Testing Probabilistic Distributed Systems*

Robert M. Hierons[1] and Manuel Núñez[2]

[1] Department of Information Systems and Computing, Brunel University
Uxbridge, Middlesex, UB8 3PH United Kingdom
rob.hierons@brunel.ac.uk
[2] Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid, Madrid, Spain
mn@sip.ucm.es

Abstract. There has been much interest in the testing of systems that have physically distributed interfaces and this has been encouraged by recent trends towards the use of such systems. Most formal work in this area has considered the testing of deterministic systems based on deterministic models. However, distributed systems are usually nondeterministic and often can be seen as probabilistic systems in which required or expected probabilities can be attached to the allowable events. This paper provides a formal testing framework for systems with physically distributed interfaces where nondeterministic decisions among alternatives are probabilistically quantified. It first considers testing from systems where there is a unique type of action. In this setting, a *generative* interpretation of probabilities is adequate and a formal framework to test these systems is provided. However, the observable events of a system are usually divided into inputs and outputs. In such situations it is necessary to use the *reactive* interpretation of probabilities.

1 Introduction

It is widely accepted that testing is an important part of the software development process and manual testing is typically expensive and error prone. This has led to interest in the testing of systems based on specifications or models, an area usually called *model based testing* (MBT). Most approaches to MBT use state-based models such as finite state machines or labelled transition systems (see, for example, [18,22,4,11,9]). In addition to providing a formal framework to reason about the correctness of systems, MBT has the enormous advantage of facilitating the automation of the main testing tasks (see [27] for a review of some tool-supported theories).

In order to develop and apply MBT techniques it is necessary to formally define what it means for an implementation to conform to a specification or

* This research was partially supported by the Spanish MEC projects WEST/FAST (TIN2006-15578-C02-01) and TESIS (TIN2009-14312-C02-01), the UK EPSRC project Testing of Probabilistic and Stochastic Systems (EP/G032572/1) and the UCM-BSCH programme to fund research groups (GR58/08 - group number 910606).

J. Hatcliff and E. Zucca (Eds.): FMOODS/FORTE 2010, LNCS 6117, pp. 63–77, 2010.

model. Such definitions are usually called *implementation relations*. For a given model, an implementation relation defines exactly which implementations are correct and thus testing is based on both the model and the implementation relation. Many implementation relations are either trace inclusion or extensions of this such as ioco [26].

In this paper we assume that the *system under test* (SUT) has physically distributed interfaces, called *observation ports* or just ports, and so that in testing we place one tester at each observation port. We assume that the testers do not directly communicate with one another during testing and also that this corresponds to the situation in use: there will be separate users or systems at the ports and while these may share information later, they do not synchronise their actions through communicating when using the SUT. The restrictions imposed on testing have the benefit of making the test infrastructure easier and cheaper to implement: there is no need to introduce an external communications network between the testers. We make one additional assumption, usually made in this context, which is that the testers do not have access to a global clock. It is known that this *decentralised* approach reduces test effectiveness and this topic has received much attention (see, for example, [24,20,23,16]).

The decentralised approach to testing is simpler to implement since there is no need to produce an external communications network through which the testers synchronise their actions. In addition, while we can test individual components without this (assuming each has at most one interface), there is still a need to test the entire system. Crucially, if the decentralised approach corresponds to the *expected use* of the system then the reduced observational power, and corresponding implementation relations, capture what it means for the SUT to be a correct implementation. Thus, even if we can synchronise the testers there is a need to define implementation relations that capture the observational power of agents that will interact with the SUT in use. There can be at least two negative consequences of not doing this. First, our testing may be inefficient: we might produce test cases that aim to find differences between the behaviour of the SUT and the specification where these differences cannot be observed in use and thus are not faults. Second, our testing might not be sound since we could declare that the SUT fails a test even though the behaviour was not observationally different from a behaviour in the specification.

Previous research on testing systems with physically distributed interfaces has not considered models with probabilities. However, distributed systems are often probabilistic in nature and this has led to significant interest in probabilistic models and to the study of semantic models for probabilistic processes. If we consider only probabilistic extensions based on testing we can mention [6,17,29,25,7,21,19,5,8,12]. Even though there are plenty of proposals to test a wide variety of probabilistic processes, as far as we are aware there has been no previous work on the problem of testing with physically distributed interfaces where nondeterministic decisions are probabilistically quantified.

This paper considers two approaches to adding probabilities. First we consider testing from labelled transition systems in which we apply a *generative approach*

[10]; the probabilities of the transitions leaving a state sum to 1. However, to properly specify most systems it is necessary to distinguish between inputs and outputs. The system determines which outputs are produced while the environment controls the inputs. For such systems the generative approach corresponds to having probabilistic information about the environment in addition to the required behaviours of the SUT. Thus, for systems with a distinction between inputs and outputs we use a combination of the *reactive* [17] and generative approaches. Our approach is reactive for inputs: given state s and input $?i$, the sum of the probabilities of the transitions leaving s with input $?i$ is 1. However, it is generative for outputs: given state s, the sum of the probabilities of the transitions leaving s and labelled by an output is 1. Intuitively, this mixed approach corresponds to having probabilistic requirements for the SUT but not placing restrictions on the behaviour of the environment. Similar ideas are used in the probabilistic models considered in [28,2].

In analysing a model we determine the probability of making particular observations. Interestingly, it transpires that this can be problematic when we distinguish between inputs and outputs as a result of races. Specifically, observations are not global traces of the system but sets of global traces that are indistinguishable when there are independent agents/testers at the ports. There can be races between events at different ports and where one or more of these events are inputs the reactive-generative setting does not provide probabilistic information regarding the outcome of such races. As a result, we outlaw these types of races and provide a condition under which such races cannot occur.

It is worth mentioning that our implementation relations are conservative extensions of previous notions for the non-distributed and/or non-probabilistic framework. For the generative approach we show that if we have only one port and we forget probabilistic information then our implementation relation is equivalent to trace inclusion. If we allow probabilities then our implementation relation is equivalent to the natural extension of trace.

This paper is structured as follows. In Section 2 we give preliminary definitions regarding observations that can be made when testing distributed systems. Section 3 then considers the problem of testing from probabilistic labelled transition systems. In Section 4 we investigate the problem of testing from a probabilistic system having a distinction between inputs and outputs. Finally, in Section 5, we draw conclusions and discuss future work.

2 Preliminaries

Throughout this paper we assume that there are m observation ports and that we identify these using the integers in $\mathcal{O} = \{1, \ldots, m\}$. If $\mathcal{A}ct$ denotes the set of actions then this is partitioned into sets $\mathcal{A}ct_1, \ldots, \mathcal{A}ct_m$: for all $o \in \mathcal{O}$, $\mathcal{A}ct_o$ denotes the set of actions that can be observed at o. In Section 4 we consider input output transition systems. When testing from an input output transition system it is common to assume that we can observe the system being in a stable (quiescent) state, in which it cannot progress without receiving further input, and this observation is denoted δ. Quiescence can be observed at all ports.

When a system interacts with its environment it does so through a sequence of actions in $\mathcal{A}ct \cup \{\delta\}$ and this is called a *global trace*. Given a global trace $\sigma \in (\mathcal{A}ct \cup \{\delta\})^*$ we can define the projection $\pi_o(\sigma)$ of σ onto port o, and this is called a *local trace*, in the following way (ϵ represents the empty sequence):

1. $\pi_o(\epsilon) = \epsilon$.
2. If $z \in \mathcal{A}ct_o \cup \{\delta\}$ then $\pi_o(z\sigma) = z\pi_o(\sigma)$.
3. If $z \notin \mathcal{A}ct_o \cup \{\delta\}$ then $\pi_o(z\sigma) = \pi_o(\sigma)$.

Consider, for example, a global trace $a_1b_2c_1$ in which a_1 and c_1 are at port 1 and b_2 is at port 2. Then $\pi_1(a_1b_2c_1) = a_1c_1$ and $\pi_2(a_1b_2c_1) = b_2$.

As stated above, we partition the set of actions so that each can occur at only one port. For systems where the same event can occur at two or more ports we can simply label an event with the port number.

Given global traces $\sigma, \sigma' \in (\mathcal{A}ct \cup \{\delta\})^*$ and observation port o we write $\sigma \sim_o \sigma'$ if σ and σ' cannot be distinguished when only observing the local traces at o. More formally, $\sigma \sim_o \sigma'$ if and only if $\pi_o(\sigma) = \pi_o(\sigma')$. For example, if a_1 and c_1 are events at port 1 and b_2 is an event at port 2 then we have that $a_1b_2c_1 \sim_1 a_1c_1 \sim_1 b_2a_1b_2c_1b_2$. However, a_1c_1 and c_1a_1 are not related under $\sim_1$. We can strengthen this to say what it means for two global traces to be indistinguishable when observing the local traces. Given global traces $\sigma, \sigma' \in (\mathcal{A}ct \cup \{\delta\})^*$ we write $\sigma \sim \sigma'$ if σ and σ' cannot be distinguished when only observing the local traces. Thus, $\sigma \sim \sigma'$ if and only if for all $o \in \mathcal{O}$ we have that $\pi_o(\sigma) = \pi_o(\sigma')$. For example, we have that $a_1b_2c_1 \sim b_2a_1c_1$ since $\pi_1(a_1b_2c_1) = a_1c_1 = \pi_1(b_2a_1c_1)$ and $\pi_2(a_1b_2c_1) = b_2 = \pi_2(b_2a_1c_1)$.

Relations $\sim$ and $\sim_o$ are equivalence relations and so define equivalence classes. Given global trace σ and port o we let $[\sigma]_o$ denote the equivalence class of σ with respect to $\sim_o$ and this is the set of global traces that are indistinguishable from σ when only observing the local trace at o. Thus,

$$[\sigma]_o = \{\sigma' \in (\mathcal{A}ct \cup \{\delta\})^* | \pi_o(\sigma') = \pi_o(\sigma)\}$$

Similarly, given global trace σ we let $[\sigma]$ denote the equivalence class of σ with respect to $\sim$ and this is the set of global traces that are indistinguishable from σ when only observing the local traces. Thus,

$$[\sigma] = \{\sigma' \in (\mathcal{A}ct \cup \{\delta\})^* | \forall o \in \mathcal{O} : \pi_o(\sigma') = \pi_o(\sigma)\}$$

Clearly we have that $\sigma' \sim \sigma$ if and only if for all $o \in \mathcal{O}$ we have that $\sigma' \sim_o \sigma$. In addition, for all σ we have that $[\sigma] = \bigcap_{o \in \mathcal{O}} [\sigma]_o$.

In this paper the set $(0, 1]$ denotes all non-zero probabilities; all real numbers that are greater than 0 and no larger than 1. In addition, $[0, 1] = \{0\} \cup (0, 1]$. In general, we will use multisets of probabilities, instead of sets, since the same probability can be associated with different transitions that we are somehow counting together. We use $\{\!\{$ and $\}\!\}$ as the delimiters for multisets.

3 Implementation Relations for Labelled Transition Systems: A Purely Generative Approach

Labelled transitions systems (LTSs) have states, actions and transitions between states. Recent work has shown how we can define such systems with multiple ports[1] (see, for example, [14,13]). Under the generative interpretation of probabilities, in a state q there is a set of possible transitions, each transition has a probability and the probabilities sum to 1.

Definition 1. *A* probabilistic labelled transition system (PLTS) *s is defined by a tuple $s = (Q, \mathcal{A}ct, T, q_{in})$ in which Q is a countable set of states, $q_{in} \in Q$ is the initial state, $\mathcal{A}ct$ is a countable set of actions, and $T \subseteq Q \times \mathcal{A}ct \times Q \times (0, 1]$, is the transition relation. A transition (q, a, q', p) means that when in state q, with probability p the next event moves s to state q' with action $a \in \mathcal{A}ct$. Naturally, we cannot have two transitions $(q, a, q', p) \in T$ and $(q, a, q', p') \in T$ in which $p \neq p'$. The set $\mathcal{A}ct$ of actions is partitioned into subsets $\mathcal{A}ct_1, \ldots, \mathcal{A}ct_m$ where for all $o \in \mathcal{O}$ we have that $\mathcal{A}ct_o$ denotes the set of actions that can occur at observation port o. We require that for every state $q \in Q$ either $\sum \{\!|\, p \,|\, \exists a, q' : (q, a, q', p) \in T \,|\!\}$ is equal to 1 or q is a deadlock state and so this sum is equal to zero. We let $\mathcal{PLTS}(\mathcal{A}ct)$ denote the set of PLTSs with action set $\mathcal{A}ct$.*

We say that the process s is finitely branching *if for every state $q \in Q$ there are only a finite number of transitions with starting state q. In this paper we only consider processes that are finitely branching.*

Any state $q \in Q$ induces an LTS derived from s by setting the initial state to q, that is, abusing the notation we consider $q = (Q, \mathcal{A}ct, T, q)$.

Let us note that all transitions have non-zero probability. An alternative is to allow the probability of a transition to be from the set $[0, 1]$ but we can simply delete any transition with probability 0 since it does not affect the behaviour of the PLTS. We now introduce notation for PLTSs that we will use in defining implementation relations. In particular, we define the probability of making a sequence of observations from a state q of a PLTS s.

Definition 2. *Given a PLTS $s = (Q, \mathcal{A}ct, T, q_{in})$, a state q of s, and $\sigma \in \mathcal{A}ct^*$, we let $prob(q, \sigma)$ denote the probability of performing the sequence σ from state q. Formally,*

$$prob(q, \sigma) = \begin{cases} 1 & \text{if } \sigma = \epsilon \\ \sum \{\!|\, p \cdot prob(q', \sigma') \,|\, (q, a, q', p) \in T \,|\!\} & \text{if } \sigma = a\sigma' \end{cases}$$

We say that $\sigma \in \mathcal{A}ct^*$ is a trace *of s if $prob(q_{in}, \sigma) > 0$. We denote by $L(s)$ the set of traces of s.*

[1] The work actually defines input output transition systems with multiple ports but it is straightforward to adapt this approach.

We will define an implementation relation for the generative case in the distributed setting. Essentially we wish to say that every $\sigma \in Act^*$ that can be observed in the specification must have the same probability of occurrence in the SUT. However, as a result of there being multiple ports, there may be alternative global traces of the specification and of the implementation that are indistinguishable. We therefore compare the probability of observing elements of equivalence classes rather than particular global traces.

Definition 3. *Let $s = (Q, I, O, T, q_{in})$ be a PLTS and $\sigma \in Act^*$. We define the* probability with which s performs the equivalence class $[\sigma]$*, denoted by $prob(s, [\sigma])$, as*

$$\sum \{\!|\, prob(q_{in}, \sigma') \,|\, \sigma' \in [\sigma] \,|\!\}$$

Let s, r be PLTSs. We write $r \sqsubseteq^G s$ if for all $\sigma \in L(s)$ we have that $prob(s, [\sigma]) = prob(r, [\sigma])$.

In order to show that our relation is a conservative extension of the *classical* non-distributed and non-probabilistic framework, we have the following result where probabilistic information can be reduced to trace containment.

Proposition 1. *Let s, r be single-port PLTSs. We have $r \sqsubseteq^G s$ implies $L(r) \supseteq L(s)$.*

Proof. The proof is easy by taking into account that, under the specified conditions, if $s = (Q, I, O, T, q_{in})$ then for all $\sigma \in \mathcal{A}ct^*$ we have $prob(s, [\sigma]) = prob(q_{in}, \sigma)$. □

However, if we retain probabilistic information we find that our relation is equivalent to the natural extension of trace inclusion.

Proposition 2. *Let s, r be single-port PLTSs. We have $r \sqsubseteq^G s$ implies that for all $\sigma \in L(s)$ we have that $prob(r, \sigma) = prob(s, \sigma)$.*

Proof. Since $r \sqsubseteq^G s$, if $\sigma \in L(s)$ then we have that $prob(s, [\sigma]) = prob(r, [\sigma])$. However, for all $\sigma \in \mathcal{A}ct^*$ we have $prob(s, [\sigma]) = prob(s, \sigma)$ and $prob(r, [\sigma]) = prob(r, \sigma)$ and so the result follows. □

We would now like to comment on a limitation of $\sqsubseteq^G$. Even though it seems to be an appropriate adaptation of the generative approach to the distributed setting, it has a main drawback: it does not properly capture the notion of *complete trace*. This problem is not related to the additional complexity introduced by distributed ports. Therefore, we will illustrate it with two single-port systems. Let us consider the following two processes shown in Figure 1:

1. Process s initially produces a with two different transitions. The first transition, with probability $\frac{1}{2}$, reaches a deadlock state while the second transition, also with probability $\frac{1}{2}$, reaches a state that can perform b and reaches a deadlock state.

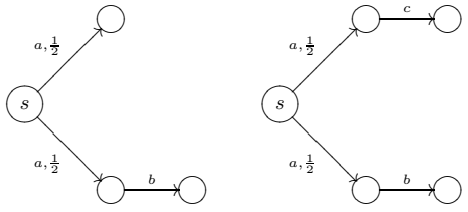

Fig. 1. Processes s and r

2. Process r initially produces a with two different transitions. The first transition, with probability $\frac{1}{2}$, reaches a state that can perform c followed by deadlock while the second transition, also with probability $\frac{1}{2}$, reaches a states that can perform b and then is also followed by deadlock.

We certainly have that $r \sqsubseteq^G s$ since we only check the probabilities associated with the traces a (equal to 1 in both processes) and ab (equal to $\frac{1}{2}$ in both processes). However, we might consider that s is offering a behaviour that r cannot *simulate*. Specifically, s can perform a to reach a deadlock state while r cannot. In order to capture this type of behaviour, we can extend the alphabet of actions with a special action that will be associated with deadlock states. We will add self-loops labelled by this special action and probability 1 in deadlock states. This is exactly the idea of *quiescence* that we will use in the next section.

Definition 4. *Let $s = (Q, \mathcal{A}ct, T, q_{in})$ be a PLTS. We can extend the set of transitions T to a new set T_δ by adding the transition $(q, \delta, q, 1)$ for each state $q \in Q$ such that $\sum \{\!|\, p \,|\, \exists a, q' : (q, a, q', p) \in T \,|\!\} = 0$. The* augmented *PLTS is given by $s_\delta = (Q, \mathcal{A}ct \cup \{\delta\}, T_\delta, q_{in})$.*

Let s, r be PLTSs and s_δ, r_δ be the corresponding augmented PLTSs. We write $r \sqsubseteq^G_\delta s$ if for all $\sigma \in L(s_\delta)$ we have that $prob(s_\delta, [\sigma]) = prob(r_\delta, [\sigma])$.

This new implementation relation properly captures complete traces since a trace σ reaching a deadlock state is *transformed* into the traces $\sigma\delta^n$, for $n \geq 0$. For example, if we consider the processes r and s that we used to motivate the new relation we have that they are not related under $\sqsubseteq^G_\delta$. In addition to cope with complete traces, the new implementation relation has a very interesting property. We have that the asymmetry in its definition, since only traces of s are considered, is not real since this order relation masks an equivalence relation as the following result shows.

Proposition 3. *Let s, r be PLTSs. We have that $r \sqsubseteq^G_\delta s$ implies $s \sqsubseteq^G_\delta r$.*

Proof. We use proof by contradiction, assuming that $r \sqsubseteq^G_\delta s$ but that $s \sqsubseteq^G_\delta r$ does not hold. Let σ be a shortest element of $L(r)$ such that $prob(r, [\sigma]) \neq prob(s, [\sigma])$. If $prob(s, \sigma) > 0$ then, since $prob(r, [\sigma]) \neq prob(s, [\sigma])$, we know that $r \sqsubseteq^G_\delta s$ does not hold, providing a contradiction. We therefore must have that $prob(s, \sigma) = 0$ and therefore $\sigma \notin L(s)$.

Let k denote the length of $\sigma \in \mathcal{A}ct^*$ and assume that in s we have l distinct equivalence classes $[\sigma_1], \ldots, [\sigma_l]$ of sequences that have length k. Let us remark that traces σ reaching a deadlock state and having a length $k_\sigma < k$ will contribute to this set of classes with the class $[\sigma\delta^{k-k_\sigma}]$.

Since probabilities are generative, it is easy to check that $\sum_{1 \leq i \leq l} prob(s, [\sigma_i]) = 1$. Further, since $r \sqsubseteq_\delta^G s$, $\sum_{1 \leq i \leq l} prob(r, [\sigma_i]) \geq \sum_{1 \leq i \leq l} prob(s, [\sigma_i])$ and so $\sum_{1 \leq i \leq l} prob(r, [\sigma_i]) = 1$.

Let us note that $prob(r, [\sigma]) \neq prob(s, [\sigma])$ and $prob(s, \sigma) = 0$ and so we have $\sum_{1 \leq i \leq l} prob(r, [\sigma_i]) + prob(r, [\sigma]) > 1$. However, for all $1 \leq i \leq l$ we have that $\sigma \not\sim \sigma_i$ and the sequences $\sigma, \sigma_1, \ldots, \sigma_l$ all have length k and so we must have that $\sum_{1 \leq i \leq l} prob(r, [\sigma_i]) + prob(r, [\sigma]) \leq 1$. This provides a contradiction and so the result follows. □

Thus, $\sqsubseteq_\delta^G$ is an equivalence relation. Normally, an implementation relation is a preorder but not an equivalence relation since it allows a range of implementation decisions. Future work will consider whether there are suitable implementation relations for the generative case that are not equivalence relations.

4 Implementation Relations for Input Output Transition Systems: A Reactive-Generative Approach

Many systems interact with their environment through inputs and outputs and in this section we consider such systems and the observations that can be made, which are sequences of inputs and output (input output sequences).

When a system interacts with its environment through inputs and outputs there is often an asymmetry between these since the environment controls the inputs while the system controls the outputs. This has led to the use of input output transition systems (IOTSs), which essentially are LTSs where we distinguish between input and output. We now define a probabilistic IOTS that has multiple ports. We use the reactive scenario for inputs and the generative for outputs. However, we do attach probabilities to inputs since there may be more than one transition leaving a state q with a given input $?x$: the environment chooses the input to supply but the system determines which transition to take.

Definition 5. *A* probabilistic input-output transition system (PIOTS) *s is defined by a tuple $s = (Q, I, O, T, q_{in})$ in which Q is a countable set of states, $q_{in} \in Q$ is the initial state, I is a countable set of inputs, O is a countable set of outputs, and $T \subseteq Q \times (I \cup O) \times Q \times (0, 1]$ is the transition relation. A transition (q, a, q', p) means that from state q it is possible to move to state q' with action $a \in I \cup O$ with probability p. Again, we cannot have two transitions $(q, a, q', p) \in T$ and $(q, a, q', p') \in T$ in which $p \neq p'$. If $a \in O$ then we should interpret the probability p of (q, a, q', p) as meaning that if an output occurs in state q before input is provided then with probability p this transition occurs. Therefore, for every state q we must have that $\sum \{\!|\, p \,|\, \exists q', a : (q, a, q', p) \in T \wedge a \in O \,|\!\}$ is either 1 or 0 (if the state cannot produce any output). Further, if $a \in I$ then*

we must have that the sum of the probabilities of transitions leaving q with input a, that is $\sum \{\!|\, p \mid \exists q' : (q, a, q', p) \in T \,|\!\}$, is either 1 or 0 (if the input is not available at that state). This means that once an available input is chosen by the environment, we can forget the other available inputs and concentrate on the probability distribution function governing the transitions labelled by a.

A state $q \in Q$ is quiescent *if all transitions from q involve input. We can extend the set of transitions T to a new set T_δ by adding the transition $(q, \delta, q, 1)$ for each quiescent state q. We let $\mathcal{A}ct = I \cup O \cup \{\delta\}$ denote the set of actions.*

We partition the set I of inputs into $I_1, \ldots, I_m$ in which for port $o \in \mathcal{O}$ we have that I_o is the set of inputs that can be received at port o. Similarly, we partition the set O of outputs into sets $O_1, \ldots, O_m$. We let $\mathcal{PIOTS}(I, O)$ denote the set of PIOTSs with input set I and output set O.

We say that the process s is input-enabled *if for all $q \in Q$ and all $?i \in I$ there exists $q' \in Q$ and probability $p \in (0, 1]$ such that $(q, ?i, q', p) \in T$. We say that the process s is* output-divergent *if it can reach a state in which there is an infinite path that contains outputs only. We say that s is* finitely branching *if for every state $q \in Q$ there are only a finite number of transitions with starting state q. In this paper we only consider processes that are finitely branching and are not output-divergent.*

Let us remark that if we consider δ as a regular output action, then we can say that the sum of the probabilities associated with a state is always equal to 1, that is, $\sum \{\!|\, p \mid \exists q', a : (q, a, q', p) \in T_\delta \wedge a \notin I \,|\!\} = 1$. In addition, as usual, we precede the name of an input by ? and we precede the name of an output by !. We will often label inputs and outputs in order to make their port clear. For example, $?i_p$ denotes an input at p and $!o_p$ denotes an output at p. An alternative, used in [14], to this notion of (probabilistic) input output transition systems is to allow outputs to be tuples of values but the formalism used in this paper has the advantage of simplifying the notation and analysis.

Traces are sequences of actions, possibly including quiescence, and are usually called *suspension traces*. In this paper we simply call them global traces. The following is standard notation in the context of **ioco**: the implementation relation usually used in testing from a single-port IOTS [26].

Definition 6. *Let $s = (Q, I, O, T, q_{in})$ be a PIOTS. We use the following notation.*

1. *If $(q, a, q', p) \in T_\delta$, for $a \in \mathcal{A}ct$, then we write $q \xrightarrow{a} q'$ and $q \xrightarrow{a}$.*
2. *We write $q \stackrel{\sigma}{\Longrightarrow} q'$ for $\sigma = a_1 \ldots a_m \in \mathcal{A}ct^*$ if there exist $q_0, \ldots, q_m$, $q = q_0$, $q' = q_m$ such that for all $0 \leq i < m$ we have that $q_i \xrightarrow{a_{i+1}} q_{i+1}$.*
3. *If there exists q' such that $q_{in} \stackrel{\sigma}{\Longrightarrow} q'$ we say that σ is a* trace *of s. We let $\mathcal{T}r^*(s)$ denote the set of traces of s.*
4. *Let $q \in Q$ be a state and $\sigma \in \mathcal{T}r^*(s)$ be a trace. We introduce the following concepts.*
   *(a) s **after** $\sigma = \{q \in Q | q_{in} \stackrel{\sigma}{\Longrightarrow} q\}$.*
   (b) $\mathbf{out}(q) = \{!o \in O | q \stackrel{!o}{\Longrightarrow}\}$.

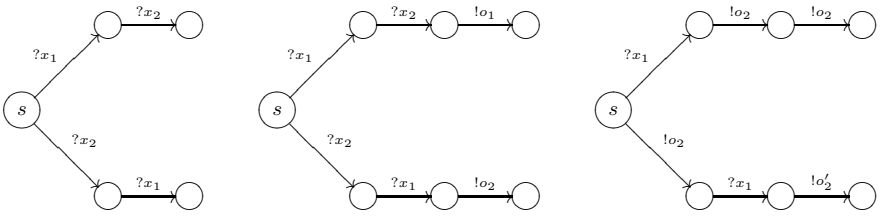

Fig. 2. Process s_1, s_2, and s_3

5. *Given an input output sequence σ, we let* $\mathbf{in}(\sigma)$ *denote the input sequence formed by removing all of elements of σ that are not inputs.*

Let us note that we have initially abstracted probabilistic information in the definition of trace. This information will be incorporated when defining our implementation relations.

Next we recall the definition of **ioco** [26] and its adaption to systems with multiple ports that we call **dioco** [14].

Definition 7. *Let $s, r \in \mathcal{PIOTS}(I, O)$. We write r* **ioco** *s if for every trace $\sigma \in \mathcal{T}r^*(s)$ we have that* $\mathbf{out}(r \textbf{ after } \sigma) \subseteq \mathbf{out}(s \textbf{ after } \sigma)$.

We write r **dioco** *s if for every trace σ such that $r \stackrel{\sigma}{\Longrightarrow} r'$ for some r' that is in a quiescent state, if there is a global trace $\sigma_1 \in \mathcal{T}r^*(s)$ such that $\mathbf{in}(\sigma_1) \sim \mathbf{in}(\sigma)$ then there exists a trace $\sigma' \in \mathcal{T}r^*(s)$ such that $s \stackrel{\sigma'}{\Longrightarrow}$ and $\sigma' \sim \sigma$.*

As expected, the previous implementation relation discarded probabilistic information. It is important to explain why we restrict attention to traces that end in quiescent states. Let us consider the following processes:

1. Process s that can produce output $!o_1$ at port 1, then output $!o_2$ at port 2 and then deadlocks; and
2. Process r that can produce output $!o_2$ at port 2, then output $!o_1$ at port 1 and then deadlocks.

If we can observe just $!o_2$ in r then we can distinguish between r and s but the environment cannot block the following $!o_1$. This is because the agents/testers at the ports can only exchange information in quiescent states as a result of not being able to synchronise their actions during testing.

An interesting, but more complicated, alternative has been recently defined in [15] where only infinite traces are used to define **dioco**. Since a finite trace is a prefixes of an infinite one (since we can always extend a trace with either an output or with δ), this notion captures the nature of **dioco**. For example, the infinite traces that the previous processes can perform are $!o_1!o_2\delta\delta\cdots$ and $!o_2!o_1\delta\delta\cdots$, respectively, and these are related by $\sim$.

In order to define an appropriate probabilistic extension of **dioco**, we need to restrict our attention to a class of systems without *pathological* behaviours. Let

us consider the process s_1 in Figure 2 left, in which $?x_1$ is input at port 1 and $?x_2$ is input at port 2. All transitions have probability 1. We might ask what probability we should associate with global traces that are indistinguishable from $?x_1?x_2$. We cannot simply sum the probabilities of the two global traces that are equivalent to $?x_1?x_2$ under $\sim$ since we obtain the result 2.

Now let us consider process s_2 in Figure 2 centre, in which $!o_1$ is at port 1 and $!o_2$ is at port 2. We might ask what probability we should associate with global traces that are indistinguishable from $?x_1?x_2!o_1$. However, whether we observe $?x_1?x_2!o_1$ depends on the outcome of a race between $?x_1$ and $?x_2$ and we have no probabilistic information regarding this race. Thus, we cannot give such a probability. However, for a model to be useful it should define the probability of an observation. Thus, we either need to use a generative approach, which provides the probabilities for the outcomes of a race, or we restrict attention to PIOTSs without such races. We now discuss how the latter can be done.

Previous work on using IOTSs for distributed systems has defined the *mioco* implementation relation where we make global observations but there are multiple ports [3]. This does not require that an IOTS is input-enabled but does insist that if no transition is defined for input $?i_o \in I_o$ in state q then there are no transitions from q with an input from I_o. This corresponds to the SUT being able to block input at an interface. We make a similar assumption, which is that for any state q we cannot have transitions from q for inputs from different ports. This corresponds to a design that avoids races by restricting input to one port at a time. This restriction is motivated by the observation that, as seen in Figure 2 left and centre, we need to avoid such races if we wish to be able to assign probabilities to observations. Interestingly, work on Message Sequence Charts (MSCs) [1] has defined a pathology in which the next events after branching are on different processes: our restriction is similar to outlawing this pathology.

It is not sufficient to outlaw races between inputs since we can have a race between an input and an output at different ports. An example of this is given in s_3 from Figure 2 right. Here we cannot assign a probability to there being a global trace indistinguishable from $?x_1!o_2!o_2$ since there is a race between $?x_1$ and $!o_2$. However, having different possible outputs at different ports or inputs and outputs at the same port, from a state q, causes no problem since PIOTSs do contain probabilistic information regarding these choices. Thus, a PIOTS in which there can be races between inputs or between inputs and outputs should not be allowed since we cannot assign probabilities to traces. We will require any PIOTS to be consistent, as defined below.

Definition 8. *Let $s = (Q, I, O, T, q_{in})$ be a PIOTS. We say that s is* consistent *if for every state $q \in Q$ if there exist $a_1, a_2 \in I \cup O$ such that $q \xrightarrow{a_1}$ and $q \xrightarrow{a_2}$ then either both of them are outputs or they are at the same port.*

Throughout the rest of this paper we only consider PIOTSs that are consistent, in addition to being finitely branching and not output-divergent. We can adapt notation defined for PLTSs; we consider a PIOTS to be a PLTS and so reuse concepts such as the probability $prob(q, \sigma)$ of performing a given sequence of

actions σ from a state q. We assume that observations are made in quiescent states and restrict attention to paths that end in quiescent states and adapt the **dioco** implementation relation to the probabilistic framework.

Definition 9. *Let $s, r \in \mathcal{PIOTS}(I, O)$. We write $r \sqsubseteq^R s$ if for every global trace σ such that $s \stackrel{\sigma}{\Longrightarrow} s'$ for some s' that is in a quiescent state, we have that $prob(s, [\sigma\delta]) = prob(r, [\sigma\delta])$.*

In the above we include δ in the global traces over which we sum probabilities to ensure that we are considering quiescent traces in each process. The notation $prob(s, [\sigma\delta])$ is given in Definition 3. Let us remark that, in contrast with $\sqsubseteq^G_\delta$ but similar to $\sqsubseteq^G$, the relation $\sqsubseteq^R$ is not symmetric.

Proposition 4. *There exist PIOTSs r and s such that $r \sqsubseteq^R s$ but we do not have that $s \sqsubseteq^R r$.*

Proof. It is sufficient to consider processes r and s such that:

1. Process s has only one transition which is a self-loop with label δ.
2. Process r has a self-loop with label δ and a transition with input $?x$ to a state r' that has only one transition, which is a self-loop with label δ.

Then $r \sqsubseteq^R s$ since the quiescent traces of s contain only δ and these are also traces of r and have the same probabilities. However, to see that $s \sqsubseteq^R r$ does not hold it is sufficient to consider the quiescent trace $?x\delta$ of r. □

If we consider single-port systems and we also forget probabilistic information then we have that our relation reduces to **ioco**. Let us remark that **ioco** is only defined for input-enabled implementations and so we have to make this restriction. However, input-enabled single-port PIOTSs can be still consistent since, obviously, they cannot perform actions at different ports.

Proposition 5. *Let s, r be single-port PIOTSs such that r is input-enabled and $r \sqsubseteq^R s$. If $\sigma \in \mathcal{T}r^*(s)$ and either $a \in \mathbf{out}(r \textbf{ after } \sigma)$ or $a \in \mathbf{out}(s \textbf{ after } \sigma)$ then $prob(r, \sigma a) = prob(s, \sigma a)$.*

Proof. We will use proof by induction on the length of σ. The base case is $\sigma = \epsilon$. First, since s is not output-divergent, the sum of the probabilities of quiescent traces of s that contain only output and exactly one δ is 1. In addition, there are a finite number of such sequences in s (since s is finitely-branching and is not output-divergent), and since $r \sqsubseteq^R s$ and there is only one port we have that these are also quiescent sequences of r and in r they have the same probabilities. It is now sufficient, for each element $a \in \mathbf{out}(r \textbf{ after } \epsilon) \cup \mathbf{out}(s \textbf{ after } \epsilon)$, to consider such traces that start with output a and sum the probabilities of the corresponding traces. The result thus holds for the base case.

Now we assume that the result holds for all traces of length less than that of $\sigma \in \mathcal{T}r^*(s)$. By the inductive hypothesis, since every prefix of σ is a trace of s we have that $prob(s, \sigma) = prob(r, \sigma)$. Since s is finitely-branching we have that $\mathbf{out}(s \textbf{ after } \sigma)$ is finite and let us suppose that $\mathbf{out}(s \textbf{ after } \sigma) = \{a_1, \ldots, a_k\}$.

Since s is not output-divergent, for each $1 \leq i \leq k$ we have that there is a finite set A_i of (minimal) traces of s that extend σ with only outputs and δ and that contain exactly one extra δ (at the end). Thus, σ' is in A_i if and only if $\sigma' \in \mathcal{T}r^*(s)$, σ' extends σ with outputs and exactly one δ, ends in δ, and σa_i is a prefix of σ'. Since outputs are generative, $prob(s, \sigma a_i) = \sum_{\sigma' \in A_i} prob(s, \sigma')$. Since $r \sqsubseteq^R s$ and there is only one port, for all $\sigma'' \in A_i$ we must have that $prob(r, \sigma'') = prob(s, \sigma'')$ and so for all $1 \leq i \leq k$ we have that $prob(r, \sigma a_i) \geq prob(s, \sigma a_i)$. Now let us note that we have $prob(s, \sigma) = prob(r, \sigma)$, $prob(s, \sigma) = \sum_{1 \leq i \leq k} prob(s, \sigma a_i)$ and $prob(r, \sigma) \geq \sum_{1 \leq i \leq k} prob(r, \sigma a_i) = \sum_{1 \leq i \leq k} prob(s, \sigma a_i)$. Thus, for all $1 \leq i \leq k$ we have $prob(r, \sigma a_i) = prob(s, \sigma a_i)$ and $\mathbf{out}(r \textbf{ after } \sigma) = \{a_1, \ldots . a_k\}$. □

The following is an immediate consequence.

Proposition 6. *Let s, r be single-port PIOTSs. If r is input-enabled then we have $r \sqsubseteq^R s$ implies that r* **ioco** *s.*

Our last result relates the implementation relations presented in this paper. If the set of inputs is empty then the relations should coincide. Under this assumption we do not have $r \sqsubseteq^R s$ if and only if $r \sqsubseteq^G s$ due to the inadequate treatment of complete traces under $\sqsubseteq^G$. Fortunately, the expected result holds with the variant of $\sqsubseteq^G$.

Proposition 7. *Let s, r be single-port PIOTSs with empty sets of inputs. We have $r \sqsubseteq^R s$ if and only if $r \sqsubseteq^G_\delta s$.*

5 Conclusions

This paper has investigated the problem of testing a probabilistic system that has distributed interfaces, called ports. This is a significant problem since distributed systems are becoming increasingly important and are often probabilistic in nature. Interestingly, while there has been much separate work on testing probabilistic systems and testing systems with distributed interfaces, this appears to be the first that considers the combination.

We assume that the *system under test (SUT)* has physically distributed ports and we place one local tester at each port. We also assume that the testers do not communicate with one another during testing and this corresponds to the situation in use: the separate users/systems at the ports might share information later but do not synchronise when using the SUT. Each local tester observes a projection of the global trace that occurs and this is called a local trace.

Initially we considered a generative situation where the probabilities on transitions leaving a state sum to 1. This is similar to a Markov Chain and corresponds to testing a closed system or one in which we know the probabilities of events from the environment that affect the SUT. In this context we explored testing from probabilistic labelled transition systems by defining an implementation relation where the local testers may return their sets of observations and a verdict can be produced from these (local traces).

Sometimes we distinguish between input and output: the SUT controls output and the environment controls input. Here, input output transition systems are more suitable that labelled transition systems and we require a reactive scenario. We investigated the problem of testing from a probabilistic input output transition system (PIOTS) that has multiple ports. In analysing such a model we need to determine the probability of observing a trace that is equivalent to a given trace σ. For some models the presence of races means that this probability is undefined and so we defined a class of model for which such problems do not occur and we also defined an implementation relation.

There are several avenues for future work. First, there is the problem of producing test generation algorithms that direct testing in order to achieve a given objective. It is also necessary to consider how tests should be applied and verdicts assigned but the latter essentially corresponds to estimating the probabilities of sequences of events in the SUT through sampling and comparing the estimate with the required probability. In addition, simulation relations have been defined for distributed systems and there is the problem of extending these to probabilistic distributed systems. Finally, it may be possible to extend the implementation relation defined for the reactive case by allowing races but using symbolic values to represent the probabilities of inputs supplied by the environment.

References

1. Ben-Abdallah, H., Leue, S.: Syntactic detection of process divergence and non-local choice in Message Sequence Charts. In: Brinksma, E. (ed.) TACAS 1997. LNCS, vol. 1217, pp. 259–274. Springer, Heidelberg (1997)
2. Bravetti, M., Aldini, A.: Discrete time generative-reactive probabilistic processes with different advancing speeds. Theoretical Computer Science 290(1), 355–406 (2003)
3. Brinksma, E., Heerink, L., Tretmans, J.: Factorized test generation for multi-input/output transition systems. In: 11th IFIP Workshop on Testing of Communicating Systems, IWTCS 1998, pp. 67–82. Kluwer Academic Publishers, Dordrecht (1998)
4. Brinksma, E., Tretmans, J.: Testing transition systems: An annotated bibliography. In: Cassez, F., Jard, C., Rozoy, B., Dermot, M. (eds.) MOVEP 2000. LNCS, vol. 2067, pp. 187–195. Springer, Heidelberg (2001)
5. Cheung, L., Stoelinga, M., Vaandrager, F.: A testing scenario for probabilistic processes. Journal of the ACM 54(6), Article 29 (2007)
6. Christoff, I.: Testing equivalences and fully abstract models for probabilistic processes. In: Baeten, J.C.M., Klop, J.W. (eds.) CONCUR 1990. LNCS, vol. 458, pp. 126–140. Springer, Heidelberg (1990)
7. Cleaveland, R., Dayar, Z., Smolka, S.A., Yuen, S.: Testing preorders for probabilistic processes. Information and Computation 154(2), 93–148 (1999)
8. Deng, Y., van Glabbeek, R., Hennessy, M., Morgan, C.: Characterising testing preorders for finite probabilistic processes. Logical Methods in Computer Science 4(4) (2008)
9. Frantzen, L., Merayo, M.G., Núñez, M.: A brief history of A-MOST. Journal of Logic and Algebraic Programming 78(6), 417–424 (2009)
10. van Glabbeek, R., Smolka, S.A., Steffen, B.: Reactive, generative and stratified models of probabilistic processes. Information and Computation 121(1), 59–80 (1995)

11. Hierons, R.M., Bogdanov, K., Bowen, J.P., Cleaveland, R., Derrick, J., Dick, J., Gheorghe, M., Harman, M., Kapoor, K., Krause, P., Luettgen, G., Simons, A.J.H., Vilkomir, S., Woodward, M.R., Zedan, H.: Using formal methods to support testing. ACM Computing Surveys 41(2) (2009)
12. Hierons, R.M., Merayo, M.G.: Mutation testing from probabilistic and stochastic finite state machines. Journal of Systems and Software 82(11), 1804–1818 (2009)
13. Hierons, R.M., Merayo, M.G., Núñez, M.: Controllable test cases for the distributed test architecture. In: Cha, S(S.), Choi, J.-Y., Kim, M., Lee, I., Viswanathan, M. (eds.) ATVA 2008. LNCS, vol. 5311, pp. 201–215. Springer, Heidelberg (2008)
14. Hierons, R.M., Merayo, M.G., Núñez, M.: Implementation relations for the distributed test architecture. In: Suzuki, K., Higashino, T., Ulrich, A., Hasegawa, T. (eds.) TestCom/FATES 2008. LNCS, vol. 5047, pp. 200–215. Springer, Heidelberg (2008)
15. Hierons, R.M., Merayo, M.G., Núñez, M.: Implementation relations and test generation for systems with distributed interfaces (submitted, 2010)
16. Hierons, R.M., Ural, H.: The effect of the distributed test architecture on the power of testing. The Computer Journal 51(4), 497–510 (2008)
17. Larsen, K., Skou, A.: Bisimulation through probabilistic testing. Information and Computation 94(1), 1–28 (1991)
18. Lee, D., Yannakakis, M.: Principles and methods of testing finite state machines: A survey. Proceedings of the IEEE 84(8), 1090–1123 (1996)
19. López, N., Núñez, M., Rodríguez, I.: Specification, testing and implementation relations for symbolic-probabilistic systems. Theoretical Computer Science 353(1-3), 228–248 (2006)
20. Luo, G., Dssouli, R., von Bochmann, G.: Generating synchronizable test sequences based on finite state machine with distributed ports. In: 6th IFIP Workshop on Protocol Test Systems, IWPTS 1993, pp. 139–153. North-Holland, Amsterdam (1993)
21. Núñez, M.: Algebraic theory of probabilistic processes. Journal of Logic and Algebraic Programming 56(1-2), 117–177 (2003)
22. Petrenko, A.: Fault model-driven test derivation from finite state models: Annotated bibliography. In: Cassez, F., Jard, C., Rozoy, B., Dermot, M. (eds.) MOVEP 2000. LNCS, vol. 2067, pp. 196–205. Springer, Heidelberg (2001)
23. Rafiq, O., Cacciari, L.: Coordination algorithm for distributed testing. The Journal of Supercomputing 24(2), 203–211 (2003)
24. Sarikaya, B., von Bochmann, G.: Synchronization and specification issues in protocol testing. IEEE Transactions on Communications 32, 389–395 (1984)
25. Segala, R.: Testing probabilistic automata. In: Sassone, V., Montanari, U. (eds.) CONCUR 1996. LNCS, vol. 1119, pp. 299–314. Springer, Heidelberg (1996)
26. Tretmans, J.: Model based testing with labelled transition systems. In: Hierons, R.M., Bowen, J.P., Harman, M. (eds.) FORTEST 2008. LNCS, vol. 4949, pp. 1–38. Springer, Heidelberg (2008)
27. Utting, M., Legeard, B.: Practical Model-Based Testing: A Tools Approach. Morgan-Kaufmann, San Francisco (2007)
28. Wu, S.-H., Smolka, S.A., Stark, E.W.: Composition and behaviors of probabilistic I/O automata. Theoretical Computer Science 176(1-2), 1–37 (1997)
29. Yi, W., Larsen, K.G.: Testing probabilistic and nondeterministic processes. In: 12th IFIP/WG6.1 Int. Symposium on Protocol Specification, Testing and Verification, PSTV 1992, pp. 47–61. North Holland, Amsterdam (1992)