A Compositional Trace-Based Semantics for Probabilistic Automata *

Roberto Segala

MIT Laboratory for Computer Science
Cambridge, MA 02139

Abstract. We extend the trace semantics for labeled transition systems to a randomized model of concurrent computation. The main objective is to obtain a compositional semantics. The role of a trace in the randomized model is played by a probability distribution over traces, called a *trace distribution*. We show that the preorder based on trace distribution inclusion is not a precongruence, and we build an elementary context, called the *principal context*, that is sufficiently powerful to characterize the coarsest precongruence that is contained in the trace distribution preorder. Finally, we introduce a notion of a *probabilistic forward simulation* and we prove that it is sound for the trace distribution precongruence. An important characteristic of probabilistic forward simulations is that they relate states to probability distributions over states.

1 Introduction

The growing interest in randomized algorithms for the solutions of problems in distributed computation [2, 3] has created a need for formal models where randomized distributed systems can be analyzed. The formal models should be able to deal at least with randomization, which is used to describe the choices of a system that are due to some random draws, and nondeterminism, which is the basic mathematical tool to express the unpredictable behavior of an external environment and the unpredictable relative speeds of two or more systems. Several formal models for randomized concurrent systems were studied in the past [5, 7, 8, 10, 13, 19–23], and among those, the models of [8, 19, 20, 22] distinguish between probability and nondeterminism.

Our long term goal is to build a model that extends Labeled Transition Systems [16], that has a strong mathematical foundation, and that can be used for the actual verification of systems. The choice of labeled transition systems is due to their successful application to model concurrency. In [20] we have extended the labeled transition systems model to account for randomization and we have extended the classical simulation and bisimulation relations to it; in [14, 17] we have shown how the model of [20] can be used for the actual analysis of randomized distributed systems. The main objects of [20] are called *probabilistic automata*, and they differ from ordinary labeled transition systems in that a transition leads to a probability distribution over states rather than to a single state. Probabilistic automata are an extension of the probabilistic automata of Rabin [18] where the occurrence of an action can lead to several probability distributions over states. Choosing a transition represents the nondeterministic behavior of a probabilistic automaton; choosing a state to reach within a transition represents the probabilistic behavior of a probabilistic automaton.

* Supported by NSF grant CCR-92-25124, by DARPA contract N00014-92-J-4033, and by AFOSR-ONR contract F49620-94-1-0199.

In this paper we show how it is possible to define a compositional semantics for probabilistic automata that relies on some form of traces rather than simulation relations; then, we show how the simulation method of [15] extends to the new framework. The problem is not simple since trace-based semantics are known to be linear, i.e., to be independent of the branching structure of a system, while in the probabilistic framework the branching structure of a system can be used to create dependencies between events in the context of a parallel composition. In other words, a trace semantics that is sufficiently expressive to capture part of the probabilistic behavior of a system must be sensitive to part of the branching structure of a system in order to be compositional. The problem, then, is to see how sensitive such a relation should be.

We define a simple and natural trace-based semantics for probabilistic automata, where the main objects to be observed are probability distributions over traces, called *trace distributions*. Then, we define a preorder on probabilistic automata, called *trace distribution preorder*, which is based on trace distribution inclusion. Our trace distributions are a conservative extension of the traces of ordinary labeled transition systems (called automata), and the trace distribution preorder is a conservative extension of the trace inclusion preorder of ordinary automata. We observe that the trace distribution preorder is not a precongruence, and thus, following standard arguments, we define the trace distribution precongruence as the coarsest precongruence that is contained in the trace distribution preorder.

Our first main theorem is that the trace distribution precongruence can be characterized in an alternative and more intuitive way. Namely, we show that there is a context, which we call the *principal context*, that is sufficient to distinguish every pair of probabilistic automata that are not in the trace distribution precongruence relation. As a consequence, the trace distribution precongruence can be characterized as inclusion of principal trace distributions, where a principal trace distribution of a probabilistic automaton is a trace distribution of the probabilistic automaton in parallel with the principal context.

We extend the simulation method of [15] by studying a new relation for probabilistic automata in the style of the forward simulations of [15]. The new relation, which is called *probabilistic forward simulation*, is coarser than the relations of [20] and relates states to probability distributions over states. Probabilistic forward simulations allow us to simulate a probabilistic transition like rolling a dice with eight faces by flipping three fair coins one after the other; this is not possible with the simulation relations of [20]. Our second main theorem is that probabilistic forward simulations are sound for the trace distribution precongruence.

We believe that our methodology can be applied to the study of other semantics based on abstract observations. In particular, in further work we plan to extend the failure semantics of [4] to the probabilistic framework, and possibly to study a related theory of testing.

The rest of the paper is organized as follows. Section 2 gives some background on measure theory; Section 3 introduces the probabilistic automata of [19, 20]; Section 4 introduces the trace distributions and the trace distribution precongruence; Section 5 introduces the principal context and the alternative characterization of the trace distribution precongruence; Section 6 introduces our new simulation relations and shows their soundness for the trace distribution precongruence; Section 7 gives some concluding remarks.

2 Preliminaries

A *probability space* is a triple $(\Omega, \mathcal{F}, P)$ where Ω is a set, $\mathcal{F}$ is a collection of subsets of Ω that is closed under complement and countable union and such that $\Omega \in \mathcal{F}$, and P is a function from $\mathcal{F}$ to $[0, 1]$ such that $P[\Omega] = 1$ and such that for any collection $\{C_i\}_i$ of at most countably many pairwise disjoint elements of $\mathcal{F}$, $P[\cup_i C_i] = \sum_i P[C_i]$. The set Ω is called the *sample space*, $\mathcal{F}$ is called the *σ-field*, and P is called the *probability measure*.

A probability space $(\Omega, \mathcal{F}, P)$ is *discrete* if $\mathcal{F} = 2^{\Omega}$ and for each $C \subseteq \Omega$, $P[C] = \sum_{x \in C} P[\{x\}]$. Given a set X, denote by $Probs(X)$ the set of discrete probability distributions whose sample space is a subset of X and such that the probability of each element is not 0.

The Dirac distribution over an element x, denoted by $\mathcal{D}(x)$, is the probability space with a unique element x. The uniform distribution over a collection of elements $\{x_1, \ldots, x_n\}$, denoted by $\mathcal{U}(x_1, \ldots, x_n)$, is the probability space that assign probability $1/n$ to each x_i.

Throughout the paper we denote a probability space $(\Omega, \mathcal{F}, P)$ by $\mathcal{P}$. As a notational convention, if $\mathcal{P}$ is decorated with indices and primes, then the same indices and primes carry to its elements. Thus, $\mathcal{P}'_i$ denotes $(\Omega'_i, \mathcal{F}'_i, P'_i)$.

The *product* $\mathcal{P}_1 \otimes \mathcal{P}_2$ of two discrete probability spaces $\mathcal{P}_1$, $\mathcal{P}_2$ is the discrete probability space $(\Omega_1 \times \Omega_2, 2^{\Omega_1 \times \Omega_2}, P)$, where $P[(x_1, x_2)] = P_1[x_1]P_2[x_2]$ for each $(x_1, x_2) \in \Omega_1 \times \Omega_2$.

A function $f : \Omega \to \Omega'$ is said to be a *measurable* function from $(\Omega, \mathcal{F})$ to $(\Omega', \mathcal{F}')$ if for each set C of $\mathcal{F}'$ the inverse image of C, denoted by $f^{-1}(C)$, is an element of $\mathcal{F}$. Let P be a probability measure on $(\Omega, \mathcal{F})$, and let P' be defined on $\mathcal{F}'$ as follows: for each element C of $\mathcal{F}'$, $P'(C) = P(f^{-1}(C))$. Then P' is a probability measure on $(\Omega', \mathcal{F}')$. The measure P' is called the measure *induced* by f, and is denoted by $f(P)$. If $\mathcal{P}$ is a discrete probability space and f is a function defined on Ω, then f can be *extended* to $\mathcal{P}$ by defining $f(\mathcal{P})$ to be the discrete probability space $(f(\Omega), 2^{f(\Omega)}, f(P))$.

3 Probabilistic Automata

In this section we introduce the probabilistic automata of [19], which appear also in [20] with a slightly different terminology. We start with an informal overview of the model.

A labeled transition system, also called an *automaton*, is a state machine with labeled transitions. Each transition leaves from a state and leads to the occurrence of a label, also called an *action*, and to a state. A probabilistic automaton is like an ordinary automaton except that each transition leads to an action and to a probability distribution over states.

Resolving the nondeterminism in an automaton leads to a linear chain of states interleaved with actions, called an *execution* or a *computation*; resolving the nondeterminism in a probabilistic automaton leads to a Markov chain structure since each transition leads probabilistically to more than one state. Such a structure is called a *probabilistic execution*. A probabilistic execution can be visualized as a probabilistic automaton that enables at most one transition from each state (a *fully probabilistic automaton*). Due to the complex structure of a probabilistic execution, it is convenient to view it as a special case of a probabilistic automaton; in this way the analysis of a probabilistic execution is simplified.

However, nondeterminism could be resolved also using randomization: a scheduler for n processes running in parallel could choose the next process to schedule by rolling an n-side dice; similarly, if some actions model the input of an external environment, the environment could provide the input at random or could provide no input with some non-zero probability. Thus, in a probabilistic execution the transition that leaves from a state may lead to a probability distribution over both actions and states and also over deadlock (no input). This new kind of transition is not part of our informal definition of a probabilistic automaton; yet, it is still convenient to view a probabilistic execution as a probabilistic automaton.

Thus, our definition of a probabilistic automaton allows for a transition to lead to probability distributions over actions and states and over a symbol δ that models deadlock; however, except for the handling of probabilistic executions, we concentrate on *simple probabilistic automata*, which allow only probabilistic choices over states within a transition.

3.1 Probabilistic Automata

Definition 1. A *probabilistic automaton* M consists of four components:

1. a set $states(M)$ of states,
2. a nonempty set $start(M) \subseteq states(M)$ of start states,
3. an action signature $sig(M) = (ext(M), int(M))$ where $ext(M)$ and $int(M)$ are disjoint sets of *external* and *internal* actions, respectively,
4. a transition relation $trans(M) \subseteq states(M) \times Probs((acts(M) \times states(M)) \cup \{\delta\})$, where $acts(M)$ denotes the set $ext(M) \cup int(M)$ of actions.

A probabilistic automaton M is *simple* if for each transition $(s, \mathcal{P})$ of $trans(M)$ there is an action a such that $\Omega \subseteq \{a\} \times states(M)$. In such a case a transition can be represented alternatively as $(s, a, \mathcal{P}')$, where $\mathcal{P}' \in Probs(states(M))$, and is called a *simple transition*.

A probabilistic automaton is *fully probabilistic* if it has a unique start state and from each state there is at most one transition enabled. □

An ordinary automaton is a special case of a probabilistic automaton where each transition leads to a Dirac distribution; the generative model of probabilistic processes of [7] is a special case of a fully probabilistic automaton; simple probabilistic automata are partially captured by the reactive model of [7] in the sense that the reactive model assumes some form of nondeterminism between different actions. However, the reactive model does not allow nondeterministic choices between transitions involving the same action. By restricting simple probabilistic automata to have finitely many states, we obtain objects with a structure similar to that of the Concurrent Labeled Markov Chains of [8]; however, in our model we do not need to distinguish between nondeterministic and probabilistic states. In our model nondeterminism is obtained by means of the structure of the transition relation. This allows us to retain most of the traditional notation that is used for automata.

3.2 Executions and Probabilistic Executions

We now move to the notion of an execution, which is the result of resolving both the nondeterministic and the probabilistic choices in a probabilistic automaton; it corresponds to the notion of an execution for ordinary automata. We introduce also a notion of an extended execution, which we use later to study the probabilistic behavior of a probabilistic automaton.

Definition 2. An *execution fragment* α of a probabilistic automaton M is a (finite or infinite) sequence of alternating states and actions starting with a state and, if the execution fragment is finite, ending in a state, $\alpha = s_0 a_1 s_1 a_2 s_2 \cdots$, where for each i there exists a probability space $\mathcal{P}$ such that $(s_i, \mathcal{P}) \in trans(M)$ and $(a_{i+1}, s_{i+1}) \in \Omega$. Denote by $fstate(\alpha)$ the first state of α, and, if α is finite, denote by $lstate(\alpha)$ the last state of α. Denote by $frag^*(M)$ and $frag(M)$ the sets of finite and all execution fragments of M, respectively. An *execution* is an execution fragment whose first state is a start state. Denote by $exec^*(M)$ and $exec(M)$ the sets of finite and all executions of M, respectively.

An extended execution (fragment) of M is either an execution (fragment) of M, or a sequence $\alpha = s_0 a_1 s_1 \cdots a_n s_n \delta$ such that $s_0 a_1 s_1 \cdots a_n s_n$ is an execution (fragment) of M.

A finite execution fragment $\alpha_1 = s_0 a_1 s_1 \cdots a_n s_n$ of M and an extended execution fragment $\alpha_2 = s_n a_{n+1} s_{n+1} \cdots$ of M can be *concatenated*. In this case the concatenation, written $\alpha_1 \frown \alpha_2$, is the extended execution fragment $s_0 a_1 s_1 \cdots a_n s_n a_{n+1} s_{n+1} \cdots$. An extended execution fragment α_1 of M is a *prefix* of an extended execution fragment α_2 of M, written $\alpha_1 \leq \alpha_2$, if either $\alpha_1 = \alpha_2$ or α_1 is finite and there exists an extended execution fragment α_1' of M such that $\alpha_2 = \alpha_1 \frown \alpha_1'$. □

As we said already, an execution is the result of resolving both the nondeterministic and the probabilistic choices in a probabilistic automaton. The result of the resolution of nondeterministic choices only is a fully probabilistic automaton, called a *probabilistic execution*, which is the entity that replaces the executions of ordinary automata. Informally, since in ordinary automata there is no probability left once the nondeterminism is resolved, the executions and probabilistic executions of an ordinary automaton describe the same objects. Before giving the formal definition of a probabilistic execution, we introduce *combined transitions*, which allow us to express the ability to resolve the nondeterminism using probability. Informally, a combined transition leaving from a state s is obtained by choosing a transition that leaves from s probabilistically, and then behaving according to the transition chosen. Among the choices it is possible not to schedule any transition. This possibility is expressed by the term $(1 - \sum_i p_i)$ in the probability of δ in the definition below.

Definition 3. Given a probabilistic automaton M, a finite or countable set $\{\mathcal{P}_i\}_i$ of probability distributions of $Probs((acts(M) \times states(M)) \cup \{\delta\})$, and a weight $p_i > 0$ for each i such that $\sum_i p_i \leq 1$, the combination $\sum_i p_i \mathcal{P}_i$ of the distributions $\{\mathcal{P}_i\}_i$ is the probability space $\mathcal{P}$ such that

- $\Omega = \begin{cases} \cup_i \Omega_i & \text{if } \sum_i p_i = 1 \\ \cup_i \Omega_i \cup \{\delta\} & \text{if } \sum_i p_i < 1 \end{cases}$
- $\mathcal{F} = 2^{\Omega}$
- for each $(a, s) \in \Omega$, $P[(a, s)] = \sum_{i \mid (a,s) \in \Omega_i} p_i P_i[(a, s)]$
- if $\delta \in \Omega$, then $P[\delta] = (1 - \sum_i p_i) + \sum_{i \mid \delta \in \Omega_i} p_i P_i[\delta]$.

A pair $(s, \mathcal{P})$ is a *combined transition* of M if there exists a finite or countable family of transitions $\{(s, \mathcal{P}_i)\}_i$ and a set of positive weights $\{p_i\}_i$ with $\sum_i p_i \leq 1$, such that $\mathcal{P} = \sum_i p_i \mathcal{P}_i$. Denote $(s, \mathcal{P})$ by $\sum_i p_i (s, \mathcal{P}_i)$. □

We are now ready to define a probabilistic execution. A technical detail is that in order to name the states of a probabilistic execution, those states are represented by finite execution fragments of a probabilistic automaton. A probabilistic execution can be seen as the result of unfolding the transition relation of a probabilistic automaton and then choosing probabilistically a transition from each state.

Definition 4. Let α be a finite execution fragment of a probabilistic automaton M. Define a function $\alpha\frown$ that applied to a pair (a, s) returns $(a, \alpha a s)$, and applied to δ returns δ. Recall from the last paragraph of Section 2 that the function $\alpha\frown$ can be extended to discrete probability spaces.

A *probabilistic execution fragment* of a probabilistic automaton M, is a fully probabilistic automaton, denoted by H, such that

1. $states(H) \subseteq frag^*(M)$. Let q range over states of probabilistic executions.
2. for each transition $tr = (q, \mathcal{P})$ of H there is a combined transition $tr' = (lstate(q), \mathcal{P}')$ of M, called the *corresponding combined transition*, such that $\mathcal{P} = q \frown \mathcal{P}'$.
3. each state of H is reachable and enables one transition, where a state q of H is reachable if there is an execution of H whose last state is q.

A *probabilistic execution* is a probabilistic execution fragment whose start state is a start state of M. Denote by $prfrag(M)$ the set of probabilistic execution fragments of M, and by $prexec(M)$ the set of probabilistic executions of M. Also, denote by q_0^H the start state of a generic probabilistic execution fragment H, and for each transition $(q, \mathcal{P})$ of H, denote the pair $(q, \mathcal{P})$ by tr_q^H, and denote $\mathcal{P}$ by $\mathcal{P}_q^H$. □

Example 1. Two examples of probabilistic executions appear in Figure 1. In particular, the probabilistic execution denoted by H is a probabilistic execution of the probabilistic automaton denoted by M. For notational convenience, in the representation of a probabilistic execution H we do not write explicitly the full names of the states of H since the full names are derivable from the position of each state in the diagram; moreover, whenever a state q enables the transition $(q, \mathcal{D}(\delta))$ we do not draw any arc leaving from the state of the diagram that represents q. □

There is a strong correspondence between the extended execution fragments of a probabilistic automaton and the extended executions of one of its probabilistic execution fragments. We express this correspondence by means of an operator $\alpha\!\downarrow$ that takes an extended execution of H and gives back the corresponding extended execution fragment of M, and an operator $\alpha \!\uparrow\! q_0^H$ that takes an extended execution fragment of M and gives back the corresponding extended execution of H if it exists.

3.3 Events

We now define a probability space $(\Omega_H, \mathcal{F}_H, P_H)$ for a probabilistic execution fragment H, so that it is possible to analyze the probabilistic behavior of a probabilistic automaton once the nondeterminism is resolved. The sample space Ω_H is the set of extended executions of M that represent complete extended execution fragments of H, where an extended execution α of H is complete iff it is either infinite or $\alpha = \alpha'\delta$ and $\delta \in \Omega^H_{lstate(\alpha')}$. For each finite extended execution fragment α of M, let C_α, the *cone* with prefix α, be the set $\{\alpha' \in \Omega_H \mid \alpha \leq \alpha'\}$, and let $\mathcal{C}_H$ be the class of cones for H. The probability $\mu_H(C_\alpha)$ of the cone C_α is the product of the probabilities associated with each edge that generates α in H. Formally, if $\alpha = q_0^H a_1 s_1 \cdots s_{n-1} a_n s_n$, then $\mu_H(C_\alpha) \triangleq P^H_{q_0}[(a_1, q_1)] \cdots P^H_{q_{n-1}}[(a_n, q_n)]$, where each q_i is defined to be $q_0^H a_1 s_1 \cdots a_i s_i$, and if $\alpha = q_0^H a_1 q_1 \cdots q_{n-1} a_n q_n \delta$, then $\mu_H(C_\alpha) \triangleq P^H_{q_0}[(a_1, q_1)] \cdots P^H_{q_{n-1}}[(a_n, q_n)] P^H_{q_n}[\delta]$, where each q_i is defined to be $q_0^H a_1 s_1 \cdots a_i s_i$. In [19] it is shown that there is a unique measure $\bar{\mu}_H$ that extends μ_H to the σ-field $\sigma(\mathcal{C}_H)$ generated by $\mathcal{C}_H$, i.e., the smallest σ-field that contains $\mathcal{C}_H$. Then, $\mathcal{F}_H$ is $\sigma(\mathcal{C}_H)$ and P_H is $\bar{\mu}_H$. With this definition it is possible to show that any union of cones is measurable.

3.4 Prefix

One of our objectives in the definition of the probabilistic model is that the standard notions defined on ordinary automata carry over to the probabilistic framework. One of this concepts is the notion of a prefix for ordinary executions. Here we just claim that it is possible to give a meaningful definition of a prefix for probabilistic executions.

Definition 5. A probabilistic execution fragment H is a prefix of a probabilistic execution fragment H', denoted by $H \leq H'$, iff H and H' have the same start state, and for each state q of H, $P_H[C_q] \leq P_{H'}[C_q]$. □

It is easy to verify that this definition of prefix coincides with the definition of prefix for ordinary executions when probability is absent. The reader is referred to [19] for a complete justification of Definition 5.

3.5 Parallel Composition

We now turn to the parallel composition operator, which is defined in the CSP style [9], i.e., by synchronizing two probabilistic automata on their common actions. As outlined in [8], it is not clear how to define a parallel composition operator for general probabilistic automata that extends the CSP synchronization style; thus, we define it only for simple probabilistic automata, and we concentrate on simple probabilistic automata for the rest of this paper. We use general probabilistic automata only for the analysis of probabilistic executions. The reader is referred to [19] for more details.

Definition 6. Two simple probabilistic automata M_1 and M_2 are said to be *compatible* if $int(M_1) \cap acts(M_2) = \emptyset$, and $int(M_2) \cap acts(M_1) = \emptyset$.

The *parallel composition* $M_1 \| M_2$ of two compatible simple probabilistic automata M_1 and M_2 is the simple probabilistic automaton M such that $states(M) = states(M_1) \times states(M_2)$, $start(M) = start(M_1) \times start(M_2)$, $ext(M) = ext(M_1) \cup ext(M_2)$, $int(M) = int(M_1) \cup int(M_2)$, and the transition relation satisfies the following: $((s_1, s_2), a, \mathcal{P}) \in trans(M)$ iff $\mathcal{P} = \mathcal{P}_1 \otimes \mathcal{P}_2$, such that

1. if $a \in acts(M_1)$ then $(s_1, a, \mathcal{P}_1) \in trans(M_1)$, else $\mathcal{P}_1 = \mathcal{D}(s_1)$, and
2. if $a \in acts(M_2)$ then $(s_2, a, \mathcal{P}_2) \in trans(M_2)$, else $\mathcal{P}_2 = \mathcal{D}(s_2)$. □

Remark. Another point in favor of these definitions is that it is possible to define the projection $H \lceil M_i$, $i = 1, 2$, of a probabilistic execution H of $M_1 \| M_2$ onto one of its components M_i. The definition is non-trivial and the interested reader is referred to [19]. □

3.6 Notation for Transitions

We conclude this section with some notation for transitions. We write $s \xrightarrow{a} \mathcal{P}$ whenever there is a simple transition $(s, a, \mathcal{P})$ in M, and we write $s \xrightarrow{a}_{\mathrm{C}} \mathcal{P}$ whenever there is a simple combined transition $(s, a, \mathcal{P})$ in M.

Similar to the non-probabilistic case, we extend the arrow notation to weak arrows ($\stackrel{a}{\Longrightarrow}_{\mathrm{C}}$) to state that $\mathcal{P}$ is reached through a sequence of combined transitions of M, some of which are internal. The main difference from the non-probabilistic case is that in our framework the transitions involved form a tree rather that a linear chain. Formally, $s \stackrel{a}{\Longrightarrow}_{\mathrm{C}} \mathcal{P}$, where a is either an external action or the empty sequence and $\mathcal{P}$ is a probability distribution over states, iff there is a probabilistic execution fragment H such that

1. the start state of H is s;
2. $P_H[\{\alpha\delta \mid \alpha\delta \in \Omega_H\}] = 1$, i.e., the probability of termination in H is 1;
3. for each $\alpha\delta \in \Omega_H$, $trace(\alpha) = a$, where $trace(\alpha)$ is the ordered sequence of external actions that occur in α;
4. $\mathcal{P} = lstate(\delta\text{-}strip(\mathcal{P}_H))$, where $\delta\text{-}strip(\mathcal{P}_H)$ is the probability space $\mathcal{P}'$ such that $\Omega' = \{\alpha \mid \alpha\delta \in \Omega_H\}$, and for each $\alpha \in \Omega'$, $P'[\alpha] = P_H[C_{\alpha\delta}]$;

Example 2. The left side of Figure 1 represents a weak transition with action a that leads to state s_1 with probability $5/12$ and to state s_2 with probability $7/12$. The action τ represents any internal action. From the formal definition of a weak transition, a tree that represents a weak transition may have an infinite branching structure, i.e., it may have transitions that lead to countably many states, and may have some infinite paths; however, each tree representing a weak transition has the property that infinite paths occur with probability 0.

The right side of Figure 1 represents a weak transition of a probabilistic automaton with cycles in its transition relation. Specifically, H represents the weak transition $s_0 \Longrightarrow \mathcal{P}$, where $P[s_0] = 1/8$ and $P[s_1] = 7/8$. If we extend H indefinitely on its right, then we obtain a new probabilistic execution fragment that represents the weak transition $s_0 \Longrightarrow \mathcal{D}(s_1)$. Observe that the new probabilistic execution fragment has an infinite path that occurs with probability 0. Furthermore, observe that there is no other way to reach state s_1 with probability 1. □

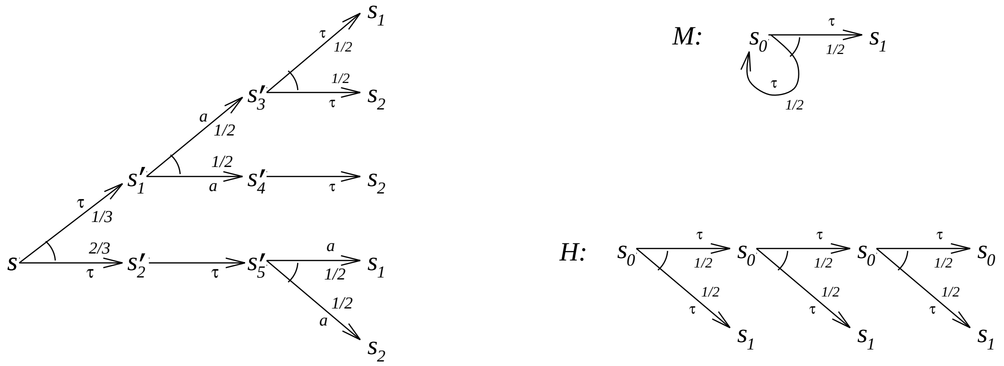

Fig. 1. Weak transitions (also probabilistic executions).

4 Trace Distribution Precongruence

The objective of this section is to extend the trace semantics to the probabilistic framework and to define a corresponding trace-based preorder. The problem is that a trace semantics is linear, i.e., it does not depend on the branching structure of a system, while in probabilistic automata the branching structure is important. Thus, the question is the following: under the condition that we want a trace semantics that describes the probabilistic behavior of a probabilistic automaton, how much of the branching structure of a probabilistic automaton do we have to know? We address the question above by defining a reasonable and natural trace semantics and by characterizing the minimum precongruence contained in it.

Definition 7. Let H be a probabilistic execution fragment of a probabilistic automaton M. For each extended execution fragment α of M, let $trace(\alpha)$ denote the ordered sequence of external actions of M that appear in α.

Let f be a function from Ω_H to $\Omega = ext(M)^* \cup ext(M)^\omega$ that assigns to each execution of Ω_H its trace. The *trace distribution* of H, denoted by $tdistr(H)$, is the probability space $(\Omega, \mathcal{F}, P)$ where $\mathcal{F}$ is the σ-field generated by the cones C_β, where β is an element of $ext(M)^*$, and $P = f(P_H)$. The fact that f is measurable follows from standard arguments. Denote a generic trace distribution by $\mathcal{D}$. A trace distribution of a probabilistic automaton M is the trace distribution of one of the probabilistic executions of M. ◻

Given two probabilistic executions H_1 and H_2, it is possible to check whether $tdistr(H_1) = tdistr(H_2)$ just by verifying that $P_{tdistr(H_1)}[C_\beta] = P_{tdistr(H_2)}[C_\beta]$ for each finite sequence of actions β. This follows from standard measure theory arguments. In [12] Jou and Smolka study a probabilistic trace semantics for generative processes; our rule above to determine whether two probabilistic executions have the same trace distribution coincides with the trace equivalence of [12] (a probabilistic execution is essentially a generative process).

Example 3. The reader may wonder why we have not defined Ω to be $trace(\Omega_H)$. This is to avoid to distinguish two trace distribution just because they have different sample spaces. Figure 2 illustrates the idea. The two probabilistic automata of Figure 2 have the same trace distributions; however, the left probabilistic automaton has a probabilistic execution where the trace a^∞ occurs with probability 0, while the right probabilistic automaton does not. Thus, by defining the sample space of $tdistr(H)$ to be $trace(\Omega_H)$, the two probabilistic automata of Figure 2 would be distinct. In Section 6 we define several simulation relations for

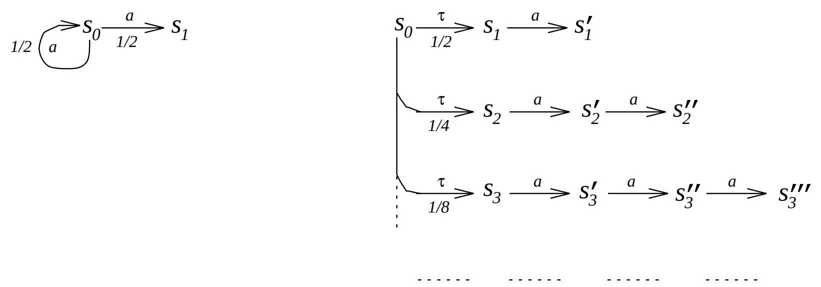

Fig. 2. Trace distribution equivalent probabilistic automata.

probabilistic automata and we show that they are sound for the trace distribution precongruence; such results would not be true with the alternative definition of Ω. □

It is easy to see that trace distributions extend the traces of ordinary automata: the trace distribution of a linear probabilistic execution fragment is a Dirac distribution. It is easy as well to see that prefix and action restriction extend to the probabilistic framework, thus enforcing our definition of a trace distribution. A trace distribution $\mathcal{D}$ is a *prefix* of a trace distribution $\mathcal{D}'$, denoted by $\mathcal{D} \leq \mathcal{D}'$, iff for each finite trace β, $P_{\mathcal{D}}[C_\beta] \leq P_{\mathcal{D}'}[C_\beta]$. Thus, two trace distributions are equal iff each one is a prefix of the other.

Lemma 8. *Let H_1 and H_2 be two probabilistic execution fragments of a probabilistic automaton M. If $H_1 \leq H_2$, then $tdistr(H_1) \leq tdistr(H_2)$.* □

Let β be a trace, and let V be a set of actions. Then $\beta \lceil V$ denotes the ordered sequence of actions from V that appear in β. Let $\mathcal{D} = (\Omega, \mathcal{F}, P)$ be a trace distribution. The *restriction* of $\mathcal{D}$ to V, denoted by $\mathcal{D} \lceil V$, is the probability space $(\Omega', \mathcal{F}', P')$ where $\Omega' = \{\beta \lceil V \mid \beta \in \Omega\}$, $\mathcal{F}'$ is the σ-field generated by the sets of cones $C_{\beta'}$ such that $\beta' \leq \beta$ for some $\beta \in \Omega'$, and P' is the inverse image of P under the function that restricts traces to V.

Lemma 9. *Let $\mathcal{D}$ be a trace distribution of $M_1 \| M_2$. Then, $\mathcal{D} \lceil ext(M_i)$, $i = 1, 2$, is a trace distribution of M_i.* □

Definition 10. Let M_1, M_2 be two probabilistic automata with the same external actions. The *trace distribution preorder* is defined as follows.

$$M_1 \sqsubseteq_D M_2 \text{ iff } tdistrs(M_1) \subseteq tdistrs(M_2).$$ □

The trace distribution preorder is a direct extension of the trace preorder of ordinary automata; however, it is not a precongruence. Consider the two probabilistic automata M_1 and M_2 of Figure 3. It is easy to see that M_1 and M_2 have the same trace distributions. Consider now the context C of Figure 3. Figure 4 shows a probabilistic execution of $M_2 \| C$ where there is a total correlation between the occurrence of actions d and f and of actions e and g. Such a correlation cannot be obtained from $M_1 \| C$, since the choice between f and g must be resolved before knowing what action among d and e is chosen probabilistically. Thus, $M_1 \| C$ and $M_2 \| C$ do not have the same trace distributions. This leads us to the following definition.

Definition 11. Let M_1, M_2 be two probabilistic automata with the same external actions. The *trace distribution precongruence*, denoted by $\sqsubseteq_{DC}$, is the coarsest precongruence that is contained in the trace distribution preorder. □

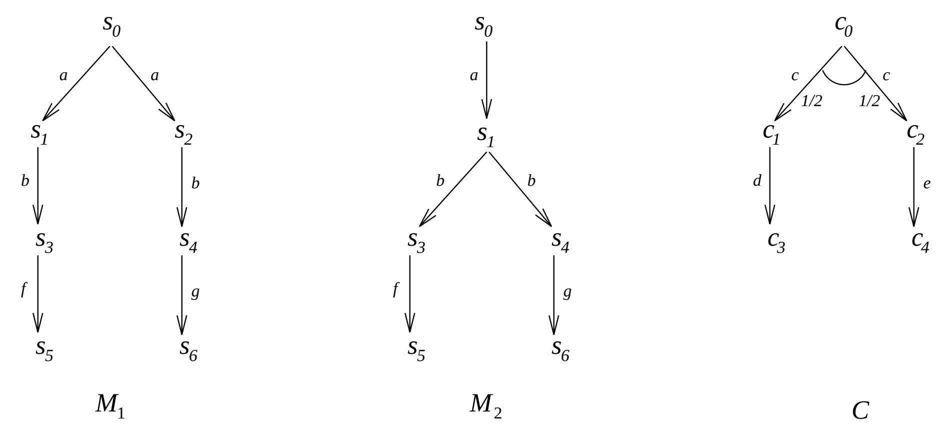

Fig. 3. The trace distribution preorder is not a precongruence.

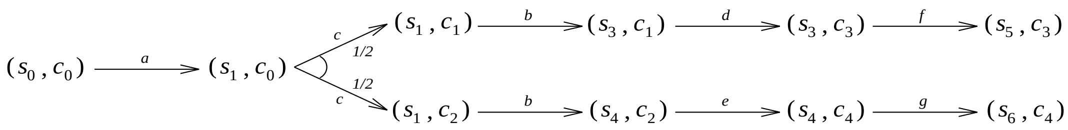

Fig. 4. A probabilistic execution of $M_2 \| C$.

The trace distribution precongruence preserves properties that resemble the safety properties of ordinary automata [1]. One example of such a property is the following.

> "*After some finite trace β has occurred, the probability that some action a occurs is not greater than p.*"

The property above means that in every trace distribution of a probabilistic automaton M the probability of the traces where action a occurs after β, conditional on the occurrence β, is not greater than p. Suppose that $M_1 \sqsubseteq_{DC} M_2$, and suppose by contradiction that M_2 satisfies the property above, while M_1 does not. Then there is a trace distribution of M_1 where the probability of a after β conditional to β is greater than p. Since $M_1 \sqsubseteq_{DC} M_2$, there is a trace distribution of M_2 where the probability of a after β conditional to β is greater than p. This contradicts the hypothesis that M_2 satisfies the property above.

5 The Principal Context

In this section we give an alternative characterization of the trace distribution precongruence that is easier to manipulate and that gives us an idea of the role of the branching structure of a probabilistic automaton. We define the *principal context*, denoted by C_P, and we show that there exists a context C that can distinguish two probabilistic automata M_1 and M_2 iff the principal context distinguishes M_1 and M_2.

The principal context is a probabilistic automaton with a unique state and three self-loop transitions labeled with actions that do not appear in any other probabilistic automaton. Two self-loop transitions are deterministic (Dirac) and are labeled with action *left* and *right*, respectively; the third self-loop transition is probabilistic, where one edge leads to the occurrence of action *pleft* with probability 1/2 and the other edge leads to the occurrence

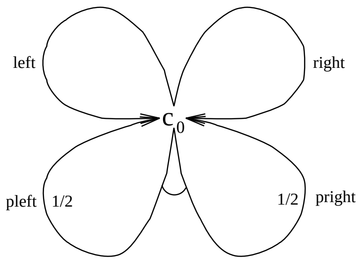

Fig. 5. The principal context.

of action *pright* with probability 1/2 (see Figure 5). The principal context is not a simple probabilistic automaton; however, since it does not have any action in common with any other probabilistic automaton, the parallel composition operator can be extended trivially: no synchronization is allowed. The main theorem is the following.

Theorem 12. *$M_1 \sqsubseteq_{DC} M_2$ iff $M_1 \| C_P \sqsubseteq_D M_2 \| C_P$.* □

As a corollary we obtain an alternative characterization of the trace distribution precongruence. Let a *principal trace distribution* of a probabilistic automaton M be a trace distribution of $M \| C_P$, and denote by $ptdistrs(M)$ the set $tdistrs(M \| C_P)$.

Corollary 13. *$M_1 \sqsubseteq_{DC} M_2$ iff $ext(M_1) = ext(M_2)$ and $ptdistrs(M_1) \subseteq ptdistrs(M_2)$.* □

We give a high level sketch of the proof of Theorem 12, which appears in [19]. The proof is structured in several steps where a generic distinguishing context C is transformed into a simpler distinguishing context C' till the point where the principal context is obtained. This shows the "if" part of the theorem. To show the "only if" part the principal context is transformed into a simple probabilistic automaton. The transformation steps are the following.

1. Ensure that C does not have any action in common with M_1 and M_2;
2. Ensure that C does not have any cycles in its transition relation;
3. Ensure that the branching structure of C is at most countable;
4. Ensure that the branching structure of C is at most binary;
5. Ensure that the probabilistic transitions of C lead to binary and uniform distributions;
6. Ensure that each action of C is external and appears exactly in one edge of the transition relation of C;
7. Ensure that each state of C enables two deterministic transitions and one probabilistic transition with a uniform binary distribution;
8. Rename all the actions of the context of 7 according to the action names of the principal context and then collapse all the states of the new context into a unique state, leading to the principal context.

We give an example of the first transformation. Let C be a distinguishing context for M_1 and M_2. Build C' as follows: for each each action a in common with M_1 and M_2, replace a with two new actions a_1, a_2, and replace each transition $(c, a, \mathcal{P})$ of C with two transitions (c, a_1, c') and $(c', a_2, \mathcal{P})$, where c' denotes a new state that is used only for the transition $(c, a, \mathcal{P})$. Denote c' by $c_{(c,a,\mathcal{P})}$.

Let $\mathcal{D}$ be a trace distribution of $M_1 \| C$ that is not a trace distribution of $M_2 \| C$. Consider a probabilistic execution H_1 of $M_1 \| C$ such that $tdistr(H_1) = \mathcal{D}$, and consider the scheduler that leads to H_1. Apply to $M_1\|C'$ the same scheduler with the following modification: whenever a transition $((s_1,c),a,\mathcal{P}_1\otimes\mathcal{P})$ is scheduled in $M_1\|C$, schedule $((s_1,c),a_1,\mathcal{D}((s_1,c')))$, where c' is $c_{(c,a,\mathcal{P})}$, followed by $((s_1,c'),a,\mathcal{P}_1\otimes\mathcal{D}(c'))$, and, for each $s'_1\in\Omega_1$, followed by $((s'_1,c'),a_2,\mathcal{D}(s'_1)\otimes\mathcal{P})$. Denote the resulting probabilistic execution by H'_1 and the resulting trace distribution by $\mathcal{D}'$. Then, we prove that $\mathcal{D}'\upharpoonright acts(M_1\|C)=\mathcal{D}$.

Suppose by contradiction that it is possible to obtain $\mathcal{D}'$ from $M_2\|C'$, and let H'_2 be a probabilistic execution of $M_2\|C'$ such that $tdistr(H'_2)=\mathcal{D}'$. Then, we show that it is possible to build a probabilistic execution H_2 of $M_2\|C$ such that $tdistr(H_2)=\mathcal{D}$. The construction of H_2 is not simple since we need to handle all the internal actions of M_2 that occur in H'_2 between each pair of actions of the form a_1,a_2.

6 Probabilistic Forward Simulations

The second main result of this paper is that the simulation method of [15] extends to the probabilistic framework. Thus, the trace distribution precongruence relation can be verified by defining a relation between the states of two probabilistic automata and checking some simple local conditions. This is one of the major verification methods for ordinary automata.

We start with the coarsest simulation relation of [20], and we show that it distinguishes too much. Then, we introduce our probabilistic forward simulation relations and we show their soundness for the trace distribution precongruence.

A *weak probabilistic simulation* between two simple probabilistic automata M_1 and M_2 is a relation $\mathcal{R}\subseteq states(M_1)\times states(M_2)$ such that

1. each start state of M_1 is related to at least one start state of M_2;
2. for each pair of states $s_1\ \mathcal{R}\ s_2$ and each transition $s_1\xrightarrow{a}\mathcal{P}_1$ of M_1, there exists a weak combined transition $s_2\overset{a\upharpoonright ext(M_2)}{\Longrightarrow}_{\mathrm{C}}\mathcal{P}_2$ of M_2 such that $\mathcal{P}_1\sqsubseteq_{\mathcal{R}}\mathcal{P}_2$,

where $\sqsubseteq_{\mathcal{R}}$ is the *lifting* of $\mathcal{R}$ to probability spaces [10, 20]. That is, $\mathcal{P}_1\sqsubseteq_{\mathcal{R}}\mathcal{P}_2$ iff there exists a function $w:\Omega_1\times\Omega_2\to[0,1]$ such that

1. for each $s_1\in\Omega_1$, $\sum_{s_2\in\Omega_2}w(s_1,s_2)=P_1[s_1]$,
2. for each $s_2\in\Omega_2$, $\sum_{s_1\in\Omega_1}w(s_1,s_2)=P_2[s_2]$,
3. for each $(s_1,s_2)\in\Omega_1\times\Omega_2$, if $w(s_1,s_2)>0$ then $s_1\ \mathcal{R}\ s_2$.

The idea behind the definition of $\sqsubseteq_{\mathcal{R}}$ is that each state of Ω_1 must be represented by some states of Ω_2, and similarly, each state of Ω_2 must represent one or more states of Ω_1.

Example 4. Weak probabilistic simulations are sound for the trace distribution precongruence (cf. Theorem 16); however, they are too strong.

Consider the two probabilistic automata of Figure 6. The probabilistic automaton M_2, which chooses internally one element out of four with probability 1/4 each, is implemented by the probabilistic automaton M_1, which flips two fair coins to make the same choice (by "implement" we mean $\sqsubseteq_D$). However, the first transition of M_1 cannot be simulated by M_2 since the probabilistic choice of M_2 is not resolved completely yet in M_1. This situation suggests a new preorder relation where a state of M_1 can be related to a probability distribution over states of M_2. The informal idea behind a relation $s_1\ \mathcal{R}\ \mathcal{P}_2$ is that s_1 represents an intermediate stage of M_1 in reaching the distribution $\mathcal{P}_2$. For example, in Figure 6 state s_1 would be related to a uniform distribution $\mathcal{P}$ over states s'_3 and s'_4 ($\mathcal{P}=\mathcal{U}(s'_3,s'_4)$), meaning that s_1 is an intermediate stage of M_1 in reaching the distribution $\mathcal{P}$.

It is also possible to create examples where the relationship between s and $\mathcal{P}$ does not mean simply that s is an intermediate stage of M_1 in reaching the distribution $\mathcal{P}$, but rather

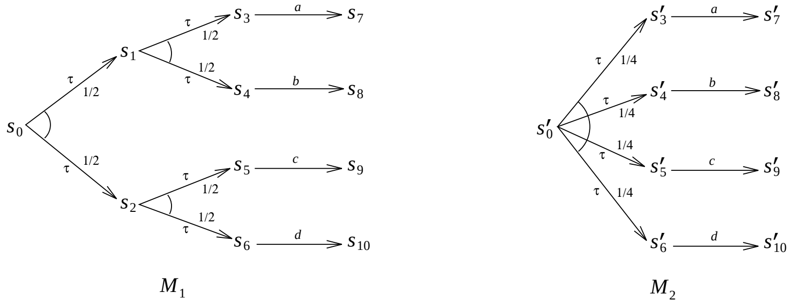

Fig. 6. Implementation of a probabilistic transition with several probabilistic transitions.

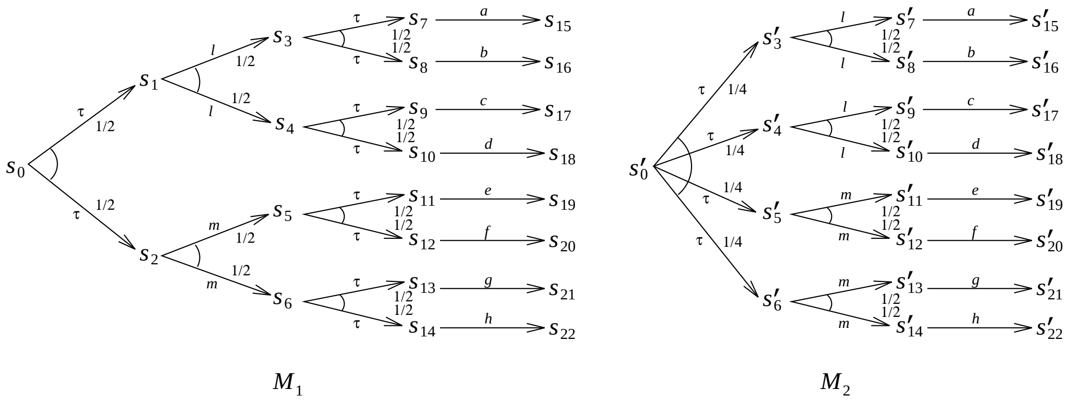

Fig. 7. A more sophisticated implementation.

that s is an intermediate stage in reaching a probability distribution that can be reached from $\mathcal{P}$. Consider the two probabilistic automata of Figure 7. Although not evident at the moment, M_1 and M_2 are in the trace distribution precongruence relation, i.e., $M_1 \sqsubseteq_{DC} M_2$. Following the same idea as for the example of Figure 6, state s_1 is related to $\mathcal{U}(s'_3, s'_4)$. However, s_1 is not an intermediate stage of M_1 in reaching $\mathcal{U}(s'_3, s'_4)$, since s_1 enables a transition labeled with an external action l, while in M_2 no external action occurs before reaching $\mathcal{U}(s'_3, s'_4)$. Rather, from s'_3 and s'_4 there are two transitions labeled with l, and thus the only way to simulate the transition $s_1 \xrightarrow{l} \mathcal{U}(s_3, s_4)$ from $\mathcal{U}(s'_3, s'_4)$ is to perform the two transitions labeled with l, which lead to the distribution $\mathcal{U}(s'_7, s'_8, s'_9, s'_{10})$. Now the question is the following: in what sense does $\mathcal{U}(s'_7, s'_8, s'_9, s'_{10})$ represent $\mathcal{U}(s_3, s_4)$? The first observation is that s_3 can be seen as an intermediate stage in reaching $\mathcal{U}(s'_7, s'_8)$, and that s_4 can be seen as an intermediate stage in reaching $\mathcal{U}(s'_9, s'_{10})$. Thus, s_3 is related to $\mathcal{U}(s'_7, s'_8)$ and s_4 is related to $\mathcal{U}(s'_9, s'_{10})$. The second observation is that $\mathcal{U}(s'_7, s'_8, s'_9, s'_{10})$ can be expressed as $1/2\mathcal{U}(s'_7, s'_8) + 1/2\mathcal{U}(s'_9, s'_{10})$. Thus, $\mathcal{U}(s'_7, s'_8, s'_9, s'_{10})$ can be seen as a combination of two probability spaces, each one representing an element of $\mathcal{U}(s_3, s_4)$. This recalls the lifting of a relation that we introduced at the beginning of this section. □

Definition 14. A *probabilistic forward simulation* between two simple probabilistic automata M_1 and M_2 is a relation $\mathcal{R} \subseteq states(M_1) \times Probs(states(M_2))$ such that

1. each start state of M_1 is related to at least one Dirac distribution over a start state of M_2;

2. for each $s \mathcal{R} \mathcal{P}'$, if $s \xrightarrow{a} \mathcal{P}_1$, then
   (a) for each $s' \in \Omega'$ there exists a probability space $\mathcal{P}_{s'}$ such that $s' \stackrel{a\lceil ext(M_2)}{\Longrightarrow_C} \mathcal{P}_{s'}$, and
   (b) there exists a probability space $\mathcal{P}'_2$ of $Probs(Probs(states(M_2)))$ satisfying $\mathcal{P}_1 \sqsubseteq_{\mathcal{R}} \mathcal{P}'_2$, such that $\sum_{s' \in \Omega'} P'[s']\mathcal{P}_{s'} = \sum_{\mathcal{P} \in \Omega'_2} P'_2[\mathcal{P}]\mathcal{P}$.

We write $M_1 \sqsubseteq_{FS} M_2$ whenever $ext(M_1) = ext(M_2)$ and there is a forward simulation from M_1 to M_2. □

Example 5. The probabilistic forward simulation for the probabilistic automata M_1 and M_2 of Figure 7 is the following: s_0 is related to $\mathcal{U}(s'_0)$; each state s_i, $i \geq 7$, is related to $\mathcal{D}(s'_i)$; each state s_i, $1 \leq i \leq 6$, is related to $\mathcal{U}(s'_{2i+1}, s'_{2i+2})$. It is an easy exercise to check that this relation is a probabilistic forward simulation. Observe also that there is no probabilistic forward simulation from M_2 to M_1. Informally, s'_3 cannot be simulated by M_1, since the only candidate state to be related to s'_1 is s_1, and s_1 does not contain all the information contained in s'_3. The formal way to see that there is no probabilistic forward simulation from M_2 to M_1 is to observe that M_2 and M_1 are not in the trace distribution precongruence relation and then use the fact that probabilistic forward simulations are sound for the trace distribution precongruence relation (cf. Theorem 16). In $M_2 \| C_P$ it is possible force action *left* to be scheduled exactly when M_2 is in s'_3, and thus it is possible to create a correlation between action *left* and actions a and b; in $M_1 \| C_P$ such a correlation cannot be created since action *left* must be scheduled before action l. □

Proposition 15. *$\sqsubseteq_{FS}$ is a preorder and is preserved by parallel composition.* □

The proof that $\sqsubseteq_{FS}$ is preserved by the parallel composition operator is standard. The proof that $\sqsubseteq_{FS}$ is transitive is much more complicated and is based on a probabilistic version of the execution correspondence lemma of [6]. The complete proofs can be found in [19].

Theorem 16. *Let $M_1 \sqsubseteq_{FS} M_2$. Then $M_1 \sqsubseteq_{DC} M_2$.* □

The proof of Theorem 16, which appears in [19], is carried out in two steps. Let $\mathcal{R}$ be a probabilistic forward simulation from M_1 to M_2. Given a probabilistic execution H_1 of M_1, we build a probabilistic execution H_2 of M_2 that *represents* H_1 via $\mathcal{R}$. The structure that describes how H_2 represents H_1 is called an *execution correspondence structure*. Then, we show that if H_2 represents H_1, then H_1 and H_2 have the same trace distribution. Thus, $M_1 \sqsubseteq_{FS} M_2$ implies $M_1 \sqsubseteq_D M_2$. Since from Proposition 15 $\sqsubseteq_{FS}$ is a precongruence, the proof of Theorem 16 is completed.

7 Concluding Remarks

We have defined a trace-based semantics for probabilistic automata that is preserved by the parallel composition operator, and we have extended the simulation method of [15] to the new framework. The main object of observation is a trace distribution, which is a probability distribution over traces. The compositionality result is obtained by studying the trace distributions of a system composed in parallel with an elementary context called the principal context. The new simulation relations have the interesting property that states are related to probability distributions over states.

In further work we plan to investigate on completeness results concerning probabilistic forward simulations and the trace distribution precongruence. We also plan to to apply the same methodology outlined in this paper to define a failure-based semantics for probabilistic automata. Finally, it is desirable to study further what can be done with general probabilistic automata and how to extend the work of this paper to models that include real-time or hybrid behavior. A trace-based semantics for probabilistic timed automata is studied in [19].

Acknowledgments. I would like to thank Nancy Lynch for useful discussion that lead to the definition of probabilistic forward simulations.

References

1. B. Alpern and F.B. Schneider. Defining liveness. *Information Processing Letters*, 21(4), 1985.
2. J. Aspnes and M.P. Herlihy. Fast randomized consensus using shared memory. *Journal of Algorithms*, 15(1):441–460, September 1990.
3. M. Ben-Or. Another advantage of free choice: completely asynchronous agreement protocols. In *Proceedings of the* 2^{nd} *Annual ACM PODC*, 1983.
4. S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, 1984.
5. I. Christoff. *Testing Equivalences for Probabilistic Processes.* PhD thesis, Department of Computer Science, Uppsala University, 1990.
6. R. Gawlick, R. Segala, J.F. Søgaard-Andersen, and N.A. Lynch. Liveness in timed and untimed systems. In *Proceedings* 21^{th} *ICALP,* Jerusalem, LNCS 820, 1994. A full version appears as MIT Technical Report number MIT/LCS/TR-587.
7. R.J. van Glabbeek, S.A. Smolka, B. Steffen, and C.M.N. Tofts. Reactive, generative, and stratified models of probabilistic processes. In *Proceedings* 5^{th} *Annual Symposium on Logic in Computer Science,* Philadelphia, USA, pages 130–141. IEEE Computer Society Press, 1990.
8. H. Hansson. *Time and Probability in Formal Design of Distributed Systems*, volume 1 of *Real-Time Safety Critical Systems.* Elsevier, 1994.
9. C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, 1985.
10. B. Jonsson and K.G. Larsen. Specification and refinement of probabilistic processes. In *Proceedings of the 6th IEEE Symposium on Logic in Computer Science*, pages 266–277, July 1991.
11. B. Jonsson and J. Parrow, editors. *Proceedings of CONCUR 94*, LNCS 836, 1994.
12. C.C. Jou and S.A. Smolka. Equivalences, congruences, and complete axiomatizations for probabilistic processes. In *Proceedings of CONCUR 90*, LNCS 458, pages 367–383, 1990.
13. K.G. Larsen and A. Skou. Compositional verification of probabilistic processes. In *Proceedings of CONCUR 92* LNCS 630, pages 456–471, 1992.
14. N.A. Lynch, I. Saias, and R. Segala. Proving time bounds for randomized distributed algorithms. In *Proceedings of the* 13^{th} *Annual ACM PODC*, pages 314–323, 1994.
15. N.A. Lynch and F.W. Vaandrager. Forward and backward simulations for timing-based systems. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Proceedings of the REX Workshop "Real-Time: Theory in Practice"*, LNCS 600, pages 397–446, 1991.
16. G.D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer science Department, Aarhus University, 1981.
17. A. Pogosyants and R. Segala. Formal verification of timed properties of randomized distributed algorithms. In *Proceedings of the* 14^{th} *Annual ACM PODC*, 1995.
18. M.O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.
19. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems.* PhD thesis, MIT, Dept. of Electrical Engineering and Computer Science, 1995.
20. R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. In Jonsson and Parrow [11], pages 481–496.
21. K. Seidel. Probabilistic communicating processes. Technical Report PRG-102, Ph.D. Thesis, Programming Research Group, Oxford University Computing Laboratory, 1992.
22. M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings of 26th IEEE Symposium on Foundations of Computer Science*, pages 327–338, 1985.
23. S.H. Wu, S. Smolka, and E.W. Stark. Composition and behaviors of probabilistic I/O automata. In Jonsson and Parrow [11].