$  and  $\mathbb{P}_M^{s*}(\mathbb{C}(F_{rej})) \leq \mathbb{P}_M^{s*}(\llbracket \neg\varphi \rrbracket)$  for all labelled Markov chains  $M = (S, s_*, \Delta)$ .*

Definition 8 fulfills criterion (1), since the monotonicity property of probability measures, which states that if  $A \subseteq B$ , then  $\mathbb{P}(A) \leq \mathbb{P}(B)$ , gives us that if  $\mathbb{C}(F_{acc}) \subseteq \llbracket \varphi \rrbracket$ , then  $\mathbb{P}_M^{s_*}(\mathbb{C}(F_{acc})) \leq \mathbb{P}_M^{s_*}(\llbracket \varphi \rrbracket)$ , and likewise for rejection. It also fulfills criterion (2), since probabilistic soundness ensures that the probability of the monitor accepting is an underapproximation of the probability of the property being satisfied, and likewise for rejection.

*Example 4.* Assume  $\text{Act} = \{a, b, c\}$ . Recall the formula  $\varphi = [a]\langle a \rangle \text{tt} \wedge [b]\text{ff}$  we considered in Example 2. Let

$$F_{acc} = \{w \in \text{Act}^* \mid \exists w'. w = aaw' \text{ or } w = cw'\} \quad \text{and} \\ F_{rej} = \emptyset.$$

For any  $M = (S, s_*, \Delta)$ , Examples 1–2 tell us that

$$\mathbb{P}_M^{s_*}(\mathbb{C}(F_{acc})) = \mathbb{P}_M^{s_*}(\{\pi \in \text{Act}^\omega \mid (\pi\langle 1 \rangle = a = \pi\langle 2 \rangle) \text{ or } (\pi\langle 1 \rangle = c)\}) = \mathbb{P}(\llbracket \varphi \rrbracket).$$

Moreover,  $0 = \mathbb{P}_M^{s_*}(\emptyset) = \mathbb{P}_M^{s_*}(F_{rej}) \leq \mathbb{P}_M^{s_*}(\llbracket \neg\varphi \rrbracket)$ , so  $m = (F_{acc}, F_{rej})$  is sound for  $\varphi$ .

**Definition 9 (Probabilistic completeness).** *A monitor  $m = (F_{acc}, F_{rej})$  is*

– *probabilistically satisfaction complete for a formula  $\varphi$  if*

$$\mathbb{P}_M^{s_*}(\mathbb{C}(F_{acc})) \geq \mathbb{P}_M^{s_*}(\llbracket \varphi \rrbracket) \quad \text{for all labelled Markov chains } M = (S, s_*, \Delta),$$

– *probabilistically violation complete for a formula  $\varphi$  if*

$$\mathbb{P}_M^{s_*}(\mathbb{C}(F_{rej})) \geq \mathbb{P}_M^{s_*}(\llbracket \neg\varphi \rrbracket) \quad \text{for all labelled Markov chains } M = (S, s_*, \Delta),$$

– *probabilistically complete for a formula  $\varphi$  if it is both probabilistically satisfaction and violation complete for the given formula.*

This definition also fulfills both of the stated criteria. Criterion (1) is satisfied for the same reason as for probabilistic soundness, and criterion (2) is satisfied because the probability that the monitor accepts is an overapproximation of the probability that the property is satisfied, and likewise for rejection.

*Remark 2.* At first glance, Definitions 8 and 9 for probabilistic soundness and completeness may seem arbitrary. Indeed, the comparison of the probabilities of two events does not in general indicate any causal relation between them. Note, however, that if the goal is to estimate the probability that a labelled Markov chain generates a trace that satisfies or violates a property, Definitions 8 and 9 describe exactly the appropriate notions of under- and over-approximation. Furthermore, as Theorem 1 demonstrates later in this section, in this case, due to the quantification over all labelled Markov chains, our probability requirements *do imply* a causal relation between the monitor verdict and the satisfaction of the formula.



*Example 5.* Assume  $\text{Act} = \{a, b, c\}$  and recall  $\varphi$  from Example 4 with

$$\begin{aligned} F_{acc} &= \{w \in \text{Act}^* \mid \exists w'. w = aaw' \text{ or } w = cw'\}, \text{ and} \\ F_{rej} &= \{w \in \text{Act}^* \mid \exists w'. w = bw' \text{ or } w = abw' \text{ or } w = acw'\}. \end{aligned}$$

Then, for any system described by a labelled Markov chain  $M = (S, s_*, \Delta)$ , we get

$$\begin{aligned} \mathbb{P}_M^{s_*}(\mathbb{C}(F_{acc})) &= \mathbb{P}_M^{s_*}(\{\pi \in \text{Act}^\omega \mid (\pi(1) = a = \pi(2)) \text{ or } (\pi(1) = c)\}) = \mathbb{P}_M^{s_*}(\llbracket \varphi \rrbracket), \text{ and} \\ \mathbb{P}_M^{s_*}(\mathbb{C}(F_{rej})) &= \mathbb{P}_M^{s_*}(\{\pi \in \text{Act}^\omega \mid (\pi(1) = b) \text{ or } (\pi(1) = a \text{ and } (\pi(2) = b \text{ or } \pi(2) = c))\}) \\ &= \mathbb{P}_M^{s_*}(\{\pi \in \text{Act}^\omega \mid (\pi(1) \neq a \text{ or } \pi(2) \neq a) \text{ and } (\pi(1) \neq c)\}) = \mathbb{P}_M^{s_*}(\llbracket \neg\varphi \rrbracket), \end{aligned}$$

so  $m = (F_{acc}, F_{rej})$  is both probabilistically sound and complete for  $\varphi$ .

Soundness and completeness together would then imply  $\mathbb{P}_M^{s_*}(\mathbb{C}(F_{acc})) = \mathbb{P}_M^{s_*}(\llbracket \varphi \rrbracket)$  and  $\mathbb{P}_M^{s_*}(\mathbb{C}(F_{rej})) = \mathbb{P}_M^{s_*}(\llbracket \neg\varphi \rrbracket)$  for all labelled Markov chains  $M = (S, s_*, \Delta)$ . This describes the probabilistic monitorability of a formula.

**Definition 10 (Probabilistic monitorability).** *A formula  $\varphi$  is*

- *probabilistically satisfaction monitorable if there exists a monitor  $m$  which is probabilistically sound and probabilistically satisfaction complete for  $\varphi$ ,*
- *probabilistically violation monitorable if there exists a monitor  $m$  which is probabilistically sound and probabilistically violation complete for  $\varphi$ , and*
- *probabilistically fully monitorable if there exists a monitor  $m$  which is probabilistically sound and probabilistically complete for  $\varphi$ .*

It is interesting to consider the connections between the probabilistic and non-probabilistic version of soundness and completeness. Because probabilistic soundness and completeness are conservative extensions of their non-probabilistic counterparts, if  $m$  monitors soundly for  $\varphi$  in the non-probabilistic setting, then  $m$  also monitors soundly for  $\varphi$  in the probabilistic setting, and likewise for the various flavours of completeness. Surprisingly, it turns out that the reverse implication also holds.

**Theorem 1.** *For every monitor  $m$  and formula  $\varphi$ ,*

- *$m$  is sound for  $\varphi$  if and only if  $m$  is probabilistically sound for  $\varphi$ ;*
- *$m$  is satisfaction complete for  $\varphi$  if and only if  $m$  is probabilistically satisfaction complete for  $\varphi$ ;*
- *$m$  is violation complete for  $\varphi$  if and only if  $m$  is probabilistically violation complete for  $\varphi$ ; and*
- *$m$  is complete for  $\varphi$  if and only if  $m$  is probabilistically complete for  $\varphi$ .*

*Proof.* Soundness and completeness imply their probabilistic counterparts by monotonicity of probability measures. For the other direction, we only present the proof for the case of soundness, as all the others are similar—and the case of completeness is a consequence of the other ones. We prove the contrapositive, so assume that  $m = (F_{acc}, F_{rej})$  is not sound for  $\varphi$ . Assume, without loss of

generality, that  $\mathbb{C}(F_{acc}) \not\subseteq \llbracket \varphi \rrbracket$ . This means that there exists a trace  $\pi \in \mathbb{C}(F_{acc})$  such that  $\pi \notin \llbracket \varphi \rrbracket$ . It is now immediate to exhibit a labelled Markov chain  $M$  such that  $\mathbb{P}_M^{s_*}(\mathbb{C}(F_{acc})) = 1$  but  $\mathbb{P}_M^{s_*}(\llbracket \varphi \rrbracket) = 0$  by constructing  $M$  such that it generates only the trace  $\pi$ . Then  $1 = \mathbb{P}_M^{s_*}(\mathbb{C}(F_{acc})) \not\leq \mathbb{P}_M^{s_*}(\llbracket \varphi \rrbracket) = 0$ , so  $m$  is not probabilistically sound for  $\varphi$ .  $\square$

A corollary of Theorem 1 is that the probabilistically fully monitorable formulas are exactly those that are also non-probabilistically fully monitorable. In [4, Theorem 4.8] it was shown that the Hennessy-Milner logic [28], which is the recursion-free fragment of the  $\mu$ -calculus, expresses exactly the fully monitorable properties—without assuming that these properties are expressed in the  $\mu$ -calculus.

**Corollary 1.** *A property is probabilistically fully monitorable if and only if it can be expressed with a formula generated by the following grammar:*

$$\varphi, \psi ::= \mathbf{tt} \mid \mathbf{ff} \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid [a]\varphi \mid \langle a \rangle \varphi.$$

The satisfaction- and violation-monitorable formulas were also characterized in [4] with maximally expressive fragments. However, in contrast to the case of full monitorability, there are violation-monitorable properties that cannot be expressed in the  $\mu$ -calculus, as long as one assumes sufficiently powerful monitors.

**Corollary 2.** *A  $\mu$ -calculus formula is*

- *probabilistically satisfaction monitorable, if and only if is equivalent to a formula of the fragment*

$$\varphi, \psi ::= \mathbf{tt} \mid \mathbf{ff} \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid [a]\varphi \mid \langle a \rangle \varphi \mid \nu X.\varphi, \quad \text{and}$$

- *probabilistically violation monitorable, if and only if is equivalent to a formula of the fragment*

$$\varphi, \psi ::= \mathbf{tt} \mid \mathbf{ff} \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid [a]\varphi \mid \langle a \rangle \varphi \mid \mu X.\varphi.$$

### 3.2 Other Monitor Requirements

Theorem 1 may seem to imply that Definitions 8 and 9 are very restrictive. However, the theorem holds for other interpretations of soundness and completeness in a probabilistic setting. Fix two parameters  $c, d > 0$ .

**Definition 11 (Probabilistic soundness and completeness with a margin of error).** *A monitor  $m = (F_{acc}, F_{rej})$  is probabilistically sound for  $\varphi$  with margin of error  $c$  if  $\mathbb{P}_M^{s_*}(\mathbb{C}(F_{acc})) \leq c \cdot \mathbb{P}_M^{s_*}(\llbracket \varphi \rrbracket)$  and  $\mathbb{P}_M^{s_*}(\mathbb{C}(F_{rej})) \leq c \cdot \mathbb{P}_M^{s_*}(\llbracket \neg \varphi \rrbracket)$  for all labelled Markov chains  $M = (S, s_*, \Delta)$ . Likewise,  $m$  is*

- *probabilistically satisfaction complete for  $\varphi$  with margin of error  $d$  if*

$$\mathbb{P}_M^{s_*}(\mathbb{C}(F_{acc})) \geq d \cdot \mathbb{P}_M^{s_*}(\llbracket \varphi \rrbracket) \quad \text{for all labelled Markov chains } M = (S, s_*, \Delta),$$

- probabilistically violation complete for  $\varphi$  with margin of error  $d$  if
 
$$\mathbb{P}_M^{s_*}(\mathbb{C}(F_{rej})) \geq d \cdot \mathbb{P}_M^{s_*}(\llbracket \neg\varphi \rrbracket) \quad \text{for all labelled Markov chains } M = (S, s_*, \Delta),$$
- probabilistically complete for  $\varphi$  with margin of error  $d$  if it is both satisfaction and violation complete for  $\varphi$  with margin of error  $d$ .

The two parameters, when  $c > 1$  and  $d < 1$ , allow the monitor to occasionally give more or fewer verdicts than it should, but always within a set margin of error. Another candidate for soundness and completeness, parameterised with respect to  $c$  and  $d$ , is conditional soundness and completeness.

**Definition 12 (Conditional soundness and completeness).** *A monitor  $m = (F_{acc}, F_{rej})$  is conditionally sound for  $\varphi$  with margin of error  $c$  if it holds that  $\mathbb{P}_M^{s_*}(\llbracket \varphi \rrbracket \mid \mathbb{C}(F_{acc})) \geq c$  and  $\mathbb{P}_M^{s_*}(\llbracket \neg\varphi \rrbracket \mid \mathbb{C}(F_{rej})) \geq c$  for all labelled Markov chains  $M = (S, s_*, \Delta)$ . Likewise,  $m$  is*

- conditionally satisfaction complete with margin of error  $d$  for a formula  $\varphi$  if
 
$$\mathbb{P}_M^{s_*}(\mathbb{C}(F_{acc}) \mid \llbracket \varphi \rrbracket) \geq d \quad \text{for all labelled Markov chains } M = (S, s_*, \Delta),$$
- conditionally violation complete with margin of error  $d$  for a formula  $\varphi$  if
 
$$\mathbb{P}_M^{s_*}(\mathbb{C}(F_{rej}) \mid \llbracket \neg\varphi \rrbracket) \geq d \quad \text{for all labelled Markov chains } M = (S, s_*, \Delta),$$
- conditionally complete with margin of error  $d$  for a formula  $\varphi$  if it is both conditionally satisfaction and violation complete for  $\varphi$ .

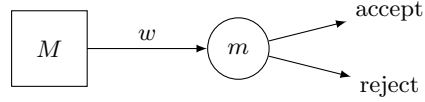
We observe that for these variations of probabilistic soundness and completeness, the arguments used in the proof of Theorem 1 can also be applied.

**Theorem 2.** *All the variants of soundness and completeness are equivalent. This means that Definitions 5, 8, 11, and 12 are equivalent, and that Definitions 6, 9, 11, and 12 are also equivalent.*

*Proof.* The first two items, both for soundness and completeness are equivalent, by Theorem 1. To show that each other item is equivalent to the first, we follow the proof of Theorem 1.  $\square$

Theorem 2 allows us to treat monitorability uniformly for all the approaches described by Definitions 5, 6, 8, 9 and 11 to 12. For instance, the monitor synthesis defined in [4,26] and implemented in [1,8,9] applies directly to the probabilistic setting (with margins of error).

*Remark 3.* One conclusion from the results of this section is that using the qualitative monitoring framework studied in [4,6] to monitor for properties of labelled Markov chains allows one to monitor the same collections of properties that are monitorable for labelled transition systems. Therefore, to use the probabilistic setting to monitor for more properties than in the qualitative setting, one must increase the power of monitors, e.g., by using randomisation (as done in, e.g., [34]), or by giving them some information about the monitored system.



**Fig. 2.** A setup for estimating probabilities.  $M$  is a probabilistic system being monitored by the monitor  $m$ , which reads the trace  $w$  emitted by  $S$  to provide a verdict.

## 4 An Application: Estimating Probabilities

The theory we have described in Section 3 allows us to estimate the probabilities of properties over infinite traces, even if the system itself is a black box. To see this, consider the setup depicted in Figure 2. Here we have a probabilistic system  $M = (S, s_*, \Delta)$ , of which we do not know the internal workings, and hence should be viewed as a black box. Using the monitor synthesis from [4], we can generate a monitor  $m = (F_{acc}, F_{rej})$  which is both sound and complete for a monitorable property  $\varphi$ , whose probability in  $M$  we are interested in estimating. As  $m$  observes the behaviour of  $M$  given by a sequence of outputs  $w = a_1 \dots a_n$ ,  $m$  will eventually, in finite time, produce either an accept or a reject verdict. This is guaranteed because  $m$  is both sound and complete.

In a setting where a system is executed repeatedly (e.g., once every morning), we can estimate the probability  $\mathbb{P}_M^{s_*}(\llbracket \varphi \rrbracket)$ . Concretely, every time the system  $M$  is run (with passive monitor  $m$ ), the verdict reached for an exhibited trace is recorded (here we assume that we can reset the system to its initial state, as is done in, for instance, [19]). After some number of iterations, say  $n$  iterations, we will have observed some number  $n_{acc}$  of accept verdicts and some number  $n_{rej}$  of reject verdicts. We can then estimate the probabilities  $\mathbb{P}_M^{s_*}(\mathbb{C}(F_{acc}))$  and  $\mathbb{P}_M^{s_*}(\mathbb{C}(F_{rej}))$  by  $\frac{n_{acc}}{n}$  and  $\frac{n_{rej}}{n}$ , respectively. By Theorem 1, the probability of satisfying the property is equal to the probability of the monitor accepting, and likewise for not satisfying the property and rejecting. This means that  $\frac{n_{acc}}{n}$  and  $\frac{n_{rej}}{n}$  are also estimates of  $\mathbb{P}_M^{s_*}(\llbracket \varphi \rrbracket)$  and  $\mathbb{P}_M^{s_*}(\llbracket \neg \varphi \rrbracket)$ , respectively, so we can use these to estimate the probability that  $\varphi$  is satisfied in  $M$ .

This approach to estimating only works for the monitorable fragment of the logic (see Corollary 1). However, even for non-monitorable properties, we can use the approach to give estimates of the probability in terms of lower and upper bounds. For some non-monitorable property  $\varphi$ , one could construct a sound monitor  $m_1 = (F_{acc}^1, F_{rej}^1)$  and a complete monitor  $m_2 = (F_{acc}^2, F_{rej}^2)$ . Then  $\mathbb{P}_M^{s_*}(\mathbb{C}(F_{acc}^1)) \leq \mathbb{P}_M^{s_*}(\llbracket \varphi \rrbracket) \leq \mathbb{P}_M^{s_*}(\mathbb{C}(F_{acc}^2))$ , and similarly for  $\llbracket \neg \varphi \rrbracket$  and the rejection parts of the monitors. Hence  $m_1$  gives a lower bound on the probability of  $\varphi$ , and  $m_2$  gives an upper bound. Now we can use the approach from before to estimate the probabilities of  $m_1$  accepting and rejecting and of  $m_2$  accepting and rejecting, thus giving us estimates on lower and upper bounds on the probability that  $M$  satisfies  $\varphi$ . The downside is that in this case we have no guarantee that  $m_1$  will give a verdict in finite time.

## 5 Conclusions

In this article, we have applied the monitoring framework studied in [4,6] to verify properties of labelled Markov chains at runtime. We have considered a variety of notions of probabilistic monitorability that arise naturally in this setting and proved that they are all equivalent to the corresponding ‘purely qualitative’ notions in the above-mentioned references. The main conclusion of our study is, therefore, that the power of monitors observing systems with probabilistic dynamics must be increased if one wants to monitor more properties of such systems than in the qualitative setting from [4,6].

*Related work* Runtime monitoring for probabilistic systems has been an active research area for some time and is currently the subject of considerable activity—see, for instance, the papers [14,19,27,31,34,35] to name but a few. To our mind, the work that is closest to our study is the one presented by Sistla and Srinivas in [34]. In that article, the authors investigate runtime monitoring of qualitative properties for systems modelled as Hidden Markov Chains, namely Markov chains that have outputs associated with their states. (The labelled Markov chains we consider can be viewed as an action-based counterpart of Hidden Markov Chains.) In the above-mentioned paper, Sistla and Srinivas study both deterministic and probabilistic monitors. They give deterministic monitors that use counters to monitor properties that can be expressed as deterministic Büchi automata with a desired accuracy. However, the monitoring algorithm needs to know the Hidden Markov Chain defining the system and is therefore not black box. This deficiency is remedied in *op. cit.* by means of probabilistic monitors. A probabilistic monitor for a property is a randomised algorithm that rejects with probability one every system computation that does not satisfy the property. On the other hand, a strong probabilistic monitor for a property is a probabilistic monitor that accepts every system computation that satisfies the property with some positive probability. Sistla and Srinivas prove an expressive completeness result for strong probabilistic monitors that characterises the class of properties that have such monitors, namely the class of properties whose complements can be recognised by (infinite-state) Büchi automata. The above-mentioned paper also gives some techniques that can be used to combine deterministic and probabilistic approaches to monitoring Hidden Markov Chains.

Junges *et al.* study runtime monitors for systems modelled as Markov Decision Processes in [31]. These are systems that are partially observable and, unlike (Hidden) Markov Chains, exhibit both nondeterministic and probabilistic dynamics. The observation function for Markov Decision Processes, which describes the observations that can be made at each system state, is also probabilistic. Moreover, each system state has an associated non-negative real number that describes how risky that state is. In the above-mentioned paper, Junges *et al.* study the following monitoring problem:

Decide whether, for any possible scheduler used to resolve the nondeterminism in the observed system, the ‘weighted trace risk’ of a given trace of observations is larger than some given threshold.

Two algorithms are given for solving the above problem and are evaluated on a range of benchmark applications. One is based on using forward filtering and employs vertices of a convex hull to represent a possibly exponential set of distributions. The other is based on model checking and runs in polynomial time. Both algorithms, however, require knowledge of the observed Markov Decision Process and therefore do not treat the system as a black box.

The runtime monitoring problem studied in [31] is conceptually related to the predictive monitoring problem for hybrid systems [15], namely the problem of predicting, at runtime, whether the system can reach some unsafe state in the future within a given time bound. Unlike other approaches to runtime verification, predictive monitoring aims at detecting potentially bad system executions *before* a violation occurs. In [33], Phan *et al.* have proposed a method they call Neural State Classification to train deep neural networks to classify observed executions of hybrid systems as unsafe if they can be extended to reach an unsafe state. Bortolussi *et al.* build on that work in [13] to develop a framework called Neural Predictive Monitoring that provides efficiency, accuracy, and statistical guarantees on the prediction error, which were not provided by the methods from [33]. It would be very interesting to apply some of the methods in those two papers to the setting described in Section 4.

Esparza *et al.* study the enforcement of  $\omega$ -regular properties over labelled Markov chains by means of universal restarting strategies in [19]. The key requirement on the restarting strategy is that, for each Markov chain, the number of restarts is finite and the execution of the Markov chain after the last restart satisfies the desired property, with probability 1. Two algorithms are given for this task, a *cautious* and a more efficient *bold* one, and are evaluated experimentally using models from the PRISM Benchmark Suite [32]. In the work by Esparza *et al.*, both the Markov chain and its set of states are unknown to the algorithms. However, the authors assume that the algorithms can detect whether the current state has been observed previously, but cannot pinpoint which state of the observed chain it is.

In [14], Bartolo *et al.* investigate monitoring for probabilistic, automata-based specifications expressed as (binary) session types where choice points are augmented with a probability distribution. Their monitors employ statistical inference techniques to detect (partial) executions that deviate considerably from the prescribed probabilities as they pass repeatedly through these choice points. Since detections in this work are interpreted with respect to a pre-specified confidence level, it is worth investigating whether Definition 12 can be used to assess the soundness and completeness of the approach; the margin of error  $c$  might be used to accommodate errors induced by said confidence level.

*Future work* The work presented in this paper paves the way to several interesting avenues for future research. First of all, following [34], it is natural to study the expressive power of randomised versions of the monitors considered in [4,6] and to characterise precisely the collection of properties they can monitor and with what guarantees. It would also be interesting to investigate how repeated monitor observations of some probabilistic system can be used to learn models of

it and how well those models approximate the system under observation. Moreover, building on the work by Esparza *et al.* on the enforcement of  $\omega$ -regular properties over labelled Markov chains in [19], one might attempt to develop a general theory of runtime enforcement for probabilistic systems akin to the one presented in, for instance, [20]. Last, but by no means least, it would be fitting to accompany the above mentioned theoretical work with tool development, and applications to benchmarks from the literature and to case studies. We would not be surprised if Thomas A. Henzinger and his group at IST Austria did all this work and much more. VAMOS Tom!

*Acknowledgments* We thank the anonymous reviewers for their comments, and Thomas A. Henzinger for the inspiration his scientific work has provided over many years and for his leadership within the scientific community.

## References

1. Luca Aceto, Antonis Achilleos, Duncan Paul Attard, Léo Exibard, Adrian Francalanza, and Anna Ingólfssdóttir. A Monitoring Tool for Linear-Time  $\mu$ HML. In Maurice ter Beek and Marjan Sirjani, editors, *Proceedings of Coordination Models and Languages — 24th IFIP WG 6.1 International Conference, COORDINATION 2022*, Lecture Notes in Computer Science. Springer, 2022. (to appear).
2. Luca Aceto, Antonis Achilleos, Adrian Francalanza, and Anna Ingólfssdóttir. Monitoring for silent actions. In Satya V. Lokam and R. Ramanujam, editors, *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India*, volume 93 of *LIPICs*, pages 7:1–7:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
3. Luca Aceto, Antonis Achilleos, Adrian Francalanza, and Anna Ingólfssdóttir. A framework for parameterized monitorability. In Christel Baier and Ugo Dal Lago, editors, *Foundations of Software Science and Computation Structures - 21st International Conference, FOSSACS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*, volume 10803 of *Lecture Notes in Computer Science*, pages 203–220. Springer, 2018.
4. Luca Aceto, Antonis Achilleos, Adrian Francalanza, Anna Ingólfssdóttir, and Karoliina Lehtinen. Adventures in monitorability: from branching to linear time and back again. *Proc. ACM Program. Lang.*, 3(POPL):52:1–52:29, 2019.
5. Luca Aceto, Antonis Achilleos, Adrian Francalanza, Anna Ingólfssdóttir, and Karoliina Lehtinen. The best a monitor can do. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic, CSL 2021, January 25-28, 2021, Ljubljana, Slovenia (Virtual Conference)*, volume 183 of *LIPICs*, pages 7:1–7:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
6. Luca Aceto, Antonis Achilleos, Adrian Francalanza, Anna Ingólfssdóttir, and Karoliina Lehtinen. An operational guide to monitorability with applications to regular properties. *Software and Systems Modeling*, 20(2):335–361, 2021.
7. Robert B. Ash and Catherine A. Doléans-Dade. *Probability & Measure Theory*. Harcourt/Academic Press, 2nd edition, 1999.

8. Duncan Paul Attard, Luca Aceto, Antonis Achilleos, Adrian Francalanza, Anna Ingólfssdóttir, and Karoliina Lehtinen. Better late than never or: Verifying asynchronous components at runtime. In Kirstin Peters and Tim A. C. Willemse, editors, *Formal Techniques for Distributed Objects, Components, and Systems - 41st IFIP WG 6.1 International Conference, FORTE 2021, Held as Part of the 16th International Federated Conference on Distributed Computing Techniques, DisCoTec 2021, Valletta, Malta, June 14-18, 2021, Proceedings*, volume 12719 of *Lecture Notes in Computer Science*, pages 207–225. Springer, 2021.
9. Duncan Paul Attard and Adrian Francalanza. Trace partitioning and local monitoring for asynchronous components. In Alessandro Cimatti and Marjan Sirjani, editors, *Software Engineering and Formal Methods - 15th International Conference, SEFM 2017, Trento, Italy, September 4-8, 2017, Proceedings*, volume 10469 of *Lecture Notes in Computer Science*, pages 219–235. Springer, 2017.
10. Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.
11. Ezio Bartocci, Yliès Falcone, Adrian Francalanza, and Giles Reger. Introduction to runtime verification. In Ezio Bartocci and Yliès Falcone, editors, *Lectures on Runtime Verification - Introductory and Advanced Topics*, volume 10457 of *Lecture Notes in Computer Science*, pages 1–33. Springer, 2018.
12. Patrick Billingsley. *Probability And Measure*. Wiley-Interscience, 3rd edition, 1995.
13. Luca Bortolussi, Francesca Cairoli, Nicola Paoletti, Scott A. Smolka, and Scott D. Stoller. Neural predictive monitoring. In Bernd Finkbeiner and Leonardo Mariani, editors, *Proceedings of Runtime Verification - 19th International Conference, RV 2019*, volume 11757 of *Lecture Notes in Computer Science*, pages 129–147. Springer, 2019.
14. Christian Bartolo Burlò, Adrian Francalanza, Alceste Scalas, Catia Trubiani, and Emilio Tuosto. Towards probabilistic session-type monitoring. In Ferruccio Damiani and Ornella Dardha, editors, *Proceedings of Coordination Models and Languages — 23rd IFIP WG 6.1 International Conference, COORDINATION 2021*, volume 12717 of *Lecture Notes in Computer Science*, pages 106–120. Springer, 2021.
15. Xin Chen and Sriram Sankaranarayanan. Model predictive real-time monitoring of linear systems. In *Proceedings of the 2017 IEEE Real-Time Systems Symposium, RTSS 2017*, pages 297–306. IEEE Computer Society, 2017.
16. Mads Dam. Fixed points of Büchi automata. In R. K. Shyamasundar, editor, *Foundations of Software Technology and Theoretical Computer Science, 12th Conference, New Delhi, India, December 18-20, 1992, Proceedings*, volume 652 of *Lecture Notes in Computer Science*, pages 39–50. Springer, 1992.
17. Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Equivalence of labeled Markov chains. *International Journal of Foundations of Computer Science*, 19(3):549–563, 2008.
18. Adrián Elgyütt, Thomas Ferrère, and Thomas A. Henzinger. Monitoring temporal logic with clock variables. In David N. Jansen and Pavithra Prabhakar, editors, *Proceedings of Formal Modeling and Analysis of Timed Systems - 16th International Conference, FORMATS 2018*, volume 11022 of *Lecture Notes in Computer Science*, pages 53–70. Springer, 2018.
19. Javier Esparza, Stefan Kiefer, Jan Kretínský, and Maximilian Weininger. Enforcing  $\omega$ -regular properties in Markov chains by restarting. In Serge Haddad and Daniele Varacca, editors, *Proceedings of the 32nd International Conference on Concurrency Theory, CONCUR 2021*, volume 203 of *LIPICs*, pages 5:1–5:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.



20. Yliès Falcone, Jean-Claude Fernandez, and Laurent Mounier. What can you verify and enforce at runtime? *International Journal on Software Tools for Technology Transfer*, 14(3):349–382, 2012.
21. Thomas Ferrère, Thomas A. Henzinger, and Bernhard Kragl. Monitoring event frequencies. In Maribel Fernández and Anca Muscholl, editors, *Proceedings of the 28th EACSL Annual Conference on Computer Science Logic, CSL 2020*, volume 152 of *LIPICs*, pages 20:1–20:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
22. Thomas Ferrère, Thomas A. Henzinger, and N. Ege Saraç. A theory of register monitors. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018*, pages 394–403. ACM, 2018.
23. Adrian Francalanza. Consistently-Detecting Monitors. In Roland Meyer and Uwe Nestmann, editors, *28th International Conference on Concurrency Theory, CONCUR 2017, September 5-8, 2017, Berlin, Germany*, volume 85 of *LIPICs*, pages 8:1–8:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
24. Adrian Francalanza. A Theory of Monitors. *Inf. Comput.*, 281:104704, 2021.
25. Adrian Francalanza, Luca Aceto, Antonis Achilleos, Duncan Paul Attard, Ian Cas-sar, Dario Della Monica, and Anna Ingólfssdóttir. A foundation for runtime monitoring. In Shuvendu K. Lahiri and Giles Rege, 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, 2017.
26. Adrian Francalanza, Luca Aceto, and Anna Ingólfssdóttir. Monitorability for the Hennessy-Milner logic with recursion. *Formal Methods Syst. Des.*, 51(1):87–116, 2017.
27. Kalpana Gondi, Yogeshkumar Patel, and A. Prasad Sistla. Monitoring the full range of omega-regular properties of stochastic systems. In Neil D. Jones and Markus Müller-Olm, editors, *Proceedings of Verification, Model Checking, and Abstract Interpretation, 10th International Conference, VMCAI 2009*, volume 5403 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 2009.
28. Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, 1985.
29. Thomas A. Henzinger and N. Ege Saraç. Monitorability under assumptions. In Jyotirmoy Deshmukh and Dejan Nickovic, editors, *Proceedings of Runtime Verification - 20th International Conference, RV 2020*, volume 12399 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2020.
30. Thomas A. Henzinger and N. Ege Saraç. Quantitative and approximate monitoring. In *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021*, pages 1–14. IEEE, 2021.
31. Sebastian Junges, Hazem Torfah, and Sanjit A. Seshia. Runtime monitors for Markov decision processes. In Alexandra Silva and K. Rustan M. Leino, editors, *Proceedings of Computer Aided Verification - 33rd International Conference, CAV 2021, Part II*, volume 12760 of *Lecture Notes in Computer Science*, pages 553–576. Springer, 2021.
32. Marta Z. Kwiatkowska, Gethin Norman, and David Parker. The PRISM benchmark suite. In *Proceedings of the Ninth International Conference on Quantitative Evaluation of Systems, QEST 2012*, pages 203–204. IEEE Computer Society, 2012.
33. Dung T. Phan, Nicola Paoletti, Timothy Zhang, Radu Grosu, Scott A. Smolka, and Scott D. Stoller. Neural state classification for hybrid systems. In Shuvendu K.

- Lahiri and Chao Wang, editors, *Proceedings of Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018*, volume 11138 of *Lecture Notes in Computer Science*, pages 422–440. Springer, 2018.
34. A. Prasad Sistla and Abhigna R. Srinivas. Monitoring temporal properties of stochastic systems. In Francesco Logozzo, Doron A. Peled, and Lenore D. Zuck, editors, *Proceedings of Verification, Model Checking, and Abstract Interpretation, 9th International Conference, VMCAI 2008*, volume 4905 of *Lecture Notes in Computer Science*, pages 294–308. Springer, 2008.
  35. A. Prasad Sistla, Milos Zefran, and Yao Feng. Runtime monitoring of stochastic cyber-physical systems with hybrid state. In Sarfraz Khurshid and Koushik Sen, editors, *Proceedings of Runtime Verification — Second International Conference, RV 2011, Revised Selected Papers*, volume 7186 of *Lecture Notes in Computer Science*, pages 276–293. Springer, 2011.
  36. Terence Tao. *An Introduction to Measure Theory*. Graduate studies in mathematics. American Mathematical Society, 2013.
  37. Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 327–338. IEEE Computer Society, 1985.
  38. Moshe Y. Vardi. A temporal fixpoint calculus. In Jeanne Ferrante and P. Mager, editors, *Conference Record of the Fifteenth Annual ACM Symposium on Principles of Programming Languages, San Diego, California, USA, January 10-13, 1988*, pages 250–259. ACM Press, 1988.