Innovations in Systems and Software Engineering (2019) 15:207–233
https://doi.org/10.1007/s11334-019-00349-z

S.I. : NFM2018

# Model-based testing of stochastically timed systems

Marcus Gerhold[1] · Arnd Hartmanns[1] · Mariëlle Stoelinga[1]

Received: 20 September 2018 / Accepted: 11 June 2019 / Published online: 18 June 2019

**Abstract**
Many systems are inherently stochastic: they interact with unpredictable environments or use randomised algorithms. Classical model-based testing is insufficient for such systems: it only covers functional correctness. In this paper, we present two model-based testing frameworks that additionally cover the stochastic aspects in hard and soft real-time systems. Using the theory of Markov automata and stochastic automata for specifications, test cases, and a formal notion of conformance, they provide clean mechanisms to represent underspecification, randomisation, and stochastic timing. Markov automata provide a simple memoryless model of time, while stochastic automata support arbitrary continuous and discrete probability distributions. We cleanly define the theoretical foundations, outline practical algorithms for statistical conformance checking, and evaluate both frameworks' capabilities by testing timing aspects of the Bluetooth device discovery protocol. We highlight the trade-off of simple and efficient statistical evaluation for Markov automata versus precise and realistic modelling with stochastic automata.

**Keywords** Model-based testing · Markov automata · Stochastic automata · Ioco conformance

## 1 Introduction

Model-based testing (MBT) [50] is a technique to automatically generate, execute, and evaluate test suites on black-box *implementations under test* (IUT). The theoretical ingredients of an MBT framework are a formal *model* that specifies the desired system behaviour, often in terms of (some extension of) input–output transition systems; a notion of *conformance* that specifies when an IUT is considered a valid implementation of the model; and a precise definition of what a *test case* is. For the framework to be applicable in practice, we also need algorithms to *derive* test cases from the model, *execute* them on the IUT, and *evaluate* the results, i.e. decide conformance. They need to be sound (i.e. every implementation that fails a test case does not conform to the model), and ideally also complete (i.e. for every non-conforming implementation, there theoretically exists a failing test case). MBT is attractive due to its high degree of automation: given a model, the otherwise labour-intensive and error-prone derivation, execution and evaluation steps can be performed in a fully automatic way.

Model-based testing originally gained prominence for input–output transition systems (IOTS) using the **ioco** relation for *input–output conformance* [49]. IOTS partition the observable actions of the IUT (and thus of the model and test cases) into *inputs* (or *stimuli*) that can be provided at any time, e.g. pressing a button or receiving a network message, and *outputs* that are signals or activities that the environment can observe, e.g. delivering a product or sending a network message. IOTS include nondeterministic choices, allowing underspecification: the IUT may implement any or all of the modelled alternatives. MBT with IOTS tests for *functional* correctness: the IUT shall only exhibit behaviours allowed by the model. In the presence of nondeterminism, the IUT is allowed to use *any* deterministic or randomised policy to decide between the specified alternatives.

Stochastic behaviour and requirements are an important aspect of today's complex systems: network protocols extensively rely on randomised algorithms, cloud providers commit to service level agreements, probabilistic robotics [46] allows the automation of complex tasks via simple randomised strategies (as seen in, e.g. vacuuming and lawn

This work is supported by the 3TU.BSR project, by NWO projects BEAT and SUMBAT, and by the NWO VENI Grant No. 639.021.754.

✉ Arnd Hartmanns
a.hartmanns@utwente.nl

Marcus Gerhold
m.gerhold@utwente.nl

Mariëlle Stoelinga
m.i.a.stoelinga@utwente.nl

[1] University of Twente, Enschede, The Netherlands

mowing robots), and we see a proliferation of probabilistic programming languages [23]. Stochastic systems must satisfy stochastic requirements. Consider the example of exponential backoff in Ethernet: an adapter that, after a collision, sometimes retransmits earlier than prescribed by the standard may not impact the overall functioning of the network, but may well gain an unfair advantage in throughput at the expense of overall network performance. In the case of cloud providers, the service level agreements are inherently stochastic when guaranteeing a certain availability (i.e. *average* uptime) or a certain distribution of maximum response times for different tasks. This has given rise to extensive research in stochastic *model checking* techniques [30]. However, in practice, *testing* remains the dominant technique to evaluate and certify systems outside of a limited area of highly safety-critical applications.

In this paper, we present two MBT frameworks based on input–output Markov automata [17] (IOMA) and stochastic automata [11,12] (IOSA), which are transition systems augmented with discrete probabilistic choices and stochastic delays. Markov automata are a memoryless continuous-time model, essentially the extension of continuous-time Markov chains with nondeterminism: the time spent in any state of the automaton follows some exponential distribution. In stochastic automata, on the other hand, the progress of time is governed by clock variables whose expiration times follow general probability distributions. By using IOMA or IOSA models, we can quantitatively specify stochastic aspects of a system, in particular, w.r.t. timing. While IOMA are more suitable for the abstract specification of soft real-time systems, IOSA enable precise modelling of both hard and soft real-time systems and requirements. Since both models extend transition systems, nondeterminism is available for underspecification as usual. After introducing the models and their semantics (Sect. 3), we formally define the notions of Markovian and stochastic ioco (**mar-ioco** and **sa-ioco**, respectively), and of test cases as restrictions of IOMA and IOSA (Sect. 4). We then outline practical algorithms for conformance testing (Sect. 5). The latter combines per-trace functional verdicts as in standard ioco with a statistical evaluation that builds upon confidence interval estimation for IOMA and the Kolmogorov–Smirnov test [29] for IOSA. We finally exemplify our frameworks' capabilities and the tradeoffs between the IOMA and IOSA approaches by testing timing aspects of different implementation variants of the Bluetooth device discovery protocol (Sect. 6).

## 1.1 Related work

Our **mar-ioco** and **sa-ioco** frameworks generalise the **pioco** framework [20] for probabilistic automata (or Markov decision processes), which only supports discrete probabilistic choices and has no notion of time at all.

Early influential work on model-based testing had only deterministic time [4,31,33,34], later extended with timeouts/quiescence [5]. Probabilistic testing relations and equivalences are well studied [9,14,42]. Probabilistic bisimulation via hypothesis testing was first introduced in [35]. Our work is largely influenced by [8], which introduced a way to compare trace frequencies with collected samples. A more restricted approach is given in the work on stochastic finite state machines [28,40]: stochastic delays are specified similarly, but discrete probability distributions over target states are not included. Closely related to our testing relation for Markov automata are the studies of bisimulation relations [17], which inspired further work on weak bisimulation [15] and late-weak bisimulation [43]. By studying relations based on trace distribution semantics, rather than equivalence relations, we grant vastly more implementation freedom.

Probabilistic and non-probabilistic MBT are part of a greater ecosystem of formal methods developed to improve the correctness, dependability, and trustworthiness of various types of systems, ranging from software over cyber-physical systems to, for example, organisational processes and biological applications. Model checking [1], probabilistic model checking [30], and statistical model checking [26,54] serve to prove or disprove the conformance of a (probabilistic) model of a system to a (probabilistic) specification usually given in terms of temporal logics formulas. Notable probabilistic model checkers include PRISM [32], STORM [13], and the MCSTA tool of the MODEST TOOLSET [25], while two current examples of statistical model checkers are PLASMA LAB [36] and the MODEST TOOLSET's MODES simulator [6]. These techniques and tools are complimentary to MBT, which establishes a relation between a model (which now acts as a specification, and may earlier have been verified with model checking) and the real implementation. Notably, the MODEST TOOLSET also includes an MBT tool [24], thus providing all three techniques for probabilistic systems in one package. The "opposite" of MBT, deriving a model from an implementation using automata learning [51,53], is also gaining popularity and is especially well suited for the analysis of legacy systems [41]. Automata learning typically uses MBT internally to check whether the model learned so far is approximately equivalent to the implementation under learning.

## 1.2 Previous work

This paper provides a new integrated presentation of our previous papers on model-based testing for Markov automata [21] and stochastic automata [19]. We explain the differences and tradeoffs between the two frameworks in theory and practice. We added examples and more detailed explanations throughout the paper. Test cases for both models are now effectively IOTS (Sect. 4.2), where our previous work used probabilistic test cases, providing a clean distinction between test generation and test selection.

Specifically compared to [21], we use a more standard definition of IOMA (Definition 1) that does not rely on being input-reactive and output-generative [52]. We discuss how to implement quiescence in a Markovian setting in a way that does not affect the statistical evaluation yet minimises the testing runtime and the chance for errors of the second kind (Sect. 5.2). Finally, we study an additional protocol mutant with IOMA in the Bluetooth case study (Sect. 6).

Compared to [19], we adapted the **sa-ioco** conformance relation such that it now properly extends **ioco**. That is, where [19] relied on trace distribution inclusion of closed systems, we now utilise schedulers for open systems. As a result, **sa-ioco** is in line with **mar-ioco** and with earlier work on untimed probabilistic systems [20]. We also present full proofs for the soundness and completeness of the IOSA MBT framework (Sect. 4.4).

## 2 Preliminaries

### 2.1 Mathematical notation

$\mathbb{N}$ is $\{0, 1, \ldots\}$, the set of natural numbers. $\mathbb{R}$, $\mathbb{R}^+$, and $\mathbb{R}^+_0$ are the sets of all, all positive, and all nonnegative real numbers, respectively. We write closed intervals as $[a, b] \overset{\text{def}}{=} \{x \in \mathbb{R} \mid a \leq x \leq b\}$, open intervals as $]a, b[ \overset{\text{def}}{=} \{x \in \mathbb{R} \mid a < x < b\}$, and half-open intervals analogously as $]a, b]$ and $[a, b[$. For a given set $\Omega$, we denote its powerset by $\mathcal{P}(\Omega)$. A multiset is written as $\{\!|\ \ldots\ |\!\}$. Let the function $\mathbb{1} \in \{\mathit{true}, \mathit{false}\} \to \{0, 1\}$ be defined by $\mathbb{1}(\mathit{true}) = 1$ and $\mathbb{1}(\mathit{false}) = 0$. We write $\mathbb{1}_b$ to denote $\mathbb{1}(b)$.

We use angled brackets $\langle\cdot\rangle$ to denote tuples, and define $\Omega^* \overset{\text{def}}{=} \cup_{i\in\mathbb{N}}\Omega^i$, the set of all finite tuples or *sequences* consisting of elements from $\Omega$. Correspondingly, we write $\Omega^\omega$ for the set of all infinite sequences, $\Omega^{\leq\omega}$ for the set of all finite and infinite sequences, and $\Omega^{\leq k}$ for the set of all sequences of length at most $k$. For a sequence

$$\sigma = \omega_0 \ldots \omega_n \overset{\text{def}}{=} \langle\omega_0, \ldots, \omega_n\rangle \in \Omega^{n+1},$$

we write $\sigma.\omega_{n+1}$ for $\omega_0 \ldots \omega_n\, \omega_{n+1} \in \Omega^{n+2}$, i.e. $\sigma$ extended by $\omega_{n+1} \in \Omega$. We also use the generalisation of the . operator to the concatenation of two sequences.

### 2.2 Probability theory

For a given set $\Omega$, a *probability subdistribution* is a function $\mu \in \Omega \to [0, 1]$ such that

$$\text{support}(\mu) \overset{\text{def}}{=} \{\omega \in \Omega \mid \mu(\omega) > 0\}$$

is countable. Its probability mass is $|\mu| \overset{\text{def}}{=} \sum_{\omega\in\text{support}(\mu)} \mu(\omega)$. If $|\mu| = 1$, then $\mu$ is a *probability distribution*. We write SubDistr$(\Omega)$ and Distr$(\Omega)$ for the sets of all probability subdistributions and distributions over $\Omega$, respectively. The *Dirac* distribution for $\omega$ is $\mathcal{D}(\omega)$, defined by $\mathcal{D}(\omega) = 1$ and $\mathcal{D}(\omega') = 0$ for all $\omega' \neq \omega$. Given probability distributions $\mu_1$ and $\mu_2$, we denote by $\mu_1 \otimes \mu_2$ the *product distribution*, which is the unique probability distribution defined by

$$(\mu_1 \otimes \mu_2)(\langle\omega_1, \omega_2\rangle) = \mu_1(\omega_1) \cdot \mu_2(\omega_2)$$

for all $\langle\omega_1, \omega_2\rangle \in \text{support}(\mu_1) \times \text{support}(\mu_2)$.

Let $\Omega$ be endowed with a $\sigma$-algebra $\sigma(\Omega)$: a collection of *measurable* subsets of $\Omega$. A *probability measure* over $\Omega$ is a function $\mu \in \sigma(\Omega) \to [0, 1]$ such that

$$\mu(\Omega) = 1 \quad \text{and} \quad \mu(\cup_{i\in I} B_i) = \sum_{i\in I} \mu(B_i)$$

for any countable index set $I$ and pairwise disjoint measurable sets $B_i \subseteq \Omega$. Meas$(\Omega)$ is the set of probability measures over $\Omega$. Each $\mu \in$ Distr$(\Omega)$ induces a probability measure, and we also write $\mathcal{D}(\cdot)$ for the Dirac measure.

### 2.3 Valuations

$\mathit{Val} \overset{\text{def}}{=} V \to \mathbb{R}^+_0$ is the set of *valuations* for an (implicit) set $V$ of (nonnegative real-valued) variables. Valuation $\mathbf{0}$ assigns value zero to all variables. Given $X \subseteq V$ and $v \in \mathit{Val}$, we write $v[X \mapsto 0]$ for the valuation defined by $v[X \mapsto 0](x) = 0$ if $x \in X$ and $v[X \mapsto 0](y) = v(y)$ otherwise. For $t \in \mathbb{R}^+_0$, $v + t$ is the valuation defined by $(v + t)(x) = v(x) + t$ for all $x \in V$.

## 3 Automata with stochastic time

We now present the formal automata-based models underlying our model-based testing approaches: Markov automata for memoryless time and stochastic automata for general stochastic time. In addition to their syntax and semantics (in terms of paths, traces and trace distributions), we define parallel composition operators to formally capture the interaction between implementations and test cases.

### 3.1 Markov automata

Our approach to testing memoryless stochastic-timed systems builds upon the framework of Markov automata [17]. They are a formal model that unifies the discrete probabilistic and nondeterministic choices of Markov decision processes (MDP) with the exponentially distributed delays of continuous-time Markov chains (CTMC) in a compositional way. The exponential distribution provides an appropriate approximation of reality if only the mean durations of activities are known, as is often the case in practice.

In Markov automata, we distinguish between *probabilistic* and *Markovian* transitions. The former take place as soon as possible and lead into a probability distribution over successor states (as in MDP). The latter are defined via a *rate* parameter in $\mathbb{R}^+$: the time until the transition is taken follows the exponential distribution with that rate (as in CTMC).

**Definition 1** (*IOMA*) An *input–output Markov automaton* (IOMA) is a tuple

$$\mathcal{M} = \langle S, s_0, Act, T_P, T_M \rangle$$

where

- $S$ is a finite set of *states*,
- $s_0 \in S$ is the *initial state*,
- $Act = Act_I \uplus Act_O \uplus \{ \tau \}$ is the set of *actions* partitioned into inputs, outputs, and the internal action $\tau$, respectively, with $\delta \in Act_O$ being the distinct *quiescence* action,
- $T_P \in S \to \mathcal{P}(Act \times \mathrm{Distr}(S))$ is the finite *probabilistic transition function*, and
- $T_M \in S \to \mathcal{P}(\mathbb{R}^+ \times S)$ is the finite *Markovian transition function*.

If $\langle \lambda, s' \rangle \in T_M(s)$, we say that $\langle s, \lambda, s' \rangle$ is a (Markovian) transition (of $\mathcal{M}$), also written $s \overset{\lambda}{\rightsquigarrow} s'$. If $\langle a, \mu \rangle \in T_P(s)$, we say that $\langle s, a, \mu \rangle$ is a (probabilistic) transition (of $\mathcal{M}$), also written $s \xrightarrow{a} \mu$. We say that $s$ is Markovian if $|T_M(s)| \neq 0$; $s$ is probabilistic if $|T_P(s)| \neq 0$. We write $s \to a$ if $\exists\, \mu : s \xrightarrow{a} \mu$, and $s \nrightarrow a$ if $\nexists\, \mu : s \xrightarrow{a} \mu$. In the former case, we also say that action $a$ is *enabled in* $s$. The set *enabled*$(s)$ contains all enabled actions in $s$. We write $s \xrightarrow{a}_{\mathcal{M}} \mu$, etc., to clarify that a transition belongs to IOMA $\mathcal{M}$ if ambiguities arise. For brevity, whenever we refer to an IOMA $\mathcal{M}$, we assume it to be a tuple with components $\langle S, s_0, Act, T_P, T_M \rangle$ as in the above definition unless otherwise noted. $\mathcal{M}$ is *input-enabled* if all inputs are enabled in all states, i.e. we have that $\forall\, a \in Act_I, s \in S : s \to a$.

We partition the action alphabet into inputs and outputs. This captures communication ports of a system with its environment (e.g. a tester). $\tau$ represents internal progress of a system that is not visible to an external observer. The existence of a distinct quiescence action $\delta$ is required to explicitly characterise the absence of any other output for an *indefinite* amount of time. The combination of exponentially distributed delays and quiescence poses a particular challenge to an MBT framework since quiescence in practice is frequently judged by waiting a finite amount of time [5]. We further investigate this challenge in Sect. 5.2.

A Markov automaton starts in its initial state and then progresses through the state space, incurring exponentially distributed delays and jumping between states. When in state $s$, the next transition to take is selected as follows: if there is an outgoing probabilistic transition labelled with an action in $Act_O \cup \{ \tau \}$, we apply the *maximal progress* assumption [27]: no time can pass, and one of these transitions is selected nondeterministically. We also say that outputs and internal actions are *urgent*. Otherwise, time passes until a Markovian transition takes place or an input arrives. The sum of the rates of all outgoing Markovian transitions of $s$ is called its *exit rate*, denoted $\mathbf{E}(s)$. Multiple Markovian transitions represent a *race* between exponential distributions. Thus, the time until any Markovian transition takes place is exponentially distributed with rate $\mathbf{E}(s)$; at that point, the actual transition to take is selected probabilistically, with the probability of each transition being its rate divided by $\mathbf{E}(s)$. We define $\mathbf{R}\left(s, s'\right) = \sum_{\langle \lambda, s' \rangle \in T_M(s)} \lambda$, the rate from $s$ to $s'$.

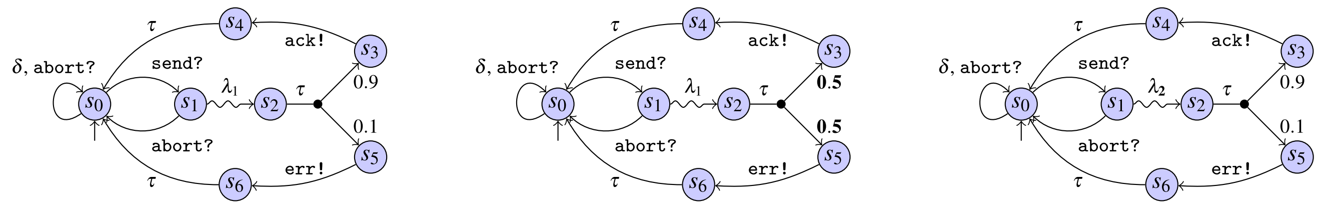
***Example 1*** Figure 1 shows three IOMA describing a protocol that associates a delay with every `send` action, followed by an acknowledgement or error. As a convention, we indicate inputs by a `?` suffix and outputs by a `!` suffix. Discrete probability distributions follow an intermediate dot. Markovian transitions are presented as wavy arrows.

After the `send?` input is received by the specification in Fig. 1a, there is an exponentially distributed delay with rate $\lambda_1$: the probability to go from $s_1$ to $s_2$ in at most $T$ time units is $1 - \mathrm{e}^{-\lambda_1 T}$. State $s_2$ has one probabilistic transition. The specification requires that only 10% of all messages end in an error report and the remaining 90% are delivered correctly. After a message is delivered, the automaton goes back to its initial state where it stays quiescent until input is provided. The $\delta$ self-loop marks the absence of outputs.

The "unfair" implementation model in Fig. 1b has the same structure, except for altered probabilities in the distribution out of $s_2$. While the delay conforms to the one prescribed in the specification model, sufficiently many executions of the implementation should reveal that an error is reported more frequently than required. The "slow" implementation model of Fig. 1c assigns rate $\lambda_2$ to the exponential delay between input and output. This is conforming iff $\lambda_1 = \lambda_2$; if $\lambda_2 < \lambda_1$, it would be slower than required. This paper aims at establishing an MBT framework capable of identifying that implementations like these two do not conform to the given specification model.

## 3.2 Stochastic automata

We use stochastic automata [11] to develop an MBT approach for general stochastic-timed systems. They are MDP augmented with real-time clocks that expire after delays governed by general (continuous) probability distributions. In



**(a)** Specification **(b)** Unfair implementation **(c)** Slow implementation

**Fig. 1** Protocol specification IOMA and two erroneous implementations

this way, they allow every stochastic delay to be modelled precisely, without the need for exponential or phase-type approximation as with Markov automata.

The progress of time is governed and tracked across locations and edges explicitly by *clocks*. This is necessary because, working in general continuous time not restricted to exponential distributions, delays in stochastic automata do not have the memoryless property. Clocks are real-valued variables that increase synchronously with rate 1 over time and *expire* some random amount of time after they have been *restarted*. The expiration time is drawn from a probability distribution specified for each clock. Stochastic automata are thus a symbolic model, so they consist of *locations* and *edges* rather than states and transitions.

**Definition 2** (*IOSA*) An *input–output stochastic automaton* (IOSA) is a tuple

$$\mathcal{I} = \langle Loc, \ell_0, \mathcal{C}, Act, E, F \rangle$$

where

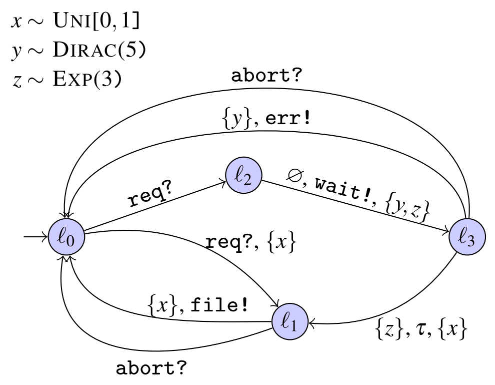
- $Loc$ is a finite set of *locations*,
- $\ell_0 \in Loc$ is the *initial location*,
- $\mathcal{C}$ is a finite set of *clocks*,
- $Act = Act_I \uplus Act_O \uplus \{ \tau \}$ is the set of *actions* partitioned into inputs, outputs, and the internal action $\tau$, respectively, with $\delta \in Act_O$ being the distinct *quiescence* action,
- $E \in Loc \to \mathcal{P}(Edges)$ with $Edges \overset{\text{def}}{=} \mathcal{P}(\mathcal{C}) \times Act \times \mathrm{Distr}(T)$ and $T \overset{\text{def}}{=} \mathcal{P}(\mathcal{C}) \times Loc$ is the *edge function* mapping each location to a finite set of edges that in turn consist of a *guard* set, an action *label*, and a distribution over *targets* in $T$ consisting of a *restart* set of clocks and target locations, and
- $F \in \mathcal{C} \to \mathrm{Meas}(\mathbb{R}_0^+)$ is the *delay measure function* that maps each clock to a probability measure.

We write $pdf(c)$ to refer to the probability density function associated with the measure $F(c)$ for $c \in \mathcal{C}$. As for Markov automata, we use an input–output variant of stochastic automata, along the lines of [12]. We transfer the notation used for transitions in IOMA to edges in IOSA. We call an IOSA $\mathcal{I}$ *input-enabled* if all inputs are available in every location at every time, i.e. $\exists\, \mu \colon \ell \xrightarrow{\varnothing, a_I} \mu$ for all $\ell \in Loc$ and $a_I \in Act_I$.

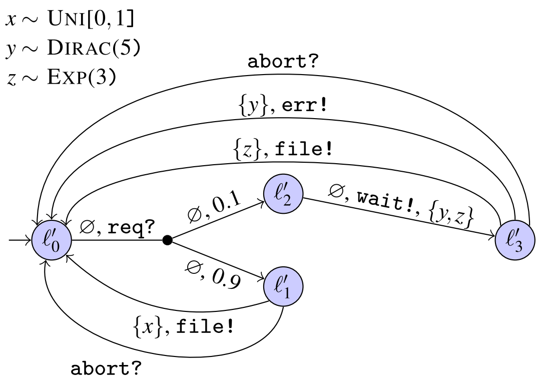
Intuitively, a stochastic automaton starts in the initial location with all clocks expired. An edge may be taken only if all clocks in its guard set $G$ are expired. If *any* output or internal edge is enabled, *some* edge must be taken, i.e. all outputs and internal actions are urgent. When an edge $\ell \xrightarrow{G, a} \mu$ is taken, its action is $a$, we select a target $\langle R, \ell' \rangle \in T$ randomly according to the discrete distribution $\mu$, all clocks in $R$ are restarted, and we move to successor location $\ell'$. There, another edge may be taken immediately or we may need to wait until some further clocks expire, and so on. When a clock $c$ is restarted, the time until it expires is chosen randomly according to the probability measure $F(c)$.

***Example 2*** Figure 2a shows an example IOSA specifying the behaviour of a file server with archival storage. We omit empty restart sets and the empty guard sets of inputs. Upon receiving a request in the initial location $\ell_0$, the specification allows implementations to either move to $\ell_1$ or $\ell_2$. The edge, i.e. the element of $E(\ell_0)$, corresponding to the move to $\ell_1$ is $\langle \varnothing, \texttt{req?}, \mathcal{D}(\langle \{ x \}, \ell_2 \rangle) \rangle$, where $\varnothing$ is the edge's empty guard set—it must be empty since `req?` is an input. The move to $\ell_2$ represents the case of a file in archive: the server must immediately deliver a `wait!` notification and then attempt to retrieve the file from the archive. Clocks $y$ and $z$ are restarted, and used to specify that retrieving the file shall take on average $\frac{1}{3}$ of a time unit, exponentially distributed, but no more than 5 time units. In location $\ell_3$, there is thus a race between retrieving the file and a deterministic timeout. In case of timeout, an error message (action `err!`) is returned; otherwise, the file can be delivered as usual from location $\ell_1$. Clock $x$ is used to specify the transmission time of the file: it shall be uniformly distributed between 0 and 1 time units.

In Fig. 2b, we show an implementation of this specification. One out of ten files randomly requires to be fetched from the archive. This is allowed by the specification: it is one particular (randomised) resolution of the nondeterminism, i.e. underspecification, defined in $\ell_0$. The implementation also manages to transmit files from archive directly while fetching them, as evidenced by the direct edge from $\ell_3$ back to $\ell_0$ labelled `file!`. This violates the timing prescribed by the



**(a)** File server specification



**(b)** File server implementation

**Fig. 2** File server specification and implementation IOSA

specification, and must be detected by an MBT procedure for IOSA.

In the remainder of this paper, whenever a statement applies to both IOMA and IOSA, we will say that it applies to an *automaton* $\mathcal{A}$ for brevity.

## 3.3 Parallel composition

To give a semantics for synchronisation and communication between components of a system, we define a binary parallel composition operator. Two components synchronise on inputs and outputs, and otherwise evolve independently. Our operators are defined w.r.t. a binary input–output relation $M$ that associates outputs of one component with inputs of the other component, and vice versa. Wherever we use the `!`/`?`-suffix convention for action labels, we assume that $M$ relates every output $a\,!$ with the input $a\,?$ and vice versa.

**Markov automata** IOMA interact via probabilistic transitions, while Markovian transitions evolve independently, with the single technical exception of Markovian self-loops:

$$\frac{s_1 \xrightarrow{a}_{\mathcal{M}_1} \mu \quad a = \tau \vee \nexists b \in Act_2\colon \langle a,b\rangle \in M}{\langle s_1,s_2\rangle \xrightarrow{a} \mu \otimes \mathcal{D}(s_2)}\ indep_1$$

$$\frac{s_1 \xrightarrow{a_O}_{\mathcal{M}_1} \mu_1 \quad s_2 \xrightarrow{a_I}_{\mathcal{M}_2} \mu_2 \quad a_O \in Act_{O_1} \wedge \langle a_O,a_I\rangle \in M}{\langle s_1,s_2\rangle \xrightarrow{a_O} \mu_1 \otimes \mu_2}\ sync_1$$

$$\frac{s_1 \overset{\lambda}{\rightsquigarrow}_{\mathcal{M}_1} s_1' \quad s_1 \neq s_1'}{\langle s_1,s_2\rangle \overset{\lambda}{\rightsquigarrow} \langle s_1',s_2\rangle}\ mar_1$$

$$\frac{s_1 \overset{\lambda}{\rightsquigarrow}_{\mathcal{M}_1} s_1}{\langle s_1,s_2\rangle \overset{\mathbf{R}(s_1,s_1)+\mathbf{R}(s_2,s_2)}{\rightsquigarrow} \langle s_1,s_2\rangle}\ marloop_1$$

**Fig. 3** Inference rules for IOMA parallel composition

**Definition 3** (*parallel composition, IOMA*) For two IOMA

$$\mathcal{M}_i = \langle S_i, s_{0_i}, Act_i, T_P^i, T_M^i\rangle,$$

$i \in \{1, 2\}$, and an input–output relation

$$M \subseteq (Act_{O_1} \times Act_{I_2}) \cup (Act_{I_1} \times Act_{0_2}),$$

the parallel composition of $\mathcal{M}_1$ and $\mathcal{M}_2$ w.r.t. $M$ is

$$\mathcal{M}_1 \parallel_M \mathcal{M}_2 \stackrel{\text{def}}{=} \langle S_1 \times S_2, \langle s_{0_1}, s_{0_2}\rangle, Act, T_P, T_M\rangle$$

with $Act \stackrel{\text{def}}{=} Act_I \uplus Act_O \uplus \{\tau\}$, $Act_O = Act_{O_1} \cup Act_{O_2}$, and

$$Act_I \stackrel{\text{def}}{=} (Act_{I_1} \cup Act_{I_2}) \backslash (\sqcap^{Act_{I_1}}_{Act_{O_2}}(M) \cup \sqcap^{Act_{I_2}}_{Act_{O_1}}(M^{-1}))$$

where $\sqcap^I_O(M)$ are the inputs in $I$ that are matched to an output in $O$ by $M$:

$$\sqcap^I_O(M) \stackrel{\text{def}}{=} \{ a_I \in I \mid \exists a_O \in O\colon \langle a_I, a_O\rangle \in M \}.$$

The transition functions $T_P$ and $T_M$ are the smallest functions satisfying the inference rules given in Fig. 3 plus symmetric rules *indep*$_2$, *sync*$_2$, *mar*$_2$, and *marloop*$_2$ for the corresponding independent steps, synchronising outputs, Markovian transitions, and Markovian loops of $\mathcal{M}_2$.

In the action alphabet only those inputs carry over that do not have a synchronising output in the other component associated with them via $M$. If $s_1 \rightarrow_{\mathcal{M}_1} a_1$ and $\langle a_1, a_2\rangle \in M$, an $a_1$-labelled transition can only take place in synchronisation with an $a_2$-labelled transition from the second component (assuming no other action is associated with $a_1$ by $M$). In particular, if $s_1 \nrightarrow_{\mathcal{M}_1} a_2$, then $\langle s_1, s_2\rangle$ has no $a_1$-$a_2$-synchronising transition: synchronisation waits for all partners to be ready. We later restrict to input-enabled models to make sure that outputs cannot be prevented from occurring immediately.

$$\frac{\ell_1 \xrightarrow{G,a}_{\mathcal{I}_1} \mu \quad a = \tau \vee \nexists b \in Act_2 \colon \langle a, b\rangle \in M}{\langle \ell_1, \ell_2\rangle \xrightarrow{G,a} \{\, \langle R, \langle \ell_1', \ell_2\rangle\rangle \mapsto \mu(\langle R, \ell_1'\rangle) \mid R \subseteq \mathcal{C}, \ell_1' \in Loc_1 \,\}}$$

$$\frac{\ell_1 \xrightarrow{G_1,a_O}_{\mathcal{I}_1} \mu_1 \quad \ell_2 \xrightarrow{G_2,a_I}_{\mathcal{I}_2} \mu_2 \quad a_O \in Act_{O_1} \wedge \langle a_O, a_I\rangle \in M}{\langle \ell_1, \ell_2\rangle \xrightarrow{G_1 \cup G_2, a_O} \{\, \langle R_1 \cup R_2, \langle \ell_1', \ell_2'\rangle\rangle \mapsto \mu_1(\langle R_1, \ell_1'\rangle) \cdot \mu_2(\langle R_2, \ell_2'\rangle) \,\}}$$

**Fig. 4** Inference rules for IOSA parallel composition

**Stochastic automata** The definition of parallel composition for IOSA is similar: while there are no Markovian transitions, the synchronisation of probabilistic edges now requires building the unions of the involved guard and restart sets. This means that a synchronising edge in the parallel composition only takes places as soon as both of its constituent edges are enabled: synchronisation partners wait, just as in IOMA.

**Definition 4** (*parallel composition, IOSA*) For two IOSA

$$\mathcal{I}_i = \langle Loc_i, \ell_{0_i}, \mathcal{C}_i, Act_i, E_i, F_i\rangle,$$

$i \in \{1, 2\}$, with $\mathcal{C}_1 \cap \mathcal{C}_2 = \varnothing$ and an input–output relation $M$ as in Definition 3, the parallel composition of $\mathcal{I}_1$ and $\mathcal{I}_2$ w.r.t. $M$ is

$$\mathcal{I}_1 \parallel \mathcal{I}_2 \overset{\text{def}}{=} \langle Loc_1 \times Loc_2, \langle \ell_{0_1}, \ell_{0_2}\rangle, \mathcal{C}_1 \cup \mathcal{C}_2, Act, E, F_1 \cup F_2\rangle$$

with $Act$ as in Definition 3 and $E$ being the smallest function satisfying the inference rules given in Fig. 4, plus symmetric rules for the corresponding steps of $\mathcal{I}_2$.

## 3.4 Qualitative semantics

The non-probabilistic aspects of the semantics of IOMA and IOSA are captured in the notion of a path, which precisely represents a single execution of an automaton.

### 3.4.1 Paths

A concrete execution of an automaton—the exact amount of time spent in each state, the transition/edge taken, and the selected successor state/location—is captured by a path.

**Markov automata** The definition of paths for IOMA is based on the automaton's states and transitions:

**Definition 5** (*path, IOMA*) The set of all *paths* of an IOMA $\mathcal{M}$ is

$$paths(\mathcal{M}) \subseteq S \times (\mathbb{R}_0^+ \times T \times \{\varnothing\} \times S)^{\leq \omega},$$

with $T \overset{\text{def}}{=} (Act \times \text{Distr}(S)) \cup \mathbb{R}^+$ serving to characterise transitions, and contains precisely the sequences $\pi$ of the form

$$\pi = s_0\, t_1\, \alpha_1\, \varnothing\, s_1\, t_2\, \alpha_2\, \varnothing \ldots$$

where, for all applicable $i \geq 1$, for the $\alpha_i \in T$ we have that either $\alpha_i = \langle a_i, \mu_i\rangle \in Act \times \text{Distr}(S)$ such that

$$\langle a_i, \mu_i\rangle \in T_P(s_{i-1}) \wedge \mu_i(s_i) > 0,$$

i.e. $\alpha_i$ is a probabilistic transition, or $\alpha_i = \lambda_i \in \mathbb{R}^+$ with $\langle \lambda_i, s_i\rangle \in T_M(s_{i-1})$, i.e. it is a Markovian transition.

By definition, every finite path ends in a state, and either $s_i \xrightarrow{a_{i+1}} \mu_{i+1}$ or $s_i \overset{\lambda_{i+1}}{\rightsquigarrow} s_{i+1}$ for every non-final state $s_i$. A subsequence $s_{i-1}\, t_i\, \alpha_i\, \varnothing\, s_i$ means that $\mathcal{M}$ resided $t_i$ time units in state $s_{i-1}$ before moving to $s_i$ via $\alpha_i$. The empty sets $\varnothing$ are for consistent notation with paths for IOSA (see below).

**Stochastic automata** IOSA comprise real-valued clocks; to define a path through an IOSA $\mathcal{I}$, we need to keep track of their values and expiration times. We do so by defining the *state* of $\mathcal{I}$to include these values: the set of *states* of an IOSA $\mathcal{I}$ is $S \overset{\text{def}}{=} Loc \times Val \times Val$. Each state $\langle \ell, v, x\rangle \in S$ consists of the current location $\ell$ and the values $v$ and expiration times $x$ of all clocks. Consequently, the state space of an IOSA is uncountably infinite.

**Definition 6** (*path, IOSA*) Let us define the predicate

$$\text{Ex}(G, v, x) \overset{\text{def}}{=} \forall c \in G \colon v(c) \geq x(c)$$

that indicates whether all clocks in $G$ are expired. Then, the set of all *paths* of an IOSA $\mathcal{I}$ is

$$paths(\mathcal{I}) \subseteq S \times (\mathbb{R}_0^+ \times Edges \times \mathcal{P}(\mathcal{C}) \times S)^{\leq \omega}$$

and contains precisely the sequences $\pi$ of the form

$$\pi = \langle \ell_0, v_0, x_0\rangle\, t_1\, \langle G_1, a_1, \mu_1\rangle\, R_1\, \langle \ell_1, v_1, x_1\rangle\, t_2 \ldots$$

where $v_0 = x_0 = \mathbf{0}$ and, for all applicable $i \geq 1$, we have

- $\ell_{i-1} \xrightarrow{G_i, a_i} \mu_i$,
- $v_i = (v_{i-1} + t)[R_i \mapsto 0]$,
- $\text{Ex}(G_i, v_{i-1} + t, x_{i-1})$ is satisfied,
- $\mu_i(\langle R_i, \ell_i\rangle) > 0$,
- the expiration times satisfy

$$x_i \in \{\, x \in Val \mid \forall c \in \mathcal{C} \setminus R_i \colon x(c) = x_{i-1}(c) \\ \wedge \forall c \in R_i \colon x(c) \geq 0 \,\},$$

– and if $a_i \notin Act_I$, then additionally

$$\nexists t' \in [0, t[ \colon \exists \ell_{i-1} \xrightarrow{G, a} \mu \colon \mathrm{Ex}(G, v_{i-1} + t', x_{i-1}).$$

The last condition implements the urgency of outputs and internal actions. We require that every path starts in the initial location with all clocks and expiration times set to zero. An edge may only be taken if all clocks in its guard set are expired (which is the case when predicate Ex is satisfied). The clock values in the successor state are obtained by resetting exactly those clocks in the restart set $R_i$ to zero. All other clocks keep their value and expiration time.

We write $last(\pi)$ to denote the last state of a finite path. We write $\pi' \sqsubseteq \pi$ if $\pi'$ is a *prefix* of $\pi$. The set of all finite paths of an automaton $\mathcal{A}$ is $paths^{fin}(\mathcal{A})$. The set of *complete paths*, denoted $paths^{com}(\mathcal{A})$, contains every path ending in a deadlock, i.e. in a state $s$ where $T_P(s) = T_M(s) = \varnothing$ (for IOMA) or a location $\ell$ where $E(\ell) = \varnothing$ (for IOSA).

### 3.4.2 Traces

A trace is the projection of a path to its delays and actions, recording the path's visible behaviour:

**Definition 7** (*trace*) The *trace* of $\pi$ is

$$tr(\pi) \in (\mathbb{R}_0^+ \times Act \backslash \{\tau\})^{\leq\omega}$$

given as the projection of

$$\pi = s_0\, t_1\, \alpha_1\, R_1\, s_1\, t_2\, \alpha_2\, R_2 \ldots$$

to the $t_i$ and the actions $a_i \neq \tau$ of those $\alpha_i$ that are of the form $\langle a_i, \mu_i\rangle \in Act \times \mathrm{Distr}(S)$ for IOMA or $\langle G_i, a_i, \mu_i\rangle \in Edges$ for IOSA, summing up the $t_i$ over all subsequent steps where $\alpha_i$ is of another form (i.e. internal and Markovian transitions for IOMA and internal edges for IOSA). The *length* of $\pi$, denoted $|\pi|$, is the number of actions on $tr(\pi)$. The set $tr^{-1}(\sigma)$ is the set of all paths that have trace $\sigma$. The set of all traces of an automaton $\mathcal{A}$ is $traces(\mathcal{A})$, while $traces^{fin}(\mathcal{A})$ is the set of all of its finite traces. Finally, $traces^{com}(\mathcal{A})$ is the set of all its *complete traces*, i.e. those $\sigma$ for which $tr^{-1}(\sigma)$ contains at least one complete path.

### 3.4.3 Abstract traces

When delays are governed by continuous probability distributions, the probability of any single time point is zero. Hence, we will need a notion that represents an automaton's behaviour over time intervals instead of points.

**Definition 8** (*abstract trace*) An *abstract trace* is a trace where each delay $t_i$ is replaced by an interval $I_i \subseteq \mathbb{R}_0^+$ with $t_i \in I_i$.

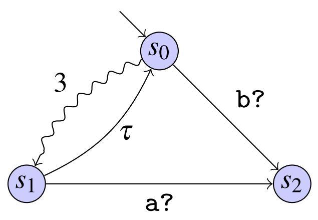


**Fig. 5** Example IOMA for paths and traces

W.l.o.g. we only consider non-empty intervals of the form $[0, t]$ in the remainder of this paper. Consequently, every trace can be replaced by its abstract trace by changing all $t_i$ to $[0, t_i]$ and vice versa, defining a bijection between traces and their abstract counterparts. Hence, for a trace $\sigma$ we denote by $\Sigma$ its corresponding abstract trace. $AbsTraces(\mathcal{A})$ is the set of all abstract traces of automaton $\mathcal{A}$, and $AbsTraces^{fin}(\mathcal{A})$ is the set of all its finite abstract traces. For $\Sigma$ and $\Sigma'$ with $\Sigma = I_1\, a_1\, I_2\, a_2 \ldots a_n$ and $\Sigma' = I_1'\, a_1'\, I_2'\, a_2' \ldots$, we say $\Sigma$ is a *prefix* of $\Sigma'$, denoted $\Sigma \sqsubseteq \Sigma'$, if $I_i = I_i'$ and $a_i = a_i'$ for $i = 1, 2, \ldots, n$. That is, $\Sigma$ and $\Sigma'$ coincide on the first $n$ steps. Finally, we define $act(\sigma)$ as the *action trace* of $\sigma$, obtained by removing all time values $t_i$ from $\sigma$, i.e. $act(\sigma)$ consists of actions in $Act \backslash \{\tau\}$ only.

***Example 3*** Consider the IOMA $\mathcal{M}$ given in Fig. 5. Let the three Dirac distributions of the transitions labelled $\tau$, `a?`, and `b?` be $\mu_\tau$, $\mu_a$ and $\mu_b$, respectively. For the path

$$\pi = s_0\; 2.9\; 3\; \varnothing\; s_1\; 0\; \langle \tau, \mu_\tau\rangle\; \varnothing\; s_0\; 0\; \langle \mathtt{b?}, \mu_b\rangle\; \varnothing\; s_2$$

we have $\pi \in paths^{com}(\mathcal{M})$, trace $tr(\pi) = \sigma = 2.9\ \mathtt{b?}$, abstract trace $\Sigma = [0, 2.9]\ \mathtt{b?}$, action trace $act(\sigma) = \mathtt{b?}$, and path length $|\pi| = 1$. Note that the trace is much shorter than the path since it omits the internal $\tau$ steps and then merges all the delay steps between any two consecutive remaining (i.e. non-$\tau$) actions.

## 3.5 Quantitative semantics

Our goal is now to quantify the frequency of observed traces. For this purpose, we first define schedulers, which resolve all nondeterministic choices, and then a probability space and measure over the remaining paths. The space and measure will allow us to specify trace distributions.

### 3.5.1 Schedulers

IOMA and IOSA comprise nondeterministic choices, discrete probability distributions, and delays following continuous probability distributions. Due to the nondeterminism, we cannot assign probabilities to paths and traces directly. Rather, we resort to *schedulers* that resolve nondeterminism, and consequently yield a purely probabilistic system.

Given any finite history leading to a state/location, a scheduler returns a discrete probability distribution over the set of next transitions/edges. In order to model termination, we define schedulers such that they can continue paths with a halting extension $\perp$, after which only quiescence is observed.

**Definition 9** (*scheduler, IOMA*) A *scheduler* of an IOMA $\mathcal{M}$ is a function

$$\mathfrak{S} \in paths^{fin}(\mathcal{M}) \to \text{SubDistr}(Act \times \text{Distr}(S) \cup \{\perp\})$$

such that, with $last(\pi) = s$, $\mathfrak{S}(\pi)(\langle a, \mu\rangle) > 0$ implies $s \xrightarrow{a} \mu$, and if $s \to a$ for $a \in Act_O \cup \{\tau\}$ then $|\mathfrak{S}(\pi)| = 1$. The probability to *halt* is $\mathfrak{S}(\pi)(\perp)$; we say that $\mathfrak{S}$ *halts* on $\pi$ if $\mathfrak{S}(\pi)(\perp) = 1$, and that $\mathfrak{S}$ is of *length* $k \in \mathbb{N}$ if it halts on all paths $\pi$ with $|\pi| \geq k$ and for every complete path of length less than $k$. The set of all schedulers of $\mathcal{M}$ of length $k$ is $Sched(\mathcal{M})^{\leq k}$; the set of all schedulers of finite length is $Sched(\mathcal{M})$.

The definition of schedulers ensures that only enabled transitions are chosen. We use subdistributions, as opposed to distributions, such that the probability mass a scheduler did not assign to actions in *Act* is left for Markovian transitions. That is, a scheduler chooses an action, halts *immediately* ($\perp$), or leaves a chance for Markovian actions to take place. Schedulers for IOSA are defined similarly:

**Definition 10** (*scheduler, IOSA*) A *scheduler* of an IOSA $\mathcal{I}$ is a measurable function

$$\mathfrak{S} \in paths^{fin}(\mathcal{I}) \to \text{Distr}(Edges \cup \{\perp\})$$

such that, with $last(\pi) = \langle \ell, v, x\rangle$, $\mathfrak{S}(\pi)(\langle G, a, \mu\rangle) > 0$ implies $\ell \xrightarrow{G,a} \mu \wedge \text{Ex}(G, v + t, x)$ where $t \in \mathbb{R}_0^+$ is the minimal delay for which no other transition was available before, i.e.

$$\nexists t' \in [0, t[\colon \bigvee_{\ell \xrightarrow{G',a'} \mu'} \text{Ex}(G', v + t', x).$$

$\mathfrak{S}(\pi)(\perp)$ is the probability to *halt*. $\mathfrak{S}$ *halts* on $\pi$ if $\mathfrak{S}(\pi)(\perp) = 1$. $\mathfrak{S}$ is of *length* $k \in \mathbb{N}$ if it halts on all paths $\pi$ with $|\pi| \geq k$ and for every complete path of length less than $k$. The set of all schedulers of $\mathcal{I}$of length $k$ is $Sched(\mathcal{I})^{\leq k}$; the set of all schedulers of finite length is $Sched(\mathcal{I})$.

A scheduler for an IOSA can only choose between the edges enabled at the points where *any* edge just became enabled. While actions (via probabilistic transitions) and the passage of time (via Markovian transitions) were decoupled in IOMA, edges in IOSA directly govern delays. Schedulers thus return distributions, not subdistributions.

***Remark 1*** We use schedulers in the context of MBT in an open environment, yet schedule both inputs and outputs. This is in contrast to similar approaches in the literature; for instance, [7] use a partial scheduler for each component and an *arbiter scheduler* that tells precisely how progress of the composed system is determined. Our approach is non-compositional (see, for example, [44]). However, we utilise schedulers only to determine the probabilities of paths and traces, which does not require compositionality.

For both IOMA and IOSA, we restrict to finite-length schedulers in the remainder of the paper. As is usual, we also consider only schedulers that let time diverge with probability 1.

### 3.5.2 Probabilities of paths

By resolving all nondeterminism, a scheduler makes it possible to calculate the probability for measurable sets of paths via step probability functions. A scheduler schedules without delay. Hence, there are no additional races between Markovian transitions or edges and scheduler decisions.

**Definition 11** (*step probability, IOMA*) Let $\mathfrak{S}$ be a scheduler of an IOMA $\mathcal{M}$. We define the *step probability function* $Q^{\mathfrak{S}}$ from $paths^{fin}(\mathcal{M})$ to

$$\text{Meas}((\mathbb{R}_0^+ \times T \times \{\varnothing\} \times S) \cup \{\perp\}),$$

with $T \overset{\text{def}}{=} (Act \times \text{Distr}(S)) \cup \mathbb{R}^+$ by $Q^{\mathfrak{S}}(\pi)(\perp) = \mathfrak{S}(\pi)(\perp)$ and, for $\pi$ with $last(\pi) = s$, by

$$Q^{\mathfrak{S}}(\pi)(I \times A_Q \times \{\varnothing\} \times S_Q) =$$

$$\sum_{s' \in S_Q} \Big( \sum_{\alpha_P \in T_P(s) \cap A_Q} P_\pi(I, \alpha_P, s') + \sum_{\alpha_M \in T_M(s) \cap A_Q} M_\pi(I, \alpha_M, s') \Big)$$

with $P_\pi(I, \langle a, \mu\rangle, s') \overset{\text{def}}{=} \mathbb{1}_{0 \in I} \cdot \mathfrak{S}(\pi)(\langle \alpha, \mu\rangle) \cdot \mu(s')$

and $M_\pi(I, \langle \lambda, s''\rangle, s') \overset{\text{def}}{=} \mathbb{1}_{s''=s'} \cdot (1 - |\mathfrak{S}(\pi)|) \cdot \displaystyle\int_{t \in I} \lambda\, e^{-\mathbf{E}(s)\cdot t}$.

The probability to halt right after $\pi$ is inferred from the probability a scheduler assigns to the halting extension $\perp$. Otherwise, this function defines, for every path $\pi$, a measure quantifying the probability to continue from state $last(\pi) = s$ by incurring a delay in the interval $I \subseteq \mathbb{R}_0^+$, taking a transition in $A_Q$, and ending up in a state in $S_Q$. Auxiliary function $P_\pi$ calculates the probability of doing so via a probabilistic transition while $M_\pi$ considers Markovian transitions. The integral in $M_\pi$ implements the exponential distribution of delays.

**Definition 12** (*step probability, IOSA*) Let $\mathfrak{S}$ be a scheduler of an IOSA $\mathcal{I}$. We define the *step probability function* $Q^{\mathfrak{S}}$ in

$$paths^{fin}(\mathcal{I}) \to \text{Meas}((\mathbb{R}_0^+ \times Edges \times \mathcal{P}(\mathcal{C}) \times S) \cup \{\perp\})$$

by $Q^{\mathfrak{S}}(\pi)(\bot) = \mathfrak{S}(\pi)(\bot)$ and, for $\pi$ with $last(\pi) = \langle \ell, v, x \rangle$ and $t$ the minimal delay in $\ell$ as in Definition 6,

$$Q^{\mathfrak{S}}(\pi)(I \times E_Q \times R_Q \times S_Q) = \mathbb{1}_{t \in I} \cdot \sum_{e \in E_Q} Y^{S_Q}_{R_Q}(\pi, e)$$

where

$$Y^{S_Q}_{R_Q}(\pi, e) \stackrel{\text{def}}{=} \mathfrak{S}(\pi)(e) \cdot \sum_{R \in R_Q, \ell' \in Loc} \mu(\langle R, \ell' \rangle) \cdot \int_{\langle \ell', v', x' \rangle \in S_Q} X^x_R(v', x')$$

and

$$X^x_R(v', x') \stackrel{\text{def}}{=} \mathbb{1}_{v'=(v+t)[R \mapsto 0]} \prod_{c \in \mathcal{C}} \begin{cases} 1 \text{ if } c \notin R \wedge x(c) = x'(c) \\ 0 \text{ if } c \notin R \wedge x(c) \neq x'(c) \\ pdf(c)(x'(c)) \;\text{ if } c \in R. \end{cases}$$

This function defines, for every path $\pi$, a measure quantifying the probability to continue from state $last(\pi) = \langle \ell, v, x \rangle$ by incurring a delay in the interval $I \subseteq \mathbb{R}^+_0$, taking an edge in $E_Q$, resetting a set of clocks in $R_Q$, and ending up in a state in $S_Q$. First, the factor $\mathbb{1}_{t \in I}$ ensures that only delays in $I$ have positive probability. We then sum the probabilities over all edges, with the value for each edge being given by auxiliary function $Y^{S_Q}_{R_Q}$. In that function, we multiply the probability that the scheduler selects this edge, the probability for each probabilistic branch, and the probability to end up in a state in $S_Q$ by following that branch. States are uncountable, so we integrate the probability density for every state as given by auxiliary function $X^x_R$. A state can only have positive probability if the values it assigns to clocks are the previous values plus the selected delay plus the branch's clock restarts (factor $\mathbb{1}_{v'=(v+t)[R \mapsto 0]}$). The final multiplication in $X^x_R$ assigns the correct probability mass (via $pdf(c)(x'(c))$) to sampling new expiration times for the clocks that are restarted (identified by $c \in R$); all other clocks retain their expiration times (as enforced by the first two lines of the case distinction).

### 3.5.3 Trace distributions

Overall, the two-step probability functions induce unique probability measures $P_{\mathfrak{S}}$ over $paths^{fin}(\mathcal{A})$ for an automaton $\mathcal{A}$and a scheduler $\mathfrak{S}$. We can define the *trace distribution* for $\mathcal{A}$ and a scheduler as the probability measure over traces (using abstract traces to construct the corresponding $\sigma$-algebra) induced by these probability measures over paths in the usual way. The probability of a set of abstract traces $X$ is the probability of all paths whose trace is in $X$.

**Definition 13** (*trace distribution*) The trace distribution $\mathcal{T}$ of a scheduler $\mathfrak{S} \in Sched(\mathcal{M})$, denoted $\mathcal{T} = trd(\mathfrak{S})$, is given by the probability space $\langle \Omega_{\mathcal{T}}, \mathcal{F}_{\mathcal{T}}, P_{\mathcal{T}} \rangle$ where

– $\Omega_{\mathcal{T}} \stackrel{\text{def}}{=} AbsTraces(\mathcal{M})$,
– $\mathcal{F}_{\mathcal{T}}$ is the smallest $\sigma$-field generated by the sets

$$\{ C_\Sigma \mid \Sigma \in AbsTraces^{fin}(\mathcal{M}) \}$$

with $C_\Sigma \stackrel{\text{def}}{=} \{ \Sigma' \in \Omega_{\mathcal{T}} \mid \Sigma \sqsubseteq \Sigma' \}$, and
– $P_{\mathcal{T}}$ is the unique probability measure on $\mathcal{F}_{\mathcal{T}}$ defined by $P_{\mathcal{T}}(X) = P_{\mathfrak{S}}(tr^{-1}(X))$ for $X \in \mathcal{F}_{\mathcal{T}}$.

We can also use trace distributions to relate two automata: $\mathcal{A}_1$ and $\mathcal{A}_2$ are related if they induce the same trace distributions. In particular, a trace distribution $\mathcal{T}$ of $\mathcal{A}_1$ is contained in the set of trace distributions of $\mathcal{A}_2$ if there is a scheduler $\mathfrak{S}$ in $\mathcal{A}_2$ such that $\mathcal{T} = trd(\mathfrak{S})$. We write $trd(\mathcal{A}, k)$ for the set of trace distributions based on a scheduler of length $k$ and $trd(\mathcal{A})$ for the set of all finite trace distributions. Finally, we write $\mathcal{A}_1 \sqsubseteq^k_{TD} \mathcal{A}_2$ if $trd(\mathcal{A}_1, k) \subseteq trd(\mathcal{A}_2, k)$ for $k \in \mathbb{N}$, and $\mathcal{A}_1 \sqsubseteq^{fin}_{TD} \mathcal{A}_2$ if $\mathcal{A}_1 \sqsubseteq^k_{TD} \mathcal{A}_2$ for some $k \in \mathbb{N}$. This induces an equivalence relation $=_{TD}$: $\mathcal{A}_1$ and $\mathcal{A}_2$ are trace distribution equivalent, written $\mathcal{A}_1 =_{TD} \mathcal{A}_2$, iff $trd(\mathcal{A}_1) = trd(\mathcal{A}_2)$.

## 4 Stochastic testing theory

Model-based testing comprises automatic test case generation, execution, and evaluation based on a requirements model. We now establish this three-step procedure for IOMA and IOSA. As a first step, we define formal conformance between two models via two conformance relations akin to **ioco** [49], called **mar-ioco** and **sa-ioco**. We then specify what a test case is, and when an observed trace should be judged as correct via test annotations. Working in a stochastic environment also necessitates a statistical verdict. We describe the sampling process for an IUT and then define verdict functions. Finally, we prove the correctness of the framework.

The main difference of our stochastic test theory, compared to the probabilistic test theory of [20], lies in the sampling process and its resulting observations, in particular, in the trace frequency counting functions. We carefully defined IOMA and IOSA in such a way that many of the notions in the remainder of this section apply to both settings. For this reason, we will write ***-ioco**, $\sqsubseteq^*_{ioco}$, etc., to summarise a definition for both **mar-ioco** and **sa-ioco**, $\sqsubseteq^{mar}_{ioco}$ and $\sqsubseteq^{sa}_{ioco}$, etc.

### 4.1 Stochastic conformance relations

The purpose of the conformance relation is to judge whether an implementation model conforms to the requirements specification model. We define our relations for IOMA and IOSA such that they only rely on trace distributions. Trace distribution equivalence $=_{TD}$ is the probabilistic counterpart of *trace* *equivalence* for transition systems. However, trace equivalence or inclusion is too fine as a conformance relation for testing [48]. The **ioco** relation for *functional* conformance solves this problem by allowing underspecification of functional behaviour: an implementation $\mathcal{I}$ is conforming to a specification $\mathcal{S}$ if every experiment derived from $\mathcal{S}$ executed on $\mathcal{I}$ leads to an output that was foreseen in $\mathcal{S}$:

$$\mathcal{I} \sqsubseteq_{ioco} \mathcal{S} \;\Leftrightarrow\; \forall \sigma \in traces^{fin}(\mathcal{S}): out_{\mathcal{I}}(\sigma) \subseteq out_{\mathcal{S}}(\sigma)$$

where $out_{\mathcal{I}}(\sigma)$ is the set of outputs in $\mathcal{I}$ that is enabled after trace $\sigma$. To extend ioco testing to stochastic systems, we need two auxiliary concepts that mirror trace prefixes and the set *out* stochastically:

**Definition 14** (*prefix and output continuation*) For trace distributions $\mathcal{T}$ of length $k$ and $\mathcal{T}'$ of length $\geq k$, the prefix relation $\sqsubseteq_k$ is defined by

$$\mathcal{T} \sqsubseteq_k \mathcal{T}' \;\Leftrightarrow\; \forall \sigma \in (\mathbb{R}_0^+ \times Act)^{\leq k}: P_{\mathcal{T}}(\Sigma) = P_{\mathcal{T}'}(\Sigma).$$

For an automaton $\mathcal{A}$, the *output continuation* of trace distribution $\mathcal{T}$ of length $k$ is $outcont_{\mathcal{A}}(\mathcal{T})$ defined as the set of all $\mathcal{T}' \in trd(\mathcal{A}, k+1)$ such that

$$\mathcal{T} \sqsubseteq_k \mathcal{T}' \wedge \forall \sigma \in (\mathbb{R}_0^+ \times Act)^k \times \mathbb{R}_0^+ \times Act_I: P_{\mathcal{T}'}(\Sigma) = 0.$$

The prefix relation extends the one for traces to trace distributions. The output continuation of $\mathcal{T}$ of length $k$ in $\mathcal{M}$ contains all trace distributions $\mathcal{T}'$ of length $k+1$ such that $\mathcal{T} \sqsubseteq_k \mathcal{T}'$ and $\mathcal{T}'$ assigns probability zero to every abstract trace of length $k+1$ that ends with an input.

We can now define the **mar-ioco** and **sa-ioco** conformance relations that relate input-enabled implementations $\mathcal{I}$ to specifications $\mathcal{S}$. Intuitively, $\mathcal{I}$ conforms to $\mathcal{S}$ if the probability of every output trace of $\mathcal{I}$ can be matched by $\mathcal{S}$ under some scheduler. This includes the functional behaviour, probabilistic behaviour, and stochastic timing, as accounted for in the definition of output continuations.

**Definition 15** (**mar-ioco***and* **sa-ioco**) Let $\mathcal{I}$ and $\mathcal{S}$ be automata over the same action signature with $\mathcal{I}$ input-enabled. $\mathcal{I}$ is ***-ioco**-conforming to $\mathcal{S}$, written $\mathcal{I} \sqsubseteq^*_{ioco} \mathcal{S}$, if for all $k \in \mathbb{N}$ we have

$$\forall \mathcal{T} \in trd(\mathcal{S}, k) : outcont_{\mathcal{I}}(\mathcal{T}) \subseteq outcont_{\mathcal{S}}(\mathcal{T}).$$

***Example 4*** Recall the protocol models of Fig. 1. After the `send?` input, there is a delay before the file transmission is either acknowledged or an error is reported. Let $\mathcal{S}$ be the leftmost automaton and $\mathcal{I}$ be the rightmost one. Consider now the scheduler of $\mathcal{S}$ that schedules `send?` with probability 1. Its set of output continuations in $\mathcal{S}$ contains all trace distributions that schedule the outgoing distribution leading to `ack!` and `err!` with probability $p$ and halt with $1-p$, for $p \in [0,1]$. This holds for the set of output continuations in $\mathcal{I}$, but the probability to reach $s_2$ within a certain amount of time $t$ differs from $\mathcal{S}$ whenever $\lambda_1 \neq \lambda_2$. Hence, there are trace distributions in $\mathcal{I}$ such that the probability of, for example,

$[0, 0]$ `send?` $[0, t]$ `ack!`

cannot be matched. The implementation is therefore not conforming with respect to **mar-ioco** in this case.

**Relationship to other relations** If $\mathcal{A}$ is an IOMA without Markovian transitions or an IOSA where $\mathcal{C} = \varnothing$, then $\mathcal{A}$ is a probabilistic input–output transition system (pIOTS). Under this restriction, **mar-ioco** and **sa-ioco** coincide with **pioco** of [20] and are thus extensions of **pioco**:

**Theorem 1** *For two pIOTS $\mathcal{I}$ and $\mathcal{S}$ with $\mathcal{I}$ input-enabled, we have $\mathcal{I} \sqsubseteq^*_{ioco} \mathcal{S} \Leftrightarrow \mathcal{I} \sqsubseteq_{pioco} \mathcal{S}$.*

***Proof sketch*** All three relations are defined in the same way over trace distributions and schedulers, the notions for which coincide if $T_M = \varnothing$ or $\mathcal{C} = \varnothing$, respectively. □

Consequently, the relationships already established between **pioco** and other relations in [20] carry over as well: **mar-ioco** and **sa-ioco** extend **ioco** (i.e. the relations coincide on IOTS), and for trace distribution inclusion, we have the following result:

**Theorem 2** *Let $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ be automata and let $\mathcal{A}$ and $\mathcal{B}$ be input-enabled, then*

$$\mathcal{A} \sqsubseteq^*_{ioco} \mathcal{B} \;\Leftrightarrow\; \mathcal{A} \sqsubseteq^{fin}_{TD} \mathcal{B}$$
$$and \mathcal{A} \sqsubseteq^*_{ioco} \mathcal{B} \wedge \mathcal{B} \sqsubseteq^*_{ioco} \mathcal{C} \;\Rightarrow\; \mathcal{A} \sqsubseteq^*_{ioco} \mathcal{C}.$$

***Proof sketch*** The fact that finite trace distribution inclusion implies conformance with respect to $\sqsubseteq^*_{ioco}$ is immediate if we consider that the relation is defined via trace distributions. The opposite direction follows from the fact that all abstract traces of $\mathcal{A}$ ending in output assuredly can get assigned the same probabilities in $\mathcal{B}$ by $\sqsubseteq^*_{ioco}$. All abstract traces ending in input are taken care of because $\mathcal{A}$ and $\mathcal{B}$ are input-enabled, and all such distributions are input-reactive. The second result is a direct consequence of the first. □

## 4.2 Test cases and annotations

The advantage of MBT over manual testing is that test cases can be automatically generated from the specification and automatically executed on an implementation. We are interested in the result of a parallel composition of a test case and an implementation model. We define test cases over an action signature $\langle Act_I, Act_O\rangle$. A test case is a collection of traces that represent the possible behaviour of a tester. It is summarised by an IOMA without Markovian transitions, or an IOSA without clocks, whose graph is a tree. The action signature describes the potential interaction with the implementation. In each state/location, the test may either stop, wait for a response of the system, or provide some stimulus. When a test is waiting for a response, it has to take into account all potential outputs including the situation that the system provides no response at all, modelled by quiescence $\delta$. A single test case may provide multiple options, giving rise to multiple concrete testing sequences. It may also prescribe different reactions to different outputs.

**Definition 16** (*test case, test suite*) A *test case* over an action signature $\langle Act_I, Act_O\rangle$ of system inputs $Act_I$ and system outputs $Act_O$ is an IOMA

$$\mathfrak{t} = \langle S, s_0, Act^{\mathfrak{t}}, T_P, \varnothing\rangle$$

or an IOSA

$$\mathfrak{t} = \langle Loc, \ell_0, \varnothing, Act^{\mathfrak{t}}, E, \varnothing\rangle$$

where $Act^{\mathfrak{t}} = Act^{\mathfrak{t}}_I \uplus Act^{\mathfrak{t}}_O$ with inputs $Act^{\mathfrak{t}}_I = Act_O \cup \{\delta\}$ and outputs $Act^{\mathfrak{t}}_O = Act_I \setminus \{\delta\}$ that is a finite, internally deterministic, and connected tree. In addition, all discrete distributions of the transitions or edges must be Dirac, and for every state or location $s$ we require that either

(1) $enabled(s) = \emptyset$ (stop the test) or
(2) $enabled(s) = Act^{\mathfrak{t}}_I$ (wait for some response) or
(3) $enabled(s) \subseteq Act^{\mathfrak{t}}_O \wedge |enabled(s) = 1|$ (provide a single stimulus, deterministically).

A *test suite* $\mathfrak{T}$ is a set of test cases. A test case (suite) for an automaton $\mathcal{S}$ with inputs $Act_I$ and outputs $Act_O$ is a test case (suite) if it is defined over action signature $\langle Act_I, Act_O\rangle$ and if we additionally require in item 3 above that, if a transition or edge labelled $a \in Act^{\mathfrak{t}}_O$ can lead to state or location $s'$ with positive probability, then there exists a $\sigma \in traces(\mathcal{S})$ such that $\sigma\ .\ t\ a \in traces(\mathcal{S})$ for some $t \in \mathbb{R}^+_0$.

Test cases are, in effect, IOMA or IOSA that are IOTS. The *inputs* of a test case are the *outputs* of the action signature, i.e. the outputs of the implementation or specification, and vice versa. The last requirement in the definition ensures that only specified inputs are provided: a test may only judge the correctness of specified behaviour. This is referred to as being *input minimal* in the literature [47].

In order to identify the behaviour which we deem as functionally acceptable/correct, each complete trace of a test, i.e. every leaf state or location, is annotated with a *pass* or *fail* verdict. We annotate exactly the traces that are present in the specification with the *pass* verdict, formally:

**Definition 17** (*test annotation*) For a test $\mathfrak{t}$, a *test annotation* is a function

$$ann \in traces^{com}(\mathfrak{t}) \to \{pass, fail\}.$$

A pair $\hat{\mathfrak{t}} = \langle \mathfrak{t}, ann\rangle$ consisting of a test and a test annotation is an *annotated test*. The set of all such $\hat{\mathfrak{t}}$, denoted by $\hat{\mathfrak{T}} = \{(t_i, ann_i)_{i\in\mathcal{I}}\}$ for some index set $\mathcal{I}$, is an *annotated test suite*. If $\mathfrak{t}$ is a test case for a specification $\mathcal{S}$ with signature $\langle Act_I, Act_O\rangle$, we define

$$ann^{\mathcal{S}}_{*\text{-}ioco} \in traces^{com}(\mathfrak{t}) \to \{pass, fail\}$$

by $ann^{\mathcal{S}}_{*\text{-}ioco}(\sigma) = fail$ if there exist $\rho \in traces^{fin}(\mathcal{S}), t \in \mathbb{R}^+_0$ and $a \in Act_O$ such that

$$\rho\ .\ t\ a \sqsubseteq \sigma \ \wedge\ \rho\ .\ t\ a \notin traces^{fin}(\mathcal{S})$$

and $ann^{\mathcal{S}}_{*\text{-}ioco}(\sigma) = pass$ otherwise.

Annotations decide functional correctness only. The correctness of discrete probabilistic choices and stochastic delays is assessed in a separate second step.

***Example 5*** Figure 6 presents a test suite for the file server specification IOSA of Fig. 2. Test case $\hat{\mathfrak{t}}_1$ uses the quiescence observation $\delta$ to assure no output is given in the initial state. $\hat{\mathfrak{t}}_2$ checks for *eventual* delivery of the file, which may be archived, requiring the intermediate `wait!` notification, or may be sent directly. Finally, $\hat{\mathfrak{t}}_3$ tests the `abort?` edge.

### 4.3 Sampling and verdicts

Functional conformance is assessed via test annotations in the same way as in classical ioco theory [47]. However, we test stochastic systems; thus, executing a test case once is insufficient to establish ***-ioco** conformance. We now focus on the statistical evaluation of the probabilistic and stochastic-timed behaviour based on a *sample* of multiple traces.

#### 4.3.1 Sampling

We perform a statistical hypothesis test on the implementation based on the outcome of a push-button experiment in the sense of [37]. We assume a black-box timed trace machine with inputs, a time and an action window, and a reset button, as illustrated in Fig. 7. An observer records each individual execution before the reset button is pressed and a new execution starts. A clock that increases is started, and is stopped once the next visible action is recorded. We assume

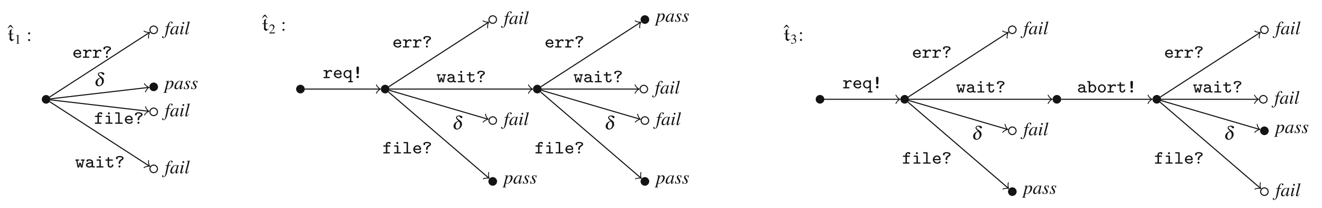


**Fig. 6** Three test cases for the file server specification

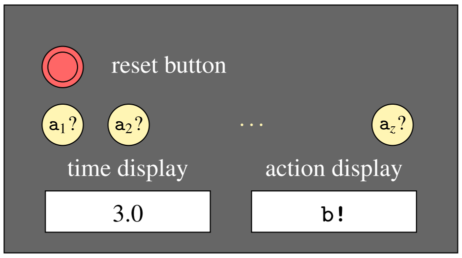


**Fig. 7** Black-box timed trace machine

that recording an action resets the clock. Thus, the recordings of the external observer match the notion of (abstract) traces. After a sample of sufficient size has been collected, we compare the collected frequencies of abstract traces to their expected frequencies according to the specification. If the empiric observations are close to the expectations, we accept the probabilistic behaviour of the implementation.

Before the experiment, we fix the parameters for sample length $k \in \mathbb{N}$ (the length of the individual test executions), sample size $m \in \mathbb{N}$ (how many test executions to observe), and level of significance $\alpha \in ]0, 1[$ (the probability of erroneously rejecting a correct implementation). Checking the abstract trace frequencies contained in the sample versus their expectancy w.r.t. the specification $\mathcal{S}$ requires a scheduler due to the presence of nondeterminism in $\mathcal{S}$. In order for any statistical reasoning to work, we assume each iteration of the sampling process to be governed by the same scheduler, which induces a trace distribution $\mathcal{T} \in trd(\mathcal{I})$.

### 4.3.2 Frequencies and expectations

To quantify how *close* a sample is to its expectations, we require a notion of *distance*. Our goal is to evaluate the deviation of a collected sample to the expected distribution. Thus, we require (1) a metric space for the quantification of distances between measures, (2) the frequency measure of abstract traces in a sample, and (3) the expected measure of abstract traces in the specification under $\mathcal{T}$.

For automaton $\mathcal{A}$, we use metric space $\langle \mathrm{Meas}(\mathcal{A}), dist\rangle$ where the metric

$$dist(u, v) \overset{\text{def}}{=} \sup_{\sigma \in (\mathbb{R}_0^+ \times Act)^{\leq k}} |u(\Sigma) - v(\Sigma)|$$

is the maximal variation distance of two measures $u$ and $v$. (Recall we denote by $\Sigma$ the abstract trace corresponding to the trace $\sigma$.) We next define the two measures—the *frequency measure* for a sample $O = \{\!| \sigma_1, \ldots, \sigma_m |\!\}$ and the *expected measure* according to the specification—that need to be compared. Our definitions for the former differ between IOMA and IOSA due to their different models of stochastic time.

**Memoryless time** For IOMA, our frequency measure can assume the independence of all time intervals since the delays are memoryless. Thus, we order the $i$-th time intervals of all $\rho$ increasingly and compare them to $\sigma$. We achieve this by grouping traces into classes based on the same visible action behaviour. For a given trace $\sigma$, its class $\Sigma_\sigma$ is the set of all traces $\rho \in O$ such that $act(\rho) = act(\sigma)$. A sample of length $k$ and width $m$ then induces the frequency measure

$$freq \in ((\mathbb{R}_0^+ \times \mathbb{R}_0^+) \times Act)^{\leq k \times m} \to \mathrm{Meas}((\mathbb{R}_0^+ \times Act)^{\leq k})$$

defined by

$$freq(O)(\Sigma) = \frac{|\Sigma_\sigma|}{m} \prod_{i=1}^{k} \frac{|\{\!| \rho \in \Sigma_\sigma \mid t_i^\rho \leq t_i^\sigma |\!\}|}{|\Sigma_\sigma|}$$

where $t_i^\rho$ denotes the $i$-th time stamp of trace $\rho$. In this way, the distributions for each time stamp in a trace converge to the true underlying distribution by the Glivenko–Cantelli theorem [22].

**General stochastic time** For IOSA, we define the frequency measure by

$$freq(O)(\Sigma) = \frac{|\{\!| \rho \in O \mid \forall i : t_i^\rho \in I_i^\Sigma |\!\}|}{m},$$

i.e. the fraction of traces in $O$ that are in $\Sigma$. Specifically, we require all time stamps to be contained in the intervals given in $\Sigma$. In contrast to IOMA, this function does *not* assume the independence of clock valuations from locations.

**Expected measure** The last missing ingredient is the expected measure according to a specification. Let $\mathcal{T}$ be the trace distribution resulting from the resolution of all nondeterministic choices. We treat each iteration of the sampling process of the implementation as Bernoulli trial. Recall that a Bernoulli trial has two outcomes: *success* with probability $p$ and *failure* with probability $1 - p$. For any trace $\sigma$, we say that success occurred at position $i$ of the sample if $\sigma = \sigma_i$. Therefore, let $X_i \sim Ber(P_{\mathcal{T}}(\Sigma))$ be Bernoulli distributed random variables for $i = 1, \dots, m$. Let $Z = \frac{1}{m}\Sigma_{i=1}^m X_i$ be the empiric mean with which we observe $\sigma$ in a sample. The expected probability under $\mathcal{T}$ is then calculated as

$$\mathbb{E}^{\mathcal{T}}(Z) = \mathbb{E}^{\mathcal{T}}\left(\frac{1}{m}\Sigma_{i=1}^m X_i\right) = \frac{1}{m}\Sigma_{i=1}^m \mathbb{E}^{\mathcal{T}}(X_i) = P_{\mathcal{T}}(\Sigma).$$

Hence, the expected probability for each abstract trace $\Sigma$ is the probability of $\Sigma$ under trace distribution $\mathcal{T}$, as expected.

***Example 6*** Returning to the example of

$$\sigma_1 = 0.5\, \texttt{a?}\ 0.6\, \texttt{b!} \quad \text{and} \quad \sigma_2 = 0.6\, \texttt{a?}\ 0.5\, \texttt{b!},$$

assume $O = \{\sigma_1, \sigma_2\}$. Then,

$$\begin{aligned}
freq(O)([0, 0.5]\, \texttt{a?}\, [0, 0.5]\, \texttt{b!}) &= \frac{2}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4},\\
freq(O)([0, 0.5]\, \texttt{a?}\, [0, 0.6]\, \texttt{b!}) &= \frac{2}{2} \cdot \frac{1}{2} \cdot \frac{2}{2} = \frac{1}{2},\\
freq(O)([0, 0.6]\, \texttt{a?}\, [0, 0.6]\, \texttt{b!}) &= \frac{2}{2} \cdot \frac{2}{2} \cdot \frac{2}{2} = 1.
\end{aligned}$$

### 4.3.3 Acceptable outcomes

We accept a sample $O$ if $freq(O)$ lies within some distance $r_\alpha$ of the expected measure $\mathbb{E}^{\mathcal{T}}$. All measures deviating at most $r_\alpha$ from the expected measures are contained within the ball $B_{r_\alpha}(\mathbb{E}^{\mathcal{T}})$. The actual $r_\alpha$ is chosen such that the error of accepting an erroneous sample is limited while keeping the error of rejecting a correct sample smaller than $\alpha$, i.e.

$$r_\alpha = \inf\{r \in \mathbb{R}_0^+ \mid P_{\mathcal{T}}(freq^{-1}(B_r(\mathbb{E}^{\mathcal{T}}))) \geq 1 - \alpha\}.$$

**Definition 18** (*acceptable outcomes*) For $k, m \in \mathbb{N}$ and an automaton $\mathcal{A}$, the set of *acceptable outcomes* under $\mathcal{T} \in trd(\mathcal{A}, k)$ of significance level $\alpha \in (0, 1)$ is $Obs(\mathcal{T}, \alpha, k, m) =$

$$\{\, O \in (\mathbb{R}_0^+ \times Act)^{\leq k \times m} \mid dist(freq(O), \mathbb{E}^{\mathcal{T}}) \leq r_\alpha \,\}.$$

We obtain the set of acceptable outcomes of $\mathcal{A}$ by

$$Obs(\mathcal{A}, \alpha, k, m) = \bigcup_{\mathcal{T} \in trd(\mathcal{A}, k)} Obs(\mathcal{T}, \alpha, k, m).$$

The set of acceptable outcomes consists of all possible samples that we are willing to accept as *close enough* to the expectations. Note that this takes *all possible* trace distributions of $\mathcal{A}$ into consideration. The set of acceptable outcomes has two properties reflecting the error of false rejection and the error of false acceptance, respectively: first, if a sample was generated under a trace distribution of $\mathcal{A}$ or a trace distribution-equivalent automaton, we correctly accept it with probability higher than $1 - \alpha$, i.e.

$$P_{\mathcal{T}}(Obs(\mathcal{T}, \alpha, k, m)) \geq 1 - \alpha;$$

second, if a sample was generated by a non-admitted trace distribution, the chance of erroneously accepting it is smaller than some $\beta_m$. Again, $\alpha$ is the *a priori* defined level of significance, and $\beta_m$ is unknown, but minimal by construction. Additionally, $\beta_m \to 0$ as $m \to \infty$: the error of falsely accepting an observation decreases with increasing sample size.

***Remark 2*** The set of acceptable outcomes comprises samples of the form $O \in (\mathbb{R}_0^+ \times Act)^{\leq k \times m}$. In order to align observations with the ***-ioco** relations, we define the set of acceptable *output* outcomes $OutObs(\mathcal{T}, \alpha, k, m)$ as the set of those $O \in ((\mathbb{R}_0^+ \times Act)^{\leq k-1} \times \mathbb{R}_0^+ \times Act_O)^m$ for which we have $dist(freq(O), \mathbb{E}^{\mathcal{T}}) \leq r_\alpha$.

**Verdict functions** With all necessary components in place, the following decision process summarises whether an implementation fails a test case or test suite based on a functional or statistical verdict. The overall *pass* verdict is given iff both sub-verdicts yield a *pass*. Let $Aut_*$ denote the set of all IOMA or IOSA, respectively.

**Definition 19** (*verdicts*) Given a specification automaton $\mathcal{S}$, an annotated test $\hat{\mathfrak{t}}$ for $\mathcal{S}$, $k, m \in \mathbb{N}$ where $k$ is the length of the longest trace of $\hat{\mathfrak{t}}$, and $\alpha \in (0, 1)$, we define the *functional verdict* as the function

$$v_{func} \in Aut_* \times Aut_* \to \{pass, fail\}$$

with $v_{func}(\mathcal{I}, \hat{\mathfrak{t}}) = pass$ if

$$\forall \sigma \in traces^{com}(\mathcal{I} \,\|\, \hat{\mathfrak{t}}) \colon ann^{\mathcal{S}}_{*\text{-}ioco}(\sigma) = pass$$

and $v_{func}(\mathcal{I}, \hat{\mathfrak{t}}) = fail$ otherwise, the *statistical verdict* as

$$v_{prob} \in Aut_* \times Aut_* \to \{pass, fail\}$$

with $v_{prob}(\mathcal{I}, \hat{\mathfrak{t}}) = pass$ if for all $\mathcal{T} \in trd(\mathcal{I} \,\|\, \hat{\mathfrak{t}})$ there exists a $\mathcal{T}' \in trd(\mathcal{S}, k)$ such that

$$P_{\mathcal{T}'}(OutObs(\mathcal{T}, \alpha, k, m)) \geq 1 - \alpha$$

and $v_{prob}(\mathcal{I}, \hat{\mathfrak{t}}) = \mathit{fail}$ otherwise, and the *overall verdict* as

$$V \in Aut_* \times Aut_* \rightarrow \{pass, \mathit{fail}\}$$

with $V(\mathcal{I}, \hat{\mathfrak{t}}) = \begin{cases} pass & \text{if } v_{func}(\mathcal{I}, \hat{\mathfrak{t}}) = v_{prob}(\mathcal{I}, \hat{\mathfrak{t}}) = pass \\ \mathit{fail} & \text{otherwise.} \end{cases}$

An implementation passes a test suite $\hat{\mathfrak{T}}$ if it passes the overall verdict for all annotated tests $\hat{\mathfrak{t}} \in \hat{\mathfrak{T}}$.

Although IOMA and IOSA include three properties in terms of (1) functional behaviour, (2) discrete probabilistic behaviour, and (3) continuous time, we only have two verdicts. This is because continuous time is only present in the form of stochastic delays. Thus, on the purely mathematical level, the decision whether or not a delay in the implementation adheres to the one specified is covered by the probabilistic verdict $v_{prob}$. Only on the practical side of things do we need a new decision procedure. We study this in Sect. 5.

## 4.4 Soundness and completeness

Ideally, only ***-ioco**correct implementations pass a test suite. However, due to the stochastic nature of our models, there remains a degree of uncertainty upon giving verdicts. This is phrased as *errors of first and second kind* in hypothesis testing: the probability to reject a true hypothesis and to accept a false one, respectively. They are reflected as the probability to reject a correct implementation and to accept an erroneous one in the context of probabilistic MBT. The relevance of these errors becomes evident when we consider the correctness of our test frameworks. Correctness comprises *soundness* and *completeness*: every conforming implementation passes, and there is a test case to expose every non-conforming one. A test suite can only be considered correct with some guaranteed (high) probability.

**Definition 20** (*sound, complete*) Let $\mathcal{S}$ be a specification automaton over action signature $\langle Act_I, Act_O \rangle$, $\alpha \in ]0, 1[$ the level of significance, and $\hat{\mathfrak{T}}$ an annotated test suite for $\mathcal{S}$. Then, $\hat{\mathfrak{T}}$ is *sound* for $\mathcal{S}$ with respect to $\sqsubseteq^*_{ioco}$ if, for all input-enabled automata $\mathcal{I}$ and sufficiently large $m \in \mathbb{N}$, it holds for all $\hat{\mathfrak{t}} \in \hat{\mathfrak{T}}$ that

$$\mathcal{I} \sqsubseteq^*_{ioco} \mathcal{S} \;\Rightarrow\; V(\mathcal{I}, \hat{\mathfrak{t}}) = pass.$$

$\hat{\mathfrak{T}}$ is *complete* for $\mathcal{S}$ with respect to $\sqsubseteq^*_{ioco}$ if, for all input-enabled automata $\mathcal{I}$ and sufficiently large $m \in \mathbb{N}$, there is at least one $\hat{\mathfrak{t}} \in \hat{\mathfrak{T}}$ such that

$$\mathcal{I} \not\sqsubseteq^*_{ioco} \mathcal{S} \;\Rightarrow\; V(\mathcal{I}, \hat{\mathfrak{t}}) = \mathit{fail}.$$

Soundness expresses for a given $\alpha \in ]0, 1[$ that there is a $1-\alpha$ chance that a correct system passes the annotated test suite for sufficiently large sample size $m$. This relates to false rejection of a correct hypothesis in statistical hypothesis testing, or rejection of a correct implementation, respectively.

For the following theorems, we provide full proofs for **sa-ioco**. The proofs for **mar-ioco** use the exact same arguments and only lack some of the technical complications of the more general IOSA setting. The interested reader may find the full proofs for **mar-ioco** in [18].

**Theorem 3** *Each annotated test case for an automaton $\mathcal{S}$ is sound for every level of significance $\alpha \in (0, 1)$ with respect to $\sqsubseteq^*_{ioco}$.*

***Proof*** Let $\mathcal{I}$ be an input-enabled IOSA and $\hat{\mathfrak{t}}$ be a test for $\mathcal{S}$. Assume that $\mathcal{I} \sqsubseteq^{sa}_{ioco} \mathcal{S}$. We want to show $V(\mathcal{I}, \hat{\mathfrak{t}}) = pass$. By Definition 19, we have that $V(\mathcal{I}, \hat{\mathfrak{t}}) = pass$ if and only if $v_{func}(\mathcal{I}, \hat{\mathfrak{t}}) = v_{prob}(\mathcal{I}, \hat{\mathfrak{t}}) = pass$. We proceed by showing $v_{func}(\mathcal{I}, \hat{\mathfrak{t}}) = pass$ and $v_{prob}(\mathcal{I}, \hat{\mathfrak{t}}) = pass$ in separate steps:

**Functional verdict** By Definition 19, we need to show that

$$ann^{\mathcal{S}}_{sa\text{-}ioco}(\sigma) = pass \text{ for all } \sigma \in traces^{com}(\mathcal{I} \,\|\, \hat{t}).$$

Let $\sigma \in traces^{com}(\mathcal{I} \parallel \hat{t})$ and use Definition 17. Assume $\sigma' \in traces^{fin}(\mathcal{S})$ and $a \in Act_O$ such that $\sigma'.\,t\,a \sqsubseteq \sigma$ for some $t \in \mathbb{R}^+_0$. We observe that (a) since the empty trace is a trace and is in $traces^{fin}(\mathcal{S})$, $\sigma'$ always exists, and (b) if no such $a \in Act_O$ exists, then $\sigma$ only consists of inputs, and by Definition 17 consequently $ann^{\mathcal{S}}_{sa\text{-}ioco}(\sigma) = pass$. By construction of $\sigma$, we have $\sigma'.\,t\,a \in traces^{fin}(\mathcal{I} \parallel \hat{\mathfrak{t}})$ and therefore also $\sigma'.\,t\,a \in traces^{fin}(\mathcal{I})$. In particular, the parallel composition with a test case does not alter the guard sets on edges. We conclude that $\sigma' \in traces^{fin}(\mathcal{I}) \cap traces^{fin}(\mathcal{S})$. Our goal is to show $\sigma'.\,t\,a \in traces^{fin}(\mathcal{S})$.

Let $l = |\sigma'|$ be the length of $\sigma'$. W.l.o.g. we can now choose $\mathcal{T} \in trd(\mathcal{S}, l)$ such that $P_{\mathcal{T}}(\Sigma') > 0$. In particular, this choice is not invalidated by urgent transitions. If a transition has a guard set with a clock that can never expire in a location due to another urgent output, then this transition is never part of a path (Definition 6). With the previous observation, this yields $outcont_{\mathcal{I}}(\mathcal{T}) \neq \varnothing$. Again, w.l.o.g. we choose $\mathcal{T}' \in outcont_{\mathcal{I}}(\mathcal{T})$ such that $P_{\mathcal{T}'}(\Sigma'.\,[0, t]\,a) > 0$. Finally, we assumed $\mathcal{I} \sqsubseteq^{sa}_{ioco} \mathcal{S}$; hence,

$$outcont_{\mathcal{I}}(\mathcal{T}) \subseteq outcont_{\mathcal{S}}(\mathcal{T}).$$

We conclude $\mathcal{T}' \in trd(\mathcal{S}, l+1)$ and $P_{\mathcal{T}'}(\Sigma'.\,[0, t]\,a) > 0$. By Definition 13, this implies $\sigma'.\,t\,a \in traces^{fin}(\mathcal{S})$. If additionally $\sigma'.\,t\,a \in traces^{com}(\mathcal{I} \parallel \hat{\mathfrak{t}})$, then $\sigma = \sigma'.\,t\,a$. Consequently, $ann^{\mathcal{S}}_{sa\text{-}ioco}(\sigma) = pass$ by Definition 17 and $v_{func}(\mathcal{I}, \hat{\mathfrak{t}}) = pass$.

**Statistical verdict** By Definition 19, we must show that for all $\mathcal{T} \in trd(\mathcal{I} \,\|\, \hat{\mathfrak{t}}, k)$ there exists a $\mathcal{T}' \in trd(\mathcal{S}, k)$ such that

$$P_{\mathcal{T}'}\left(OutObs\left(\mathcal{T}, \alpha, k, m\right)\right) \geq 1 - \alpha.$$

Let $\mathcal{T} \in trd(\mathcal{I} \,\|\, \hat{\mathfrak{t}}, k)$. By Remark 2, $OutObs(\mathcal{T}, \alpha, k, m)$ is the set of all $O \in ((\mathbb{R}_0^+ \times Act)^{\leq k-1} \times \mathbb{R}_0^+ \times Act_O)^m$ such that $dist(freq(O), \mathbb{E}^{\mathcal{T}}) \leq r_\alpha$. There exists $\mathcal{T} \in trd(\mathcal{I} \,\|\, \hat{\mathfrak{t}}, k)$ with

$$P_{\mathcal{T}'}(\Sigma) = \begin{cases} 0 & \text{if } \sigma \in (\mathbb{R}_0^+ \times Act)^{k-1} \times \mathbb{R}_0^+ \times Act_I \\ P_{\mathcal{T}}(\Sigma) & \text{if } \sigma \in (\mathbb{R}_0^+ \times Act)^{\leq k-1} \times \mathbb{R}_0^+ \times Act_O. \end{cases} \tag{1}$$

To see why, consider the scheduler that assigns all probability to halting instead of inputs for traces of length $k$ while assigning the same probability to outputs as the scheduler of $\mathcal{T}$. By construction of *OutObs* (Remark 2), observe that

$$\begin{aligned} P_{\mathcal{T}'}(OutObs(\mathcal{T}', \alpha, k, m)) &= P_{\mathcal{T}'}(OutObs(\mathcal{T}, \alpha, k, m)) \\ &= P_{\mathcal{T}}(OutObs(\mathcal{T}, \alpha, k, m)) \\ &\geq 1 - \alpha \end{aligned}$$

since only traces ending in output are measured.

It is now sufficient to show that $\mathcal{T}' \in trd(\mathcal{S}, k)$. As an intermediate step, we first show that $\mathcal{T}' \in trd(\mathcal{I}, k)$, as this will let us make use of the assumption $\mathcal{I} \sqsubseteq_{ioco}^{sa} \mathcal{S}$. Consider the mapping

$$f \in paths^{fin}(\mathcal{I} \,\|\, \hat{\mathfrak{t}}) \longrightarrow paths^{fin}(\mathcal{I})$$

where for every fragment of the path we have

$$\begin{aligned} &f(\ldots \langle\langle \ell, q\rangle, v_0, x_0\rangle \langle t_1, e_1, R_1, |\langle \ell, q\rangle, v_1, x_1\rangle\rangle \ldots) \\ &\quad = \ldots \langle \ell, \bar{v}_0, \bar{x}_0\rangle \langle \bar{t}_1, \bar{e}_1, \bar{R}_1 \langle \ell', \bar{v}_1, \bar{x}_1\rangle\rangle \ldots \end{aligned}$$

This is possible because test cases do not contain clocks and parallel composition thus does not change guard sets, restart sets, or expiration times (Definition 4) and implies $v_i = \bar{v}_i \wedge x_i = \bar{x}_i$ for $i = 0, 1$ and $t_1 = \bar{t}_1 \wedge R_1 = \bar{R}_1$. For $\bar{e}_1$ consider $g \in E_{\mathcal{I}\|\hat{\mathfrak{t}}} \rightarrow E_{\mathcal{I}}$ such that

$$g(e) = g(C, a, \mu(R, (\ell, q))) = (C, a, \bar{\mu}(R, \ell)) = \bar{e}$$

where $\mu(R, \langle \ell, q\rangle) = \bar{\mu}(R, \ell)$ for all $\ell$. This construction of $\mu$ is possible because tests only contain Dirac distributions and discrete probabilities thus directly transfer. Hence, $q$ is uniquely determined by parallel composition. Since $\hat{\mathfrak{t}}$ is internally deterministic, $f$ is an injective mapping, i.e.

$$f(\pi_1) = f(\pi_2) \Rightarrow \pi_1 = \pi_2.$$

By Definition 13, there is a scheduler $\mathfrak{S}' \in Sched(\mathcal{I} \,\|\, \hat{\mathfrak{t}})^{\leq k}$ such that $trd(\mathfrak{S}') = \mathcal{T}'$. With the help of $f$, we show the existence of a scheduler $\mathfrak{S}'' \in Sched(\mathcal{I})$ such that for all traces $\sigma$ we have $P_{trd(\mathfrak{S}')}(\Sigma) = P_{trd(\mathfrak{S}'')}(\Sigma)$, i.e. $trd(\mathfrak{S}'') = \mathcal{T}'$.

For every path $\pi \in paths^{fin}(\mathcal{I})$ with

$$f^{-1}(\pi) \in paths^{fin}(\mathcal{I} \,\|\, \hat{\mathfrak{t}}),$$

we define $\mathfrak{S}''$ as $\mathfrak{S}''(\pi)(\bar{e}) \stackrel{\text{def}}{=} \mathfrak{S}'(f^{-1}(\pi))(e)$. $P_{\mathfrak{S}''}(\Pi) = 0$ if $\pi \notin paths^{fin}(\mathcal{I} \,\|\, \hat{\mathfrak{t}})$. The construction of $\mathfrak{S}''$ is straightforward: due to the construction of test cases, $\mathcal{I} \,\|\, \hat{\mathfrak{t}}$ is internally deterministic. In particular, there is no interleaving. This means that $\mathfrak{S}''$ can copy the behaviour of $\mathfrak{S}'$ step by step. We set $\mathcal{T}'' = trd(\mathfrak{S}'')$ and conclude $\mathcal{T}'' \in trd(\mathcal{I}, k)$. By construction $P_{\mathcal{T}''}(\Sigma) = P_{\mathcal{T}'}(\Sigma)$ for all traces $\sigma$. Further,

$$\begin{aligned} P_{\mathcal{T}''}(OutObs(\mathcal{T}'', \alpha, k, m)) &= P_{\mathcal{T}''}(OutObs(\mathcal{T}', \alpha, k, m)) \\ &= P_{\mathcal{T}''}(OutObs(\mathcal{T}, \alpha, k, m)) \\ &= P_{\mathcal{T}'}(OutObs(\mathcal{T}, \alpha, k, m)) \\ &= P_{\mathcal{T}}(OutObs(\mathcal{T}, \alpha, k, m)) \\ &\geq 1 - \alpha. \end{aligned}$$

We proceed to show that $\mathcal{T}'' \in trd(\mathcal{S}, k)$. The proof is by induction over trace distribution length of prefixes of $\mathcal{T}''$ up to $k$. Trivially, if $\mathcal{T}'' \in trd(\mathcal{I}, 0)$, then also $\mathcal{T}'' \in trd(\mathcal{S}, 0)$. Assume this has been shown for length $n$. We proceed by showing that the statement holds for $n + 1 \leq k$. Let $\mathcal{T}'' \in trd(\mathcal{I}, n + 1)$ and take $\mathcal{T}''' \sqsubseteq_n \mathcal{T}''$. By induction assumption $\mathcal{T}''' \in trd(\mathcal{S}, n)$. Together with $\mathcal{I} \sqsubseteq_{ioco}^{sa} \mathcal{S}$, we have

$$outcont_{\mathcal{I}}(\mathcal{T}''') \subseteq outcont_{\mathcal{S}}(\mathcal{T}''').$$

Since $\mathcal{T}'' \in outcont_{\mathcal{I}}(\mathcal{T}''')$ (Eq. 1), we also have that $\mathcal{T}'' \in outcont_{\mathcal{S}}(\mathcal{T}''')$, and consequently $\mathcal{T}'' \in trd(\mathcal{S}, n + 1)$. We showed $\mathcal{T}'' \in trd(\mathcal{S}, k)$ and conclude

$$P_{\mathcal{T}''}(OutObs(\mathcal{T}, \alpha, k, m)) \geq 1 - \alpha.$$

Ultimately, this yields $v_{prob}(\mathcal{I}, \hat{\mathfrak{t}}) = pass$ by Definition 19). □

Completeness of a test suite is an inherently theoretical result. Infinite behaviour of the implementation, for instance, via loops, would require an infinite test suite. Moreover, the possibility of accepting an erroneous implementation by chance, i.e. committing an error of the second kind, remains. However, the latter is bounded from above by construction, and decreases with increasing sample size (Definition 18).

**Theorem 4** *The set of all annotated test cases for an automaton $\mathcal{S}$ is complete for every level of significance $\alpha \in (0, 1)$ with respect to $\sqsubseteq_{ioco}^{sa}$ for sufficiently large sample size.*

***Proof*** Assume $\mathcal{I} \not\sqsubseteq_{ioco}^{sa} \mathcal{S}$. We want to show that $V(\mathcal{I}, \hat{\mathfrak{T}}) = \mathit{fail}$. By the definition of verdicts (Definition 19), this is the case iff $v_{func}(\mathcal{I}, \hat{\mathfrak{t}}) = \mathit{fail}$ or $v_{prob}(\mathcal{I}, \hat{\mathfrak{t}}) = \mathit{fail}$ for some $\hat{\mathfrak{t}} \in \hat{\mathfrak{T}}$. Since $\mathcal{I} \not\sqsubseteq_{ioco}^{sa} \mathcal{S}$, there is a $k \in \mathbb{N}$ such that there is a $\mathcal{T}^* \in \mathit{trd}(\mathcal{S}, k)$ for which $\mathit{outcont}_{\mathcal{I}}(\mathcal{T}^*) \not\subseteq \mathit{outcont}_{\mathcal{S}}(\mathcal{T}^*)$. More specifically, there exists a $\mathcal{T} \in \mathit{outcont}_{\mathcal{I}}(\mathcal{T}^*)$ such that

$$\forall \mathcal{T}' \in \mathit{outcont}_{\mathcal{S}}(\mathcal{T}^*) \colon \exists \sigma \in \mathfrak{C} \colon P_{\mathcal{T}}(\Sigma) \neq P_{\mathcal{T}'}(\Sigma) \qquad (2)$$

where $\mathfrak{C} \overset{\text{def}}{=} \mathit{traces}^{\mathit{fin}}(\mathcal{I}) \cap (\mathbb{R}_0^+ \times \mathit{Act})^k \times \mathbb{R}_0^+ \times \mathit{Act}_O$ and $\Sigma$ is the abstract trace of $\sigma$. W.l.o.g. we can assume $k$ to be minimal. There are two cases to consider: (1) $\exists \sigma \in \mathfrak{C} : \sigma \notin \mathit{traces}^{\mathit{fin}}(\mathcal{S})$, or (2) $\forall \sigma \in \mathfrak{C} : \sigma \in \mathit{traces}^{\mathit{fin}}(\mathcal{S})$. We will relate the two cases to the functional and the probabilistic verdict (Definition 19): we prove that case 1 implies that $v_{func}(\mathcal{I}, \hat{\mathfrak{T}}) = \mathit{fail}$ and that case 2 implies $v_{prob}(\mathcal{I}, \hat{\mathfrak{T}}) = \mathit{fail}$. Now let $\mathcal{T} \in \mathit{outcont}_{\mathcal{I}}(\mathcal{T}^*)$ such that Eq. 2 holds for all $\mathcal{T}' \in \mathit{outcont}_{\mathcal{S}}(\mathcal{T}^*)$.

**Functional verdict** By Definition 19, we need to show

$$\exists \sigma \in \mathit{traces}^{\mathit{com}}(\mathcal{I} \,\|\, \hat{\mathfrak{t}}) \colon \mathit{ann}^{\mathcal{S}}_{\mathit{sa\text{-}ioco}}(\sigma) = \mathit{fail}$$

for some $\hat{\mathfrak{t}} \in \hat{\mathfrak{T}}$. Assume there is a $\sigma \in \mathfrak{C}$ such that $\sigma \notin \mathit{traces}^{\mathit{fin}}(\mathcal{S})$. Our goal is to show that there is $\hat{\mathfrak{t}} \in \hat{\mathfrak{T}}$ for which $\sigma \in \mathit{traces}^{\mathit{com}}(\mathcal{I} \,\|\, \hat{\mathfrak{t}})$ and $\mathit{ann}^{\mathcal{S}}_{\mathit{pioco}}(\sigma) = \mathit{fail}$.

Without loss of generality, we assume $P_{\mathcal{T}}(\Sigma) > 0$. To see why, assume $P_{\mathcal{T}}(\Sigma) = 0$. Then, we can find a trace distribution in $\mathit{outcont}_{\mathcal{S}}(\mathcal{T}^*)$ with an underlying scheduler $\mathit{Sched}(\mathcal{S})$ that does not assign positive probability to the last action in $\sigma$ to obtain overall probability zero. This violates the assumption that $P_{\mathcal{T}}(\Sigma) \neq P_{\mathcal{T}'}(\Sigma)$ for all $\mathcal{T}' \in \mathit{trd}(\mathcal{S})$. We conclude $\sigma = \sigma' .\, t\, a$, for some $\sigma' \in (\mathbb{R}_0^+ \times \mathit{Act})^k$, $a \in \mathit{Act}_O$ and $t \in \mathbb{R}_0^+$. The prefix $\sigma'$ is in $\mathit{traces}^{\mathit{fin}}(\mathcal{S})$ because it is of length $k$ and since $\mathcal{T}^* \in \mathit{trd}(\mathcal{S}, k)$. Since $\mathcal{T}$ and all $\mathcal{T}' \in \mathit{outcont}_{\mathcal{S}}(\mathcal{T}^*)$ are continuations of $\mathcal{T}^*$, we conclude that $P_{\mathcal{T}^*}(\Sigma') = P_{\mathcal{T}}(\Sigma') = P_{\mathcal{T}'}(\Sigma')$, i.e. that all trace distributions of the respective sets assign every prefix of $\sigma$ the same probability by merit of $\mathit{outcont}$. We conclude $\sigma' \in \mathit{traces}^{\mathit{fin}}(\mathcal{S})$, but $\sigma' .\, t\, a \notin \mathit{traces}^{\mathit{fin}}(\mathcal{S})$.

By initial assumption $\hat{\mathfrak{T}}$ contains *all* annotated test cases. Let $\hat{\mathfrak{t}} \in \hat{\mathfrak{T}}$ such that $\sigma \in \mathit{traces}^{\mathit{com}}(\hat{\mathfrak{t}})$. This is possible because $\sigma' \in \mathit{traces}^{\mathit{fin}}(\mathcal{S})$. By Definition 17, $\mathit{ann}^{\mathcal{S}}_{\mathit{sa\text{-}ioco}}(\sigma) = \mathit{fail}$. Recall that the set of clocks in test cases in empty. Since $\sigma \in \mathit{traces}^{\mathit{fin}}(\mathcal{I})$ and $\sigma \in \mathit{traces}^{\mathit{com}}(\hat{\mathfrak{t}})$, we consequently also have $\sigma \in \mathit{traces}^{\mathit{com}}(\mathcal{I} \,\|\, \hat{\mathfrak{t}})$ as no guard or restart sets are changed under parallel composition with a test case. Ultimately, this yields $v_{func}(\mathcal{I}, \hat{\mathfrak{t}}) = \mathit{fail}$.

**Statistical verdict** By Definition 19, we must show that there is $\mathcal{T} \in \mathit{trd}(\mathcal{I} \,\|\, \hat{\mathfrak{t}}, l)$ such that for all $\mathcal{T}' \in \mathit{trd}(\mathcal{S}, l)$ we have

$$P_{\mathcal{T}'}(\mathit{OutObs}(\mathcal{T}, \alpha, l, m)) < 1 - \alpha,$$

for some $\hat{\mathfrak{t}} \in \hat{\mathfrak{T}}$ and some $l \in \mathbb{N}$.

Together with Eq. 2 and Definition 18, we conclude that for all $\mathcal{T}' \in \mathit{outcont}_{\mathcal{S}}(\mathcal{T}^*)$ we have

$$P_{\mathcal{T}'}(\mathit{OutObs}(\mathcal{T}, \alpha, k+1, m)) < \beta_m \qquad (3)$$

for some $\beta_m \to 0$ as $m \to \infty$. Observe that

$$\begin{aligned} &\sup_{\mathcal{T}' \in \mathit{trd}(\mathcal{S}, k+1)} P_{\mathcal{T}'}(\mathit{OutObs}(\mathcal{T}, \alpha, k+1, m)) \\ &= \sup_{\mathcal{T}' \in \mathit{outcont}_{\mathcal{S}}(\mathcal{T}^*)} P_{\mathcal{T}'}(\mathit{OutObs}(\mathcal{T}, \alpha, k+1, m)), \end{aligned} \qquad (4)$$

by Remark 2. *OutObs* only comprises traces ending in output; thus, its measure under any trace distribution of $\mathit{trd}(S, k+1)$ cannot be larger than the measure of the ones already contained in $\mathit{outcont}_{\mathcal{S}}(\mathcal{T}^*)$. Together with Eq. 3, this yields that for all $\mathcal{T}' \in \mathit{trd}(\mathcal{S}, k+1)$ we have

$$P_{\mathcal{T}'}(\mathit{OutObs}(\mathcal{T}, \alpha, k+1, m)) < \beta_m \qquad (5)$$

for some $\beta_m \to 0$ as $m \to \infty$. We are left to show that $\mathcal{T} \in \mathit{trd}(\mathcal{I} \,\|\, \hat{\mathfrak{t}}, k+1)$ for some $\hat{\mathfrak{t}} \in \hat{\mathfrak{T}}$. Let

$$\mathfrak{K} = \{\sigma \in \mathit{traces}^{\mathit{fin}}(\mathcal{I}) \mid P_{\mathcal{T}}(\Sigma) > 0\},$$

i.e. the set of all traces assigned positive probability under $\mathcal{T}$. Obviously $\mathfrak{C} \subseteq \mathfrak{K}$. By initial assumption, we know that all $\sigma \in \mathfrak{C}$ are contained in $\mathit{traces}^{\mathit{fin}}(\mathcal{S})$. Hence, all $\sigma \in \mathfrak{K}$ are necessarily in $\mathit{traces}^{\mathit{fin}}(\mathcal{S})$. Thus, there is a test case $\hat{\mathfrak{t}}$ for $\mathcal{S}$ such that all $\sigma \in \mathfrak{K}$ are in $\mathit{traces}^{\mathit{com}}(\hat{\mathfrak{t}})$. In particular, all $\sigma$ end in output by assumption. Hence, the last stage of every test case is item 2 in Definition 16. We now construct a scheduler $\mathfrak{S}' \in \mathit{Sched}(\mathcal{I} \,\|\, \hat{\mathfrak{t}})^{\leq k+1}$ such that $\mathit{trd}(\mathfrak{S}') = \mathcal{T}$.

Consider the mapping $f \in \mathit{tr}^{-1}(\mathfrak{K}) \to \mathit{paths}^{\mathit{fin}}(\mathcal{I} \,\|\, \hat{\mathfrak{t}})$ where for every path fragment we have

$$\begin{aligned} &f(\ldots \langle \ell, v_0, x_0 \rangle \langle t_1, e_1, R_1 \langle \ell', v_1, x_1 \rangle \rangle \ldots) \\ &= \ldots \langle \langle \ell, q \rangle, \bar{v}_0, \bar{x}_0 \rangle \langle \bar{t}_1, \bar{e}_1, \bar{R}_1, \langle \langle \ell, q \rangle, \bar{v}_1, \bar{x}_1 \rangle \rangle \ldots . \end{aligned}$$

By Definition 16, $v_i = \bar{v}_i \wedge x_i = \bar{x}_i$ for $i = 0, 1$ and $t_1 = \bar{t}_1 \wedge R_1 = \bar{R}_1$, because test cases do not have clocks. Further, we define $g \in E_{\mathcal{I}} \to E_{\mathcal{I} \| \mathfrak{t}}$ such that

$$g(e) = g(C, a, \mu(\langle R, \ell \rangle)) = (C, a, \bar{\mu}(\langle R, \langle \ell, q \rangle \rangle)) = \bar{e}$$

where $\mu(\langle R, \langle \ell, q \rangle \rangle) = \bar{\mu}(\langle R, \ell \rangle)$ for all $\ell$. $q$ is uniquely determined because tests are internally deterministic and every distribution is the Dirac distribution. Thus, discrete probabilities carry over from $\mu$ to $\bar{\mu}$. In particular, $q = q'$ if $a = \tau$. Then, $f$ is an injection, i.e. $f(\pi_1) = f(\pi_2) \Rightarrow \pi_1 = \pi_2$.

We now construct $\mathfrak{S}'$. Let $\mathfrak{S}$ be the scheduler that induces $\mathcal{T}$ by Definition 13. For every $\pi \in tr^{-1}(\mathfrak{K})$, we define

$$\mathfrak{S}'(\pi)(\bar{e}) \overset{\text{def}}{=} \mathfrak{S}(f^{-1}(\pi))(e).$$

The construction of $\mathfrak{S}'$ is straightforward: since $\hat{\mathfrak{t}}$ is internally deterministic, and every of its discrete distributions is the Dirac distribution, there is no interleaving in $\mathcal{I} \parallel \hat{\mathfrak{t}}$. Hence, a scheduler of $\mathcal{I} \parallel \hat{\mathfrak{t}}$ may copy the decisions of $\mathfrak{S}$ step by step. In particular, $P_{trd(\mathfrak{S}')}(\Sigma) = 0$ for $\sigma \notin \mathfrak{K}$. We conclude $trd(\mathfrak{S}') = \mathcal{T}$ and therefore $\mathcal{T} \in trd(\mathcal{I} \parallel \hat{\mathfrak{t}}, k+1)$.

Together with Eq. 4, we have found a scheduler $\mathfrak{S}'$ such that $trd(\mathfrak{S}') \in trd(\mathcal{I} \parallel \hat{\mathfrak{t}}, k+1)$, and for all $\mathcal{T}' \in trd(\mathcal{S}, k+1)$ we have

$$P_{\mathcal{T}'}(OutObs(trd(\mathfrak{S}'), \alpha, k+1, m)) < \beta_m.$$

Now iff $\alpha \leq 1 - \beta_m$, we estimate this further to

$$P_{\mathcal{T}'}(OutObs(trd(\mathfrak{S}'), \alpha, k+1, m)) < \beta_m \leq 1 - \alpha.$$

However, the inequality $\alpha \leq 1 - \beta_m$ always holds for sufficiently large $m$, since $\beta_m \to 0$ as $m \to \infty$ by Definition 18. Ultimately, this yields $v_{prob}(\mathcal{I}, \hat{\mathfrak{t}}) = fail$. □

## 5 Implementing stochastic testing

We now present practical procedures to implement the concepts defined in the previous section. First, we propose a goodness-of-fit method in the form of Pearson's $\chi^2$ test enriched with confidence interval analysis on the time stamps to evaluate the stochastic behaviour of the observed traces in the IOMA setting. Waiting times recorded in traces are grouped and compared to the prescribed rate parameters in the specification. Some additional assumptions are necessary to enable a clean and efficient framework. Since IOSA are not limited to exponential distributions, we need more powerful ways to infer if a sample was drawn from a particular distribution. In the IOSA setting, we thus apply the Kolmogorov–Smirnov (KS) test, which is able to infer general probability distributions, in place of interval estimation. Next, we discuss the interplay of stochastic delays and quiescence. Finally, we summarise the overall stochastic MBT procedure from test case generation to final verdicts.

### 5.1 Goodness of fit

We need practically applicable methods to decide about the verdicts given by Definition 19. While the functional verdict is determined via test annotations in the same straightforward way as in traditional ioco testing, we also need a procedure to decide the probabilistic verdict. We propose a two-step procedure consisting of Pearson's $\chi^2$ hypothesis test for the discrete probabilities followed by interval estimation (in the IOMA setting) or multiple KS tests (in the IOSA setting) for the time stamps resulting from the stochastic delays.

Our method is based on a theorem known from the literature [8] relating trace distributions to the set of acceptable outcomes. However, neither is readily available to us in case of a real black-box implementation—only experiments and samples give evidence about its inner workings. Therefore, we pose a null-hypothesis test based on a gathered sample of the *implementation*. Should the sample turn out to be an acceptable outcome of the *specification*, too, then we accept the hypothesis that all observations of the implementation are also observations of the specification. In tandem with the theorem by Cheung et al. [8], this would imply an embedding on the set of trace distributions. Consequently, the resulting probabilistic verdict in Definition 19 would be *pass*.

#### 5.1.1 Pearson's $\chi^2$ test

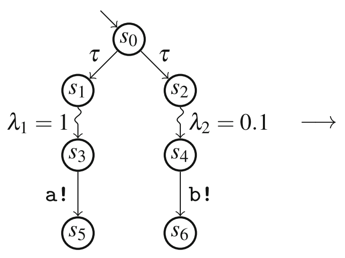
In previous work for pIOTS models [20], we used the $\chi^2$ hypothesis test to judge discrete probabilistic behaviour. Its outcome is based on a sample $O$ taken from the implementation under test. Should $O$ prove to be a sample of the set $OutObs(\mathcal{S}, \alpha, k, m)$ for some $\alpha \in (0,1)$, we are willing to accept the hypothesis of the embeddings of observations. In the continuous-time stochastic case, we argue along the same lines. However, only applying the $\chi^2$ hypothesis test is insufficient, as it does not take into account the delays observed in abstract traces. Nonetheless, passing the $\chi^2$ test is a necessary condition for an implementation to be accepted.

For a finite trace $\sigma = t_1 a_1 t_2 a_2 \ldots t_n a_n$, we define its time closure as $\bar{\sigma} = \mathbb{R}_0^+ a_1 \mathbb{R}_0^+ a_2 \ldots \mathbb{R}_0^+ a_n$. Then, the empiric $\chi^2$ score is given as

$$\chi^2 \overset{\text{def}}{=} \sum_{\bar{\sigma} \in \{\bar{\rho} \mid \rho \in O\}} \frac{\left(|\{\!\!\{ \bar{\rho} \mid \rho \in O \wedge \bar{\rho} = \bar{\sigma} \}\!\!\}| - m\mathbb{E}^{\mathcal{T}}(\bar{\sigma})\right)^2}{m\mathbb{E}^{\mathcal{T}}(\bar{\sigma})}, \tag{6}$$

essentially comparing observed traces to their respective expected counterparts. We use the time closure of traces to ignore time stamps for the $\chi^2$ analysis. The empirical $\chi^2$ value is compared to critical values of given degrees of freedom and levels of significance. The degrees of freedom are given by the number of different timed closures in $O$ minus one. The critical value can be calculated, or looked up in a $\chi^2$ table. In case the empiric $\chi^2$ score is below the given threshold $\chi^2_{crit}$, the hypothesis is accepted, and otherwise, it is rejected.

However, the expected value $\mathbb{E}^{\mathcal{T}}$ depends on the resulting trace distribution of a scheduler. Thus, finding a scheduler such that $\chi^2 \leq \chi^2_{crit}$ turns (6) into a minimisation problem



| ID | Trace $\sigma$ |
|---|---|
| 1 | 0.03 a! |
| 2 | 0.30 a! |
| 3 | 0.37 a! |
| 4 | 0.64 a! |
| 5 | 1.38 a! |
| 6 | 1.79 a! |
| 7 | 1.82 a! |
| 8 | 2.69 a! |
| 9 | 2.28 b! |
| 10 | 4.52 b! |
| 11 | 8.28 b! |
| 12 | 8.71 b! |
| 13 | 14.27 b! |
| 14 | 19.01 b! |

**Fig. 8** Specification IOMA and observation sample

(or satisfaction problem, respectively):

$$\min_{\mathcal{T}\in trd(\mathcal{S},k)} \sum_{\bar{\sigma}\in\{\bar{\rho}\mid\rho\in O\}} \frac{|\{\!|\, \bar{\rho} \mid \rho \in O \wedge \bar{\rho} = \bar{\sigma} \,|\!\}| - m\mathbb{E}^{\mathcal{T}}(\bar{\sigma})^2}{m\mathbb{E}^{\mathcal{T}}(\bar{\sigma})}. \tag{7}$$

The probability of a trace is given by a scheduler and the corresponding path probability function. Hence, we need to find probabilities $p$ used by a scheduler to resolve nondeterminism. This turns (7) into a minimisation or constraint solving problem of a rational function $f(p)/g(p)$ with inequality constraints on the vector $p$. This type of problem is NP-hard in general [39].

### 5.1.2 Interval estimation for IOMA

In addition to the $\chi^2$ test defined above, we need a metric to decide whether the observed delays correspond to exponential distributions prescribed by the specification in the IOMA setting. For this purpose, we use interval estimation on the parameters of the exponential distributions.

In general, assume values $x_1, \ldots, x_n$ are given, and suppose we ought to test whether the values follow an exponential distribution with rate $\lambda$. Our goal is to construct the confidence interval of these values for a given $\alpha \in ]0, 1[$, i.e. upon further sampling and estimations, there is a $1-\alpha$ chance that the true parameter $\lambda_{real}$ is contained in the interval. The $1 - \alpha$ confidence interval is given by

$$\left[\frac{\chi^2_{1-\alpha/2,2n}}{2\Sigma_{i=1}^{n}x_i}, \frac{\chi^2_{\alpha/2,2n}}{2\Sigma_{i=1}^{n}x_i}\right] \tag{8}$$

where $\chi^2_{\alpha,2n}$ is the $1-\alpha$ quantile of the $\chi^2$ distribution of $2n$ degrees of freedom.

***Example 7*** Figure 8 shows an example specification model alongside an example observation sample from an implementation. State $s_0$ has two outgoing $\tau$ transitions, followed by one Markovian transition in each of $s_1$ and $s_2$. In states $s_3$ and $s_4$, we either observe action a! or b!, respectively. The sample shows 14 recorded traces of length one, thus $m = 14$ and $k = 1$. There are two steps to assess whether the observed data are a truthful sample of the specification model with a confidence of $\alpha = 0.1$: first find a trace distribution that minimises the $\chi^2$ statistic, then evaluate two confidence intervals to assess whether the observed time data are a sample of $\lambda_1 = 1$ and $\lambda_2 = 0.1$, respectively.

There are two classes of traces solely based on the action signature: ID 1-8 with a! and ID 9-14 with b!. Let $p$ be the probability that a scheduler assigns to taking the left branch in $s_0$, and $1 - p$ the probability for the right branch. Upon drawing a sample with $m = 14$ we expect $m \cdot p$ as frequency for a! and $m \cdot (1 - p)$ as frequency for b!. The empirical $\chi^2$ score therefore calculates as

$$\chi^2 = \frac{8 - 14 \cdot p}{14 \cdot p} + \frac{6 - 14 \cdot (1 - p)}{14 \cdot (1 - p)}.$$

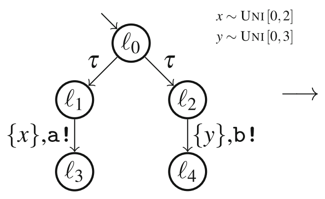
This yields $\chi^2 = 0$ for $p = 8/14$, which is obviously smaller than the value $\chi^2_{crit} = \chi^2_{0.1,1} = 2.706$. We thus proceed to confidence interval estimation.

$t_1 = 0.03, \ldots, t_8 = 2.69$ is the data associated with $\lambda_1$ and $t_1' = 2.28, \ldots, t_6' = 19.01$ the data associated with $\lambda_2$. Calculating the confidence intervals according to Eq. 8 yields $C_1 = [0.441, 1.458]$ and $C_2 = [0.092, 0.368]$. We see that $\lambda_1 \in C_1$ and $\lambda_2 \in C_2$ and are therefore willing to accept that the recorded sample was drawn under the prescribed parameters.

These two steps do not yet make a sound statement about the acceptance of the hypothesis $O \in OutObs(\mathcal{S}, 0.05, 1, 14)$ since we test multiple hypotheses at once. We need to adjust the individual level of significance for the statistical tests, to conclude the overall acceptance with $\alpha = 0.1$. This inflation of the error of first kind is discussed in Sect. 5.1.4.

Example 7 highlights the necessity of two assumptions if we are to apply confidence intervals as the method of choice:

– We must be able to uniquely identify every recorded trace. Assume for illustration that the transition currently labelled b! was labelled a! instead. It would not directly be possible to associate values $t_i$ with $\lambda_1$ and $t_i'$ with $\lambda_2$; we would need to check all possible permutations. This becomes infeasible in practice even for moderate sample sizes or moderately sized models; we therefore assume all specification models to be internally deterministic, i.e. there must be a bijection between paths and traces.
– The sum of exponential distributions is not an exponential distribution. Hence, confidence interval estimation would be flawed for two sequential Markovian actions. We would need to deal with *phase-type distributions* instead, which are dense in the set of all positively valued



| ID | Trace $\sigma$ |
|---|---|
| 1 | 0.26 a! |
| 2 | 0.33 a! |
| 3 | 0.55 a! |
| 4 | 0.77 a! |
| 5 | 1.18 a! |
| 6 | 1.41 a! |
| 7 | 1.46 a! |
| 8 | 1.97 a! |

| ID | Trace $\sigma$ |
|---|---|
| 9 | 0.29 b! |
| 10 | 0.38 b! |
| 11 | 1.90 b! |
| 12 | 2.44 b! |
| 13 | 2.71 b! |
| 14 | 2.74 b! |

**Fig. 9** Specification IOSA and observation sample

distributions. We thus assume models to contain an input or output between any two Markovian transitions.

### 5.1.3 Kolmogorov–Smirnov tests for IOSA

Working with IOSA means that specifications and implementations are not limited to the exponential distribution. Since they neither comprise *one specific distribution* nor *one specific parameter* to test for, we use the nonparametric KS test to validate that the observed delays were drawn from the specified clocks and distributions. The KS test assesses whether observed data matches a hypothesised *continuous* probability measure. We thus restrict the practical application of our approach to IOSA where the $F(c)$ for all clocks $c$ are continuous distributions.

Let $t_1, \ldots, t_n$ be the delays observed for a certain edge over multiple traces in ascending order and $F_n$ be the resulting step function, i.e. the right-continuous function $F_n$ defined by

$$F_n(t) = \begin{cases} 0 & \text{if } t < t_1 \\ n_i/n & \text{if } t_i \leq t < t_{i+1} \\ 1 & \text{if } t \geq t_n \end{cases}$$

where $n_i$ is the number of $t_j$ that are smaller or equal to $t_i$. Further, let $c$ be a clock with CDF $F_c$ for the measure $F(c)$. Then the $n$-th KS statistic is given by

$$K_n \overset{\text{def}}{=} \sup_{t \in \mathbb{R}_0^+} |F_c(t) - F_n(t)|. \tag{9}$$

If the sample values $t_1, \ldots, t_n$ are truly drawn from the CDF $F_c$, then $K_n \to 0$ almost surely as $n \to \infty$ by the Glivenko–Cantelli theorem [22]. Hence, for given $\alpha$ and sample size $n$, we accept the hypothesis that the $t_i$ were drawn from $F_c$ iff $K_n \leq K_{crit}$, where $K_{crit}$ is a critical value given by the Kolmogorov distribution. Again, the critical values can be calculated or found in tables.

***Example 8*** The left-hand side of Fig. 9 shows a tiny example specification IOSA with clocks $x$ and $y$. The expiration times of both are uniformly distributed with different parameters. In $\ell_0$ there is a nondeterministic choice to either take the left or the right branch. The right-hand side depicts a sample from this IOSA. There are two steps to assess whether the observed data are a truthful sample of the specification with a confidence of $\alpha = 0.05$: first find a trace distribution that minimises the $\chi^2$ statistic, and then evaluate two KS tests to assess whether the observed time data are a truthful sample of UNI[0, 2] and UNI[0, 3], respectively.

In the same way as in Example 7, the empirical $\chi^2$ value calculates as

$$\chi^2 = \frac{(8 - 14 \cdot p)^2}{(14 \cdot p)} + \frac{(6 - 14 \cdot (1 - p))^2}{(14 \cdot (1 - p))},$$

which is minimal for $p = 8/14$ and smaller than $\chi^2_{crit} = 3.84$. We thus found a scheduler that maximises the likelihood of the observed frequencies.

For the second step, $t_1 = 0.26, \ldots, t_8 = 1.97$ is the data associated with clock $x$ and $t'_1 = 0.29, \ldots, t'_6 = 2.74$ is the data associated with clock $y$. Since there is no time that was recorded twice, the step function of the $t_i$ is

$$F_8(t) = \begin{cases} 0 & \text{if } t < t_0 \\ \frac{k}{8} & \text{if } t_k \leq t < t_{k+1}, k = 1, \ldots, 7 \\ 1 & \text{if } t \geq t_8. \end{cases}$$

$D_8 = 0.145$ is the maximal distance between this empirical step function and UNI[0, 2]. The critical value of the Kolmogorov distribution for $n = 8$ and $\alpha = 0.05$ is $K_{crit} = 0.46$. With $K_8 < K_{crit}$, the empiric value is below the given threshold. Hence, the inferred measure is sufficiently close to the specification. The KS test for $t'_i$ and UNI[0, 3] can be performed analogously. To conclude overall acceptance with $\alpha = 0.1$, we again need to adjust the level of significance due to performing multiple tests; see Sect. 5.1.4.

Our intention is to provide a general and universally applicable procedure. The KS test is conservative for general distributions, but can be made precise [10]. Specialised and thus more efficient tests exist for specific distributions, e.g. the Lilliefors test [29] for Gaussian distributions, and parametric tests are generally preferred due to higher power at equal sample size. The KS test requires a comparably large sample size, an alternative being, e.g. the Anderson–Darling test [29].

***Remark 3*** The connection of two nonparametric tests is immensely more difficult in the presence of internal nondeterminism in a specification, cf. Example 8 with only `a!` on both visible edges. Time values can no longer be unambiguously addressed to unique distributions, and no confidence bound for the measured time data can be given. In this case, the scheduler probability decisions $p$ are used as parameters for mixture distributions, e.g. $F(p) \overset{\text{def}}{=} p \cdot F_x + (1 - p) \cdot F_y$ in Fig. 9. The parameterised distribution can then be used in the iterative expectation–maximisation algorithm [38], and confidence can be given upon convergence.

For the sake of simplicity, we assume that the specification is internally deterministic, i.e. there are no two paths that result in the same trace. While this decreases the space of potential specifications, we deem it a necessary compromise to come up with a feasible and general method.

### 5.1.4 Multiple comparisons

Since the $\chi^2$ test and all subsequent confidence interval estimations or KS tests are statistical hypothesis tests on their own, their errors accumulate. To illustrate: if a hypothesis test is performed at $\alpha = 0.05$ there is a 5% chance of performing an error of first kind, i.e. of erroneously rejecting a true hypothesis. If we apply 100 individual tests with $\alpha = 0.05$, we might naively *expect* to perform this error 5 times. If we assume the tests to be independent, the probability of committing at least one error of the first kind *actually* calculates as $1 - (1 - 0.05)^{100} = 99.4\%$.

There are several techniques to cope with the inflation of the error of first kind. For the remainder of this section, we use Bonferroni correction: $\alpha_{local} = \alpha_{global}/l$ where $l$ is the total number of statistical hypothesis tests to be performed.

***Example 9*** We return to Example 7. Applying Bonferroni correction for a total of three hypothesis tests with desired $\alpha = \alpha_{global} = 0.1$ tests yields a necessary $\alpha_{local} \approx 0.033$. This applies to the $\chi^2$ test and the two interval estimations. The $\chi^2$ test still passes, and the new confidence intervals are $C_1' = [0.353, 1.677]$ and $C_2' = [0.070, 0.432]$. We see that $\lambda_1 \in C_1'$ and $\lambda_2 \in C_2'$ still hold, so we give the implementation the probabilistic *pass* verdict.

## 5.2 Stochastic delays and quiescence

A test case needs to assess if an implementation is allowed to be unresponsive when output was expected [45]. In our formalism, quiescence $\delta$ models the absence of output for an *indefinite* time. It should be regarded with caution in practical testing scenarios. A common way to deal with quiescence is a global fixed timeout value set by a user [2,5]. The time progress in IOMA and IOSA is governed by continuous probability distributions; hence, a global timeout has two disadvantages: first, a timeout might occur before a specified Markovian transition or edge takes place. The average waiting time of this event might be substantially higher than the global timeout. Second, a *global* timeout might unnecessarily prolong the overall test process.

A timeout can be seen as a delay that follows a Dirac distribution. While this naturally fits into the framework of stochastic automata, it is incompatible with the IOMA approach: Dirac delays cannot be represented in IOMA, and

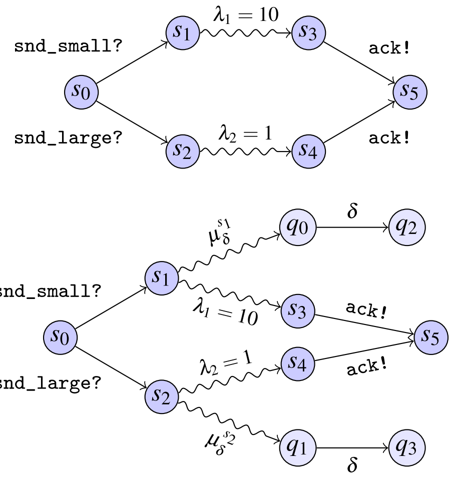


**Fig. 10** Two example specifications for quiescence timeouts

consequently, they were not considered in the statistical evaluation that we developed in Sect. 5.1.2. We now detail an approach for IOMA that avoids the problem of Dirac distributions and aims to minimise the probability of erroneously declaring quiescence while keeping the overall testing time as low as possible. While Dirac distributions are supported by IOSA, similar ideas for the latter apply to IOSA, too.

In order to avoid Dirac distributions, an MBT tool for IOMA needs to implement quiescence by racing an exponentially distributed delay with rate $\mu_\delta$ against the implementation; this *quiescence timer* winning the race is then treated as the quiescence output $\delta$. Let $\lambda > 0$ be the minimum exit rate over all Markovian states. With level of significance $\alpha \in ]0, 1[$, we would like the probability that the quiescence timer expires before a Markovian transition is executed, i.e. that we incorrectly report quiescence when the implementation could make progress, to be at most $\alpha$. Choosing $\mu_\delta = \lambda \cdot \frac{\alpha}{1-\alpha}$ as the quiescence timer's rate achieves this probability with the shortest waiting time in case of actual quiescence. We can further reduce the waiting time by using a different rate in every state: if the exit rate of state $s$ is $\lambda_s$, we use rate $\mu_\delta^s = \lambda_s \cdot \frac{\alpha}{1-\alpha}$ to judge quiescence in $s$.

The statistical evaluation only has to be adjusted to consider the new exit rate $\lambda + \mu_\delta$ and the new "Markovian transition" for quiescence. In fact, we can directly represent this approach by rewriting the specification model as shown in Example 10. For non-Markovian states, a *default* maximal waiting time is still applicable.

***Example 10*** Figure 10 (top) shows a simple specification of a file transmission protocol. Exponential distributions model the delay between sending a file and acknowledging its reception. Different delays are associated with sending small or a large files, respectively. After a file was sent, there is a chance that it gets lost, and we do not receive an acknowledgement.

In this case, the system is judged as quiescent, and therefore erroneous.

However, since $\lambda_2 \ll \lambda_1$, a test should use a quiescence timer rate of $\mu_\delta^{s_1} = 10 \cdot \frac{\alpha}{1-\alpha}$ in $s_1$ and $\mu_\delta^{s_2} = \frac{\alpha}{1-\alpha}$ in $s_2$ to minimise the probability to erroneously judge the system as quiescent while also keeping the global testing time as low as possible. Regardless, for sufficiently large sample size, an MBT tool eventually erroneously observes quiescence. Figure 10 (bottom) therefore allows some amount of quiescence observations depending on $\alpha$, i.e. on how many erroneous quiescent judgements we are willing to accept.

***Example 11*** We compare a global quiescence timer rate to individual ones by assuming $\alpha = 0.05$ and that we are to test the protocol as in Fig. 10 (top) 100 times:

Long global: A sensible long global quiescence timer rate is $\mu_d = \mu_\delta^{s_2} \approx 0.053$. Executing 100 test cases yields a *worst-case expected* waiting time (for the case where implementation is always quiescent) of $100/\mu_\delta^{s_2} = 1900$ time units. However, we are (more than) guaranteed to incorrectly judge the implementation quiescent in at most 5 % of all cases.

Short global: A sensible short global quiescence timer rate is $\mu_d = \mu_\delta^{s_1} \approx 0.526$. The worst-case expected time is now only 190 time units. However, the probability of the Markovian transition with rate $\lambda_2$ not firing before the quiescence timer becomes $\approx$ 34 %. We would then incorrectly judge the implementation quiescent even though the transition might still take place.

Individual: Using the long rate in state $s_2$ and the short one in state $s_1$ guarantees that we erroneously judge quiescence overall in at most 5% of the cases. Note that this is accounted for in the specification in Fig. 10 (bottom). The worst-case waiting time now depends on the probability $p$ of sending a small file instead of a large one; it is $p \cdot 190 + (1 - p) \cdot 1900$. Time is saved in the overall test process whenever a small file is sent.

### 5.3 Stochastic test procedure outline

Test cases for IOMA and IOSA are *essentially* IOTS. Hence, the standard test generation algorithms for ioco [47] apply directly, except for the inclusion of explicit quiescence timeouts as in Fig. 10 (bottom), if desired. We summarise all necessary steps to perform model-based testing with Markov automata or stochastic automata using our framework:

1. Generate an annotated test case (suite) for the specification automaton.
2. Execute the test case (all test cases of the test suite) $m$ times. If the functional *fail* verdict is encountered in any of the $m$ executions, then fail the implementation for *functional* reasons.
3. Calculate the number of necessary statistical hypothesis tests for each test case. Correct $\alpha$ accordingly.
4. Perform statistical analysis on the gathered sample of size $m$ for the test case (all test cases) with the new parameter $\bar{\alpha}$.

   (a) Use optimisation or constraint solving to find a scheduler such that $\chi^2 \leq \chi^2_{crit}$. If no such scheduler is found, reject the implementation for *statistical* reasons.

   ($b_1$) For IOMA, perform confidence interval estimation, and check if all Markovian parameters are contained in their respective intervals. If there is at least one parameter not contained in its confidence interval, reject the implementation for *statistical* reasons.

   ($b_2$) For IOSA, group all time stamps assigned to the same clock and perform a KS test for each clock. If any of them fail, reject $\mathcal{I}$ for *statistical* reasons.
5. Accept the implementation.

## 6 A Bluetooth device discovery example

Bluetooth is a wireless communication standard [3] aimed at low-powered devices that communicate over short distances. Before any communication can take place, Bluetooth devices organise into small networks of one *master* and up to seven *slave* devices. To cope with interference, this device discovery protocol uses a frequency hopping scheme.

To illustrate and compare our frameworks for IOMA and IOSA, we study the discovery phase for one master and one slave device. The device discovery protocol is inherently stochastic due to the initially random and unsynchronised state of the devices. We give a high-level overview of the protocol here and refer the interested reader to a verification case study performed with PRISM [16] for a more detailed description and formal analysis in a more general setting.

### 6.1 Device discovery protocol

To resolve possible interference, the master and slave device communicate via a prescribed sequence of 32 frequencies. Both devices have a 28-bit clock that ticks every 312.5 µs.

The master broadcasts on two frequencies for two consecutive ticks followed by a two-tick listening period on the same frequencies. It picks the broadcasting frequency *freq* as

$$(CLK_{16...12} + o + (CLK_{4...2,0} - CLK_{16...12}) \bmod 16) \bmod 32$$

where $CLK_{i...j}$ marks bits $i$ to $j$ of the clock and $o \in \mathbb{N}$ is an offset. The master chooses one of two *tracks* and switches to the respective other every 2.56 s. Every 1.28 s, i.e. every time the 12th bit of the clock changes, a frequency is *swapped* between the two tracks. For simplicity, we choose $o = 1$ for track one and $o = 17$ for track two, such that the two tracks initially comprise frequencies $1, \dots, 16$ and $17, \dots, 32$.

The slave device periodically scans on the 32 frequencies. It is in either a sleeping or a listening state. To ensure eventual connection, the hopping rate of the slave device is much slower. The Bluetooth standard leaves some flexibility with respect to the length of the listening period. For our study, every 0.64 s, it listens to one frequency for 11.25 ms and sleeps during the remaining time. It cycles to the next frequency after 1.28 s. This is enough time for the master device to broadcast on 16 different frequencies.

## 6.2 Specification models

The time to connect two devices is deterministic for a fixed initial state. That is, assuming we know the initial state of both devices, we can calculate the time needed until a connection is established. To study a realistic scenario, however, we have to assume that the clocks of both devices are initially desynchronised. Thus, in our models, the master starts broadcasting immediately while the slave starts listening after a uniformly chosen random waiting time. We then have four scenarios to reach synchronisation:

– Synchronisation happens during the first 16 broadcast frequencies. This happens between 0 and 1.28 s and comprises 16 frequencies.
– Synchronisation happens after the first frequency swap of the master device (1.28 to 2.56 s, one frequency).
– Synchronisation happens after the first switch of tracks and two frequency swaps of the master device (2.56 to 3.84 s, 14 frequencies).
– Synchronisation happens after the first switch of tracks and three frequency swaps of the master device (3.84 to 5.12 s, one frequency).

These four scenarios are exhaustive, i.e. the master device broadcasts on frequencies such that the slave necessarily listens to at least one of them within 5.12 s. The different scenarios yield 32 possible exact waiting times to connect, i.e. after 2 or 3 ticks, 6 or 7 ticks, etc.

This protocol specification prescribes a delay that is not exponentially distributed, as is evident by the sample CDF we collected for the specification shown in Fig. 13 (dark blue line). This is no problem for IOSA-based testing. Our IOSA specification is shown in Fig. 11; we directly incorporate the exact probability distribution to connect within a certain time as prescribed by the protocol description as the distribution $F(x)$ here. Thus, the structure of the IOSA can be extremely simple; the complexity is hidden in $F(x)$. For

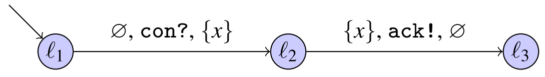

$$\begin{tikzcd} \arrow[r] & \ell_1 \arrow[r, "{\varnothing,\ \texttt{con?},\ \{x\}}"] & \ell_2 \arrow[r, "{\{x\},\ \texttt{ack!},\ \varnothing}"] & \ell_3 \end{tikzcd}$$

**Fig. 11** IOSA specification for the Bluetooth example

IOMA, we have to approximate the true distribution by an exponential distribution. Calculating the mean of all waiting times gives us the average time to connect as approximately 1.325 s and thus $\lambda = 0.755$ as the estimated rate parameter. Note that $F(x)$ in the IOSA case could also be specified as the exponential distribution with $\lambda = 0.755$ to pose the same requirement that concerns the mean time to connection only.

## 6.3 Experimental setup

Our toolchain is depicted in Fig. 12. The implementation is tested on-the-fly via the MBT tool JTorX [2], which generates tests with respect to the transition system abstraction of the specifications. JTorX returns the functional *fail* verdict if unforeseen output is observed at any time throughout the test process. Additionally, we chose a timeout of approximately 5.2 s in accordance with the protocol description: this is the time that the master device needs to broadcast *all* available frequencies at least once. We can use this fixed timeout even in the IOMA setting since we know that no correct implementation may take this long to connect; any implementation that does can be functionally rejected without the need for statistical analysis. The recorded log files of JTorX comprise the sample. We use MATLAB to calculate the statistical verdict. We implemented the correct protocol and three mutants in Java 7:

$\mathcal{M}_1$ The first master mutant never switches between tracks one and two, therefore covering far fewer different frequencies than the correct protocol in the same time. It will need a total of $16 \cdot 1.28\,\text{s} = 20.48\,\text{s}$ to cover all 32 frequencies. Hence, we expect a much longer time to connect when compared to the correct implementation.

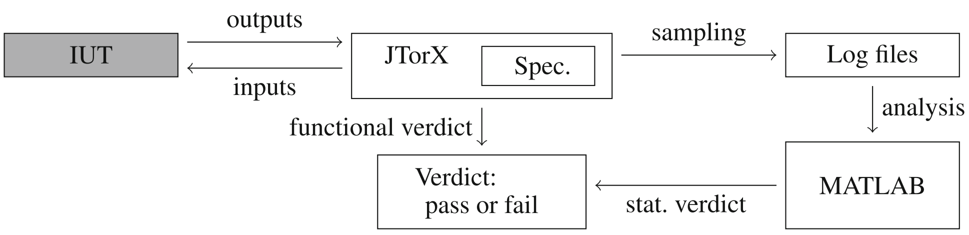
$\mathcal{M}_2$ The second master mutant never swaps frequencies, only switching between tracks one and two. The expected time to connect will therefore be around 2.56 s.

$\mathcal{S}_1$ The slave mutant has its listening period halved, and thus only listens for 5.65 ms every 1.28 s. Therefore, it has a longer sleeping period and we expect that the probability to connect is slightly reduced when compared to the correct counterpart.

## 6.4 Results

We collected $m = 100$, $m = 1000$, and $m = 10{,}000$ test executions for each of the four implementations. We set the

**Fig. 12** Experimental setup



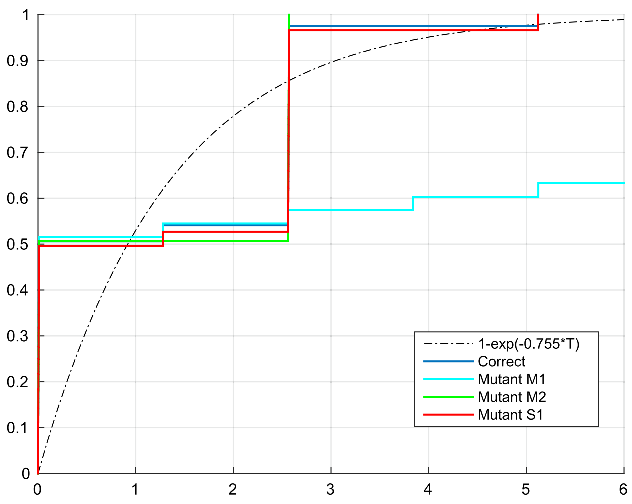


**Fig. 13** Probability to establish connection over time

level of significance to $\alpha = 0.05$. No $\chi^2$ tests are necessary due to the absence of nondeterminism and probabilistic branching in the specifications. Furthermore, we need only one statistical test in each setting and thus no $\alpha$ correction. Figure 13 shows the cumulative distribution of the sample data collected for $m = 1000$ runs of the correct implementation and mutants (coloured lines).

**IOMA** For comparison, we show as a dashed line in Fig. 13 the cumulative probability to connect within $T$ seconds for the exponential distribution with rate $\lambda = 0.755$, which is the specified distribution in the IOMA setting. Table 1 shows the confidence intervals calculated based on our samples. All intervals of the correct implementation contain the assumed value $\lambda = 0.755$, which is therefore judged as correct. $\mathcal{M}_1 \parallel \mathcal{S}$ was consistently rejected for functional reasons by JTorX due to exceeding the fixed timeout. The remaining two mutants required the statistical verdict for rejection; both were still accepted for $m = 100$, requiring at least 1000 test executions for the statistical verdict to produce a confidence interval sufficiently narrow for rejection. In particular, dividing the listening time of the slave into half had the least impact on the behaviour; it was consequently rejected with a very small margin.

**IOSA** We used MATLAB's `kstest2` function to execute a two-sample KS test to analyse the samples with respect to the specified time distribution. Table 2 shows the verdicts and the observed KS statistics $K_m$ alongside the corresponding critical values $K_{crit}$ for our experiments. The statistical verdict *pass* was given if $K_m < K_{crit}$, and *fail* otherwise. The critical values depend on $\alpha$ and $m$. The correct implementation was accepted in all three experiments. During the sampling of $\mathcal{M}_1 \parallel \mathcal{S}$, we again observed several timeouts leading to a functional *fail* verdict. It would also have failed the KS test in all three experiments. $\mathcal{M}_2 \parallel \mathcal{S}$ passed the test for $m = 100$, but was rejected with increased sample size. $\mathcal{M} \parallel \mathcal{S}_1$ is the most subtle of the three mutants and was only rejected with $m = 10{,}000$ at a narrow margin.

**Discussion** The case study was not tailored to MBT with Markov automata. The waiting time of interest is clearly not exponentially distributed, and only means of the delay until the connection is established are compared. Nonetheless, the IOMA framework is applicable and rightfully judged the correct implementation as conforming while eliminating the mutants. The confidence intervals for the slave mutant only marginally did not contain the parameter $\lambda$. Consequently, there is a relatively high probability to commit an error of second kind. On the other hand, the second master mutant was eliminated with a large margin.

In the IOSA setting, observe that the critical value decreases faster than the observed KS statistic in all three faulty implementations. We conjecture that an even larger sample is expected to have a clearer verdict, as this is in line with the decreasing error of the second kind for increasing sample size pointed out in Sect. 4. This is especially desirable in the case of $\mathcal{M} \parallel \mathcal{S}_1$, where a sample of size $m = 10{,}000$ was needed to refute the faulty implementation. This is in contrast to the IOMA setting, where $m = 1000$ sufficed, and highlights that the statistical evaluation for IOMA is in general more efficient (it needs fewer samples for clearer verdicts) than the one for IOSA. We point out that an alternate specification to the very compact one given in Fig. 11 is possible. For instance, the entire specification could comprise a probabilistic branching over 32 locations with deterministic guard sets according to the step values of the distribution of the Bluetooth specification. This illustrates the flexibility of the modelling capabilities in the IOSA test framework, and goes to show there is no *unique best* model.

**Table 1** Connection time confidence intervals (IOMA)

| | Correct | Mutants | | |
|---|---|---|---|---|
| | $\mathcal{M} \parallel \mathcal{S}$ | $\mathcal{M}_1 \parallel \mathcal{S}$ | $\mathcal{M}_2 \parallel \mathcal{S}$ | $\mathcal{M} \parallel \mathcal{S}_1$ |
| $k = 2$ | *pass* | *fail* | *pass* | *pass* |
| $m = 100$ | [0.586, 0.868] | – | [0.597, 0.885] | [0.673, 0.997] |
| Timeouts | 0 | 33 | 0 | 0 |
| $k = 2$ | *pass* | *fail* | *fail* | *fail* |
| $m = 1000$ | [0.729, 0.826] | – | [0.767, 0.868] | [0.756, 0.855] |
| Timeouts | 0 | 376 | 0 | 0 |
| $k = 2$ | *pass* | *fail* | *fail* | *fail* |
| $m = 10{,}000$ | [0.735, 0.764] | – | [0.772, 0.803] | [0.757, 0.787] |
| Timeouts | 0 | 3753 | 0 | 0 |

**Table 2** Verdicts and KS test results (IOSA)

| | Correct | Mutants | | |
|---|---|---|---|---|
| | $\mathcal{M} \parallel \mathcal{S}$ | $\mathcal{M}_1 \parallel \mathcal{S}$ | $\mathcal{M}_2 \parallel \mathcal{S}$ | $\mathcal{M} \parallel \mathcal{S}_1$ |
| $k = 2$ | *pass* | *fail* | *pass* | *pass* |
| $m = 100$ | $K_m = 0.065$ | – | $K_m = 0.110$ | $K_m = 0.065$ |
| | $K_{crit} = 0.136$ | | $K_{crit} = 0.136$ | $K_{crit} = 0.136$ |
| Timeouts | 0 | 40 | 0 | 0 |
| $k = 2$ | *pass* | *fail* | *fail* | *pass* |
| $m = 1000$ | $K_m = 0.028$ | – | $K_m = 0.050$ | $K_m = 0.020$ |
| | $K_{crit} = 0.045$ | | $K_{crit} = 0.045$ | $K_{crit} = 0.045$ |
| Timeouts | 0 | 399 | 0 | 0 |
| $k = 2$ | *pass* | *fail* | *fail* | *fail* |
| $m = 10{,}000$ | $K_m = 0.006$ | – | $K_m = 0.043$ | $K_m = 0.0193$ |
| | $K_{crit} = 0.019$ | | $K_{crit} = 0.019$ | $K_{crit} = 0.0192$ |
| Timeouts | 0 | 3726 | 0 | 0 |

Overall, there is a trade-off in expressivity and efficiency when comparing the test theory for Markov automata and stochastic automata in practical applications.

## 7 Conclusion

We presented two closely related sound and complete MBT frameworks to test probabilistic systems with stochastic delays. The underlying modelling formalisms are Markov automata and stochastic automata with a separation of their alphabet into inputs and outputs: IOMA and IOSA. The former limit delays to follow exponential distributions, but mark a relevant intermediate step between previous work on testing untimed probabilistic models [20] and the full generality—and complexity—of stochastic automata. In particular, the statistical evaluation of testing results is far simpler and more efficient in the case of IOMA. On the other hand, our Bluetooth case study shows that being able to represent arbitrary distributions over time directly as in IOSA may lead to specifications that much more closely match reality, and to provide results that are more precise and understandable.

## References

1. Baier C, Katoen JP (2008) Principles of model checking. MIT Press, Cambridge
2. Belinfante A (2014) JTorX: exploring model-based testing. Ph.D. thesis, University of Twente, Enschede, The Netherlands. http://purl.utwente.nl/publications/91781
3. Bluetooth SIG: Bluetooth specification, version 1.2. www.bluetooth.com (2003)
4. Bohnenkamp HC, Belinfante A (2005) Timed testing with TorX. In: Formal methods: international symposium of Formal Methods Europe (FM). Lecture notes in computer science, vol 3582. Springer, pp 173–188. https://doi.org/10.1007/11526841_13

5. Briones LB, Brinksma E (2004) A test generation framework for quiescent real-time systems. In: 4th international workshop on formal approaches to software testing (FATES). Lecture notes in computer science, vol 3395. Springer, pp 64–78. https://doi.org/10.1007/978-3-540-31848-4_5
6. Budde CE, D'Argenio PR, Hartmanns A, Sedwards S (2018) A statistical model checker for nondeterminism and rare events. In: 24th international conference on tools and algorithms for the construction and analysis of systems (TACAS). Lecture notes in computer science, vol 10806. Springer, pp 340–358. https://doi.org/10.1007/978-3-319-89963-3_20
7. Cheung L, Lynch NA, Segala R, Vaandrager FW (2006) Switched PIOA: parallel composition via distributed scheduling. Theor Comput Sci 365(1–2):83–108. https://doi.org/10.1016/j.tcs.2006.07.033
8. Cheung L, Stoelinga M, Vaandrager FW (2007) A testing scenario for probabilistic processes. J ACM 54(6):29. https://doi.org/10.1145/1314690.1314693
9. Cleaveland R, Dayar Z, Smolka SA, Yuen S (1999) Testing preorders for probabilistic processes. Inf Comput 154(2):93–148. https://doi.org/10.1006/inco.1999.2808
10. Conover WJ (1972) A Kolmogorov goodness-of-fit test for discontinuous distributions. J Am Stat Assoc 67(339):591–596
11. D'Argenio PR, Katoen JP (2005) A theory of stochastic systems part I: stochastic automata. Inf Comput 203(1):1–38. https://doi.org/10.1016/j.ic.2005.07.001
12. D'Argenio PR, Lee MD, Monti RE (2016) Input/output stochastic automata—compositionality and determinism. In: 14th international conference on formal modeling and analysis of timed systems (FORMATS). Lecture notes in computer science, vol 9884. Springer, pp 53–68. https://doi.org/10.1007/978-3-319-44878-7_4
13. Dehnert C, Junges S, Katoen JP, Volk M (2017) A Storm is coming: A modern probabilistic model checker. In: 29th international conference on computer aided verification (CAV). Lecture notes in computer science, vol 10427. Springer, pp 592–600. https://doi.org/10.1007/978-3-319-63390-9_31
14. Deng Y, van Glabbeek RJ, Hennessy M, Morgan C (2008) Characterising testing preorders for finite probabilistic processes. Log Methods Comput Sci 4(4):4. https://doi.org/10.2168/LMCS-4(4:4)2008
15. Deng Y, Hennessy M (2013) On the semantics of Markov automata. Inf Comput 222:139–168. https://doi.org/10.1016/j.ic.2012.10.010
16. Duflot M, Kwiatkowska MZ, Norman G, Parker D (2006) A formal analysis of Bluetooth device discovery. STTT 8(6):621–632. https://doi.org/10.1007/s10009-006-0014-x
17. Eisentraut C, Hermanns H, Zhang L (2010) On probabilistic automata in continuous time. In: 25th annual IEEE symposium on logic in computer science (LICS). IEEE Computer Society, pp 342–351. https://doi.org/10.1109/LICS.2010.41
18. Gerhold M (2018) Choice and chance—model-based testing of stochastic behaviour. Ph.D. thesis, University of Twente, Enschede, The Netherlands. https://doi.org/10.3990/1.9789036546959
19. Gerhold M, Hartmanns A, Stoelinga M (2018) Model-based testing for general stochastic time. In: 10th international NASA formal methods symposium (NFM). Lecture notes in computer science, vol 10811. Springer, pp 203–219. https://doi.org/10.1007/978-3-319-77935-5_15
20. Gerhold M, Stoelinga M (2016) Model-based testing of probabilistic systems. In: 19th international conference on fundamental approaches to software engineering (FASE). Lecture notes in computer science, vol 9633. Springer, pp 251–268. https://doi.org/10.1007/978-3-662-49665-7_15
21. Gerhold M, Stoelinga M (2017) Model-based testing of probabilistic systems with stochastic time. In: 11th international conference on tests and proofs (TAP). Lecture notes in computer science, vol 10375. Springer, pp 77–97. https://doi.org/10.1007/978-3-319-61467-0_5
22. Gibbons JD, Chakraborti S (2011) Nonparametric statistical inference. In: International encyclopedia of statistical science. Springer, pp 977–979. https://doi.org/10.1007/978-3-642-04898-2_420
23. Gordon AD, Henzinger TA, Nori AV, Rajamani SK (2014) Probabilistic programming. In: Future of software engineering (FOSE). ACM, pp 167–181. https://doi.org/10.1145/2593882.2593900
24. Graf-Brill A, Hartmanns A, Hermanns H, Rose S (2017) Modelling and certification for electric mobility. In: 15th IEEE international conference on industrial informatics (INDIN). IEEE, pp 109–114. https://doi.org/10.1109/INDIN.2017.8104755
25. Hartmanns A, Hermanns H (2014) The Modest Toolset: an integrated environment for quantitative modelling and verification. In: 20th international conference on tools and algorithms for the construction and analysis of systems (TACAS). Lecture notes in computer science, vol 8413. Springer, pp 593–598. https://doi.org/10.1007/978-3-642-54862-8_51
26. Hérault T, Lassaigne R, Magniette F, Peyronnet S (2004) Approximate probabilistic model checking. In: 5th international conference on verification, model checking, and abstract interpretation (VMCAI). Lecture notes in computer science, vol 2937. Springer, pp 73–84. https://doi.org/10.1007/978-3-540-24622-0_8
27. Hermanns H (2002) Interactive Markov chains: the quest for quantified quality. Lecture notes in computer science, vol 2428. Springer. https://doi.org/10.1007/3-540-45804-2
28. Hierons RM, Merayo MG, Núñez M (2009) Testing from a stochastic timed system with a fault model. J Log Algebr Program 78(2):98–115. https://doi.org/10.1016/j.jlap.2008.06.001
29. Hollander M, Wolfe DA, Chicken E (2013) Nonparametric statistical methods. Wiley, New York
30. Katoen JP (2016) The probabilistic model checking landscape. In: 31st annual ACM/IEEE symposium on logic in computer science (LICS). ACM, pp 31–45. https://doi.org/10.1145/2933575.2934574
31. Krichen M, Tripakis S (2009) Conformance testing for real-time systems. Form Methods Syst Des 34(3):238–304. https://doi.org/10.1007/s10703-009-0065-1
32. Kwiatkowska MZ, Norman G, Parker D (2011) PRISM 4.0: verification of probabilistic real-time systems. In: 23rd international conference on computer aided verification (CAV). Lecture notes in computer science, vol 6806. Springer, pp 585–591. https://doi.org/10.1007/978-3-642-22110-1_47
33. Larsen KG, Mikucionis M, Nielsen B (2004) Online testing of real-time systems using uppaal. In: 4th international workshop on formal approaches to software testing (FATES). Lecture notes in computer science, vol 3395. Springer, pp 79–94. https://doi.org/10.1007/978-3-540-31848-4_6
34. Larsen KG, Mikucionis M, Nielsen B (2009) Uppaal Tron user manual. CISS, BRICS, Aalborg University, Aalborg
35. Larsen KG, Skou A (1989) Bisimulation through probabilistic testing. In: Sixteenth annual ACM symposium on principles of programming languages (POPL). ACM Press, pp 344–352. https://doi.org/10.1145/75277.75307
36. Legay A, Sedwards S, Traonouez LM (2016) Plasma Lab: a modular statistical model checking platform. In: 7th international symposium on leveraging applications of formal methods, verification and validation: foundational techniques (ISoLA). Lecture notes in computer science, vol 9952, pp 77–93. https://doi.org/10.1007/978-3-319-47166-2_6
37. Milner R (1980) A calculus of communicating systems. Lecture notes in computer science, vol 92. Springer. https://doi.org/10.1007/3-540-10235-3
38. Moon TK (1996) The expectation–maximization algorithm. IEEE Signal Process Mag 13(6):47–60

39. Nie J, Demmel J, Gu M (2008) Global minimization of rational functions and the nearest GCDs. J Global Optim 40(4):697–718. https://doi.org/10.1007/s10898-006-9119-8
40. Núñez M, Rodríguez I (2003) Towards testing stochastic timed systems. In: 23rd IFIP WG 6.1 international conference on formal techniques for networked and distributed systems (FORTE). Lecture notes in computer science, vol 2767. Springer, pp 335–350. https://doi.org/10.1007/978-3-540-39979-7_22
41. Schuts M, Hooman J, Vaandrager FW (2016) Refactoring of legacy software using model learning and equivalence checking: An industrial experience report. In: 12th international conference on integrated formal methods (IFM). Lecture notes in computer science, vol 9681. Springer, pp 311–325. https://doi.org/10.1007/978-3-319-33693-0_20
42. Segala R (1995) Modeling and verification of randomized distributed real-time systems. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA
43. Song L, Zhang L, Godskesen JC (2012) Late weak bisimulation for Markov automata. CoRR. arXiv:1202.4116
44. Stoelinga M (2002) Alea Jacta Est: verification of probabilistic, real-time and parametric systems. Ph.D. thesis, University of Nijmegen, Nijmegen, The Netherlands
45. Stokkink WGJ, Timmer M, Stoelinga M (2013) Divergent quiescent transition systems. In: 7th international conference on tests and proofs (TAP). Lecture notes in computer science, vol 7942. Springer. https://doi.org/10.1007/978-3-642-38916-0_13
46. Thrun S, Burgard W, Fox D (2005) Probabilistic robotics. MIT Press, Cambridge
47. Timmer M, Brinksma E, Stoelinga M (2011) Model-based testing. In: Software and systems safety—specification and verification, NATO science for peace and security series—D: information and communication security, vol 30. IOS Press, pp 1–32. https://doi.org/10.3233/978-1-60750-711-6-1
48. Tretmans J (1996) Conformance testing with labelled transition systems: implementation relations and test generation. Comput Netw ISDN Syst 29(1):49–79. https://doi.org/10.1016/S0169-7552(96)00017-7
49. Tretmans J (2008) Model based testing with labelled transition systems. In: Formal methods and testing, an outcome of the FORTEST network, revised selected papers. Lecture notes in computer science, vol 4949. Springer, pp 1–38. https://doi.org/10.1007/978-3-540-78917-8_1
50. Utting M, Pretschner A, Legeard B (2012) A taxonomy of model-based testing approaches. Softw Test Verif Reliab 22(5):297–312. https://doi.org/10.1002/stvr.456
51. Vaandrager FW (2017) Model learning. Commun ACM 60(2):86–95. https://doi.org/10.1145/2967606
52. van Glabbeek RJ, Smolka SA, Steffen B, Tofts CMN (1990) Reactive, generative, and stratified models of probabilistic processes. In: Fifth annual symposium on logic in computer science (LICS). IEEE Computer Society, pp 130–141. https://doi.org/10.1109/LICS.1990.113740
53. Volpato M, Tretmans J (2014) Active learning of nondeterministic systems from an ioco perspective. In: 6th international symposium on leveraging applications of formal methods, verification and validation. Technologies for mastering change (ISoLA). Lecture notes in computer science, vol 8802. Springer, pp 220–235. https://doi.org/10.1007/978-3-662-45234-9_16
54. Younes HLS, Simmons RG (2002) Probabilistic verification of discrete event systems using acceptance sampling. In: 14th international conference on computer aided verification (CAV). Lecture notes in computer science, vol 2404. Springer, pp. 223–235. https://doi.org/10.1007/3-540-45657-0_17