ioco theory for probabilistic automata

Marcus Gerhold Mariëlle Stoelinga
University of Twente, Enschede, The Netherlands
m.gerhold@utwente.nl marielle@cs.utwente.nl

Model-based testing (MBT) is an well-known technology, which allows for automatic test case generation, execution and evaluation. To test non-functional properties, a number of test MBT frameworks have been developed to test systems with real-time, continuous behaviour, symbolic data and quantitative system aspects. Notably, a lot of these frameworks are based on Tretmans' classical input/output conformance (ioco) framework. However, a model-based test theory handling probabilistic behaviour does not exist yet. Probability plays a role in many different systems: unreliable communication channels, randomized algorithms and communication protocols, service level agreements pinning down up-time percentages, etc. Therefore, a probabilistic test theory is of great practical importance. We present the ingredients for a probabilistic variant of ioco and define the pioco relation, show that it conservatively extends ioco and define the concepts of test case, execution and evaluation.

1 Introduction

Model-based testing (MBT) is a way to test systems more effectively and more efficiently. By generating, executing and evaluating test cases automatically from a formal requirements model, more tests can be executed at a lower cost. A number of MBT tools have been developed, such as the Axini test manager, JTorx [1], STG [5], TorXakis [18], Uppaal-Tron [10, 16], etc.

A wide variety of model-based test theories exist: the seminal theory of Input/Output conformance [25, 27] is able to test functional properties, and has established itself as the robust core with a wide number of extensions. The correct functioning of today's complex cyberphysical systems, depends not only on functional behaviour, but largely on non-functional, quantitative system aspects, such as real-time and performance. MBT frameworks have been developed to support these aspects: To test timing requirements, such as deadlines, a number of timed ioco-variants have been developed, such as [2, 10, 15]. Symbolic data can be handled by the frameworks in [8, 14]; resources by [3], and hybrid aspects in [19].

This paper introduces pioco, a conservative extension of ioco that is able to handle discrete probabilities. Starting point is a requirements model as a *probabilistic quiescent transition system (pQTS)*, an input/output transition system, with two additional features: (1) *Quiescence*, which models the absence of outputs explicitly via a distinct δ label: quiescence is an important notion in ioco, because a system-under-test (SUT) may fail a certain test case given an output is required, but the SUT does not provide one. (2) *Discrete probabilistic choice*. We work in the input-generative / output-reactive model [9], which extend Segala's classical probabilistic automaton model [20]: upon receiving an input, a pQTS chooses probabilistically, which target state to move to. For outputs, a pQTS chooses probabilistically both which action to take, and which state to move to, see Figure 1 for an example.

An important contribution of our paper is the notion of test case execution and evaluation. In particular, we show how the use of statistical hypothesis testing can be exploited to determine the verdict of a test execution: if we execute a test case sufficiently many times and the observed trace frequencies do not

A.K. Petrenko, B.-H. Schlingloff, N. Pakulin (Eds.):
Tenth Workshop on Model-Based Testing (MBT 2015)
EPTCS 180, 2015, pp. 23–40, doi:10.4204/EPTCS.180.2

coincide with the probabilities described in the specification pQTS depending on a predefined level of significance, then we fail the test case. In this way, we obtain a clean framework for test case generation, evaluation and execution. However, being a first step, we mainly establish the theoretical background. Further Research is needed to implement this theory into a working tool for probabilistic testing

Related work. An early and influential paper on probabilistic testing is *Bisimulation Through Probabilistic Testing* [17], which not only defines the fundamental concept of probabilistic bisimulation, but also shows how different (i.e. non-bisimilar) probabilistic behaviours can be detected via statistical hypothesis testing. This idea has been taken further in our earlier work [4, 22], which shows how to observe trace probabilities via hypothesis testing.

Testing probabilistic Finite State Machines is well-studied (e.g. [13]) and coincidences to ioco theory can be found. However pQTS are more expressive than PFSMs, as they support non-determinism and underspecification, which both play a fundamental role in testing practice. Hence, they provide more suitable models for today's highly concurrent and cyberphysical systems.

A paper that is similar in spirit to ours is by Hierons et al. [11, 12], and also considers input reactive / output generative systems with quiescence. However, there are a few important differences: Our model can be considered as an extension of [11] reconsiling probabilistic and nondeterministic choices in a fully fledged way. Being more restrictive enables [11, 12] to focus on individual traces, whereas we use trace distributions.

Other work that involves the use of probability is given in [7, 28, 29], which models the behaviour of the tester, rather than of the SUT as we do, via probabilities.

Organization of the paper. We start by defining overall preliminaries in Section 2. Section 3 defines the conformance relation pioco for those systems and Section 4 provides the structure for testing and denotes what it means for an implementation to fail or pass a test suite by the means of an output and a statistical verdict. The paper ends with conclusions and future work in Section 5.

2 Probabilistic quiescent transition systems

2.1 Basic definitions

Definition 1. (Probability Distribution) A *discrete probability distribution* over a set X is a function $\mu : X \longrightarrow [0,1]$ such that $\sum_{x \in X} \mu(x) = 1$. The set of all distributions over X is denoted as $Distr(X)$. The probability distribution that assigns 1 to a certain element $x \in X$ is called the *Dirac* distribution over x and is denoted $Dirac(x)$.

Definition 2. (Probability Space) A *probability space* is a triple $(\Omega, \mathscr{F}, \mathbb{P})$, such that Ω is a set, $\mathscr{F}$ is a σ-field of Ω, and $\mathbb{P} : \mathscr{F} \to [0,1]$ a probability measure such that $\mathbb{P}(\Omega) = 1$ and $\mathbb{P}(\bigcup_{i=0}^{\infty} A_i) = \sum_{i=0}^{\infty} \mathbb{P}(A_i)$ for A_i, $i = 1, 2, \ldots$ pairwise disjoint.

2.2 Probabilistic quiescent transition systems

As stated, we consider probabilistic transitions that are *input reactive* and *output generative* [9]: upon receiving an input, the system decides probabilistically which next state to move to. However, the system cannot decide probabilistically which inputs to accept. For outputs, in contrast, a system may make a probabilistic choice over various output actions. This means that each transition in a pQTS either involves

a single input action, and a probabilistic choice over the target states; or it makes a probabilistic choice over several output actions, together with their target states. We refer to Figure 1 for an example.

Moreover, we model quiescence explicitly via a δ-label. Quiescence means absence of outputs and is essential for testing: if the SUT does not provide any outputs, a test must determine whether or not this behaviour is correct. In the non-probabilistic case, this can be done either via the suspension automaton construction [26], or via QTSs [23]. The SA construction involves determinization. However, this is an ill-defined term for probabilistic systems. Therefore, we use the quiescent-labelling approach and demand to make quiescence explicit.

Finally, we assume that our pQTSs are finite and don't contain internal steps (i.e., τ-transitions).

Definition 3. (pQTS) A *probabilistic quiescent transition system* (pQTS) is an ordered five tuple $\mathscr{A} = \left(S, s_0, L_I, L_O^\delta, \Delta\right)$ where

- S a finite set of states,
- $s_0 \in S$ the initial state,
- L_I and L_O^δ disjoint sets of input and output actions, with at least $\delta \in L_O^\delta$. We write $L := L_I \cup L_O^\delta$ for the set of all labels and let $L_O = L_O^\delta \setminus \{\delta\}$ the set of all real outputs.
- $\Delta \subseteq S \times \mathit{Distr}\,(L \times S)$ a finite transition relation such that for all $(s,\mu) \in \Delta$, $a? \in L_I$, $b \in L$, $s', s'' \in S$, if $\mu\,(a?, s') > 0$, then $\mu\,(b, s'') = 0$ for all $b \neq a?$.

We write $s \xrightarrow{\mu,a} s'$ if $(s,\mu) \in \Delta$ and $\mu\,(a,s') > 0$; and $s \to a$ if there are $\mu \in \mathit{Distr}\,(L \times S)$ and $s' \in S$ such that $s \xrightarrow{\mu,a} s'$. If it is not clear from the context about which system we are talking, we will write $s \xrightarrow{\mu,a}_{\mathscr{A}} s'$, $(s,\mu)_{\mathscr{A}}$ and $s \to_{\mathscr{A}} a$ to clarify ambiguities. Lastly we say that $\mathscr{A}$ is *input enabled* if for all $s \in S$ we have $s \to a?$ for every $a \in L_I$.

2.3 Paths and traces

We define the usual language-theoretic concepts for pQTSs.

Definition 4. Let $\mathscr{A} = \left(S, s_0, L_I, L_O^\delta, \Delta\right)$ be a pQTS.

- A *path* π of a pQTS $\mathscr{A}$ is a (possibly) infinite sequence of the form

$$\pi = s_1 \mu_1 a_1 s_2 \mu_2 a_2 s_3 \mu_3 a_3 s_4 \ldots,$$

  where $s_i \in S$, $a_i \in L$ for $i = 1, 2, \ldots$ and $\mu \in \mathit{Distr}\,(L, S)$, such that each finite path ends in a state and $s_i \xrightarrow{\mu_i, a_i} s_{i+1}$ for each nonfinal i. We use the notation $\mathit{first}\,(\pi) := s_1$ to denote the first state of a path, as well as $\mathit{last}\,(\pi) := s_n$ for a finite path ending in s_n, and $\mathit{last}\,(\pi) = \infty$ for infinite paths. The set of all finite paths of a pQTS $\mathscr{A}$ is denoted by $\mathit{Path}^*\,(\mathscr{A})$ and the set of all infinite paths by $\mathit{Path}\,(\mathscr{A})$ respectively.
- The *trace* of a path $\pi = s_1 \mu_1 a_1 s_2 \mu_2 a s_3 \ldots$ is the sequence obtained by omitting everything but the action labels, i.e. $\mathit{trace}\,(\pi) = a_1 a_2 a_3 \ldots$.
- All finite traces of $\mathscr{A}$ are summarized in $\mathit{traces}\,(\mathscr{A}) = \{\mathit{trace}\,(\pi) \in L^* \mid \pi \in \mathit{Path}^*\,(\mathscr{A})\}$.
- We write $s_1 \stackrel{\sigma}{\Rightarrow} s_n$ with $\sigma \in L^*$ for $s_1, s_n \in S$ in case there is a path $\pi = s_1 \mu_1 a_1 \ldots \mu_{n-1} a_{n-1} s_n$ with $\mathit{trace}\,(\pi) = \sigma$ and $s_i \xrightarrow{\mu_i, a_i} s_{i+1}$ for $i = 1, \ldots, n-1$.
- We write $\mathit{reach}_{\mathscr{A}}\,(S', \sigma)$ for the set of reachable states of a subset $S' \subseteq S$ via σ, i.e. $\mathit{reach}_{\mathscr{A}}\,(S', \sigma) = \left\{ s \in S \mid \exists s' \in S' : s' \stackrel{\sigma}{\Rightarrow} s \right\}$.

- All complete initial traces of $\mathscr{A}$ are denoted by $ctraces(\mathscr{A})$, which is defined as the set

$$\{trace(\pi) \mid \pi \in Path(\mathscr{A}) : first(\pi) = s_0, |\pi| = \infty \vee \forall a \in L : reach_{\mathscr{A}}(last(\pi), a) = \emptyset\}.$$

- We write $after_{\mathscr{A}}(s)$ for the set of actions, enabled from state s, i.e. $after_{\mathscr{A}}(s) = \{a \in L \mid s \to a\}$. We lift this definition to traces by defining

$$after_{\mathscr{A}}(\sigma) = \bigcup_{s \in reach_{\mathscr{A}}(s_0,\sigma)} after_{\mathscr{A}}(s).$$

- We write $out_{\mathscr{A}}(\sigma) = after_{\mathscr{A}}(\sigma) \cap L_O^\delta$ to denote the set of all output actions as well as quiescence after trace σ.

In order for a pQTS to be meaningful, [23] postulated four well-formedness rules about quiescence, stating for instance that quiescence should not be succeeded by an output action. Since our current treatment does not rely on well-formedness, we omit these rules here. Moreover, our definition of a test case is a pQTS that does not adhere to the well-formedness criteria.

2.4 Trace distributions

Very much like the visible behaviour of a labelled transition system is given by its traces, the visible behaviour of a pQTS is given by its trace distributions: each trace distribution is a probability space that assigns a probability to (sets of) traces [20]. Just as a trace in an LTS is obtained by first selecting a path in the LTS and by then removing all states and internal actions, we do the same in the probabilistic case: we first resolve all the nondeterministic choices in the pQTS via an adversary, and by then removing all states — recall that our pQTSs do not contain internal actions. The resolution of the nondeterminism via an adversary leads to a purely probabilistic structure where we can assign a probability to each finite path, by multiplying the probabilities along that path. The mathematics to handle infinite paths is more complex, but completely standard [6]: in non-trivial situations, the probability assigned to an individual trace is 0 (cf., the probability to always roll a 6 with a dice is 0). Hence, we consider the probability assigned to sets of traces (e.g., the probability that a 6 turns up in the first 100 dice rolls). A classical result in measure theory shows that it is impossible to assign a probability to all sets of traces. Therefore, we collect those sets that can be assigned a probability in a so-called σ-field $\mathscr{F}$.

Adversaries. Following the standard theory for probabilistic automata [21], we define the behavior of a pQTS via adversaries (a.k.a. policies or schedulers). These resolve the nondeterministic choices in pQTSs: in each state of the pQTSs, the adversary chooses which transition to take. Adversaries can be (1) history-dependent, i.e. the choice which transition to take can depend on the full history; (2) randomized, i.e. the adversary may make a random choice over all outgoing transitions; and (3) partial, i.e., at any point in time, a scheduler may decide, with some probability, to terminate the execution.

Thus, given any finite history leading to a current state, an adversary returns a discrete probability distribution over the set of available next transitions (distributions to be precise). In order to model termination, we define schedulers which continue the transitions of pQTSs with a halting extension.

Definition 5. (Adversary) A *(partial, randomized, history-dependent) adversary* E of a pQTS $\mathscr{A} = (S, s_0, L_I, L_O, \Delta)$ is a function

$$E : Path^*(\mathscr{A}) \longrightarrow Distr(Distr(L \times S) \cup \{\bot\}),$$

such that for each finite path π, if $E(\pi)(\mu) > 0$, then $(last(\pi), \mu) \in \Delta$. The value $E(\pi)(\bot)$ is considered as *interruption/halting*. We say that E is *deterministic*, if $E(\pi)$ assigns the Dirac distribution for every distribution after all $\pi \in Path^*(\mathscr{A})$. An adversary E halts on a path π, if it extends π to the halting state $\bot$, i.e.

$$E(\pi)(\bot) = 1.$$

We say that an adversary halts after $k \in \mathbb{N}$ steps, if it halts for every path π with $|\pi| \geq k$. We denote all such adversaries by $Adv(\mathscr{A}, k)$. Lastly E is finite, if there exists $k \in \mathbb{N}$ such that $E \in Adv(\mathscr{A}, k)$.

The probability space assigned to an adversary. Intuitively an adversary tosses a coin at every step of the computation, thus resulting in a purely probabilistic (as opposed to nondeterministic) computation tree.

Definition 6. (Path Probability) Let E be an adversary of $\mathscr{A}$. The function $Q^E : Path^*(\mathscr{A}) \to [0,1]$ is called the *path probability function* and it is defined by induction. We set $Q^E(s_0) = 1$ and $Q^E(\pi\mu as) = Q^E(\pi) \cdot E(\pi)(\mu) \cdot \mu(a,s)$.

Loosely speaking, we follow a finite path in the transition system and multiply every scheduled probability along the way, resolving every nondeterminism according to the adversary E to get the ultimate path probability. The path probability function enables us to define a probability space associated with an adversary, thus giving every path in a pQTS $\mathscr{A}$ an exact probability.

Definition 7. (Adversary Probability Space) Let E be an adversary of $\mathscr{A}$. The *unique probability space associated to* E is the probability space $(\Omega_E, \mathscr{F}_E, P_E)$ given by.

1. $\Omega_E = Path^\infty(\mathscr{A})$
2. $\mathscr{F}_E$ is the smallest σ-field that contains the set $\{C_\pi \mid \pi \in Path^*(\mathscr{A})\}$, where the cone is defined as $C_\pi = \{\pi' \in \Omega_E \mid \pi \text{ is a prefix of } \pi'\}$.
3. P_E is the unique probability measure on $\mathscr{F}_E$ s. t. $P_E(C_\pi) = Q^E(\pi)$, for all $\pi \in Path^*(\mathscr{A})$.

The set of all adversaries is denoted by $adv(\mathscr{A})$ with $adv(\mathscr{A}, k)$ being the set of adversaries halting after $k \in \mathbb{N}$ respectively.

Trace distributions. As we mentioned, a trace distribution is obtained from (the probability space assigned to) an adversary by removing all states. This means that the probability assigned to a set of traces X is defined as the probability of all paths whose trace is an element of X.

Definition 8. (Trace Distribution) The *trace distribution* H of an adversary E, denoted $H = trd(E)$ is the probability space $(\Omega_H, \mathscr{F}_H, P_H)$ given by

1. $\Omega_H = L^*_{\mathscr{A}}$
2. $\mathscr{F}_H$ is the smallest σ- field containing the set $\{C_\beta \mid \beta \in L^*_{\mathscr{A}}\}$, where the cone is defined as $C_\beta = \{\beta' \in \Omega_E \mid \beta \text{ is a prefix of } \beta'\}$
3. P_H is the unique probability measure on $\mathscr{F}_H$ such that $P_H(X) = P_E\left(trace^{-1}(X)\right)$ for $X \in \mathscr{F}_H$.

As an abbreviation, we will write $P_H(\beta) := P_H\left(C_\beta\right)$ for $\beta \in L^*_{\mathscr{A}}$

Like before, we denote the set of trace distributions based on adversaries of $\mathscr{A}$ by $trd(\mathscr{A})$ and $trd(\mathscr{A}, k)$ if it is based on an adversary halting after $k \in \mathbb{N}$ steps respectively. Lastly we write $\mathscr{A} =_{TD} \mathscr{B}$ if $trd(\mathscr{A}) = trd(\mathscr{B})$, $\mathscr{A} \sqsubseteq_{TD} \mathscr{B}$ if $trd(\mathscr{A}) \subseteq trd(\mathscr{B})$ and $\mathscr{A} \sqsubseteq^k_{TD} \mathscr{B}$ if $trd(\mathscr{A}, k) \subseteq trd(\mathscr{B}, k)$ for $k \in \mathbb{N}$,

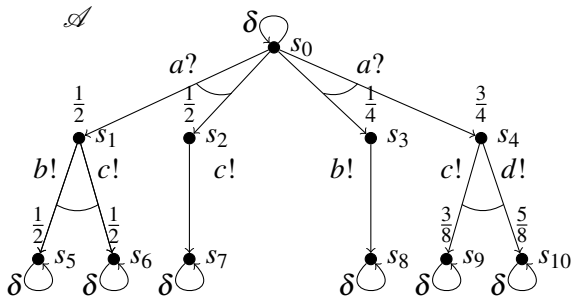


Figure 1: An example of the combination of nondeterministic and probabilistic choices.

where the embedding means that for every trace distribution H of $\mathscr{A}$ there is a trace distribution H' of $\mathscr{B}$ such that for all traces σ of $\mathscr{A}$, we have $P_H(\sigma) = P_{H'}(\sigma)$.

The fact that $(\Omega_E, \mathscr{F}_E, P_E)$, $(\Omega_H, \mathscr{F}_H, P_H)$ really define probability spaces, follows from standard measure theory arguments (see [6]).

Example 9. Consider the pQTS $\mathscr{A} = \left(S, s_0.L_I, L_O^\delta, \Delta\right)$ in Figure 1. There $S = \{s_0, s_1, \ldots, s_{10}\}$, $L_I = \{a?\}$, $L_O^\delta = \{b!, c!, d!\} \cup \{\delta\}$ and $\Delta = \{(s_0, \mu_{0_1}), (s_0, \mu_{0_2}), (s_0, \mu_{0_3}), (s_1, \mu_1), \ldots, (s_{10}, \mu_{10})\}$. We can see that this system has both probabilistic and nondeterministic choices. Observe that it has indeed only input reactive and output generative transitions as mentioned in the beginning of 2.2.

We will now consider an adversary E for $\mathscr{A}$. The only nondeterministic choice we have in this system, is located at state s_0, where we can either apply $a?$ to enter the left branch, $a?$ to enter the right branch, or do nothing (corresponding to μ_{0_1}, μ_{0_2} and μ_{0_3} respectively). Therefore consider the adversary $E(s_0)(\mu_{0_1}) = \frac{1}{2}$ and $E(s_0)(\mu_{0_2}) = \frac{1}{2}$ and $E(\pi)(\mu) = Dirac$ for every other distribution μ after a path π (i.e. those are taken with probability 1).

The adversary probability space created for this adversary assigns an unambiguous path probability to each path. Consider the path $\pi = s_0 \mu_{0_1} a? s_1 \mu_1 b! s_5$, then

$$P_E(\pi) = Q^E(\pi) = \underbrace{Q^E(s_0)}_{1} \underbrace{E(s_0)(\mu_{0_1})}_{\frac{1}{2}} \underbrace{\mu_{0_1}(a?, s_1)}_{\frac{1}{2}} \underbrace{E(s_0 \mu_{0_1} a? s_1)(\mu_1)}_{1} \underbrace{\mu_1(b!, s_5)}_{\frac{1}{2}} = \frac{1}{8}.$$

However, consider the trace distribution $H = trd(E)$. Then for $\sigma = a?b!$, we have $trace^{-1}(\sigma) = \{\pi, \eta\}$ with π as before and $\eta = s_0 \mu_{0_2} a? s_3 \mu_3 b! s_8$. Hence

$$P_H(\sigma) \quad = \quad P_{trd(E)}\left(trace^{-1}(\sigma)\right) = P_E(\{\pi, \eta\}) = P_E(\pi) + P_E(\eta) = \frac{1}{4}.$$

3 The probabilistic conformance relation pioco

3.1 The pioco relation

The classical input-output conformance relation ioco states that an implementation $\mathscr{A}_i$ conforms to a specification $\mathscr{A}_s$ if $\mathscr{A}_i$ never provides any unspecified output. In particular this refers to the observation of quiescence, when other output was expected.

Figure 2: An example illustrating pioco

Definition 10. (Input- Output Conformance) Let $\mathscr{A}_i$ and $\mathscr{A}_s$ be two QTS and let $\mathscr{A}_i$ be input enabled. Then we say $\mathscr{A}_i \sqsubseteq_{ioco} \mathscr{A}_s$, if and only if

$$\forall \sigma \in traces(\mathscr{A}_s) : out_{\mathscr{A}_i}(\sigma) \subseteq out_{\mathscr{A}_s}(\sigma).$$

To generalize ioco to pQTSs, we introduce two auxiliary concepts. For a natural number k, the prefix relation $H \sqsubseteq_k H'$ states that trace distribution H assigns exactly the same probabilities as H' to traces of length k and halts afterwards. The output continuation of a trace distribution H prolongs the traces of H with output actions. More precisely, output continuation of H wrt length k contains all trace distributions that (1) coincide with H for traces upto length k and (2) the $k+1$st action is an output label (incl δ); i.e. traces of length $k+1$ that end on an input action are assigned probability 0. Recall that $P_H(\sigma)$ abbreviates $P_H(C_\sigma)$.

Definition 11. (Notations) For a natural number $k \in \mathbb{N}$, and trace distributions $H \in trd(\mathscr{A},k)$, we say that

1. H is a *prefix* of $H' \in trd(\mathscr{A})$ up to k, denoted by $H \sqsubseteq_k H'$, iff $\forall \sigma \in L^k : P_H(\sigma) = P_{H'}(\sigma)$.
2. the *output continuation* of H in $\mathscr{A}$ is given by

$$outcont(H,\mathscr{A},k) : \quad = \quad \left\{ H' \in trd(\mathscr{A},k+1) \mid H \sqsubseteq_k H' \wedge \forall \sigma \in L^k L_I : P_{H'}(\sigma) = 0 \right\}.$$

We are now able to define the core idea of pioco. Intuitively an implementation should conform to a specification, if the probability of every trace in $\mathscr{A}_i$ specified in $\mathscr{A}_s$, can be matched in the specification. Just as in ioco, we will neglect underspecified traces continued with input actions (i.e., everything is allowed to happen after that). However, if there is unspecified output in the implementation, there is at least one adversary that schedules positive probability to this continuation, which consequently cannot be matched of output-continuations in the specification.

Definition 12. Let $\mathscr{A}_i$ and $\mathscr{A}_s$ be two pQTS. Furthermore let $\mathscr{A}_i$ be input enabled, then we say $\mathscr{A}_i \sqsubseteq_{pioco} \mathscr{A}_s$ if and only if

$$\forall k \in \mathbb{N} \forall H \in trd(\mathscr{A}_s,k) : outcont(H,\mathscr{A}_i,k) \subseteq outcont(H,\mathscr{A}_s,k).$$

Example 13. Consider the two systems of $\mathscr{A}$ and $\mathscr{B}$ shown in Figure 2 and assume that $p \in [0,1]$. It is true that $\mathscr{A} \sqsubseteq_{pioco} \mathscr{B}$, because we can always choose an adversary E of $\mathscr{B}$, which imitates the probabilistic behaviour of $\mathscr{B}$, i.e. choose $E(\varepsilon)(\mu) = \nu$ such that $\nu(a!,t_1) = p$ and $\nu(b!,t_2) = 1-p$.

However, the opposite does not hold. For example assume $p = \frac{1}{2}$, then the trace distribution H assigning $P_H(a!) = 1$ is in $outcont(H,\mathscr{B},1)$ but not in $outcont(H,\mathscr{A},1)$ and hence $\mathscr{B} \not\sqsubseteq_{pioco} \mathscr{A}$.

3.2 Properties of the p-ioco relation

As stated before, the relation pioco conservatively extends the ioco relation, i.e. both relations coincide for non-probabilistic QTSs. Moreover, we show that several other characteristic properties of ioco carry over to pioco as well. Below, a QTS is a pQTS where every occurring distribution is the Dirac distribution.

Theorem 14. *Let $\mathscr{A}_i$ and $\mathscr{A}_s$ be two QTS and let $\mathscr{A}_i$ be input enabled, then*

$$\mathscr{A}_i \sqsubseteq_{ioco} \mathscr{A}_s \Longleftrightarrow \mathscr{A}_i \sqsubseteq_{pioco} \mathscr{A}_s.$$

Intuitively it makes sense that the implementation is input enabled, since it should accept every input at any time. The following two results justify, that we assume the specification to be not input enabled, since otherwise pioco would coincide with trace distribution inclusion. Equivalently it is known that ioco coincides with trace inclusion, if we assume both the implementation and the specification were input enabled. Thus, as stated before, we can see that pioco extends ioco.

Lemma 15. *Let $\mathscr{A}_i$ and $\mathscr{A}_s$ be two pQTS, then*

$$\mathscr{A}_i \sqsubseteq_{TD} \mathscr{A}_s \Longrightarrow \mathscr{A}_i \sqsubseteq_{pioco} \mathscr{A}_s.$$

Theorem 16. *Let $\mathscr{A}_i$ and $\mathscr{A}_s$ be two input enabled pQTS, then*

$$\mathscr{A}_i \sqsubseteq_{pioco} \mathscr{A}_s \Longleftrightarrow \mathscr{A}_i \sqsubseteq_{TD} \mathscr{A}_s.$$

Next, we show that, under some input-enabledness restrictions, the pioco relation is transitive. Again, note that this is also true for ioco for non-probabilistic systems.

Theorem 17. *(Transitivity of pioco) Let $\mathscr{A}$, $\mathscr{B}$ and $\mathscr{C}$ be pQTS, such that $\mathscr{A}$ and $\mathscr{B}$ are input enabled, then*

$$\mathscr{A} \sqsubseteq_{pioco} \mathscr{B} \wedge \mathscr{B} \sqsubseteq_{pioco} \mathscr{C} \Longrightarrow \mathscr{A} \sqsubseteq_{pioco} \mathscr{C}.$$

4 Testing for pQTS

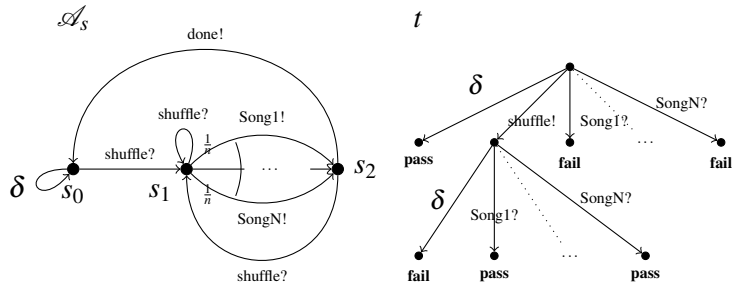
4.1 Test cases for pQTSs.

We will consider tests as sets of traces based on an action signature $\left(L_I, L_O^{\delta}\right)$, which will describe possible behaviour of the tester. This means that at each state in a test case, the tester either provides stimuli or waits for a response of the system. Additionally to output conformance testing like in [24], we introduce probabilities into our testing transition system. Thus we can represent each test case as a pQTS, albeit with a mirrored action signature $(L_O, L_I \cup \{\delta\})$. This is necessary for the parallel composition of the test pQTS and the SUT.

Since we consider tests to be pQTS, we also use all the terminology introduced earlier on. Additionally we require tests to not contain loops (or infinite paths respectively).

Definition 18. A *test (directed acyclic graph)* over an action signature $\left(L_I, L_O^{\delta}\right)$ is a pQTS of the form $t = (S, s_0, L_O, L_I \cup \{\delta\}, \Delta)$ such that

- t is internally deterministic and does not contain an infinite path;
- t is acyclic and connected;



(a) Specification $\mathscr{A}_s$ (b) An annotated test $\hat{t}$ for $\mathscr{A}_s$

Figure 3: A specification for a simple shuffle music player and a test.

- For every state $s \in S$, we either have
  - $after(s) = \emptyset$, or
  - $after(s) = L_I \cup \{\delta\}$, or
  - $after(s) = \{a!\} \cup L_I \cup \{\delta\}$ for some $a! \in L_O$.

A test suite T is a set of tests over an action signature $\left(L_I, L_O^\delta\right)$. We write $\mathscr{T}\left(L_I, L_O^\delta\right)$ to denote all the tests over an action signature $\left(L_I, L_O^\delta\right)$ and $\mathscr{TS}\left(L_I, L_O^\delta\right)$ as the set of all test suites over an action signature respectively.

For a given specification pQTS $\mathscr{A}_s = \left(S, s_0, L_I, L_O^\delta, \Delta\right)$, we say that a test t is a *test for* $\mathscr{A}_s$, if it is based on the same action signature $\left(L_I, L_O^\delta\right)$. Similar to before, we denote all tests for $\mathscr{A}_s$ by $\mathscr{T}(\mathscr{A}_s)$ and all test suites by $\mathscr{TS}(\mathscr{A}_s)$ respectively.

Note that we mirrored the action signature for tests, as can be seen in Figure 3a and Figure 3b respectively. That is, because we require tests and implementations to shake hands on shared actions. A special role is dedicated to quiescence in the context of parallel composition, since the composed system is considered quiescent if and only if the two systems are quiescent.

We will proceed to define parallel composition. Formally this means that output actions of one component are allowed to be present as input actions of the other component. These will be synchronized upon. However, keeping in mind the mirrored action signature of tests, we wish to avoid possibly unwanted synchronization, which is why we introduce system compatibility.

Definition 19. (Compatibility) Two pQTS $\mathscr{A} = \left(S, s_0, L_I, L_O^\delta, \Delta\right)$, and $\mathscr{A}' = \left(S', s_0', L_I', L_O^{\delta\prime}, \Delta'\right)$ are said to be *compatible* if $L_O^\delta \cap L_O^{\delta\prime} = \{\delta\}$.

When we put two pQTSs in parallel, they synchronize on shared actions, and evolve independently on others. Since the transitions taken by the two component of the composition are stochastically independent, we multiply the probabilities when taking shared actions.

Definition 20. (Parallel composition) Given two compatible pQTS $\mathscr{A} = \left(S, s_0, L_I, L_O^\delta, \Delta\right)$ and $\mathscr{A}' = \left(S', s_0', L_I', L_O^{\delta\prime}, \Delta'\right)$, their *parallel composition* is the tuple

$$\mathscr{A} \,||\, \mathscr{A}' = \left(S'', s_0'', L_I'', L_O^{\delta\prime\prime}, \Delta''\right),$$

where

$S'' = S \times S'$,

$s_0'' = (s_0, s_0')$,
$L_I'' = (L_I \cup L_I') \setminus (L_O \cup L_O')$,
$L_O^{\delta\prime\prime} = L_O^{\delta} \cup L_O^{\delta\prime}$,
$\Delta'' = \{((s,t),\mu) \in S'' \times Distr(L'' \times S'') \mid$

$$\mu\left(a,\left(s',t'\right)\right) \equiv \begin{cases} \mu_a(a,s')\,\nu_a(a,t') & \text{if } a \in L \cap L', \text{ where } s \xrightarrow{\mu_a,a}_{\mathscr{A}} s' \wedge t \xrightarrow{\nu_a,a}_{\mathscr{A}'} t' \\ \mu_a(a,s') & \text{if } a \in L \backslash L', \text{ where } s \xrightarrow{\mu_a,a}_{\mathscr{A}} s' \wedge t = t' \\ \nu_a(a,t') & \text{if } a \in L' \backslash L, \text{ where } s = s' \wedge t \xrightarrow{\nu_a,a}_{\mathscr{A}'} t' \\ 0 & \text{otherwise} \end{cases} \}$$

where $\mu_a \in Distr(L,S)$ and $\nu_a \in Distr(L',S')$ respectively.

Before we parallel compose a test case with a system, we obviously need to define which outcome of a test case is considered correct, and which is not (i.e., when it fails).

Definition 21. (Test case annotation) For a given test t a *test annotation* is a function

$$a : ctraces(t) \longrightarrow \{pass, fail\}.$$

A pair $\hat{t} = (t,a)$ consisting of a test and a test annotation is called an *annotated test*. The set of all such $\hat{t}$ is defined as $\hat{T} = \left\{(t_i,a_i)_{i\in\mathscr{I}}\right\}$ for some index set $\mathscr{I}$ is called *annotated Test Suite*. If t is a test case for a specification $\mathscr{A}_s$ we define the pioco test annotation $a^{pioco}_{\mathscr{A}_s,t} : ctraces(t) \longrightarrow \{pass, fail\}$ by

$$a^{pioco}_{\mathscr{A}_s,t}(\sigma) = \begin{cases} fail & \text{if } \exists \sigma_1 \in traces(\mathscr{A}_s), a! \in L_O^{\delta} : \sigma_1 a! \sqsubseteq \sigma \wedge \sigma_1 a! \notin traces(\mathscr{A}_s); \\ pass & \text{otherwise.} \end{cases}$$

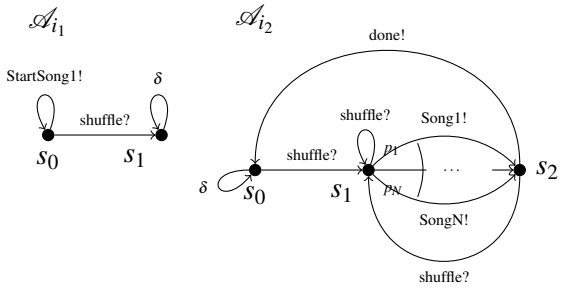
4.2 Test execution.

By taking the intersection of all complete traces within a test and all traces of an implementation, we will define the set of all traces that will be executed by an annotated test case.

Definition 22. (Test execution) Let t be a test over the action signature $\left(L_I, L_O^{\delta}\right)$ and the pQTS $\mathscr{A}_i = \left(S, s_0, L_I, L_O^{\delta}, \Delta\right)$. Then we define

$$exec_t(\mathscr{A}_i) = traces(\mathscr{A}_i) \cap ctraces(\hat{t}).$$

Example 23. Consider the specification of a shuffle music player and a derived test for it given in Figure 3. Assuming we are to test whether or not the following two implementations conform to the specification with respect to pioco:



Here $p_1,\ldots,p_N \in [0,1]$ such that $\sum_{i=1}^{N} p_i = 1$. Now when we compose $\mathscr{A}_{i_1}$ with t in Figure3*b*, we can clearly see that every complete trace of the parallel system is annotated with *fail*, as it would also have been the case for classical ioco theory. However, if we now also consider $\mathscr{A}_{i_2}$ and compose it with the same test t, every trace of the composed system would be given a *pass* label if we restricted ourselves to the annotation function and the output verdict. Note how every trace $shuffle?\cdot Song\_i!$ is given probability p_i for $i = 1,\ldots,N$. The only restriction we assumed valid for $p_1,\ldots,p_N$ is that they sum up to 1 so a correct distribution for $\mathscr{A}_{i_2}$ would be $p_1 = \frac{N-1}{N}$ and $p_2 = \ldots = p_N = \frac{1}{N^2}$. This, however, should intuitively not be given the verdict *pass*, since it differs from the uniform distribution given in the specification $\mathscr{A}_s$.

4.3 Test evaluation

In order to give a verdict of whether or not the implementation passed the test (suite), we need to extend the test evaluation process of classical ioco testing with a statistical component. Thus the idea of evaluating probabilistic systems becomes two folded. On the one hand, we want that no unexpected output (or unexpected quiescence) ever occurs during the execution. On the other hand, we want the observed frequencies of the SUT to conform in some way to the probabilities described in the specification. Thus the SUT will pass the test suite only if it passes both criteria. We will do this by augmenting classical ioco theory with zero hypothesis testing, which will be discussed in the following.

To conduct an experiment, we need to define a length $k \in \mathbb{N}$ and a width $m \in \mathbb{N}$ first. This refers to how long the traces we want to record should be and how many times we reset the machine. This will give us traces $\sigma_1,\ldots,\sigma_m \in L^k$, which we call a *sample*. Additionally, we assume that the implementation is governed by an underlying trace distribution H in every run, thus running the machine m times, gives us a sequence of possibly m different trace distributions $\vec{H} = H_1,\ldots,H_m$. So in every run the implementation makes two choices: 1) It chooses the trace distribution H and 2) H chooses a trace σ to execute. Consequently that means that once a trace distribution H_i is chosen, it is solely responsible for the trace σ_i. Thus for $i \neq j$ the choice of σ_i is independent from the choice of σ_j.

Our statistical analysis is build upon the frequencies of traces occurring in a sample O. Thus the *frequency function* will be defined as

$$freq(O)(\sigma) = \frac{|\{i \in, \{1,\ldots,m\} \,|\sigma_i = \sigma\}|}{m}.$$

Note that although every run is governed by possibly different trace distributions, we can still derive useful information from the frequency function. For fixed $k,m \in \mathbb{N}$ and $\vec{H}$, the sample O can be treated as a Bernoulli experiment of length m, where success occurs in position $i = 1,\ldots m$, if $\sigma = \sigma_i$. The success probability is then given by $P_{H_i}(\sigma)$. So for given $\vec{H}$, the expected value for σ is given by $\mathbb{E}^{\vec{H}}_{\sigma} = \frac{1}{m}\sum_{i=1}^{m} P_{H_i}(\sigma)$. Note that this expected value $\mathbb{E}^{\vec{H}}$ is the expected distribution over L^k if we assume it is based on the m trace distributions $\vec{H}$.

In order to apply zero hypothesis testing and compare an observed distribution with $\mathbb{E}^{\vec{H}}$, we will use the notion of metric spaces. This will enable us to measure deviation of two distributions. We will use the metric space $\left(L^k, dist\right)$, where *dist* is the euclidean distance of two distributions defined as $dist(\mu,\nu) = \sqrt{\sum_{\sigma\in L^k} |\mu(\sigma) - \nu(\sigma)|^2}$.

Now that we have a measure of deviation, we can say that a sample O is accepted if $freq(O)$ lies in some distance r of the expected value $\mathbb{E}^{\vec{H}}$, or equivalently if $freq(O)$ is contained in the closed ball $B_r\left(\mathbb{E}^{\vec{H}}\right) = \left\{\nu \in Distr\left(L^k\right) \mid dist\left(\nu, \mathbb{E}^{\vec{H}}\right) \leq r\right\}$. Then the set $freq^{-1}\left(B_r\left(\mathbb{E}^{\vec{H}}\right)\right)$ summarizes all samples that deviate at most r from the expected value.

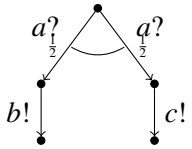


Figure 4: A probabilistic automaton representing a fair coin.

An inherent problem of hypothesis testing are the type 1 and type 2 errors, i.e. the probability of falsely accepting the hypothesis or falsely rejecting it. This problem is established in our framework by the choice of a level of significance $\alpha \in [0,1]$ and connected with it, the choice of radius r for the ball mentioned above. So for a given level of significance α the following choice of the radius will in some sense minimize the probability of false acceptance of an erroneous sample and of false rejection of a valid sample (i.e., at most α).

$$\bar{r} := \inf\left\{ r \mid P_{\vec{H}}\left(freq^{-1}\left(B_r\left(\mathbb{E}^{\vec{H}}\right)\right)\right) > 1-\alpha \right\}.$$

Thus assuming we have m different underlying trace distributions, we can determine when an observed sample seems reasonable and is declared valid. Unifying over all sets of such $\vec{H}$, we will define the total set of acceptable outcomes, called *Observations*.

Definition 24. The *acceptable outcomes* of $\vec{H}$ with significance level $\alpha \in [0,1]$ are given by the set of samples of length $k \in \mathbb{N}$ and width $m \in \mathbb{N}$, defined as

$$Obs\left(\vec{H},\alpha\right) := freq^{-1}\left(B_{\bar{r}}\left(\mathbb{E}^{\vec{H}}\right)\right) = \left\{ O \in \left(L^k\right)^m \mid dist\left(freq\left(O\right),\mathbb{E}^{\vec{H}}\right) \leq \bar{r} \right\}.$$

The set of *observations* of $\mathscr{A}$ with significance level $\alpha \in [0,1]$ is given by

$$Obs\left(\mathscr{A},\alpha\right) = \bigcup_{\vec{H} \in trd(\mathscr{A},k)^m} Obs\left(\vec{H},\alpha\right).$$

Example 25. Assume that the wanted level of significance is given by $\alpha = 0.05$ and consider the probabilistic automaton in Figure 4 representing the toss of a fair coin. Furthermore assume that we are given two samples of depth $k = 2$ and width $m = 100$.

To sample this case, assume E is the adversary that assigns probability equal to 1 to the unique outgoing transition (if there is one) and probability 1 to halting, in case there is no outgoing transition. We take $H = trd\left(E\right)$ and can see, that then $\mu_H\left(a?b!\right) = \mu_H\left(a?c!\right) = \frac{1}{2}$ and $\mu_H\left(\sigma\right) = 0$ for all other sequences σ. We define $H^{100} = \left(H_1,\ldots,H_{100}\right)$, where $H_1 = \ldots = H_{100} = H$. As we can see, we have $\mathbb{E}^{H^{100}} = \mu_H$. Since μ_H only assigns positive probability to $a?b!$ and $a?c!$, we get $P_{H^{100}}\left(B_r\left(\mu_H\right)\right) = \left(\left\{O \middle| \frac{1}{2} - r \leq freq\left(O\right)\left(a?b!\right) \leq \frac{1}{2} + r\right\}\right)$. One can show that the smallest ball, where this probability is greater or equal than 0.95 is given by the ball of radius $\bar{r} = \frac{1}{10}$.

Thus a sample O_1, which consists of 42 times $a?b!$ and 58 times $a?c!$ is an observation, and a sample O_2, which consists of 38 times $a?b!$ and 62 times $a?c!$ is not.

Thus we can finally define a verdict function, that assigns *pass* when a test case never finds erroneous behaviour (i.e. wrong output or wrong probabilistic behaviour).

Definition 26. (Output verdict) Let $\left(L_I, L_O^\delta\right)$ be an action signature and $\hat{t} = (t,a)$ an annotated test case over $\left(L_I, L_O^\delta\right)$. The *output verdict function* for $\hat{t}$ is the function $v_{\hat{t}} : pQTS \to \{pass, fail\}$, given for any pQTS $\mathscr{A}_i$

$$v_{\hat{t}}(\mathscr{A}_i) = \begin{cases} pass & \text{if } \forall \sigma \in exec_t(\mathscr{A}_i) : a(\sigma) = pass \\ fail & \text{otherwise} \end{cases}.$$

(Statistical verdict) Additionally let $\alpha \in [0,1]$ and $k,m \in \mathbb{N}$ and $O \in Obs(\mathscr{A}_i||\hat{t},\alpha) \subseteq \left(L^k\right)^m$, then the *statistical verdict function* is given by

$$v_{\hat{t}}^{\alpha}(\mathscr{A}_i) = \begin{cases} pass & \text{if } O \in Obs(\mathscr{A}_s,\alpha) \\ fail & \text{otherwise} \end{cases}.$$

(Verdict function) For any given $\mathscr{A}_i$, we assign the *verdict*

$$V_{\hat{t}}^{\alpha}(\mathscr{A}_i) = \begin{cases} pass & \text{if } v_{\hat{t}}(\mathscr{A}_i) = v_{\hat{t}}^{\alpha}(\mathscr{A}_i) = pass \\ fail & \text{otherwise} \end{cases}.$$

We extend $V_{\hat{t}}^{\alpha}$ to a function $V_{\hat{T}}^{\alpha} : pQTS \to \{pass, fail\}$, which assigns verdicts to a pQTS based on a given annotated test suite by $V_{\hat{T}}^{\alpha}(\mathscr{A}_i) = pass$ if for all $\hat{t} \in \hat{T}$ and $V_{\hat{T}}^{\alpha}(\mathscr{A}_i) = fail$ otherwise.

5 Conclusion and Future Work

We introduced the core of a probabilistic test theory by extending classical ioco theory. We defined the conformance relation pioco for probabilistic quiescent transition systems, and proved several characteristic properties. In particular, we showed that pioco is a conservative extension of ioco. Second, we have provided definitions of a test case, test execution and test evaluation. Here, test execution is crucial, since it needs to assess whether the observed behaviour respects the probabilities in the specification pQTS. Following [4], we have used statistical hypothesis testing here.

Being a first step, there is ample future work to be carried out. First, it is important to establish the correctness of the testing framework, by showing the soundness and completeness. Second, we would like to implement our framework in the MBT testing framework JTorX, and test realistic applications. Also, we would like to extend our theory to handle τ-transitions. Finally, we think that tests themselves should be probabilistic, in particular since many MBT tools in practice do already choose their next action probabilistically.

References

[1] A.E.F. Belinfante (2010): *JTorX: A Tool for On-Line Model-Driven Test Derivation and Execution*. *Lecture Notes in Computer Science* 6015, Springer, pp. 266–270, doi:10.1007/978-3-642-12002-2_21.

[2] S. Bensalem, D. Peled, H. Qu & S.S. Tripakis (2008): *Automatic generation of path conditions for concurrent timed systems*. *Theor. Comput. Sci.* 404(3), pp. 275–292, doi:10.1016/j.tcs.2008.03.012.

[3] H. C. Bohnenkamp & M.I.A. Stoelinga (2008): *Quantitative testing*. In: *Proceedings of the 8th ACM & IEEE International conference on Embedded software, (EMSOFT'08)*, ACM, pp. 227–236, doi:10.1145/1450058.1450089.

[4] L. Cheung, M.I.A. Stoelinga & F.W. Vaandrager (2007): *A testing scenario for Probabilistic Automata*. *Journal of the ACM* 54(6), doi:10.1145/1314690.1314693.

[5] D. Clarke, T. Jéron, V. Rusu & E. Zinovieva (2002): *STG: A Symbolic Test Generation Tool*. *Lecture Notes in Computer Science* 2280, Springer-Verlag, London, UK, pp. 470–475, doi:10.1007/3-540-45669-4.

[6] D. L. Cohn (1980): *Measure Theory*. Birkhäuser, doi:10.1007/978-1-4899-0399-0.

[7] W. Dulz & FF. Zhen (2003): *MaTeLo - Statistical Usage Testing by Annotated Sequence Diagrams, Markov Chains and TTCN-3*. In: *Proceedings of the 3rd International Conference on Quality Software*, IEEE Computer Society, doi:10.1109/QSIC.2003.1319119.

[8] L. Frantzen, J. Tretmans & T. A. C. Willemse (2006): *A Symbolic Framework for Model-Based Testing*. *Lecture Notes in Computer Science* 4262, Springer-Verlag, pp. 40–54, doi:10.1007/11940197_3.

[9] R. J. van Glabbeek, S. A. Smolka, B. Steffen & C.M.N. Tofts (1990): *Reactive, generative, and stratified models of probabilistic processes*. 5th Annual Symposium on Logic in Computer Science, IEEE Computer Society Press, pp. 130–141.

[10] A. Hessel, K. G. Larsen, M. Mikucionis, B. Nielsen, P. Pettersson & A. Skou (2008): *Testing Real-Time Systems Using UPPAAL*. *Lecture Notes in Computer Science* 4949, Springer, pp. 77–117, doi:10.1007/978-3-540-78917-8_3.

[11] R. M. Hierons & M Núñez (2010): *Testing Probabilistic Distributed Systems*. *Lecture Notes in Computer Science* 6117, Springer, pp. 63–77, doi:10.1007/978-3-642-13464-7_6.

[12] R. M. Hierons & M. Núñez (2012): *Using schedulers to test probabilistic distributed systems*. *Formal Asp. Comput.* 24(4-6), pp. 679–699, doi:10.1007/s00165-012-0244-5.

[13] Iksoon Hwang & Ana Cavalli (2010): *Testing a probabilistic FSM using interval estimation*. *Computer Networks* 54(7), pp. 1108 – 1125, doi:10.1016/j.comnet.2009.10.014.

[14] T. Jéron (2009): *Symbolic Model-based Test Selection*. *Electr. Notes Theor. Comput. Sci.* 240, pp. 167–184, doi:10.1016/j.entcs.2009.05.051.

[15] M. Krichen & S. Tripakis (2009): *Conformance testing for real-time systems*. *Formal Methods in System Design* 34(3), pp. 238–304, doi:10.1007/s10703-009-0065-1.

[16] K. G. Larsen, M. Mikucionis, B. Nielsen & A. Skou (2005): *Testing real-time embedded software using UPPAAL-TRON: an industrial case study*. ACM Press, pp. 299–306.

[17] K. G. Larsen & A. Skou (1989): *Bisimulation Through Probabilistic Testing*. ACM Press, pp. 344–352.

[18] W. Mostowski, E. Poll, J. Schmaltz, J. Tretmans & R. W. Schreur (2009): *Model-Based Testing of Electronic Passports*. *Lecture Notes in Computer Science* 5825, Springer, pp. 207–209.

[19] M. van Osch (2006): *Hybrid Input-Output Conformance and Test Generation*. In: *Proceedings of FATES/RV 2006*, *Lecture Notes in Computer Science* 4262, pp. 70–84.

[20] R. Segala (1995): *Modeling and Verification of Randomized Distributed Real-time Systems*. Ph.D. thesis, Cambridge, MA, USA.

[21] M.I.A. Stoelinga (2002): *Alea jacta est: Verification of Probabilistic, Real-time and Parametric Systems*. Ph.D. thesis, Radboud University of Nijmegen.

[22] M.I.A. Stoelinga & F. W. Vaandrager (2003): *A Testing Scenario for Probabilistic Automata*. *Lecture Notes in Computer Science* 2719, Springer, pp. 464–477, doi:10.1007/3-540-45061-0_38.

[23] W. G. J. Stokkink, M. Timmer & M.I.A. Stoelinga (2013): *Divergent Quiescent Transistion Sytems*. In: *Proceedings 7th conference on Tests and Proofs (TAP'13)*, Lecture Notes in Computer Science, pp. 214–231, doi:10.1007/978-3-642-38916-0_13.

[24] M. Timmer, H. Brinksma & M. I. A. Stoelinga (2011): *Model-Based Testing*. In: *Software and Systems Safety: Specification and Verification*, *NATO Science for Peace and Security Series D: Information and Communication Security* 30, IOS Press, Amsterdam, pp. 1–32.

[25] J. Tretmans (1996): *Conformance Testing with Labelled Transition Systems: Implementation Relations and Test Generation*. *Computer Networks and ISDN Systems* 29(1), pp. 49–79, doi:10.1016/S0169-7552(96)00017-7.

[26] J. Tretmans (1996): *Test Generation with Inputs, Outputs and Repetitive Quiescence. Software - Concepts and Tools* 17(3), pp. 103–120.

[27] J. Tretmans (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* 4949, Springer, pp. 1–38, doi:10.1007/978-3-540-78917-8_1.

[28] J. A. Whittaker & J. H. Poore (1993): *Markov Analysis of Software Specifications*. *ACM Trans. Softw. Eng. Methodol.* 2(1), pp. 93–106, doi:10.1145/151299.151326.

[29] J. A. Whittaker, K. Rekab & M. G. Thomason (2000): *A Markov chain model for predicting the reliability of multi-build software*. *Information & Software Technology* 42(12), pp. 889–894, doi:10.1016/S0950-5849(00)00122-1.

Appendix

Below, we present the proofs of our theorems.

Proofs

Proof of Theorem 14.

" $\Longleftarrow$ " Let $\mathscr{A}_i \sqsubseteq_{pioco} \mathscr{A}_s$ and $\sigma \in traces(\mathscr{A}_s)$. Our goal is to show $out_{\mathscr{A}_i}(\sigma) \subseteq out_{\mathscr{A}_s}(\sigma)$.

For $out_{\mathscr{A}_i}(\sigma) = \emptyset$ we are done, since $\emptyset \subseteq out_{\mathscr{A}_s}(\sigma)$ obviously.

So assume that there is $b! \in out_{\mathscr{A}_i}(\sigma)$. We want to show that $b! \in out_{\mathscr{A}_s}(\sigma)$. For this, let $k = |\sigma|$ and $H \in trd(\mathscr{A}_s, k)$ such that $P_H(\sigma) = 1$, which is possible because $\sigma \in traces(\mathscr{A}_s)$ and both $\mathscr{A}_i$ and $\mathscr{A}_s$ are non-probabilistic. The same argument gives us $outcont(H, \mathscr{A}_i, k) \neq \emptyset$, because $\sigma \in traces(\mathscr{A}_i)$.

Thus we have at least one $H' \in outcont(H, \mathscr{A}_i, k)$ such that $P_{H'}(\sigma b!) > 0$. Let $\pi \in trace^{-1}(\sigma) \cap Path^*(\mathscr{A}_s)$. Now $H' \in outcont(H, \mathscr{A}_s, k)$, because $\mathscr{A}_i \sqsubseteq_{pioco} \mathscr{A}_s$ by assumption and thus there must be at least one adversary $E' \in adv(\mathscr{A}_s, k+1)$ such that $trd(E') = H'$ and $Q^{E'}(\pi \cdot Dirac \cdot b!s') > 0$ for some $s' \in S$. Hence $E'(\pi)(Dirac)\, Dirac(b!, s') > 0$ and therefore with $s' \in reach(last(\pi), b!)$ this yields $b! \in out_{\mathscr{A}_s}(\sigma)$.

" $\Longrightarrow$ " Let $\mathscr{A}_i \sqsubseteq_{ioco} \mathscr{A}_s$, $k \in \mathbb{N}$ and $H^* \in trd(\mathscr{A}_s, k)$. Assume that $H \in outcont(H^*, \mathscr{A}_i, k)$, then we want to show that $H \in outcont(H^*, \mathscr{A}_s, k)$.

Therefore let $E \in adv(\mathscr{A}_i, k+1)$ such that $trd(E) = H$. If we can find $E' \in adv(\mathscr{A}_s, k+1)$ such that $trd(E) = trd(E')$, we are done. We will do this constructively in three steps.

1) By construction of H^* we know that there must be $E' \in adv(\mathscr{A}_s, k+1)$, such that for all $\sigma \in L^k$ we have $P_{trd(E')}(\sigma) = P_{H^*}(\sigma) = P_{trd(E)}(\sigma)$. Thus $H^* \sqsubseteq_k trd(E')$.

2) We did not specify the behaviour of E' for path of length $k+1$. Therefore we choose E' such that for all traces $\sigma \in L^k$ and $a? \in L_I$ we have $P_{trd(E')}(\sigma a?) = 0 = P_{trd(E)}(\sigma a?)$.

3) The last thing to show is that $trd(E) = trd(E')$. Therefore let us now set the behaviour of E' for traces ending in outputs. Let $\sigma \in traces(\mathscr{A}_i)$, then assume $a! \in out_{\mathscr{A}_i}(\sigma)$ (if $out_{\mathscr{A}_i}(\sigma) = \emptyset$ we are done immediately) and because $\mathscr{A}_i \sqsubseteq_{ioco} \mathscr{A}_s$, we know that $a! \in out_{\mathscr{A}_s}(\sigma)$.

Now let $p := P_{trd(E)}(\sigma) = P_{trd(E')}(\sigma)$ and $q := P_{trd(E)}(\sigma a!)$. By equality of the trace distributions for traces up to length k we know that $q \leq p \leq 1$ and therefore there is $\alpha \in [0,1]$ such that $q = p \cdot \alpha$. Let $traces(\mathscr{A}_s) \cap trace^{-1}(\sigma) = \{\pi_1, \ldots, \pi_n\}$. Without loss of generality, we choose E' such that

$$E'(\pi_i)(Dirac) = \begin{cases} \alpha & \text{if } i = 1 \\ 0 & \text{else} \end{cases}.$$

We constructed $E' \in adv(\mathscr{A}_s, k+1)$, such that for all $\sigma \in L^{k+1}$ we have $P_{trd(E')}(\sigma) = P_{trd(E)}(\sigma)$ and thus $trd(E) = trd(E')$, which finally yields $H \in outcont(H^*, \mathscr{A}_s, k)$. □

Proof of Lemma 15. Let $\mathscr{A}_i \sqsubseteq^k_{TD} \mathscr{A}_s$ then for every $H \in trd(\mathscr{A}_i, k)$ we also have $H \in trd(\mathscr{A}_s, k)$. So pick $m \in \mathbb{N}$, let $H^* \in trd(\mathscr{A}_s, m)$ and take $H \in outcont(H^*, \mathscr{A}_i, m) \subseteq trd(\mathscr{A}_i, m+1)$. We want to show that $H \in outcont(H^*, \mathscr{A}_s, m)$.

By assumption we know that $H \in trd(\mathscr{A}_s, m+1)$. In particular that means there must be at least one adversary $E \in adv(\mathscr{A}_s, m+1)$ such that $trd(E) = H$. However, for this adversary, we know that $H^* \sqsubseteq_m trd(E)$ and for all $\sigma \in L^m L_I$ we have $P_{trd(E)}(\sigma) = 0$ and by trace distribution inclusion $trd(E) = H$. Thus $H \in outcont(H^*, \mathscr{A}_s, m)$ and therefore $\mathscr{A}_i \sqsubseteq_{pioco} \mathscr{A}_s$. □

Proof of Theorem 16. "$\Longrightarrow$" Let $\mathscr{A}_i \sqsubseteq_{pioco} \mathscr{A}_s$, fix $m \in \mathbb{N}$ and take a trace distribution $H^* \in trd(\mathscr{A}_i, m)$. To show that $H^* \in trd(\mathscr{A}_s, m)$, we prove that every prefix of H^* is in $trd(\mathscr{A}_s, m)$, i.e. if $H' \sqsubseteq_k H^*$ for some $k \in \mathbb{N}$, then $H' \in trd(\mathscr{A}_s)$. The proof is by induction up to $m \in \mathbb{N}$ over the prefix trace distribution length, denoted by k.

Obviously $H' \in trd(\mathscr{A}_i, 0)$ yields both $H' \sqsubseteq_0 H^*$ and $H' \in trd(\mathscr{A}_s)$. Now assume, we know that $H' \sqsubseteq_k H^*$ for some $k < m$ and $H' \in trd(\mathscr{A}_s)$. Furthermore let $H'' \in trd(\mathscr{A}_i, k+1)$, such that $H'' \sqsubseteq_{k+1} H^*$. If we can show that $H'' \in trd(\mathscr{A}_s, k+1)$, we are done.

With $H' \in trd(\mathscr{A}_s, k)$, we take $H''' \in outcont(H', \mathscr{A}_i, k)$ such that all traces of length $k+1$ ending in an output action have the same probability, i.e. for all $\sigma \in L^k L_O^\delta$, we have $P_{H'''}(\sigma) = P_{H''}(\sigma)$. By assumption $\mathscr{A}_i \sqsubseteq_{pioco} \mathscr{A}_s$ and thus $H''' \in outcont(H', \mathscr{A}_s, k) \subseteq trd(\mathscr{A}_s)$.

Let $E \in adv(\mathscr{A}_s, k+1)$ the corresponding adversary such that $trd(E) = H'''$. By construction, we have $P_{trd(E)}(\sigma a!) = P_{H''}(\sigma a!)$ and $P_{trd(E)}(\sigma b?) = 0 \overset{\text{in general}}{\neq} P_{H''}(\sigma b?)$ for all $\sigma \in L^k$. We create yet another adversary, denoted by $E' \in adv(\mathscr{A}_s, k+1)$ such that for all $\sigma \in L^k$ and $a! \in L_O^\delta$, we have $P_{trd(E)}(\sigma) = P_{trd(E')}(\sigma)$ and $P_{trd(E)}(\sigma a!) = P_{trd(E')}(\sigma a!)$. Taking the sum over all probabilities of those traces yields

$$\sum_{a! \in L_O^\delta} P_{trd(E)}(\sigma a!) = 1 - \alpha,$$

where $\alpha \in [0,1]$ and consequently the remaining bit is covered by

$$\sum_{b? \in L_I} P_{H''}(\sigma b?) = \alpha.$$

The aim is now to set the behaviour of E' such that $\sigma \in L^k L_I$ has $P_{H''}(\sigma) = P_{trd(E')}(\sigma)$. We prove that this can indeed be done independently from σ. The input enabledness gives that for all $\sigma b? \in traces(\mathscr{A}_i)$, we also have $\sigma b? \in traces(\mathscr{A}_s)$. Assume $P_{H''}(\sigma) = p$ and thus

$$\begin{aligned} \alpha &= \sum_{b? \in L_I} P_{H''}(\sigma b?) = P_{H''}(\sigma b_1?) + \ldots + P_{H''}(\sigma b_n?) = p\alpha_1 + \ldots + p\alpha_\omega \\ &\overset{!}{=} P_{trd(E')}(\sigma b_1?) + \ldots + P_{trd(E')}(\sigma b_n?). \end{aligned}$$

However, since $trd(E) \sqsubseteq_k H''$, we also have $P_{trd(E)}(\sigma) = p$.

The last detail not yet specified about E' is the behaviour of paths of length $k+1$ ending in an input transition. We demonstrate the choice of E' for $p\alpha_1 \overset{!}{=} P_{trd(E')}(\sigma b_1?)$, and denote the associated paths $\{\pi_1, \ldots, \pi_n\} = trace^{-1}(\sigma)$. Furthermore $\pi_i' := \pi_i \mu b_1 ? s_{i_j}$ for some $s_{i_j} \in S$, $j = 1, \ldots, l$, which are reachable after π_i and distributions containing $b?$. Thus we want

$$\begin{aligned} p\alpha_1 &\stackrel{!}{=} P_{trd(E')}(\sigma b?) = \sum_{i=1}^{n} P_{E'}(\pi_i') \\ &= \sum_{i=1}^{n}\sum_{j=1}^{l} \underbrace{Q^{E'}(\pi_i)}_{=p}\underbrace{E'(\pi_i')(\mu)}_{=:\alpha_1}\mu(b_1?, s_{i_j}) \\ &= p\alpha_1 \underbrace{\sum_{i=1}^{n}\sum_{j=1}^{l}\mu(b_1?, s_{i_j})}_{=1}. \end{aligned}$$

We can do the same for all α_i for $i = 1, \ldots, \omega$. Note that the choice of the adversary does *not* depend on the chosen trace σ but solely on the presupposed behaviour of H''. Thus we have found $E' \in adv(\mathscr{A}_s, k+1)$ such that $trd(E') = H''$. Hence $H'' \in trd(\mathscr{A}_s, k+1)$, which ends the induction. Since this is possible for every $m \in \mathbb{N}$, we get $\mathscr{A}_i \subseteq_{pioco} \mathscr{A}_s$, ending the proof.

" $\Longleftarrow$ " See Lemma 15 for the proof. In particular we do not even require input enabledness for $\mathscr{A}_s$ in this case. □

Proof of Theorem 17. Let $\mathscr{A} \sqsubseteq_{pioco} \mathscr{B}$ and $\mathscr{B} \sqsubseteq_{pioco} \mathscr{C}$ and $\mathscr{A}$ and $\mathscr{B}$ be input enabled. By Theorem 16 we know, that $\mathscr{A} \sqsubseteq_{TD} \mathscr{B}$. So let $k \in \mathbb{N}$ and $H^* \in trd(\mathscr{A}, k)$. Consequently also $H^* \in trd(\mathscr{B}, k)$ and thus the following embedding holds

$$outcont(\mathscr{A}, H^*, k) \subseteq outcont(\mathscr{B}, H^*, k) \subseteq outcont(\mathscr{C}, H^*, k),$$

and thus $\mathscr{A} \sqsubseteq_{pioco} \mathscr{C}$. □