Composition and Behaviors of Probabilistic I/O Automata

Sue-Hwey Wu*, Scott A. Smolka*, Eugene W. Stark**

Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794 USA***

Abstract. We augment the I/O automaton model of Lynch and Tuttle with probability, as a step toward the ultimate goal of obtaining a useful tool for specifying and reasoning about asynchronous probabilistic systems. Our new model, called *probabilistic I/O automata*, preserves the fundamental properties of the I/O automaton model, such as the asymmetric treatment of input and output and the pleasant notion of asynchronous composition. For the class of probabilistic I/O automata without internal actions, we show that *probabilistic behavior maps*, which are an abstract representation of I/O automaton behavior in terms of a certain expectation operator, are compositional and fully abstract with respect to a natural notion of probabilistic testing.

1 Introduction

I/O automata are a kind of state machine that have been proposed by Lynch and Tuttle [LT87] as a tool for specifying and reasoning about asynchronous systems. The distinguishing features of the I/O automaton model are: an asymmetric treatment of input and output actions, a notion of asynchronous composition that takes a "compatible" collection of I/O automata and produces a new I/O automaton as a result, a simple correspondence between computations of a composite I/O automaton and certain collections of computations of its component automata, the treatment of liveness properties through the introduction of a "fairness partition" on the action set of an automaton, and the use of simulation-based techniques for proving that the set of action sequences that can be produced by one I/O automaton is a subset of the set of action sequences that can be produced by another. In this paper, we consider the problem of augmenting I/O automata with probability information, as a step toward the ultimate goal of obtaining a useful tool for specifying and reasoning about asynchronous probabilistic systems. As much as possible, we would like to preserve the characteristic features of the I/O automaton model, especially the

* Research supported in part by NSF Grants CCR-9120995 and CCR-9208585, and AFOSR Grant F49620-93-1-0250DEF.
** Research supported in part by NSF Grant CCR-8902215.
*** E-mail addresses: suewu@cs.sunysb.edu, sas@cs.sunysb.edu, stark@cs.sunysb.edu

asymmetric treatment of input and output and the associated pleasant notion of composition.

There are some interesting issues that arise when one attempts to add probability to I/O automata. These issues derive from the input/output dichotomy and also from the asynchronous notion of composition for such automata, in which for any given state of a composite automaton there can be a number of component automata "competing" with each other to control the execution of the next action. It is inadequate simply to introduce, for each state q, a single probability distribution μ on the set of all transitions from state q, because intuitively there is no good reason why the choice between input transitions (which are "externally controlled" by the environment of the automaton) and output or internal transitions (which are "locally controlled" by the automaton itself) should admit a meaningful probabilistic description independent of any particular environment. So, instead of one probability distribution for *all* transitions for state q, we introduce several probability distributions: one distribution over all the locally controlled transitions from state q, and separate distributions for each input action e. Our model is thus a kind of hybrid between the "reactive" and "generative" approaches described in [vGSST90].

The introduction of multiple probability distributions on transitions still does not solve all problems, however. Although within a single automaton we do not wish to ascribe probabilities to choices between externally controlled and locally controlled transitions, when automata are composed we do wish to have a natural probabilistic description of the outcome of the competition between component automata for control of the next action. To this end we introduce the concept of the *delay parameter* $\delta(q)$ associated with each state q. The idea is as follows: when a component automaton in a composite system arrives in state q, it draws a random delay time from an exponential distribution with parameter $\delta(q)$. This time describes the length of time the automaton will remain in state q before executing its next locally controlled action. The competition between several component automata vying for control of the next locally controlled action is won by the automaton having the least amount of delay time left. If we assume that the delay time distributions of component automata are independent, we can assign a definite probability to the event that any given component automaton will win the competition in any given system state. The "memoryless" property of the exponential distribution makes it irrelevant whether the component automata draw one delay time when they first enter their local state, or whether each component draws a new delay time after each global transition. This last feature makes it possible to give a simple definition of composition for probabilistic I/O automata.

Having obtained definitions for probabilistic I/O automata and their composition, it becomes interesting to consider their "external behaviors." The external behavior of an ordinary I/O automaton is the set of all sequences of external actions that can be produced in the various executions of the automaton. Lynch and Tuttle show that the mapping from I/O automata to external behaviors respects composition, in the sense that the external behavior of a

composite automaton is determined in a natural way by the external behaviors of the component automata. Since ordinary I/O automata have sets of action sequences as their external behaviors, one might expect probabilistic I/O automata to have probability distributions on action sequences as their external behaviors. Although this intuition can be validated to a certain extent, if one wishes the mapping from probabilistic I/O automata to external behaviors to respect composition, then the situation requires a bit more finesse than simply using probability distributions on action sequences as external behaviors. The reason is this: to compute the probability distribution on action sequences determined by a composite automaton, it is necessary to have information about the internal delays of each of the component automata as well as the probability distribution they each induce on action sequences.

Our notion of external behavior for probabilistic I/O automata is obtained as follows: Let A be a probabilistic I/O automaton having no input or internal actions, and satisfying certain finite branching conditions. Then the automaton A induces a probability distribution on the set of all its executions, and, given an action sequence $\alpha = e_0 e_1 \ldots e_{n-1}$, a conditional distribution on the subset X_α of all executions whose action sequences extend α. We may view the sequences $d_0 d_1 \ldots d_n$ of delay parameters associated with the states in such an execution as the values of an $(n+1)$-dimensional random variable D defined on X_α. We define $\mathcal{E}_\alpha^A$ to be the mapping that takes each real-valued function $g : \mathcal{R}^{n+1} \to \mathcal{R}$ to its expectation, weighted by the probability of the set X_α. Actually, the formal definition of $\mathcal{E}_\alpha^A$ given in Section 4 makes sense even if A has a nonempty set of input actions, though the interpretation as a weighted expectation only applies in the more restricted situation. We show that a compositional notion of behavior is obtained, if one takes the external behavior of an automaton A to be the mapping $\mathcal{E}^A$ that assigns to each action sequence α of length n the associated functional $\mathcal{E}_\alpha^A$ on $\mathcal{R}^{n+1} \to \mathcal{R}$.

Besides showing that our notion of behavior is compositional, we are also able to show that it is "fully abstract," in the sense that any two automata having distinct behaviors can be distinguished by a certain kind of probabilistic test. The key idea in the proof is that the success probability of tests in a certain class gives us the expectations of certain rational functions of the delay parameters. Using the uniqueness of partial fraction expansions of rational functions, we can recover full information about the functionals $\mathcal{E}_\alpha^A$ from these expectations.

The recent research literature contains a plethora of proposals for probabilistic models. Each of these proposals addresses different issues, and introduces probability in a different way. In *reactive processes* [Rab63, LS92], for each state q and action e, a separate probability distribution is associated with the set of e-labeled transitions leaving state q. In contrast, in *generative processes* [vGSST90], for each state q a single probability distribution is associated with the set of all transitions leaving state q. The *stratified processes* [vGSST90] model refines the generative model with a multi-level probabilistic choice mechanism. *Alternating processes* [HJ90, Han91] are a mixture of strictly alternating probabilistic and nondeterministic states. The *stochastic processes* of [Mol82,

GHR92, Hil93] associate a stochastic delay, represented as a random variable, with the firing of transitions. In *probabilistic specifications* [JL91] transitions are labeled by *sets* of probabilities, rather than single probabilities. There are a number of other probabilistic models worth mentioning, but are omitted due to space limitations.

The main contribution of our work is a compositional semantics for asynchronous probabilistic systems, which is fully abstract with respect to probabilistic testing. To our knowledge, we are the first to give such a result. The closest earlier work is that of Christoff [Chr90], although his approach differs from our own on a number of key aspects: Christoff considers only purely generative processes, whereas our model models both generative and reactive processes. Christoff's tests are deterministic and there is no notion of success state or success action; instead, he introduces testing equivalences based on the probabilities induced by the interaction of a process and a test on L^*, where L is a set of observable events. In our model, tests are just probabilistic I/O automata with a distinguished "success action" ω, and testing equivalence is defined in terms of the probability of a process successfully passing a test. Christoff's denotational models, which he shows to be fully abstract with respect to testing, are defined in terms of "probability functions" that map $(2^L - \emptyset)^* \times L^*$ to $[0,1]$. No composition operator on processes is defined in [Chr90] and thus the issue of compositionality of his denotational semantics is left untreated. On the other hand, our model is compositional, and to obtain this result we find it necessary to include information about the probability of internal delays in the abstract representation of a process.

The rest of this paper is organized as follows: In Section 2, we review some basic definitions and results pertaining to ordinary I/O automata and the composition operation on such automata. In Section 3, we define our probabilistic version of I/O automata, and show how the notion of composition for ordinary I/O automata extends to the probabilistic case. In Section 4, we define our notion of probabilistic behavior, and we show that the map taking each automaton to its behavior respects composition. In Section 5, we show that our notion of behavior is fully abstract with respect to probabilistic testing. Finally, in Section 6, we summarize what we have accomplished and outline plans for future investigation.

2 I/O Automata

In this section, we review some basic definitions and results pertaining to ordinary I/O automata. For further details, the reader is referred to [Tut87].

An *I/O automaton* is a quadruple $A = (Q, q^{\mathrm{I}}, E, \Delta)$, where

- Q is a set of *states*
- $q^{\mathrm{I}} \in Q$ is a distinguished *start state*.
- E is a set of *actions*, partitioned into disjoint sets of *input*, *output*, and *internal* actions, which are denoted by E^{in}, E^{out}, and E^{int}, respectively. The

set $E^{\mathrm{loc}} = E^{\mathrm{out}} \cup E^{\mathrm{int}}$ of output and internal actions is called the set of *locally controlled* actions, and the set $E^{\mathrm{ext}} = E^{\mathrm{in}} \cup E^{\mathrm{out}}$ is called the set of *external* actions.

- $\Delta \subseteq Q \times E \times Q$ is the *transition relation*, which satisfies the following *input-enabled* property: for any state $q \in Q$ and input action $e \in E^{\mathrm{in}}$, there exists a state $r \in Q$ such that $(q, e, r) \in \Delta$.

It will sometimes be convenient for us to use the notation $q \xrightarrow{e} r$ to assert that $(q, e, r) \in \Delta$.

The original definition of I/O automaton [Tut87] included an additional piece of data: a partition of the set of locally controlled actions. Such partitions are used to define a notion of *fair execution* for I/O automata, which is essential if one wishes to establish liveness properties for such automata. We do not treat liveness properties in this paper. Even so, to treat liveness in a probabilistic setting it would seem more natural to bypass fairness altogether, and instead use probability information to define a notion of "satisfies a liveness property with probability one." We shall therefore ignore the partition component of I/O automata in our discussion.

Lynch and Tuttle define a *finite execution fragment* of an I/O automaton A to be an alternating sequence of states and actions of the form

$$q_0 \xrightarrow{e_0} q_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{n-1}} q_n,$$

such that $(q_k, e_k, q_{k+1}) \in \Delta$ for $0 \leq k < n$. In this paper, we find it convenient to use a slightly more liberal definition of execution fragment, to allow such fragments to contain actions not in E. We use the term *native* to refer to an execution or execution fragment of A in which only actions from E appear. We also impose the technical condition that the set E of actions of an I/O automaton be a subset of a fixed, countable *universe of actions* U. This is not really much of a restriction in practice, since in practical situations we have to be able to explicitly denote all actions by a finite sequence of symbols. For us, then, a *finite execution fragment* is an alternating sequence of states and actions as above such that $(q_k, e_k, q_{k+1}) \in \Delta$ whenever $e_k \in E$ and such that $q_{k+1} = q_k$ whenever $e_k \in U \setminus E$. An execution fragment with $q_0 = q^{\mathrm{I}}$ (the distinguished start state) is called an *execution*.

If σ denotes an execution fragment as above, then we will use $\sigma(k)$ to denote the state q_k, for $0 \leq k \leq n$. We use the term *trace* to refer to a sequence of actions. If σ is an execution fragment as above, then the *trace* of σ, denoted $\mathrm{tr}(\sigma)$, is the sequence of actions $e_0 e_1 \ldots e_{n-1}$ appearing in σ.

A collection $\{A_i : i \in I\}$ of I/O automata, where $A_i = (Q_i, q_i^{\mathrm{I}}, E_i, \Delta_i)$, is called *compatible* if for all $i, j \in I$, $E_i^{\mathrm{out}} \cap E_j^{\mathrm{out}} = \emptyset$ and $E_i^{\mathrm{int}} \cap E_j = \emptyset$. We define the *composition* $\prod_{i \in I} A_i$ of such a collection to be the I/O automaton $(Q, q^{\mathrm{I}}, E, \Delta)$, defined as follows:

- $Q = \prod_{i \in I} Q_i$.
- $q^{\mathrm{I}} = \langle q_i^{\mathrm{I}} : i \in I \rangle$.

- $E = \bigcup_{i \in I} E_i$, where $E^{\text{out}} = \bigcup_{i \in I} E_i^{\text{out}}$, $E^{\text{int}} = \bigcup_{i \in I} E_i^{\text{int}}$ and $E^{\text{in}} = (\bigcup_{i \in I} E_i^{\text{in}}) \setminus E^{\text{out}}$.
- Δ is the set of all $(\langle q_i : i \in I \rangle, e, \langle r_i : i \in I \rangle)$ such that for all $i \in I$, if $e \in E_i$, then $(q_i, e, r_i) \in \Delta_i$, otherwise $r_i = q_i$.

With our more liberal definition of execution fragments, we have a simple correspondence between computations of a composite I/O automaton and the computations of its component automata. Suppose σ is an execution fragment for a composite automaton $\prod_{i \in I} A_i$, of the form

$$q_0 \xrightarrow{e_0} q_1 \xrightarrow{e_1} \ldots \xrightarrow{e_{n-1}} q_n,$$

where $q_k = \langle q_{k,i} : i \in I \rangle$. Then for each $i \in I$, the execution fragment σ projects in an obvious way to an execution fragment $\sigma|A_i$ for A_i by replacing each state q_k by its projection $q_{k,i}$. Suppose $\text{tr}(\sigma) = \alpha$, then $tr(\sigma|A_i) = \alpha$ for each $i \in I$. This mapping, taking each execution fragment σ of $\prod_{i \in I} A_i$ to indexed collections of execution fragments $\langle \sigma|A_i : i \in I \rangle$, is invertible, in the sense made precise by the following proposition.

Proposition 1. *Suppose $A = \prod_{i \in I} A_i$. Then for each action sequence α of the form $e_0 e_1 \ldots e_{n-1}$, the map, that takes each execution fragment σ of $\prod_{i \in I} A_i$ with $\text{tr}(\sigma) = \alpha$ to the collection of execution fragments $\{\sigma|A_i : i \in I\}$, is a bijection, from the set of execution fragments ρ of A having trace α to the set of indexed collections $\{\rho_i : i \in I\}$, where each ρ_i is an execution fragment of A_i with trace α.*

3 Probabilistic I/O Automata

A *probabilistic I/O automaton* is a sextuple $A = (Q, q^{\text{I}}, E, \Delta, \mu, \delta)$, where

- $(Q, q^{\text{I}}, E, \Delta)$ is an I/O automaton, called the *underlying* I/O automaton. The transition relation Δ is required to satisfy the following properties:
  1. The *local finite-branching* property:
     for all $q \in Q$, the set $\{(q, e, r) \in \Delta : e \in E^{\text{loc}}\}$ is finite.
  2. The *input image-finiteness* property:
     for all $q \in Q$ and all $e \in E^{\text{in}}$, the set $\{r \in Q : (q, e, r) \in \Delta\}$ is finite.
- $\mu : (Q \times E \times Q) \to [0, 1]$ is the *transition probability* function, which is required to satisfy the following conditions:
  1. $\mu(q, e, r) > 0$ iff $(q, e, r) \in \Delta$.
  2. $\sum_{r \in Q} \mu(q, e, r) = 1$, for all $q \in Q$ and all $e \in E^{\text{in}}$.
  3. For all $q \in Q$, if there exist $e \in E^{\text{loc}}$ and $r \in Q$ such that $(q, e, r) \in \Delta$, then $\sum_{r \in Q} \sum_{e \in E^{\text{loc}}} \mu(q, e, r) = 1$,
- $\delta : Q \to [0, \infty)$ is the *state delay* function, which is required to satisfy the following condition: for all $q \in Q$, we have $\delta(q) > 0$ if and only if there exist $e \in E^{\text{loc}}$ and $r \in Q$ such that $(q, e, r) \in \Delta$.

The local finite-branching condition on the transition relation Δ is imposed so that in Section 3.1 we can obtain a probability distribution on the set of all native executions of an automaton A with an empty set of input actions. This condition is also needed so that we can obtain discrete probability distributions in key situations in Section 4; thereby avoiding technical problems of measurability that would arise in a more general setting. Once we have imposed the local finite-branching condition, the input image-finiteness condition is required in order for the class of probabilistic I/O automata to be closed under the composition operation defined in Section 3.2.

The transition probability function μ describes the probability, for each state q, of choosing one transition from state q as opposed to another. As discussed in the introduction, we do not ascribe any probability to the choice between an input transition and an output or internal transition, since any such probability will be determined by the environment. Similarly, the choice between a transition for one input action and a transition for a different input action is also under the control of the environment, so we do not attempt to assign probabilities in this case either. The stochastic conditions (2) and (3) on μ reflect this point of view: Condition (2) states that for each state q and input action e, the function μ determines a probability distribution on the set of states r such that $q \xrightarrow{e} r$. Condition (3) states that if there is some locally controlled action enabled in state q, then μ determines a probability distribution on the set of all pairs (e, r) such that e is locally controlled and $q \xrightarrow{e} r$.

The state delay function δ assigns to each state q a nonnegative real number $\delta(q)$. As discussed in the introduction, the intuitive interpretation of $\delta(q)$ is as the parameter of an exponential distribution describing the length of a random "delay period" from the time state q is entered by the automaton until the time it executes its next locally controlled action. The condition on δ corresponds to the intuition that if no locally controlled action is available in state q, then the delay period will be infinite.

Function δ can be extended to finite execution fragments as follows. Let σ be a finite execution fragment of the form

$$q_0 \xrightarrow{e_0} q_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{n-1}} q_n,$$

and d_i denote the value $\delta(q_i)$ for $0 \leq i \leq n$. Then let $\delta(\sigma)$ denote the sequence $d_0 d_1 \ldots d_n$. We call $\delta(\sigma)$ the *delay sequence* of σ. We use d, d' to denote delay sequences. Let $d = d_0 d_1 \ldots d_n$ and $d' = d'_0 d'_1 \ldots d'_n$ be two arbitrary delay sequences. Then $d + d'$ is the componentwise sum of d and d'. This notation is used extensively in Sections 4 and 5.

One further convention that will be convenient in forming a probability space from the set of executions of a probabilistic I/O automaton is the following: if $A = (Q, q^{\mathrm{I}}, E, \Delta, \mu, \delta)$, and $e \in U \setminus E$, then we define $\mu(q, e, q) = 1$ and $\mu(q, e, r) = 0$ for all other $r \in Q$.

3.1 Probability Distributions on Executions and Traces

Suppose $A = (Q, q^{\mathrm{I}}, E, \Delta, \mu, \delta)$ is a probabilistic I/O automaton. In this section, we consider the problem of assigning probabilities to sets of executions of A. If A is an arbitrary probabilistic I/O automaton, it does not make much sense to ask about the probability of sets of executions of A, since we lack any sort of probabilistic description of when input actions will occur and which ones they will be. However, in case $E^{\mathrm{in}} = \emptyset$, we do not lack any such information, and the question becomes a meaningful one. We shall likewise restrict our attention to sets of native executions of A (those that contain only actions in E) since actions outside E are under the control of the environment.

In any discussion of probability, it is necessary to begin by describing the probability space. In our case, the set of basic outcomes is the set of all native executions of A (both finite and infinite). If

$$\sigma = q_0 \xrightarrow{e_0} q_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{n-1}} q_n$$

is a finite native execution, then define the set $[\sigma]$ to be the set of all finite and infinite native executions of the form

$$\rho = q_0 \xrightarrow{e_0} q_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{n-1}} q_n \xrightarrow{e_n} q_{n+1} \cdots$$

In other words, for any finite native execution σ, the set $[\sigma]$ is the set of all finite and infinite native executions ρ that extend σ. Define a set of native executions of A to be *basic measurable* if it is the union (possibly empty) of a finite disjoint collection of sets of the form $[\sigma]$ and sets of the form $\{\rho\}$, where ρ is a finite native execution with $[\rho] \neq \{\rho\}$. A basic measurable set of the form $\{\rho\}$ or of the form $[\sigma]$ is called *simple*. For each representation of a basic measurable set, $\bigcup_{i=1}^{m} S_i$, where S_i are disjoint simple measurable sets, we define the *rank* of this representation to be the number of S_i's that are of the form $\{\rho\}$.

Lemma 2. *The collection of basic measurable sets of executions of A is nonempty, and is closed under pairwise union, pairwise intersection, and complement. That is, it forms an algebra of sets.*

We now show how to assign probability to basic measurable sets of executions. Suppose

$$\sigma = q_0 \xrightarrow{e_0} q_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{n-1}} q_n$$

is a finite native execution. To the set $[\sigma]$ we assign probability as follows:

$$\Pr([\sigma]) = \prod_{k=0}^{n-1} \mu(q_k, e_k, q_{k+1}).$$

In particular $\Pr([\epsilon]) = 1$. To the singleton set $\{\rho\}$, where $[\rho] \neq \{\rho\}$ we assign probability 0.

Lemma 3. *Suppose S is a simple measurable set and $S = \bigcup_{i=1}^{m} S_i$, where S_i are disjoint simple measurable sets. Then*

$$\Pr(S) = \sum_{i=1}^{m} \Pr(S_i).$$

To extend the above assignment of probability to all basic measurable sets, we need the following result:

Lemma 4. *Suppose $B = \bigcup_{i=1}^{m} B_i$ and $B = \bigcup_{j=1}^{n} B'_j$ are two representations of B as finite unions of disjoint sets of the form $[\sigma]$ or of the form $\{\rho\}$, where $[\rho] \neq \{\rho\}$. Then*

$$\sum_{i=1}^{m} \Pr(B_i) = \sum_{j=1}^{n} \Pr(B'_j)$$

To each basic measurable set $B = \bigcup_{i=1}^{n} B_i$ represented as a a finite union of disjoint sets B_i of the form $[\sigma]$ or of the form $\{\rho\}$, where $[\rho] \neq \{\rho\}$, we assign probability as follows:

$$\Pr(B) = \sum_{i=1}^{n} \Pr(B_i).$$

The preceding lemma shows that this definition is independent of the particular choice of representation.

Lemma 5. Pr *is a measure (a countably additive set function) on the algebra of basic measurable sets.*

Proposition 6. Pr *extends to a complete measure (which we also denote by* Pr*) on a σ-algebra containing the algebra of basic measurable sets. Moreover, since* $\Pr([\epsilon]) = 1$, *it follows that* Pr *is a probability measure.*

We can also assign probabilities to sets of traces. Still working with respect to a probabilistic I/O automaton A, we define a set V of traces of A to be *measurable* if $\mathrm{tr}_A^{-1}(V)$ is a measurable set of executions of A. To each such set we assign probability as follows:

$$\Pr(V) = \Pr(\mathrm{tr}^{-1}(V)).$$

It is easy to check that these definitions determine a probability space on the set of traces.

3.2 Composition

A collection $\{A_i : i \in I\}$ of probabilistic I/O automata, where

$$A_i = (Q_i, q_i^{\mathrm{I}}, E_i, \Delta_i, \mu_i, \delta_i),$$

is called *compatible* if the corresponding collection of underlying I/O automata is compatible. The *composition* $\prod_{i \in I} A_i$ of a *finite* compatible collection of probabilistic I/O automata is defined to be the sextuple $(Q, q^{\mathrm{I}}, E, \Delta, \mu, \delta)$, where

1. Q, q^{I}, E, and Δ are defined as for composition of ordinary I/O automata.
2. $\delta(\langle q_i : i \in I \rangle) = \sum_{i \in I} \delta_i(q_i)$.
3. If $e \in E^{\mathrm{in}}$, then

$$\mu(\langle q_i : i \in I \rangle, e, \langle r_i : i \in I \rangle) = \prod_{\{i \in I : e \in E_i\}} \mu_i(q_i, e, r_i).$$

If $e \in E_k^{\mathrm{loc}}$ for some k, then

$$\mu(\langle q_i : i \in I \rangle, e, \langle r_i : i \in I \rangle) = \frac{\delta_k(q_k)}{\sum_{i \in I} \delta_i(q_i)} \prod_{\{i \in I : e \in E_i\}} \mu_i(q_i, e, r_i).$$

In this paper, we restrict our attention to the composition of finite collections only. The finiteness assumption ensures that the sum in (2) converges, and that the products appearing in the definition of μ are nonzero.

The definition of composition can be motivated as follows: In any given state $q = \langle q_i : i \in I \rangle$ of a composite automaton $A = \prod_{i \in I} A_i$, the component automata A_i participate in a race to see which one will be the next to execute a locally controlled action. Conceptually, when each component automaton A_i enters its local state q_i, it chooses a random "delay period" from an exponential distribution with parameter $\delta_i(q_i)$. It then delays for this amount of time before executing its next locally controlled action. The winner of the race from state q will be that component automaton A_i with the least amount of time to wait. Because of the "memoryless" property of the exponential distribution, it is not necessary for us to keep track in the composite automaton of how long each component automaton A_i has already delayed in state q_i—the amount of time A_i has left to delay in state q_i is described by the same exponential distribution with parameter $\delta_i(q_i)$, regardless of how long A_i has already delayed. This fact simplifies the definition of composition considerably, and would also be important if we wished to construct a real-world implementation of the probabilistic behavior modeled by these automata.

Assuming that the random delay periods associated with the component automata A_i are independent, the probability that the winner of the race from state q will be a particular component A_k will be the probability that the random delay period chosen by A_k is the minimum among all the delay periods chosen by the A_i. This probability is the ratio $\delta_k(q_k) / \sum_{i \in I} \delta_i(q_i)$. The distribution of the time that composite automaton A delays in state q before executing its next locally controlled action is the distribution of the minimum of the delay times of each of the components. Here the situation is simplified by another property of the exponential distribution: the distribution of the minimum of a finite collection $\langle x_i : i \in I \rangle$ of independent random variables, where x_i is exponentially distributed with parameter $\delta_i(q_i)$, is again exponentially distributed with parameter $\sum_{i \in I} \delta_i(q_i)$ [Tri82]. This explains the definition of δ.

The definition of μ can now be explained as follows: If it has already been determined that the next action to be executed is a particular input action e, then the probability of choosing a particular transition (q, e, r), where $q = \langle q_i : i \in I \rangle$

and $r = \langle r_i : i \in I \rangle$ is simply the joint probability that component A_i executes (q_i, e, r_i), for all $i \in I$ such that $e \in E_i$. Assuming independence, this joint probability is just the product of the individual probabilities $\mu_i(q_i, e, r_i)$. On the other hand, if it has been determined that the next action to be executed is not an input action, but rather a locally controlled action, then which locally controlled action is actually executed depends on the outcome of the race for control between the component automata. The probability that the transition executed will be (q, e, r), where $q = \langle q_i : i \in I \rangle$ and $r = \langle r_i : i \in I \rangle$, and $e \in E_k^{\mathrm{loc}}$ is locally controlled by A_k, is the joint probability that each A_i will execute transition (q_i, e, r_i), times the probability that A_k will win the race. Assuming independence, the former is just the product of the individual probabilities $\mu_i(q_i, e, r_i)$. As already discussed, the latter probability is the ratio $\delta_k(q_k) / \sum_{i \in I} \delta_i(q_i)$.

Proposition 7. *If $\{A_i : i \in I\}$ is a finite compatible collection of probabilistic I/O automata, then $\prod_{i \in I} A_i$ is also a probabilistic I/O automaton.*

4 Behaviors of Probabilistic I/O Automata

In this section and the next section, we consider the restricted class of probabilistic I/O automata $A = (Q, q^{\mathrm{I}}, E, \Delta, \mu, \delta)$ for which the set E^{int} of internal actions is empty. We call probabilistic I/O automata satisfying this condition *restricted* probabilistic I/O automata. We wish to associate with such an automaton a more abstract representation in which we ignore the details of the particular state set and transition relation of the automaton, and focus instead on externally observable aspects of its probabilistic behavior.

Suppose A is a restricted probabilistic I/O automaton. Given a trace $\alpha = e_0 e_1 \ldots e_{n-1}$, for each delay sequence $d = d_0 d_1 \ldots d_n$ define the quantity $p_\alpha^A(d)$ by:

$$p_\alpha^A(d) = \sum_\sigma \prod_{k=0}^{n-1} \mu_A(\sigma(k), e_k, \sigma(k+1)),$$

where the summation is taken over all executions σ of A having trace α and delay sequence d. Observe that convergence of the summation is automatic, since by the local finite-branching and input image-finiteness properties of A, the set $\{\sigma : tr(\sigma) = \alpha\}$ is finite. The same reasoning also shows that, for a fixed α, the set of all d for which $p_\alpha^A(d)$ is nonzero, is finite. In case the set of input actions of A is empty, and α contains only actions in E_A, the quantity $p_\alpha^A(d)$ is the probability of the set of all native executions of A having α as a prefix of their trace and d as a prefix of their delay sequence.

Now, if $g : \mathcal{R}^{n+1} \to \mathcal{R}$ is a real-valued function, define

$$\mathcal{E}_\alpha^A[g(D)] = \sum_d g(d) p_\alpha^A(d),$$

where the sum ranges over all $(n+1)$-tuples $d = (d_0, d_1, \ldots, d_n)$ of nonnegative real numbers. We may view $\mathcal{E}_\alpha^A$ as a functional

$$\mathcal{E}_\alpha^A : (\mathcal{R}^{n+1} \to \mathcal{R}) \to \mathcal{R}.$$

In case the set of input actions of A is nonempty, we may regard the sequences d as the values of an $(n+1)$-dimensional random variable $D = (D_0, D_1, \ldots, D_n)$ defined on the conditional probability space X_α of native executions of A whose traces extend α. In this case, the quantity $\mathcal{E}_\alpha^A[g(D)]$ is just the expectation of $g(D)$, times the probability p_α^A of the set X_α.

Our abstract representation for probabilistic I/O automata assigns, to each restricted probabilistic I/O automaton A, the mapping $\mathcal{E}^A$ that takes each trace $\alpha \in U^*$ of length n to the functional $\mathcal{E}_\alpha^A$ on $\mathcal{R}^{n+1} \to \mathcal{R}$. We call the mapping $\mathcal{E}^A$ the *probabilistic behavior map* associated with A.

The compositionality of the representation of automata by probabilistic behavior maps is established in the following result:

Theorem 8. *Suppose A and B are compatible probabilistic I/O automata and $\alpha = e_0 e_1 \ldots e_{n-1}$. Then*

$$\mathcal{E}_\alpha^{A|B}[g(D)] = \mathcal{E}_\alpha^B[\mathcal{E}_\alpha^A[g(D^A + D^B) \cdot h(D^A, D^B)]\,],$$

where

$$h(D^A, D^B) = \left(\prod_{k \in K_A^{\mathrm{loc}}} \frac{D_k^A}{D_k^A + D_k^B} \right) \left(\prod_{k \in K_B^{\mathrm{loc}}} \frac{D_k^B}{D_k^A + D_k^B} \right)$$

$K_A^{\mathrm{loc}} = \{k : 0 \le k < n, \;\; e_k \in E_A^{\mathrm{loc}}\}$ and $K_B^{\mathrm{loc}} = \{k : 0 \le k < n, \;\; e_k \in E_B^{\mathrm{loc}}\}$

Proof Sketch – Write out the summation that defines $\mathcal{E}_\alpha^{A|B}$, then use the definition of $\mu_{A|B}$ to rewrite this expression in terms of μ_A and μ_B. By Proposition 1 and the definition of composition for probabilistic I/O automata, the executions σ of $A|B$ with trace α and delay sequence d are in bijective correspondence with pairs (σ_A, σ_B), where σ_A is an execution of A having α as its trace and d_A as its delay sequence, σ_B is an execution of B having α as its trace and d_B as its delay sequence, and $d = d_A + d_B$. Finally, apply the definitions of $\mathcal{E}_\alpha^A$ and $\mathcal{E}_\alpha^B$. □

5 Testing Equivalence and Full Abstraction

In this section we show that probabilistic behavior maps are fully abstract with respect to a notion of probabilistic testing equivalence. That is to say, probabilistic I/O automata A and B determine the same probabilistic behavior map if and only if in a certain sense they cannot be distinguished by any probabilistic test.

Formally, a *test* is simply a probabilistic I/O automaton T that has a distinguished output action ω. We interpret the occurrence of ω in a computation of T as an indication that the test has succeeded. A test is called *closed* if its set of input actions is empty. For closed tests T, it makes sense (see Section 3.1) to talk about the probability of sets of native executions of T. In Lemma 9 we show that the set of all successful native executions of a closed test T is measurable, and we call the probability of this set the *success probability* of T.

Lemma 9. *Suppose T is a closed test. Then the set of all successful native executions of T is measurable. The probability of this set is given by the formula*

$$\sum_{\alpha \in \bar{\Omega}} \mathcal{E}^T_{\alpha\omega}[1],$$

where $\bar{\Omega}$ is the set of all traces that do not contain ω.

Suppose $A = (Q_A, q^{\mathrm{I}}_A, E_A, \Delta_A, \mu_A, \delta_A)$ is a probabilistic I/O automaton. A *proper test* for A is a test $T = (Q_T, q^{\mathrm{I}}_T, E_T, \Delta_T, \mu_T, \delta_T)$ such that $E^{\mathrm{in}}_T \subseteq E^{\mathrm{out}}_A$, $E^{\mathrm{in}}_A \subseteq E^{\mathrm{out}}_T \setminus \{\omega\}$, and $E^{\mathrm{loc}}_A \cap E^{\mathrm{loc}}_T = \emptyset$. If T is a proper test for A, then the collection $\{A, T\}$ is compatible. Let $A|T$ denote its composition, then $A|T$ is a closed test.

If A and B are probabilistic I/O automata with the same set of actions, then we call A and B *testing equivalent* if for all proper tests T for A and B, the success probability of $A|T$ equals the success probability of $B|T$.

We now define a particular class of tests that will be useful for distinguishing probabilistic I/O automata. Let a set of actions $E = E_0 \cup E_1$ be fixed. For each trace $\alpha = e_0 e_1 \ldots e_{n-1}$ with $e_k \in E$ for $0 \leq k < n$, and for each sequence $x = x_0, x_1, \ldots, x_n$ of positive real numbers, we define a test $T_{\alpha,x} = (Q, q^{\mathrm{I}}, E_T, \Delta, \mu, \delta)$ as follows:

- $Q = \{0, 1, 2, \ldots, n, n+1\}$.
- $q^{\mathrm{I}} = 0$.
- $E_T = \{e_0, e_1, \ldots, e_{n-1}\} \cup \{\omega, *\}$, with $E^{\mathrm{in}}_T = E_1$ and $E^{\mathrm{out}}_T = E_0 \cup \{\omega, *\}$.
- Δ is the union of the following sets:
  1. $\{(k, e_k, k+1) : 0 \leq k < n\}$
  2. $\{(n, \omega, n+1)\}$
  3. $\{(k, *, n+1) : 0 \leq k < n\}$
  4. $\{(k, e, n+1) : 0 \leq k < n, e \in E^{\mathrm{in}}_T, e \neq e_k\}$
  5. $\{(n, e, n+1) : e \in E^{\mathrm{in}}_T\}$.
  6. $\{(n+1, e, n+1) : e \in E^{\mathrm{in}}_T\}$.
- μ is defined as follows:
  1. If $0 \leq k < n$, then $\mu(k, e_k, k+1) = \begin{cases} 1, & \text{if } e_k \in E^{\mathrm{in}}_T \\ 1/2 & \text{otherwise.} \end{cases}$
  2. $\mu(n, \omega, n+1) = 1$.
  3. If $0 \leq k < n$, then $\mu(k, *, n+1) = \begin{cases} 1, & \text{if } e_k \in E^{\mathrm{in}}_T \\ 1/2 & \text{otherwise.} \end{cases}$
  4. If $0 \leq k < n$, $e \in E^{\mathrm{in}}_T$, and $e \neq e_k$, then $\mu(k, e, n+1) = 1$.
  5. If $e \in E^{\mathrm{in}}_T$, then $\mu(n, e, n+1) = 1$, and $\mu(n+1, e, n+1) = 1$.
- $\delta(k) = x_k$ for $0 \leq k \leq n$ and $\delta(n+1) = 0$.

Intuitively, the test $T_{\alpha,x}$ succeeds when it manages to produce the trace $\alpha\omega$ by passing successively through states $0, 1, 2, \ldots, n$ and finally to $n+1$. For $0 \leq k \leq n$, the state k has delay parameter x_k, so the delay sequence $\delta(\sigma)$ assocated with a successful execution σ of $T_{\alpha,x}$ is the sequence $x_0 x_1 \ldots x_n 0$. This is the only way executions of $T_{\alpha,x}$ can succeed; executions that deviate from α

in the initial section cause $T_{\alpha,x}$ to enter the state $n+1$ without performing the success action ω. In each state k, where $0 \leq k < n$, the test $T_{\alpha,x}$ has a nonzero chance of failing by performing the action $*$ and going directly to state $n+1$. This gives $T_{\alpha,x}$ a certain sensitivity to the delays of its environment.

Lemma 10. *Suppose A is a probabilistic I/O automaton. Then for each trace $\alpha = e_0 e_1 \ldots e_{n-1} \in E_A^*$, and for each sequence $x = x_0, x_1, \ldots, x_n$ of positive real numbers, the test $T_{\alpha,x}$ (with $E_0 = E_A^{\text{in}}$ and $E_1 = E_A^{\text{out}}$) is a proper test for A. Moreover, the success probability of $A|T_{\alpha,x}$ is given by:*

$$2^{-c} \cdot \mathcal{E}^A_{\alpha\omega}[\prod_{k=0}^{n} \frac{y_k}{x_k + D_k^A}],$$

where for all $0 \leq k < n$ we have

$$y_k = \begin{cases} D_k^A, & \text{if } e_k \in E_A^{\text{out}} \\ x_k & \text{otherwise,} \end{cases}$$

$y_n = x_n$, and c is the number of $k \in \{0, 1, \ldots, n-1\}$ for which $e_k \in E_A^{\text{in}}$.

The following lemma is a uniqueness theorem for partial fraction expansions of rational functions in several variables. It is a key component of the proof of full abstraction (Theorem 12).

Lemma 11. *Suppose f and f' are two rational functions of variables $x_0, x_1, \ldots, x_{n-1}$ ($n \geq 0$), defined as follows:*

$$f = \sum_{i \in I} \frac{c_i}{\prod_{k=0}^{n-1}(x_k + d_{i,k})} \qquad\qquad f' = \sum_{i' \in I'} \frac{c'_{i'}}{\prod_{k=0}^{n-1}(x_k + d'_{i',k})},$$

where I and I' are finite sets, $c_i \in (0, \infty)$ for $i \in I$, $c'_i \in (0, \infty)$ for $i \in I'$, and for each distinct $i, j \in I$ the sets $\{(k, d_{i,k}) : 0 \leq k < n\}$ and $\{(k, d_{j,k}) : 0 \leq k < n\}$ are distinct. If $f = f'$, then there exists a bijection $(\ \text{-}\ )' : I \to I'$ such that $c_i = c'_{i'}$ and $d_{i,k} = d'_{i',k}$ for all $i \in I$ and $0 \leq k < n$.

Theorem 12. *Suppose A and B are restricted probabilistic I/O automata with the same set of actions. Then A and B are testing equivalent if and only if the associated probabilistic behavior maps $\mathcal{E}^A$ and $\mathcal{E}^B$ are equal.*

Proof Sketch – If $\mathcal{E}^A = \mathcal{E}^B$, then the fact that A and B are testing equivalent follows directly from Theorem 8 and Lemma 9.

Conversely, suppose A and B are testing equivalent. For each trace $\alpha \in U^*$ of length n that contains no occurrences of the special action ω, in Lemma 10 we show that the tests $T_{\alpha,x}$ are proper tests for A and B and the success probability of $A|T_{\alpha,x}$ is given by an expression $\mathcal{E}^A_{\alpha\omega}[g(D, x)]$, where g is a certain rational function of the $n+1$ random variables D and the $n+1$ unknowns x. The expression $\mathcal{E}^A_{\alpha\omega}[g(D, x)]$ is thus a rational function of x alone. The success probability of $B|T_{\alpha,x}$ is given by a similar rational function. Since A and B are

testing equivalent, these two rational functions must have the same values for all positive x, hence they are the same rational function. Using Lemma 11, we show that the quantities $p^A_{\alpha\omega}(d)$ and $p^B_{\alpha\omega}(d)$, used in the definitions of $\mathcal{E}^A_\alpha$ and $\mathcal{E}^B_\alpha$, can be recovered uniquely from these rational functions, thus showing that $\mathcal{E}^A_\alpha = \mathcal{E}^B_\alpha$ for all traces α that do not contain ω. A final step in the proof removes the restriction on α, thus showing that $\mathcal{E}^A = \mathcal{E}^B$. □

6 Summary and Conclusion

In this paper, we have presented a framework in which probability can be added to I/O automata. To capture the asymmetric treatment of input and output indigenous to I/O automata, a separate distribution is associated with each input action, in the reactive style, and a single distribution is associated with all locally controlled actions, in the generative style. No relative probabilities are defined among different input actions nor between input and locally controlled actions. Moreover, the pleasant notion of I/O automaton asynchronous composition is retained, in part, through the introduction of state delay parameters. Delay parameters admit a natural probabilistic description of the outcome of the competition between automata vying for control of the next action.

As is the practice with ordinary I/O automata, we introduced a more abstract representation of the external behaviors of probabilistic I/O automata (without internal actions), *probabilistic behavior maps*. This representation maps finite action sequences to a set of expectation functionals which give information not only about the probabilities of action sequences but also delay sequences. This latter information is essential for achieving compositionality. We also showed that probabilistic behavior maps are fully abstract with respect to a natural notion of probabilistic testing.

As future work, we plan to generalize probabilistic behavior maps to probabilistic I/O automata with internal actions. Additionally, we would like to extend the entire setup to handle time as well as probability. The presence of the state delay function in our model provides a convenient mechanism on which to base this work.

References

[Chr90] I. Christoff. Testing equivalences and fully abstract models for probabilistic processes. In J. C. M. Baeten and J. W. Klop, editors, *Proceedings of CONCUR '90 – First International Conference on Concurrency Theory*, Lecture Notes in Computer Science, Volume 458, pages 126–140. Springer-Verlag, 1990.

[GHR92] N. Götz, U. Herzog, and M. Rettlebach. TIPP — a language for timed processes and performance evaluation. Technical Report 4/92, University of Erlangen-Nürnberg, Germany, November 1992.

[Han91] H. A. Hansson. *Time and Probability in Formal Design of Distributed Systems*. PhD thesis, Department of Computer Systems, Uppsala University, 1991.

[Hil93] J. Hillston. PEPA: Performance enhanced process algebra. Technical Report CSR-24-93, Department of Computer Science, University of Edinburgh, Edinburgh, Great Britain, March 1993.

[HJ90] H. A. Hansson and B. Jonsson. A calculus for communicating systems with time and probabilities. In *Proceedings of the 11th IEEE Symposium on Real-Time Systems*, 1990.

[JL91] B. Jonsson and K. G. Larsen. Specification and refinement of probabilistic processes. In *Proceedings of the 6th IEEE Symposium on Logic in Computer Science*, Amsterdam, July 1991.

[LS92] K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, September 1992. Preliminary versions of this paper appeared as University of Aalborg technical reports R 88-18 and R 88-29, and in *Proceedings of the 16th Annual ACM Symposium on Principles of Programming Languages*, Austin, Texas, 1989.

[LT87] N. A. Lynch and M. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, 1987.

[Mol82] M. K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Trans. Comput.*, C-31(9), September 1982.

[Rab63] M. O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.

[Tri82] K. S. Trivedi. *Probability & Statistics with Reliability, Queuing, and Computer Science Applications.* Prentice Hall, Englewood Cliffs, New Jersey, 1982.

[Tut87] M. Tuttle. Hierarchical correctness proofs for distributed algorithms. Master's thesis, MIT, April 1987.

[vGSST90] R. J. van Glabbeek, S. A. Smolka, B. Steffen, and C. M. N. Tofts. Reactive, generative, and stratified models of probabilistic processes. In *Proceedings of the 5th IEEE Symposium on Logic in Computer Science*, pages 130–141, Philadelphia, PA, 1990.

This article was processed using the LaTeX macro package with LLNCS style