Model-Based Testing for General Stochastic Time

Marcus Gerhold(✉), Arnd Hartmanns, and Mariëlle Stoelinga

University of Twente, Enschede, The Netherlands
{m.gerhold,a.hartmanns}@utwente.nl, marielle@cs.utwente.nl

Abstract. Many systems are inherently stochastic: they interact with unpredictable environments or use randomised algorithms. Then classical model-based testing is insufficient: it only covers functional correctness. In this paper, we present a new model-based testing framework that additionally covers the stochastic aspects in hard and soft real-time systems. Using the theory of stochastic automata for specifications, test cases and a formal notion of conformance, it provides clean mechanisms to represent underspecification, randomisation, and stochastic timing. Supporting arbitrary continuous and discrete probability distributions, the framework generalises previous work based on purely Markovian models. We cleanly define its theoretical foundations, and then outline a practical algorithm for statistical conformance testing based on the Kolmogorov-Smirnov test. We exemplify the framework's capabilities and tradeoffs by testing timing aspects of the Bluetooth device discovery protocol.

1 Introduction

Model-based testing (MBT) [29] 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, usually 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* [28]. IOTS partition the observable actions of the IUT (and thus of the model and test cases)

This work is supported by projects 3TU.BSR, NWO BEAT and NWO SUMBAT.

A. Dutle et al. (Eds.): NFM 2018, LNCS 10811, pp. 203–219, 2018.
https://doi.org/10.1007/978-3-319-77935-5_15

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 models may 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 only exhibits 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 [26] allows the automation of complex tasks via simple randomised strategies (as seen in e.g. vacuuming and lawn mowing robots), and we see a proliferation of probabilistic programming languages [15]. 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 [18]. 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 a new MBT framework based on input-output stochastic automata (IOSA) [9], which are transition systems augmented with discrete probabilistic choices and timers whose expiration is governed by general probability distributions. By using IOSA models, we can quantitatively specify stochastic aspects of a system, in particular w.r.t. timing. We support discrete as well as continuous probability distributions, so our framework is suitable for both hard and soft real-time requirements. Since IOSA extend transition systems, nondeterminism is available for underspecification as usual. Test cases are IOSA, too, so they can naturally include waiting. We formally define the notions of stochastic ioco (*sa-ioco*), and of test cases as a restriction of IOSA (Sect. 3). We then outline practical algorithms for test generation and *sa-ioco* conformance testing (Sect. 4). The latter combines per-trace functional verdicts as in standard ioco with a statistical evaluation that builds upon the Kolmogorov-Smirnov test [17]. While our theory of IOSA and *sa-ioco* is very general w.r.t. supported probability distributions and nondeterminism, we need to assume some restrictions to arrive at practically feasible algorithms. We finally exemplify our framework's capabilities and its inherent tradeoffs by testing timing aspects of different implementation variants of the Bluetooth device discovery protocol (Sect. 5).

Related Work. Our new *sa-ioco* framework generalises two previous stochastic MBT approaches: the *pioco* framework [13] for probabilistic automata (or Markov decision processes) and *marioco* [14] for Markov automata (MA [12],

which extend continuous-time Markov chains with nondeterminism). The former only supports discrete probabilistic choices and has no notion of time at all. The latter operates under the assumption that all timing is memoryless, i.e. all delays are exponentially distributed and fully characterised by means.

Early influential work had only deterministic time [3,19,21], later extended with timeouts/quiescence [4]. Probabilistic testing preorders and equivalences are well-studied [7,10,24]. Probabilistic bisimulation via hypothesis testing was first introduced in [20]. Our work is largely influenced by [5], which introduced a way to compare trace frequencies with collected samples. Closely related is work on stochastic finite state machines [16,23]: stochastic delays are specified similarly, but discrete probability distributions over target states are not included.

2 Background

Notation. $\mathbb{R}^+$ and $\mathbb{R}_0^+$ are the positive and non-negative real numbers. For a given set Ω, its powerset is $\mathcal{P}(\Omega)$. A multiset is written as $\{\!|\ \ldots\ |\!\}$. $\mathrm{Dist}(\Omega)$ is the set of *probability distributions* over Ω: functions $\mu \in \Omega \to [0,1]$ s.t. $\mathrm{support}(\mu) \stackrel{\text{def}}{=} \{\, \omega \in \Omega \mid \mu(\omega) > 0 \,\}$ is countable and $\sum_{\omega \in \mathrm{support}(\mu)} \mu(\omega) = 1$. Ω is *measurable* if it is endowed with a σ-algebra $\sigma(\Omega)$: a collection of *measurable* subsets of Ω. A *probability measure* over Ω is a function $\mu \in \sigma(\Omega) \to [0,1]$ s.t. $\mu(\Omega) = 1$ and $\mu(\cup_{i \in I} B_i) = \sum_{i \in I} \mu(B_i)$ for any countable index set I and pairwise disjoint measurable sets $B_i \subseteq \Omega$. $\mathrm{Prob}(\Omega)$ is the set of probability measures over Ω. Each $\mu \in \mathrm{Dist}(\Omega)$ induces a probability measure. Let $\mathit{Val} \stackrel{\text{def}}{=} V \to \mathbb{R}_0^+$ be the set of valuations for an (implicit) set V of (non-negative real-valued) variables. $\mathbf{0} \in \mathit{Val}$ assigns value zero to all variables. For $X \subseteq V$ and $v \in \mathit{Val}$, we write $v[X]$ for the valuation defined by $v[X](x) = 0$ if $x \in X$ and $v[X](y) = v(y)$ otherwise. For $t \in \mathbb{R}_0^+$, $v + t$ is the defined by $(v+t)(x) = v(x) + t$ for all $x \in V$.

Stochastic automata extend Markov decision processes with stochastic *clocks*: real-valued variables that increase synchronously with rate 1 over time and expire some random amount of time after they have been *restarted*. We define SA with input/output actions along the lines of [9]:

Definition 1. *An* input-output stochastic automaton *(IOSA) is a 6-tuple* $\mathcal{I} = \langle \mathit{Loc}, \mathcal{C}, \mathcal{A}, E, F, \ell_{\mathit{init}} \rangle$ *where* Loc *is a countable set of* locations, $\mathcal{C}$ *is a finite set of* clocks, $\mathcal{A} = \mathcal{A}^I \uplus \mathcal{A}^O$ *is the finite* action alphabet *partitioned into* inputs *in* $\mathcal{A}^I$ *(marked by a* ? *suffix) and* outputs *in* $\mathcal{A}^O$ *(marked by a* ! *suffix),* $E \in \mathit{Loc} \to \mathcal{P}(\mathit{Edges})$ *with* $\mathit{Edges} \stackrel{\text{def}}{=} \mathcal{P}(\mathcal{C}) \times \mathcal{A} \uplus \{\, \tau, \delta \,\} \times \mathrm{Dist}(T)$ *and* $T \stackrel{\text{def}}{=} \mathcal{P}(\mathcal{C}) \times \mathit{Loc}$ *is the* edge function *mapping each location to a finite set of edges that in turn consist of a* guard *set, a* label *that may be the* internal action τ *or quiescence* δ*, and a distribution over* targets *in* T *consisting of a* restart *set of clocks and target locations,* $F \in \mathcal{C} \to \mathrm{Prob}(\mathbb{R}_0^+)$ *is the* delay measure function *that maps each clock to a probability measure, and* $\ell_{\mathit{init}} \in \mathit{Loc}$ *is the* initial location. $\mathcal{I}$ *is* input-enabled *if* $\forall\, \ell \in \mathit{Loc}, a \in \mathcal{A}^I\, \exists\, \mu\colon \langle \varnothing, a, \mu \rangle \in E(\ell)$. $\mathcal{I}$ *is* closed *if* $\mathcal{A}^I = \varnothing$.

We also write $\ell \xrightarrow{G,a}_E \mu$ for $\langle G, a, \mu \rangle \in E(\ell)$. Whenever an IOSA $\mathcal{I}_i$ or $\mathcal{S}_i$ (where index i may be absent) is given in the remainder of this paper, it has the form $\langle Loc_i, \mathcal{C}_i, \mathcal{A}_i, E_i, F_i, \ell_{init_i} \rangle$ unless noted otherwise. Intuitively, a stochastic automaton starts its execution in the initial location with all clocks expired. An edge $\ell \xrightarrow{G,a}_E \mu$ may be taken only if all clocks in its guard set G are expired. If *any* output edge (i.e. with $a \in \mathcal{A}^O$) is enabled, *some* edge must be taken (i.e. all outputs are *urgent*). When an edge is taken, (1) its action is a, (2) we select a target $\langle R, \ell' \rangle \in T$ randomly according to μ, (3) all clocks in R are restarted and other expired clocks remain expired, and (4) we move to successor location ℓ'. 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 $F(c)$.

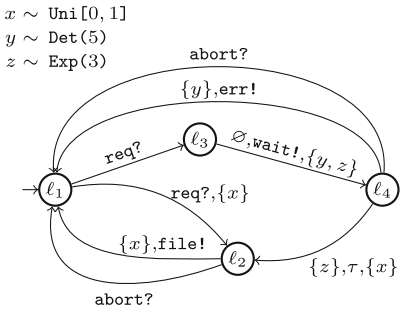


Fig. 1. File server specification.

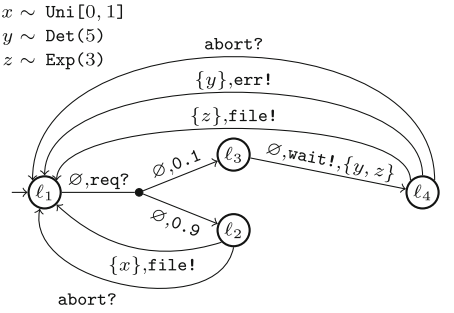


Fig. 2. File server implementation.

Example 1. Figure 1 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 ℓ_1, an implementation may either move to ℓ_2 or ℓ_3. The latter 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 ℓ_4, 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 ℓ_2. 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. 2, we show an implementation of this specification. 1 out of 10 files randomly requires to be fetched from the archive. This is allowed by the specification: it is one particular (randomised) resolution of the nondeterministic choice, i.e. underspecification, defined in ℓ_1. The implementation also manages to transmit files from archive directly while fetching them, as evidenced by the direct edge from ℓ_4 back to ℓ_1 labelled `file!`. This violates the timing prescribed by the specification, and must be detected by an MBT procedure for IOSA.

Definition 2. *Given IOSA $\mathcal{I}_1, \mathcal{I}_2$ with $\mathcal{C}_1 \cap \mathcal{C}_2 = \varnothing$, and $M \subseteq \mathcal{A}_1 \times \mathcal{A}_2$, their* parallel composition *is $\mathcal{I}_1 \parallel \mathcal{I}_2 \stackrel{\text{def}}{=} \langle Loc_1 \times Loc_2, \mathcal{C}_1 \cup \mathcal{C}_2, \mathcal{A}_\parallel, E_\parallel, F_1 \cup F_2, \langle \ell_{init_1}, \ell_{init_2} \rangle \rangle$ where $\mathcal{A}_\parallel \stackrel{\text{def}}{=} \mathcal{A}_\parallel^I \uplus \mathcal{A}_\parallel^O$ with outputs $\mathcal{A}_\parallel^O = \mathcal{A}_1^O \cup \mathcal{A}_2^O$ and inputs $\mathcal{A}_\parallel^I = \big(\mathcal{A}_1^I \cup \mathcal{A}_2^I\big) \setminus \big(\{\, a_I \in \mathcal{A}_1^I \mid \exists\, a_O \in \mathcal{A}_2^O \colon \langle a_I, a_O \rangle \in M \,\} \cup \{\, a_I \in \mathcal{A}_2^I \mid \exists\, a_O \in \mathcal{A}_1^O \colon \langle a_O, a_I \rangle \in M \,\}\big)$ and $E_\parallel$ is the smallest edge function satisfying the inference rules*

$$\frac{\ell_1 \xrightarrow{G,a}_{E_1} \mu \qquad a = \tau \vee \nexists\, a_2 \in \mathcal{A}_2 \colon \langle a, a_2 \rangle \in M}{\langle \ell_1, \ell_2 \rangle \xrightarrow{G,a}_{E_\parallel} \{\, \langle R, \langle \ell_1', \ell_2 \rangle \rangle \mapsto \mu(\langle R, \ell_1' \rangle) \mid R \subseteq \mathcal{C}, \ell_1' \in Loc_1 \,\}} \; (indep_1)$$

$$\frac{\ell_1 \xrightarrow{G_1,a_1}_{E_1} \mu_1 \qquad \ell_2 \xrightarrow{G_2,a_2}_{E_2} \mu_2 \qquad a_1 \in \mathcal{A}_1^O \wedge \langle a_1, a_2 \rangle \in M}{\langle \ell_1, \ell_2 \rangle \xrightarrow{G_1 \cup G_2, a_1}_{E_\parallel} \{\, \langle R_1 \cup R_2, \langle \ell_1', \ell_2' \rangle \rangle \mapsto \mu(\langle R_1, \ell_1' \rangle) \cdot \mu(\langle R_2, \ell_2' \rangle) \,\}} \; (sync_1)$$

plus symmetric rules $indep_2$ and $sync_2$ for the corresponding steps of $\mathcal{I}_2$.

We use the convention that two actions a_1 and a_2 match, i.e. $\langle a_1, a_2 \rangle \in M$, if they are the same except for the suffix (e.g. `a!` matches `a?` but not `b?` or `a!`).

Definition 3. *The* states *of IOSA $\mathcal{I}$ are $S \stackrel{\text{def}}{=} Loc \times Val \times Val$. Each $\langle \ell, v, x \rangle \in S$ consists of the current location ℓ and the values v and expiration times x of all clocks. The set of* paths *of $\mathcal{I}$ is $Paths_\mathcal{I} \stackrel{\text{def}}{=} S \times (\mathbb{R}_0^+ \times Edges \times \mathcal{P}(\mathcal{C}) \times S)^\omega$ where the first state is $\langle \ell_{init}, \mathbf{0}, \mathbf{0} \rangle$. $Paths_\mathcal{I}^{fin}$ is the set of all finite paths. For $\pi \in Paths_\mathcal{I}^{fin}$, $last(\pi)$ is its last state, and its length is the number of edges with actions $\neq \tau$.*

Definition 4. *A* scheduler *of a closed IOSA $\mathcal{I}$ is a measurable function $\mathfrak{S} \in Sched(\mathcal{I}) \stackrel{\text{def}}{=} Paths_\mathcal{I}^{fin} \to Dist(Edges \cup \{\, \bot \,\})$ such that $\mathfrak{S}(\pi)(\langle G, a, \mu \rangle) > 0$ with $last(\pi) = \langle \ell, v, x \rangle$ implies $\ell \xrightarrow{G,a} \mu$ and $\mathrm{Ex}(G, v + t, x)$ where $t \in \mathbb{R}_0^+$ is the minimal delay for which $\nexists\, t' \in [0, t[\colon \bigvee_{\ell \xrightarrow{G',a'}_{\mu'}} \mathrm{Ex}(G, v + t', x)$. We define $\mathrm{Ex}(G, v, x) \stackrel{\text{def}}{=} \forall\, c \in G \colon v(c) \geq x(c)$, i.e. all clocks in G are expired. $\mathfrak{S}(\pi)(\bot)$ is the probability to* halt. *$\mathfrak{S}$ is of length $k \in \mathbb{N}$ if $\mathfrak{S}(\pi)(\bot) = 1$ for all paths π of length $\geq k$. $Sched(\mathcal{I}, k)$ is the set of all schedulers of $\mathcal{I}$ of length k.*

A scheduler can only choose between the edges enabled at the points where *any* edge just became enabled in a closed IOSA. It removes all nondeterminism. The probability of each step on a path is then given by the *step probability function*:

Definition 5. *Given IOSA $\mathcal{I}$ and $\mathfrak{S} \in Sched(\mathcal{I})$, the* step probability function

$$Pr_\mathfrak{S} \in Paths_\mathcal{I}^{fin} \to \mathrm{Prob}(\{\, \bot \,\} \cup (\mathbb{R}_0^+ \times Edges \times \mathcal{P}(\mathcal{C}) \times S))$$

is defined by $Pr_\mathfrak{S}(\pi)(\bot) = \mathfrak{S}(\pi)(\bot)$ and, for π with $last(\pi) = \langle \ell, v, x \rangle$,

$$Pr_\mathfrak{S}(\pi)([t_1, t_2] \times E_{Pr} \times C_{Pr} \times S_{Pr}) = \mathbb{1}_{t \in [t_1, t_2]} \cdot \sum_{e = \langle G, a, \mu \rangle \in E_{Pr}} \mathfrak{S}(\pi)(e) \cdot \sum_{C \in C_{Pr}, \ell' \in Loc} \mu(\langle C, \ell' \rangle) \cdot \int_{\langle \ell', v', x' \rangle \in S_{Pr}} X_C^x(v', x')$$

where t is the minimal delay in ℓ as in Definition 4 and

$$X_C^x(v', x') = \mathbb{1}_{v' = (v+t)[C]} \prod_{c \in \mathcal{C}} \begin{cases} 1 & \text{if } c \notin C \wedge x(c) = x'(c) \\ 0 & \text{if } c \notin C \wedge x(c) \neq x'(c) \\ F(c)(t_2) - F(c)(t_1) & \text{if } c \in C. \end{cases}$$

The step probability function induces a probability measure over $Paths_\mathcal{I}$. As is usual, we restrict to schedulers that let time diverge with probability 1.

A path lets us follow exactly how an IOSA was traversed. Traces represent the knowledge of external observers. In particular, they cannot see the values of individual clocks, but only the time passed since the run started. Formally:

Definition 6. *The* trace *of a (finite) path π is its projection $tr(\pi)$ to the delays in $\mathbb{R}_0^+$ and the actions in $\mathcal{A}$. τ-steps are omitted and their delays are added to that of the next visible action. The set of traces of $\mathcal{I}$ is $Traces_{\mathcal{I}}$. An* abstract trace *in $AbsTraces_{\mathcal{I}}$ is a sequence $\Sigma = I_1\, a_1\, I_2\, a_2 \ldots$ with the I_i closed intervals over $\mathbb{R}_0^+$. Finite (abstract) traces are defined analogously. $Traces_{\mathcal{I}}^{\max}$ is the set of maximal finite traces for $\mathcal{I}$ with terminal locations. σ represents the set of traces $\{\, t_1\, a_1 \ldots \mid t_i \in I_i \,\}$. We identify trace $t_1\, a_1 \ldots$ with abstract trace $[0, t_1]\, a_1 \ldots$.*

We can define the *trace distribution* for an IOSA $\mathcal{I}$ and a scheduler as the probability measure over traces (using abstract traces to construct the corresponding σ-algebra) induced by the probability measure over paths in the usual way. The set of all finite trace distributions is $Trd(\mathcal{I})$. It induces an equivalence relation $\equiv_{TD}$: two IOSA $\mathcal{I}$ and $\mathcal{S}$ are trace distribution equivalent, written $\mathcal{I} \equiv_{TD} \mathcal{S}$, if and only if $Trd(\mathcal{I}) = Trd(\mathcal{S})$. A trace distribution is of length $k \in \mathbb{N}$ if it is based on a scheduler of length k. The set of all such trace distributions is $Trd(\mathcal{I}, k)$.

3 Stochastic Testing Theory

We define the theoretical concepts of our MBT framework: test cases, the *sa-ioco* conformance relation, the evaluation of test executions, and correctness notions. The specifications $\mathcal{S}$ are IOSA as in Definition 1, and we equally assume the IUT to be an input-enabled IOSA $\mathcal{I}$ with the same alphabet as $\mathcal{S}$.

3.1 Test Cases

A test case describes the possible behaviour of a tester. The advantage of MBT over manual testing is that test cases can be automatically generated from the specification, and automatically executed on an implementation. In each step of the execution, the tester may either (1) send an input to the IUT, (2) wait to observe output, or (3) stop testing. A single test may provide multiple options, giving rise to multiple concrete testing sequences. It may also prescribe different reactions to different outputs. Formally, test cases for a specification $\mathcal{S}$ are IOSA whose inputs are the outputs of $\mathcal{S}$ and vice-versa. The parallel composition of either $\mathcal{S}$ or $\mathcal{I}$ with a test case thus results in a closed IOSA. By including discrete probability distributions on edges, IOSA allow making the probabilities of the three choices (input, wait, stop) explicit[1]. Moreover, we can use clocks for explicit waiting times in test cases. Sending input can hence be delayed, which is especially beneficial to test race conditions. A test can also react to *no* output being supplied, modelled by *quiescence* δ, and check if that was expected.

[1] Tests are often *implicitly* generated probabilistically in classic *ioco* settings, too, without the support to make this explicit in the underlying theory. We fill this gap.

Definition 7. *A* test $\mathcal{T}$ *for a specification* $\mathcal{S}$ *with alphabet* $\mathcal{A}^I \uplus \mathcal{A}^O$ *is an IOSA* $\langle Loc, \mathcal{C}, \mathcal{A}^O \uplus \mathcal{A}^I, E, F, \ell_{init} \rangle$ *that has the specification's outputs as inputs and vice-versa, and that is a finite, internally deterministic, acyclic and connected tree such that for every location* $\ell \in Loc$*, we either have* $E(\ell) = \varnothing$ *(stop testing), or* $\forall\, \ell \xrightarrow{G,a} \mu \colon a = \tau$ *(an internal decision), or if* $\exists\, \ell \xrightarrow{G,a} \mu \colon a \in \mathcal{A}^I \cup \{\delta\}$ *(we can send an input or observe quiescence) then:* $\forall\, a_O \in \mathcal{A}^O \colon \exists\, G', \mu' \colon \ell \xrightarrow{G',a_O} \mu'$ *(all outputs can be received) and* $\forall\, \ell \xrightarrow{G',a'} \mu' \colon a' \in \mathcal{A}^O \vee a' = a$ *(we cannot send a different input or observe quiescence in addition to an input). Whenever* $\mathcal{T}$ *sends an input, this input must be present in* $\mathcal{S}$*, too, i.e.*

$$\forall\, \sigma \in Traces^{\text{fin}}_{\mathcal{T}} \text{ with } \sigma = \sigma_1\, t\, a\, \sigma_2 \text{ and } a \in \mathcal{A}^I \colon \sigma_1\, t\, a \in Traces^{\text{fin}}_{\mathcal{S}}.$$

3.2 Stochastic Input-Output Conformance and Annotations

Trace distribution equivalence $\equiv_{TD}$ is the probabilistic counterpart of *trace equivalence* for transition systems: it shows that there is a way for the traces of two different models, e.g. the IOSA $\mathcal{S}$ and $\mathcal{I}$, to all have the same probability via *some* resolution of nondeterminism. However, trace equivalence or inclusion is too fine as a conformance relation for testing [27]. The *ioco* relation [28] for *functional* conformance solves the problem of fineness by allowing underspecification of functional behaviour: an implementation $\mathcal{I}$ is *ioco*-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 $\mathcal{S}$. Formally:

$$\mathcal{I} \sqsubseteq_{ioco} \mathcal{S} \Leftrightarrow \forall\, \sigma \in Traces^{\text{fin}}_{\mathcal{S}} \colon out_{\mathcal{I}}(\sigma) \subseteq out_{\mathcal{S}}(\sigma)$$

where $out_{\mathcal{I}}(\sigma)$ is the set of outputs in $\mathcal{I}$ that is enabled after the trace σ.

Stochastic ioco. To extend ioco testing to IOSA, we need an auxiliary concept that mirrors trace prefixes stochastically: Given a trace distribution D of length k, and a trace distribution D' of length greater or equal than k, we say D is a *prefix* of D', written $D \sqsubseteq_k D'$, if both assign the same probability to all abstract traces of length k. We can then define:

Definition 8. *Let* $\mathcal{S}$ *and* $\mathcal{I}$ *be two IOSA. We say* $\mathcal{I}$ *is sa-ioco-conforming to* $\mathcal{S}$*, written* $\mathcal{I} \sqsubseteq^{sa}_{ioco} \mathcal{S}$*, if and only if for all tests* $\mathcal{T}$ *for* $\mathcal{S}$ *we have* $\mathcal{I} \,\|\, \mathcal{T} \sqsubseteq_{TD} \mathcal{S} \,\|\, \mathcal{T}$*.*

Intuitively, $\mathcal{I}$ is conforming if, no matter how it resolves nondeterminism (i.e. underspecification) under a concrete test, $\mathcal{S}$ can mimic its behaviour by resolving nondeterminism for the same test, such that all traces have the same probability.

The original *ioco* relation takes an experiment derived from the specification and executes it on the implementation. While Definition 8 does not directly mirror this, the property implicitly carries over from the parallel composition with tests specifically *designed for* specifications: an input is provided to the IUT if that input is present in the specification model only.

Open schedulers. The above difference in approach between *ioco* and *sa-ioco* is due to schedulers and their resulting trace distributions being solely defined for closed systems in our work (cf. Definition 4). An alternative is to also define

them for open systems. However, where schedulers for closed systems choose discretely between possible actions, their counterparts for open systems additionally schedule over continuous time, i.e. *when* an action is to be taken. This poses an additional layer of difficulty in tracing which scheduler was likely used to resolve nondeterminism, which we need to do in our *a posteriori* statistical analysis of the testing results (see Sect. 3.3).

Moreover, it is known [1,6,25] that trace distributions of probabilistic systems under "open" schedulers are not compositional in general, i.e. $A \sqsubseteq_{TD} B$ does not imply $A \parallel C \sqsubseteq_{TD} B \parallel C$. This would mean that, even when an implementation conforms to a specification, the execution of a probabilistic test case might tamper with the observable probabilities and yield an untrustworthy verdict. A general counterexample for the above implication is presented in [25], where however there is no requirement on input-enabledness of the composable systems. Our framework requires both implementation and test case to be input-enabled, cf. Definitions 7 and 8. The authors of [1] provide a counterexample for synchronous systems even in the presence of input-enabledness. Our framework works with input-enabled *asynchronous* systems; we thus believe that *sa-ioco* could also be defined in a way that more closely resembles the original definition of *ioco* by using open schedulers, but care has to be taken in defining those schedulers in the right way. We thus designed *sa-ioco* conservatively such that it is only based on trace semantics of closed systems, while still maintaining the possibility of underspecification as in *ioco* due to the way tests are used.

Annotations. To assess whether observed behaviour is functionally correct, each complete trace of a test is annotated with a verdict: all leaf locations of test cases are labelled with either *pass* or *fail*. We annotate exactly the traces that are present in the specification with the *pass* verdict; formally:

Definition 9. *Given a test $\mathcal{T}$ for specification $\mathcal{S}$, its* test annotation *is the function $ann \in Traces_{\mathcal{T}}^{\max} \to \{pass, fail\}$ such that $ann(\sigma) = fail$ if and only if $\exists\, \varrho \in Traces_{\mathcal{S}}^{\text{fin}}, t \in \mathbb{R}_0^+, a \in \mathcal{A}^O \colon \varrho\, t\, a$ is a prefix of $\sigma \wedge \varrho\, t\, a \notin Traces_{\mathcal{S}}^{\text{fin}}$.*

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

Example 2. Figure 3 presents three test cases for the file server specification of Ex. 1. $\mathcal{T}_1$ uses the quiescence observation δ to assure no output is given in the initial state. $\mathcal{T}_2$ tests for *eventual* delivery of the file, which may be in archive, requiring the intermediate `wait!` notification, or may be sent directly. $\mathcal{T}_3$ utilises a clock on the `abort!` transition: it waits for some time (depending on what $\mathcal{T}_3$ specifies for $F(x)$) before sending the input. This highlights the ability to test for race conditions, or for the possibility of a file arrival before a specified time.

3.3 Test Execution and Sampling

We test stochastic systems: executing a test case $\mathcal{T}$ once is insufficient to establish *sa-ioco* conformance. We need many executions for an overall statistical verdict

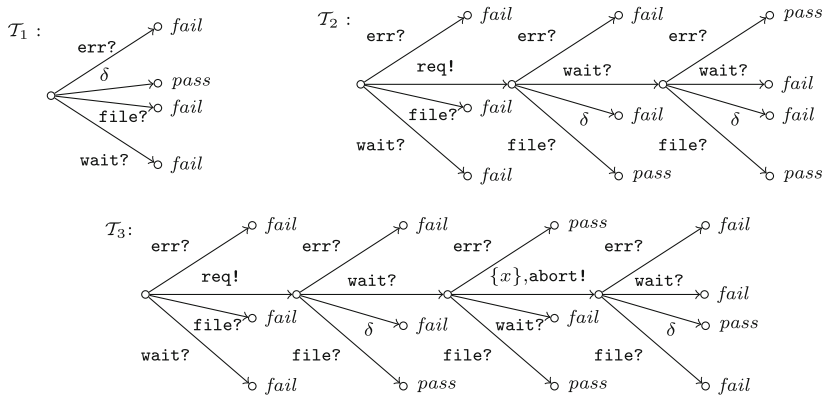


Fig. 3. Three test cases $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ for the file server specification.

about the stochastic behaviour in addition to the functional verdict obtained from the annotation on each execution. As establishing the functional verdict is the same as in standard ioco testing, we focus on the statistical evaluation here.

Sampling. We perform a statistical hypothesis test on the implementation based on the outcome of a push-button experiment in the sense of [22]. Before the experiment, we fix the parameters for sample length $k \in \mathbb{N}$ (the length of the individual test executions), sample width $m \in \mathbb{N}$ (how many test executions to observe), and level of significance $\alpha \in (0, 1)$. The latter is a limit for the *statistical error of first kind*, i.e. the probability of rejecting a correct implementation. The statistical analysis is performed after collecting the sample for the chosen parameters, while functional correctness is checked during the sampling process.

Frequencies. Our goal is to determine the deviation of a sample of traces $O = \{\, \sigma_1, \ldots, \sigma_m \,\}$ taken from $\mathcal{I} \parallel \mathcal{T}$ vs. the results expected for $\mathcal{S} \parallel \mathcal{T}$. If it is too large, O was likely not generated by an IOSA conforming to $\mathcal{S}$ and we reject $\mathcal{I}$. If the deviation is within bounds depending on k, m and α, we have no evidence to suspect an underlying IOSA other than $\mathcal{S}$ and accept $\mathcal{I}$ as a conforming IUT. We compare the frequencies of traces in O with their probabilities according to $\mathcal{S} \parallel \mathcal{T}$. Since $\mathcal{I}$ is a concrete implementation, the scheduler is the same for all executions, resulting in trace distribution D for $\mathcal{I} \parallel \mathcal{T}$ and the probability of abstract trace Σ is given directly by $D(\Sigma)$. We define $\mathit{freq}_O(\Sigma) \stackrel{\text{def}}{=} |\{\!|\, \sigma \in O \mid \sigma \in \Sigma \,|\!\}|/m$, i.e. the fraction of traces in O that are in Σ. $\mathcal{I}$ is rejected on statistical evidence if the distance of the two measures D and freq_O exceeds a threshold based on α.

Acceptable outcomes. We accept a sample O if freq_O lies within some radius, say r_α, around D. To minimise the error of false acceptance, we choose the smallest r_α that guarantees that the error of false rejection is not greater than α, i.e.

$$r_\alpha \stackrel{\text{def}}{=} \inf \{\, r \in \mathbb{R}^+ \mid D(\mathit{freq}^{-1}(B_r(D))) > 1 - \alpha \,\}, \tag{1}$$

where $B_y(x)$ is the closed ball centred at $x \in X$ with radius $y \in \mathbb{R}^+$ and X a metric space. The set of all measures defines a metric space together with the total variation distance of measures $dist(u, v) \stackrel{\text{def}}{=} \sup_{\sigma \in (\mathbb{R}_0^+ \times \mathcal{A})^k} |u(\sigma) - v(\sigma)|$.

Definition 10. *For k, $m \in \mathbb{N}$ and $\mathcal{I} \parallel \mathcal{T}$, the* observations *under a trace distribution $D \in Trd(\mathcal{I} \parallel \mathcal{T}, k)$ of level of significance $\alpha \in (0, 1)$ are given by the set*

$$Obs(D, \alpha, k, m) = \{\, O \in (\mathbb{R}_0^+ \times \mathcal{A})^{k \times m} \mid dist(freq_O, D) \leq r_\alpha \,\}.$$

The set of observations *of $\mathcal{I} \parallel \mathcal{T}$ with $\alpha \in (0, 1)$ is then given by the union over all trace distributions of length k, and is denoted $Obs(\mathcal{I} \parallel \mathcal{T}, \alpha, k, m)$.*

These sets limit the statistical error of first and second kind as follows: if a sample was generated under a trace distribution of $\mathcal{I} \parallel \mathcal{T}$ or a trace distribution equivalent IOSA, we accept it with probability higher than $1 - \alpha$; and for all samples generated under a trace distribution by non-equivalent IOSA, the chance of erroneously accepting it is smaller than some β_m, where β_m is unknown but minimal by construction, cf. (1). Note that $\beta_m \to 0$ as $m \to \infty$, i.e. the error of accepting an erroneous sample decreases as sample size increases.

3.4 Test Evaluation and Correctness

Definition 11. *Given an IOSA $\mathcal{S}$, an annotated test case $\mathcal{T}$, k and $m \in \mathbb{N}$, and a level of significance $\alpha \in (0, 1)$, we define (1) the* functional verdict *as given by $v_{func} \in IOSA^2 \to \{\, pass, fail \,\}$ where $v_{func}(\mathcal{I}, \mathcal{T}) = pass$ if and only if $\forall\, \sigma \in Traces^{\max}_{\mathcal{I} \parallel \mathcal{T}} \cap Traces^{\max}_{\mathcal{T}}$: $ann(\sigma) = pass$, and (2) the* statistical verdict *as given by $v_{prob} \in IOSA^2 \to \{\, pass, fail \,\}$ where $v_{prob}(\mathcal{I}, \mathcal{T}) = pass$ iff $\exists D \in Trd(\mathcal{S} \parallel \mathcal{T}, k)$ s.t.*

$$D(\, Obs(\mathcal{I} \parallel \mathcal{T}, \alpha, k, m) \cap Traces_{\mathcal{S} \parallel \mathcal{T}}) > 1 - \alpha.$$

$\mathcal{I}$ passes a test suite *if $v_{prob}(\mathcal{I}, \mathcal{T}) = v_{func}(\mathcal{I}, \mathcal{T}) = pass$ for all annotated test cases $\mathcal{T}$ of the test suite.*

The above definition connects the previous two subsections to implement a *correct* MBT procedure for the *sa-ioco* relation introduced in Sect. 3.2. Correctness comprises *soundness* and *completeness* (or *exhaustiveness*): the first means that every conforming implementation passes a test, whereas the latter implies that there is a test case to expose every erroneous (i.e. nonconforming) implementation. A test suite can only be considered correct with a guaranteed (high) probability $1 - \alpha$ (as inherent in Definition 11).

Definition 12. *Let $\mathcal{S}$ be a specification IOSA. Then a test case $\mathcal{T}$ is* sound *for $\mathcal{S}$ with respect to sa-ioco for every $\alpha \in (0, 1)$ iff for every input enabled IOSA $\mathcal{I}$ we have that $\mathcal{I} \sqsubseteq^{sa}_{ioco} \mathcal{S}$ implies $v_{func}(\mathcal{I}, \mathcal{T}) = v_{prob}(\mathcal{I}, \mathcal{T}) = pass$.*

Completeness of a test suite is inherently a theoretical result. Infinite behaviour of the IUT, for instance caused by loops, hypothetically requires a test suite of infinite size. Moreover, there remains a possibility of accepting an erroneous implementation by chance, i.e. making an error of second kind. However, the latter is bounded from above and decreases with increasing sample size.

Definition 13. *Let $\mathcal{S}$ be a specification IOSA. Then a test suite is called* complete *for $\mathcal{S}$ with respect to sa-ioco for every $\alpha \in (0,1)$ iff for every input-enabled IOSA $\mathcal{I}$ we have that $\mathcal{I} \not\sqsubseteq^{sa}_{ioco} \mathcal{S}$ implies the existence of a test $\mathcal{T}$ in the test suite such that $v_{func}(\mathcal{I},\mathcal{T}) = fail$ or $v_{prob}(\mathcal{S},\mathcal{T}) = fail$.*

4 Implementing Stochastic Testing

The previous section laid the theoretical foundations of our new IOSA-based testing framework. Several aspects were specified very abstractly, for which we now provide practical procedures. There are already several ways to generate, annotate and execute test cases in batch or on-the-fly in the classic *ioco* setting [28], which can be transferred to our framework. The statistical analysis of gathered sample data in MBT, on the other hand, is largely unexplored since few frameworks include probabilities or even stochastic timing. Determining verdicts according to Definition 11 requires concrete procedures to implement the statistical tests described in Sect. 3.3 with level of significance α. We now present practical methods to evaluate test cases in line with this theory. In particular, we need to find a scheduler for $\mathcal{S}$ that makes the observed traces O most likely, and test that the stochastic timing requirements are implemented correctly.

4.1 Goodness of Fit

Since our models neither comprise only *one specific distribution*, nor *one specific parameter* to test for, we resort to nonparametric goodness of fit tests. Nonparametric statistical procedures allow to test hypotheses that were designed for ordinal or nominal data [17], matching our intention of (1) testing the overall distribution of trace frequencies in a sample $O = \{\sigma_1, \ldots, \sigma_m\}$, and (2) validating that the observed delays were drawn from the specified clocks and distributions. We use Pearson's χ^2 test for (1) and multiple Kolmogorov-Smirnov tests for (2).

Pearson's χ^2 test [17] compares empirical sample data to its expectations. It allows us to check the hypothesis that observed data indeed originates from a specified distribution. The cumulative sum of squared errors is compared to a critical value, and the hypothesis is rejected if the empiric value exceeds the threshold. We can thus check whether trace frequencies correspond to a specification under a certain trace distribution. For a finite trace $\sigma = t_1\, a_1\, t_2\, a_2 \ldots t_k\, a_k$, we define its timed closure as $\bar{\sigma} \stackrel{\text{def}}{=} \mathbb{R}^+ a_1 \ldots \mathbb{R}^+ a_k$. Applying Pearson's χ^2 is done in general via $\chi^2 = \sum_{i=1}^{n} |\text{obs}_i - \text{exp}_i|^2/\text{exp}_i$, i.e. in our case

$$\chi^2 \stackrel{\text{def}}{=} \sum_{\bar{\sigma} \in \{\bar{\sigma} \mid \sigma \in O\}} \frac{(|\{\!|\, \bar{\varrho} \mid \varrho \in O \wedge \bar{\varrho} = \bar{\sigma} \,|\!\}|/m - D(\bar{\sigma}))^2}{D(\bar{\sigma})}. \tag{2}$$

We need to find a D that gives a high likelihood to a sample, i.e. such that $\chi^2 < \chi^2_{crit}$, where χ^2_{crit} depends on α and the degrees of freedom. The latter is given by the number of different timed closures in O minus 1. The critical values can be calculated or found in standard tables.

Recall that a trace distribution is based on a scheduler that resolves nondeterministic choices randomly. This turns (2) into a satisfaction problem of a

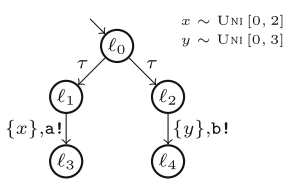
probability vector p over a rational function $f(p)/g(p)$, where f and g are polynomials. Finding a resolution such that $\chi^2 < \chi^2_{crit}$ ensures that the error of rejecting a correct IUT is at most α. This can be done via SMT solving.

The Kolmogorov-Smirnov test. While Pearson's χ^2 test assesses the existence of a scheduler that explains the observed trace frequencies, it does not take into account the observed delays. For this purpose, we use the non-parametric Kolmogorov-Smirnov test [17] (the KS test). It 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 $F_n(t) = 0$ for $t < t_1$, $F_n(t) = n_i/n$ for $t_i \leq t < t_{i+1}$, and $F_n(t) = 1$ for $t \geq t_n$ 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. Then the n-th KS statistic is given by

$$K_n \stackrel{\text{def}}{=} \sup_{t \in \mathbb{R}_0^+} |F_c(t) - F_n(t)| . \tag{3}$$

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

Example 3. The left-hand side of Fig. 4 shows a tiny example specification IOSA with clocks x and y. The expiration times of both are uniformly distributed with different parameters. The right-hand side depicts a sample from this IOSA. There are two steps to assess whether the observed data is a truthful sample of the specification with a confidence of $\alpha = 0.05$: (1) find a trace distribution that minimises the χ^2 statistic and (2) evaluate two KS tests to assess whether the observed time data is a truthful sample of $\textsc{Uni}[0, 2]$ and $\textsc{Uni}[0, 3]$, respectively.



Sampling

ID	Trace σ
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 σ
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. 4. Tiny example implementation IOSA and sample observation.

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 ℓ_0 and $1 - p$ that assigned to taking the right branch. Drawing a sample of size m, we expect $p \cdot m$ times `a!` and $(1 - p) \cdot m$ times `b!`. The empirical χ^2 value therefore calculates as

$\chi^2 = (8 - 14 \cdot p)^2/(14 \cdot p) + (6 - 14 \cdot (1-p))^2/(14 \cdot (1-p))$, which is minimal for $p = 8/14$. Since it is smaller than $\chi^2_{crit} = 3.84$, we found a scheduler that explains the observed frequencies.

Let $t_1 = 0.26, \ldots, t_8 = 1.97$ be the data associated with clock x and $t'_1 = 0.29, \ldots, t'_6 = 2.74$ be the data associated with clock y. $D_8 = 0.145$ is the maximal distance between the empirical step function of the t_i and $\textsc{Uni}[0,2]$. The critical value of the Kolmogorov distribution for $n = 8$ and $\alpha = 0.05$ is $K_{crit} = 0.46$. Hence, the inferred measure is sufficiently close to the specification. The KS test for t'_i and $\textsc{Uni}[0,3]$ can be performed analogously.

The acceptance of both the χ^2 and the KS test results in the overall statistical acceptance of the implementation based on the sample data at $\alpha = 0.05$.

Our intention is to provide general and universally applicable statistical tests. The KS test is conservative for general distributions, but can be made precise [8]. More specialised and thus efficient tests exist for specific distributions, e.g. the Lilliefors test [17] 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, and e.g. the Anderson-Darling test is an alternative.

Error propagation. A level of significance $\alpha \in (0,1)$ limits type 1 error by α. Performing several statistical experiments inflates this probability: if one experiment is performed at $\alpha = 0.05$, there is a 5% probability to incorrectly reject a true hypothesis. Performing 100 experiments, we expect to see a type 1 error 5 times. If all experiments are independent, the chance is thus 99.4%. This is the *family-wise error rate* (FWER). There are two approaches to control the FWER: *single step* and *sequential* adjustments. The most prevalent example for the first is Bonferroni correction, while a prototype of the latter is Holm's method. Both methods aim at limiting the *global* type I error in the statistical testing process.

4.2 Algorithm Outline

The overall practical procedure to perform MBT for *sa-ioco* is then as follows:

1. Generate an annotated test case $\mathcal{T}$ of length k for the specification IOSA $\mathcal{S}$.
2. Execute $\mathcal{T}$ on the IUT $\mathcal{I}$ m times. If the *fail* functional verdict is encountered in any of the m test executions then fail $\mathcal{I}$ for functional reasons.
3. Calculate the number of KS tests and e.g. adjust α to avoid error propagation.
4. Use SMT solving to find a scheduler s.t. the χ^2 statistic of the sample is below the critical value. If no scheduler is found, fail $\mathcal{I}$ for probabilistic reasons.
5. Group all time stamps assigned to the same clock and perform a KS test for each clock. If any of them fails, reject $\mathcal{I}$ for probabilistic reasons.
6. Otherwise, accept $\mathcal{I}$ as conforming to $\mathcal{S}$ according to $\mathcal{T}$.

Threats to validity. Step 5 has the potential to vastly grow in complexity if traces cannot be uniquely identified in the specification model. Recall Fig. 4 and assume $\mathtt{a}! = \mathtt{b}!$: it is now infeasible to differentiate between time values belonging to the left and to the right branch. To avoid this, we have to avoid this scenario at the time of modelling, or check *all possible combinations* of time value assignments.

5 Experiments

Bluetooth is a wireless communication protocol for low-power devices communicating over short distances. Its devices organise in small networks consisting of one *master* and up to seven *slave* devices. In this initialisation period, Bluetooth uses a frequency hopping scheme to cope with inferences. To illustrate our framework, we study the initialisation for one master and one slave device. It is inherently stochastic due to the initially random unsynchronised state of the devices. We give a high level overview and refer the reader to [11] for a detailed description and formal analysis of the protocol in a more general scenario.

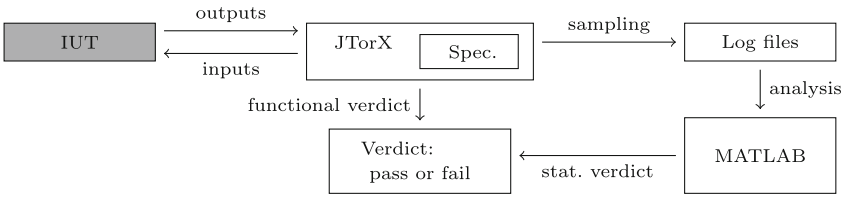


Fig. 5. Experimental setup.

Device discovery protocol. Master and slave try to connect via 32 prescribed frequencies. Both 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, which are selected according to

$$\mathit{freq} = [CLK_{16-12} + \mathit{off} + (CLK_{4-2,0} - CLK_{16-12}) \mod 16] \mod 32$$

where CLK_{i-j} marks the bits $i, \ldots, j$ of the clock and $\mathit{off} \in \mathbb{N}$ is an offset. The master switches between two *tracks* every 2.56 s. When the 12th bit of the clock changes, i.e. every 1.28 s, a frequency is swapped between the tracks. We use $\mathit{off} = 1$ for track 1 and $\mathit{off} = 17$ for track 2, i.e. the tracks initially comprise frequencies 1-16 and 17-32. The slave scans the 32 frequencies and is either in sleeping or listening state. The Bluetooth standard leaves some flexibility w.r.t. the length of the former. For our study, the slave listens for 11.25 ms every 0.64 s and sleeps for the remaining time. It picks the next frequency after 1.28 s, enough for the master to repeatedly cycle through 16 frequencies.

Experimental setup. Our toolchain is depicted in Fig. 5. The IUT is tested on-the-fly via the MBT tool JTorX [2], which generates tests w.r.t. a transition system abstraction of our IOSA specification modelling the protocol described above. JTorX returns the functional *fail* verdict if unforeseen output or a timeout (quiescence) is observed at any time throughout the test process. We chose a timeout of approx. 5.2 s in accordance with the specification. JTorX's log files comprise the sample. We implemented the protocol and three mutants in Java 7:

$\mathcal{M}_1$ Master mutant $\mathcal{M}_1$ never switches tracks 1 and 2, slowing the coverage of frequencies: new frequencies are only added in the swap every 1.28 s.

$\mathcal{M}_2$ Master mutant $\mathcal{M}_2$ never swaps frequencies, only switching between tracks 1 and 2. The expected time to connect will therefore be around 2.56 s.

$\mathcal{S}_1$ Slave mutant $\mathcal{S}_1$ has its listening period halved: it is only in a receiving state for 5.65 ms every 0.64 s.

In all cases, we expect an increase in average waiting time until connection establishment. We anticipate that the increase leads to functional *fail* verdicts due to timeouts or to stochastic *fail* verdicts based on differing connection time distributions compared to the specification. We collected $m = 100$, $m = 1000$ and $m = 10000$ test executions for each implementation, and used $\alpha = 0.05$.

Table 1. Verdicts and Kolmogorov-Smirnov test results for Bluetooth initialisation.

	Correct	Mutants		
	$\mathcal{M} \parallel \mathcal{S}$	$\mathcal{M}_1 \parallel \mathcal{S}$	$\mathcal{M}_2 \parallel \mathcal{S}$	$\mathcal{M} \parallel \mathcal{S}_1$
$k = 2$	`Accept`	`Reject`	`Accept`	`Accept`
$m = 100$	$D_m = 0.065$ $D_{crit} = 0.136$	—	$D_m = 0.110$ $D_{crit} = 0.136$	$D_m = 0.065$ $D_{crit} = 0.136$
Timeouts	0	40	0	0
$k = 2$	`Accept`	`Reject`	`Reject`	`Accept`
$m = 1000$	$D_m = 0.028$ $D_{crit} = 0.045$	—	$D_m = 0.05$ $D_{crit} = 0.045$	$D_m = 0.020$ $D_{crit} = 0.045$
Timeouts	0	399	0	0
$k = 2$	`Accept`	`Reject`	`Reject`	`Reject`
$m = 10000$	$D_m = 0.006$ $D_{crit} = 0.019$	—	$D_m = 0.043$ $D_{crit} = 0.019$	$D_m = 0.0193$ $D_{crit} = 0.0192$
Timeouts	0	3726	0	0

Results. Table 1 shows the verdicts and the observed KS statistics D_m alongside the corresponding critical values D_{crit} for our experiments. Statistical verdict `Accept` was given if $D_m < D_{crit}$, and `Reject` otherwise. Note that the critical values depend on the level of significance α and the sample size m. The correct implementation was accepted in all three experiments. During the sampling of $\mathcal{M}_1$, we observed several timeouts leading to a functional *fail* verdict. It would also also have failed the KS test in all three experiments. $\mathcal{M}_2$ passed the test for $m = 100$, but was rejected with increased sample size. $\mathcal{S}_1$ is the most subtle of the three mutants: it was only rejected with $m = 10000$ at a narrow margin.

6 Conclusion

We presented an MBT setup based on stochastic automata that combines probabilistic choices and continuous stochastic time. We instantiated the theoretical

framework with a concrete procedure using two statistical tests and explored its applicability on a communication protocol case study.

References

1. de Alfaro, L., Henzinger, T.A., Jhala, R.: Compositional methods for probabilistic systems. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 351–365. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44685-0_24
2. Belinfante, A.: JTorX: a tool for on-line model-driven test derivation and execution. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 266–270. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12002-2_21
3. Bohnenkamp, H., Belinfante, A.: Timed testing with TorX. In: Fitzgerald, J., Hayes, I.J., Tarlecki, A. (eds.) FM 2005. LNCS, vol. 3582, pp. 173–188. Springer, Heidelberg (2005). https://doi.org/10.1007/11526841_13
4. Briones, L.B., Brinksma, E.: A test generation framework for *quiescent* real-time systems. In: Grabowski, J., Nielsen, B. (eds.) FATES 2004. LNCS, vol. 3395, pp. 64–78. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31848-4_5
5. Cheung, L., Stoelinga, M., Vaandrager, F.: A testing scenario for probabilistic processes. J. ACM **54**(6), 29 (2007)
6. Cheung, L., Lynch, N., Segala, R., Vaandrager, F.: Switched PIOA: parallel composition via distributed scheduling. Theor. Comput. Sci. **365**(1), 83–108 (2006)
7. Cleaveland, R., Dayar, Z., Smolka, S.A., Yuen, S.: Testing preorders for probabilistic processes. Inf. Comput. **154**(2), 93–148 (1999)
8. Conover, W.J.: A Kolmogorov goodness-of-fit test for discontinuous distributions. J. Am. Stat. Assoc. **67**(339), 591–596 (1972)
9. D'Argenio, P.R., Lee, M.D., Monti, R.E.: Input/output stochastic automata. In: Fränzle, M., Markey, N. (eds.) FORMATS 2016. LNCS, vol. 9884, pp. 53–68. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44878-7_4
10. Deng, Y., Hennessy, M., van Glabbeek, R.J., Morgan, C.: Characterising testing preorders for finite probabilistic processes. CoRR (2008)
11. Duflot, M., Kwiatkowska, M., Norman, G., Parker, D.: A formal analysis of bluetooth device discovery. STTT **8**(6), 621–632 (2006)
12. Eisentraut, C., Hermanns, H., Zhang, L.: On probabilistic automata in continuous time. In: LICS, pp. 342–351. IEEE Computer Society (2010)
13. Gerhold, M., Stoelinga, M.: Model-based testing of probabilistic systems. In: Stevens, P., Wąsowski, A. (eds.) FASE 2016. LNCS, vol. 9633, pp. 251–268. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49665-7_15
14. Gerhold, M., Stoelinga, M.: Model-based testing of probabilistic systems with stochastic time. In: Gabmeyer, S., Johnsen, E.B. (eds.) TAP 2017. LNCS, vol. 10375, pp. 77–97. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61467-0_5
15. Gordon, A.D., Henzinger, T.A., Nori, A.V., Rajamani, S.K.: Probabilistic programming. In: FOSE, pp. 167–181. ACM (2014)
16. Hierons, R.M., Merayo, M.G., Núñez, M.: Testing from a stochastic timed system with a fault model. J. Log. Algebr. Program. **78**(2), 98–115 (2009)
17. Hollander, M., Wolfe, D.A., Chicken, E.: Nonparametric Statistical Methods. Wiley, Hoboken (2013)
18. Katoen, J.P.: The probabilistic model checking landscape. In: LICS. ACM (2016)

19. Krichen, M., Tripakis, S.: Conformance testing for real-time systems. Form. Methods Syst. Des. **34**(3), 238–304 (2009)
20. Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. ACM (1989)
21. Larsen, K.G., Mikucionis, M., Nielsen, B.: Online testing of real-time systems using Uppaal. In: Grabowski, J., Nielsen, B. (eds.) FATES 2004. LNCS, vol. 3395, pp. 79–94. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31848-4_6
22. Milner, R. (ed.): A Calculus of Communicating Systems. LNCS, vol. 92. Springer, Heidelberg (1980). https://doi.org/10.1007/3-540-10235-3
23. Núñez, M., Rodríguez, I.: Towards testing stochastic timed systems. In: König, H., Heiner, M., Wolisz, A. (eds.) FORTE 2003. LNCS, vol. 2767, pp. 335–350. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39979-7_22
24. Segala, R.: Modeling and verification of randomized distributed real-time systems. Ph.D. thesis, Cambridge, MA, USA (1995)
25. Stoelinga, M.: Alea jacta est: verification of probabilistic, real-time and parametric systems. Ph.D. thesis, Radboud University of Nijmegen (2002)
26. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT press, Cambridge (2005)
27. Tretmans, J.: Conformance testing with labelled transition systems: implementation relations and test generation. Comput. Netw. ISDN Syst. **29**(1), 49–79 (1996)
28. Tretmans, J.: Model based testing with labelled transition systems. In: Hierons, R.M., Bowen, J.P., Harman, M. (eds.) Formal Methods and Testing. LNCS, vol. 4949, pp. 1–38. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78917-8_1
29. Utting, M., Pretschner, A., Legeard, B.: A taxonomy of model-based testing approaches. Softw. Test. Verif. Reliab. **22**(5), 297–312 (2012)