Observing Branching Structure Through Probabilistic Contexts*

Nancy Lynch†
Computer Science and Artificial Intelligence Laboratory
MIT, Cambridge, MA 02139, USA
lynch@theory.csail.mit.edu

Roberto Segala‡
Dipartimento di Informatica, Università di Verona, Italy
roberto.segala@univr.it

Frits Vaandrager§
Institute for Computing and Information Sciences
Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
F.Vaandrager@cs.ru.nl

October 13, 2006

Abstract

Probabilistic automata (PAs) constitute a general framework for modeling and analyzing discrete event systems that exhibit both nondeterministic and probabilistic behavior, such as distributed algorithms and network protocols. The behavior of PAs is commonly defined using schedulers (also called adversaries or strategies), which resolve all nondeterministic choices based on past history. From the resulting purely probabilistic structures, trace distributions can be extracted, whose intent is to capture the observable behavior of a PA. However, when PAs are composed via an (asynchronous) parallel composition operator, a global scheduler may establish strong correlations between the behavior of system components and, for example, resolve nondeterministic choices in one PA based on the outcome of probabilistic choices in the other. It is well known that, as a result of this, the (linear-time) trace distribution precongruence is not compositional for PAs. In his PhD thesis from '95, Segala has shown that the (branching-time) probabilistic simulation preorder is compositional for PAs. In this paper, we establish that the simulation preorder is in fact the coarsest refinement of the trace distribution preorder that is compositional.

We prove our characterization result by providing (1) a context of a given PA $\mathcal{A}$, called the *tester*, that may announce the state of $\mathcal{A}$ to the outside world, and (2) a specific global scheduler, called the *observer*, which ensures that the state information that is announced is actually correct. Now when another PA $\mathcal{B}$ is composed with the tester, it may generate the same external behavior as the observer only when it is able to simulate $\mathcal{A}$ in the sense that whenever $\mathcal{A}$ goes to some state s, $\mathcal{B}$ can go to a corresponding state u from which it may generate the same

*A preliminary version of this paper appeared as [24].

†Supported by AFOSR contract #F49620-00-1-0097, AFRL Award #FA9550-04-1-0121, NSF grants #CCR-0121277 and #CCR-0326277, NSF Award #CNS-0614414, NSF/ITR grants #CCR-0121277 and #CCR-0326277, USAF/AFRL Award #FA9550-04-1-0121, and DARPA/AFOSR MURI #F49620-02-1-0325.

‡Supported by MURST projects MEFISTO and CoVer, MIUR project AIDA, Inria project ProNoBis.

§Supported by PROGRESS project TES4999: Verification of Hard and Softly Timed Systems (HaaST) and DFG/NWO bilateral cooperation project Validation of Stochastic Systems (VOSS and VOSS2).

external behavior. Our result shows that probabilistic contexts together with global schedulers are able to exhibit the branching structure of PAs.

1 Introduction

Labeled transition systems (automata) are studied extensively within concurrency theory as underlying operational models of concurrent systems [27]: a system is described as a state machine whose transitions are labeled by *actions*, where each action describes potential communication with the external environment. An important aspect of concurrency theory is the study of relationships between systems, namely equivalence and preorder relations, with the objective of understanding whether a system can be used in place of another one or as an implementation of some more abstract description. Several relations are studied in the literature, but the most important classes of relations are represented by simulations and bisimulations [27] and by language (trace) inclusion and equivalence [17]. An extensive classification of existing relations appears in [12] where in particular relations are classified as *branching*, which observe the places where nondeterminism is resolved, and *linear*, which are insensitive to the actual places where nondeterminism is resolved. For instance, language inclusion and language equivalence are linear relations, while simulations and bisimulations are branching relations.

During the last fifteen years there has been a growing interest in the extension of concurrent models with probabilities, mainly motivated by the fact that several applications included randomized behaviors. Some of the most relevant proposals of operational models with probability and nondeterminism are reactive, generative and stratified systems [13], concurrent labeled Markov chains [15], alternating automata [40, 29], probabilistic automata [32], and probabilistic reactive modules [10]. Extensive comparative studies that include these models appear in [36, 4, 35].

Simulation, bisimulation, and language inclusion relations have been extended to the probabilistic case as well. In particular [22] defines strong bisimulation on reactive systems, [34] defines strong and weak simulation and bisimulation relations on probabilistic automata, including a notion of branching bisimulation, [15] defines strong bisimulation on labeled concurrent Markov chains, [29] defines strong and weak bisimulation on alternating automata, and [2] defines branching bisimulation on alternating automata. Although the above definitions are quite different, it turns out that they can all be seen in a uniform way by viewing reactive systems, labeled concurrent Markov chains, and alternating automata as special cases of probabilistic automata [35]. For extensive comparative studies we refer the reader again to [36, 4, 35].

In this paper we are interested in extensions of language inclusion to the probabilistic case. On ordinary nondeterministic automata the resolution of nondeterminism produces sequences of alternating states and actions called *executions*; then, by restricting those sequences to visible actions, we obtain the so called *traces*. Implementation and equivalence of nondeterministic automata can be defined in terms of inclusion and equality of sets of traces. This approach was first proposed in the context of process algebras [17] and is used extensively in the area of I/O automata [25].

An attempt to extend language inclusion to probabilistic automata appears in [31], where it is proposed that the probabilistic extension of a trace should be a probability measure over traces. Indeed, the resolution of nondeterminism on probabilistic automata produces a stochastic process that induces a probability measure over executions (a *probabilistic execution*), and the restriction of a probabilistic execution to the externally visible actions leads to a probability measure over traces (a *trace distribution*). Then, the proposal of [31] is to compare probabilistic automata based on inclusion and equality of sets of trace distributions. This is consistent with ordinary nondeterministic automata since an execution can be seen as a probabilistic execution that assigns

probability 1 to a single element, and similarly a trace can be seen as a trace distribution that assigns probability 1 to a single element.

There are several arguments in favor of the point of view that probabilistic executions in probabilistic automata should play the role that executions play in ordinary nondeterministic automata, and thus in favor of the notion of trace distribution as well. One element is that this point of view lead to the definition of *weak transitions*, used to extend weak simulations and bisimulation to the probabilistic case. Another element of evidence comes from the area of distributed algorithms, where the probability of termination of an algorithm is studied under any scheduling policy: in this context a scheduler is the entity that resolves nondeterminism, for example, by choosing the order processes take steps, and a probabilistic execution is the natural object where the probability of termination can be computed, as demonstrated by several case studies [23, 30, 1, 39, 20, 28, 6] and by the ongoing research on automatic verification tools for probabilistic systems [16]. Finally, again in the area of distributed algorithms, the approach of [31] to language inclusion turned out to be useful for the modular analysis of complex algorithms [30].

An important requirement for an implementation relation on systems is *compositionality*, that is, the relation is preserved by parallel composition. For labelled transition systems, the trace, simulation and bisimulation preorders are all compositional [17, 27]. For probabilistic automata, various simulation and bisimulation preorders are known to be compositional [34]. A problem with the trace-based relations proposed in [31] is that they are not compositional, that is, they are not preserved by parallel composition. A typical solution to the problem, followed by [31], is to define a notion of *trace distribution precongruence* as the coarsest precongruence included in the trace distribution inclusion. Unfortunately, such implicit definition does not provide much insight about the structure of the relation. For this reason, there have been several attempts to characterize it in more concrete terms. In [32] trace distribution precongruence is characterized in terms of the set of trace distributions observable in a certain *principal context*—a rudimentary probabilistic automaton that makes very limited nondeterministic and probabilistic choices; in [33] a testing scenario is proposed. However, these indirect characterizations still do not provide much insight into the structure of trace distribution precongruence; for example, they do not explain its branching structure. Indeed, trace distribution precongruence is not a linear relation since it distinguishes ordinary nondeterministic automata that are trace equivalent.

In this paper, we provide an explicit characterization of the trace distribution precongruence, $\leq_{DC}$, for probabilistic automata, which completely explains its branching structure. Namely, we show that $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ if and only if there exists a *weak probabilistic (forward) simulation relation* from $\mathcal{P}_1$ to $\mathcal{P}_2$. Moreover, we provide a similar characterization of $\leq_{DC}$ for nondeterministic automata in terms of the existence of a weak (non-probabilistic) simulation relation. It was previously known that simulation relations are sound for $\leq_{DC}$ [32], for both nondeterministic and probabilistic automata; we show the surprising fact that they are also *complete*. That is, we show that, for both nondeterministic and probabilistic automata, probabilistic contexts can observe all the distinctions that can be expressed using simulation relations.

Our proofs of completeness rely on special contexts for probabilistic automata, called *testers*. The tester of a probabilistic automaton $\mathcal{P}$, under the action of an appropriate scheduler, can reveal the branching structure of $\mathcal{P}$ via a trace distribution. Such a scheduler is called an *observer scheduler*. Informally, the tester $\mathcal{C}$ of a probabilistic automaton $\mathcal{P}$ announces the outcome of each probabilistic choice of $\mathcal{P}$ by performing an action with the name of the state reached, and flips coins to propose and announce how $\mathcal{P}$ should resolve its nondeterministic choices. The ability of another probabilistic automaton $\mathcal{P}'$ to comply with the requirements of $\mathcal{C}$, that is, that the trace distribution induced by the observer scheduler is also a trace distribution of $\mathcal{P}' \| \mathcal{C}$ (the parallel composition of

$\mathcal{P}'$ and $\mathcal{C}$), reveals whether $\mathcal{P}'$ has at least the same possibilities to solve nondeterministic choices as $\mathcal{P}$. If $\mathcal{P} \leq_{DC} \mathcal{P}'$, then we extract a probabilistic forward simulation from $\mathcal{P}$ to $\mathcal{P}'$ by observing how $\mathcal{P}' \| \mathcal{C}$ produces the trace distribution induced by the observer scheduler.

An interesting observation about tester automata, is that probabilistic choices of a probabilistic automaton are observed via nondeterministic choices of the tester automaton, while nondeterministic choices of a probabilistic automaton are observed via probabilistic choices of the tester automaton. Thus, the branching structure of a probabilistic automaton is observed via a probabilistic context.

The rest of the paper is structured as follows. Sections 2 and 3 contain basic definitions and results for nondeterministic and probabilistic automata, respectively, and for the preorders we consider. These sections contain no new material, but recall definitions and theorems from the literature. For a more leisurely introductions see [25, 26, 38, 36]. Section 4 introduces the concept of tester automaton and the scheduler for a probabilistic automaton $\mathcal{P}$ and its tester that reveals the structure of $\mathcal{P}$. Sections 5 and 6 contain, respectively, our characterization results for nondeterministic and probabilistic automata. Since the proof of the characterization result for the general case of probabilistic automata with internal actions is highly complex, we first present a proof for the special case of nondeterministic automata without internal actions (Section 5.1). Then we successively show how we can also handle internal actions (Section 5.2) and probabilistic choice (Section 6.1) before dealing with the general case of probabilistic automata with internal actions (Section 6.2). Section 7 contains our conclusions.

2 Definitions and Basic Results for Nondeterministic Automata

In this section we recall definitions and basic results for nondeterministic automata. We impose a few restrictions to avoid confusion and unnecessary complications in the rest of the paper. For more information the reader is referred to [25, 26].

2.1 Nondeterministic Automata, Executions, and Traces

A *nondeterministic automaton* is a tuple $\mathcal{A} = (Q, \bar{q}, E, H, D)$, where

- Q is a countable set of *states*,
- $\bar{q} \in Q$ is a *start state*,
- E is a countable set of *external actions*,
- H is a countable set of *internal (hidden) actions* with $E \cap H = \emptyset$, and
- $D \subseteq Q \times (E \cup H) \times Q$ is a *transition relation*.

We denote $E \cup H$ by A and we refer to it as the set of *actions*. We denote a transition (q, a, q') of D by $q \xrightarrow{a} q'$. We write $q \rightarrow q'$ if $q \xrightarrow{a} q'$ for some a, and we write $q \rightarrow$ if $q \rightarrow q'$ for some q'.

We assume finite branching: for each state q the number of pairs (a, q') such that $q \xrightarrow{a} q'$ is finite. We denote the elements of a nondeterministic automaton $\mathcal{A}$ by $Q_{\mathcal{A}}, \bar{q}_{\mathcal{A}}, E_{\mathcal{A}}, H_{\mathcal{A}}, D_{\mathcal{A}}, A_{\mathcal{A}}, \xrightarrow{a}_{\mathcal{A}}$. Often we use the name $\mathcal{A}$ for a generic nondeterministic automaton; in this case, we usually omit the subscripts, writing simply Q, $\bar{q}$, E, H, D, A, and $\xrightarrow{a}$. We extend this convention to allow indices and primes as well; thus, the set of states of a nondeterministic automaton $\mathcal{A}'_i$ is denoted by Q'_i.

Remark 2.1 *In the definition of nondeterministic automaton above we have imposed some restrictions that are not strictly necessary for this paper, but rather avoid unnecessary complications. The restriction on the cardinality of the sets of states and actions is imposed to ensure that a nondeterministic automaton has at most countably many finite execution fragments (see definition later), which simplifies the use of measure theory later. The finite branching restriction is imposed to simplify the construction of the tester automaton in Section 4; however, the results of this paper generalize to countable branching at the cost of adding complexity to the proofs (cf. Remark 4.5). We have also chosen to define nondeterministic automata with a single initial state rather than a set of initial states. Sets of initial states do not add any technical insight, but they complicate notation slightly.*

An *execution fragment* of a nondeterministic automaton $\mathcal{A}$ is a finite or infinite sequence $\alpha = q_0 a_1 q_1 a_2 q_2 \cdots$ of alternating states and actions, starting with a state and, if the sequence is finite, ending in a state, where each $(q_i, a_{i+1}, q_{i+1}) \in D$. State q_0, the first state of α, is denoted by $fstate(\alpha)$. If α is a finite sequence, then the last state of α is denoted by $lstate(\alpha)$. An *execution* of $\mathcal{A}$ is an execution fragment whose first state is the start state $\bar{q}$. We let $frags(\mathcal{A})$ denote the set of execution fragments of $\mathcal{A}$ and $frags^*(\mathcal{A})$ the set of finite execution fragments. Similarly, we let $execs(\mathcal{A})$ denote the set of executions of $\mathcal{A}$ and $execs^*(\mathcal{A})$ the set of finite executions.

Execution fragment α is a *prefix* of execution fragment α', denoted by $\alpha \leq \alpha'$, if sequence α is a prefix of sequence α'. Finite execution fragment $\alpha_1 = q_0 a_1 q_1 \cdots a_k q_k$ and execution fragment α_2 can be concatenated if $fstate(\alpha_2) = q_k$. In this case the *concatenation* of α_1 and α_2, $\alpha_1 \frown \alpha_2$, is the execution fragment $q_0 a_1 q_1 \cdots a_k \alpha_2$. Given an execution fragment α and a finite prefix α', $\alpha \triangleright \alpha'$ (read as "α after α'") is defined to be the unique execution fragment α'' such that $\alpha = \alpha' \frown \alpha''$.

The *trace* of an execution fragment α of a nondeterministic automaton $\mathcal{A}$, written $trace_{\mathcal{A}}(\alpha)$, or just $trace(\alpha)$ when $\mathcal{A}$ is clear from context, is the sequence obtained by restricting α to the set of external actions of $\mathcal{A}$. For a set S of executions of a nondeterministic automaton $\mathcal{A}$, $traces_{\mathcal{A}}(S)$, or just $traces(S)$ when $\mathcal{A}$ is clear from context, is the set of traces of the executions in S. We say that β is a trace of a nondeterministic automaton $\mathcal{A}$ if there is an execution α of $\mathcal{A}$ with $trace(\alpha) = \beta$. Let $traces(\mathcal{A})$ denote the set of traces of $\mathcal{A}$. We define the *trace preorder* relation on nondeterministic automata as follows: $\mathcal{A}_1 \leq_T \mathcal{A}_2$ iff $E_1 = E_2$ and $traces(\mathcal{A}_1) \subseteq traces(\mathcal{A}_2)$. We use $\equiv_T$ to denote the kernel of $\leq_T$. That is, $\mathcal{A}_1 \equiv_T \mathcal{A}_2$ iff $\mathcal{A}_1 \leq_T \mathcal{A}_2$ and $\mathcal{A}_2 \leq_T \mathcal{A}_1$. A similar convention will be adopted to denote the kernels of other preorder relations used in the paper.

If $\beta \in A^*$, then $q \stackrel{\beta}{\Longrightarrow} q'$ iff there exists an execution fragment α such that $fstate(\alpha) = q$, $lstate(\alpha) = q'$, and $trace(\alpha) = trace(\beta)$. (Here and elsewhere, we abuse notation slightly by extending the *trace* function to arbitrary sequences.) We call $q \stackrel{\beta}{\Longrightarrow} q'$ a *weak transition*. If β is the empty sequence, then we write alternatively $q \Longrightarrow q'$. Observe that the definition of $q \stackrel{\beta}{\Longrightarrow} q'$ depends only on the external actions that occur in β. We have chosen to define weak transitions for any sequence β, including internal actions as well, for notational convenience in later definitions.

We let tr range over either transitions or weak transitions. For a transition $tr = (q, a, q')$, we denote q by $source(tr)$ and q' by $target(tr)$.

2.2 Composition

We define composition of nondeterministic automata by synchronizing them on common external actions. There are several ways to do this, but the simplest approach that is followed in several papers is to synchronize nondeterministic automata on common actions and impose the restriction that no internal action of a component is an action of the other component as well. This restriction can easily be eliminated, for example, by renaming internal actions if necessary.

Nondeterministic automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are *compatible* if $H_1 \cap A_2 = A_1 \cap H_2 = \emptyset$. The *(parallel) composition* of compatible nondeterministic automata $\mathcal{A}_1$ and $\mathcal{A}_2$, denoted by $\mathcal{A}_1 \| \mathcal{A}_2$, is the nondeterministic automaton $\mathcal{A} \stackrel{\Delta}{=} (Q_1 \times Q_2, (\bar{q}_1, \bar{q}_2), E_1 \cup E_2, H_1 \cup H_2, D)$ where D is the set of triples (q, a, q') such that, for $i \in \{1, 2\}$:

$$a \in A_i \Rightarrow (\pi_i(q), a, \pi_i(q')) \in D_i \text{ and } a \notin A_i \Rightarrow \pi_i(q) = \pi_i(q'),$$

where π_i is the projection function on states of $\mathcal{A}$ defined by $\pi_i(q_1, q_2) = q_i$.

Let α be an execution fragment of $\mathcal{A}_1 \| \mathcal{A}_2$, $i \in \{1, 2\}$. Then $\pi_i(\alpha)$, the i^{th} projection of α, is the sequence obtained from α by projecting each state onto its i^{th} component, and removing each action not in A_i together with its following state. Sometimes we denote this projection by $\alpha \lceil \mathcal{A}_i$.

Proposition 2.2 *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be nondeterministic automata, with $\mathcal{A}_1 \leq_T \mathcal{A}_2$. Then, for each nondeterministic automaton $\mathcal{C}$ compatible with both $\mathcal{A}_1$ and $\mathcal{A}_2$, $\mathcal{A}_1 \| \mathcal{C} \leq_T \mathcal{A}_2 \| \mathcal{C}$.*

2.3 Simulation Relations

We define two kinds of simulation relations: forward simulations, which provide a step-by-step correspondence, and weak forward simulations, which are insensitive to the occurrence of internal steps. Namely, relation $R \subseteq Q_1 \times Q_2$ is a *forward simulation* (resp., *weak forward simulation*) from $\mathcal{A}_1$ to $\mathcal{A}_2$ iff $E_1 = E_2$ and both of the following hold:

1. $\bar{q}_1 \; R \; \bar{q}_2$.
2. If $q_1 \; R \; q_2$ and $q_1 \xrightarrow{a} q_1'$, then there exists q_2' such that $q_2 \xrightarrow{a} q_2'$ (resp., $q_2 \stackrel{a}{\Longrightarrow} q_2'$) and $q_1' \; R \; q_2'$.

We write $\mathcal{A}_1 \leq_F \mathcal{A}_2$ (resp., $\mathcal{A}_1 \leq_{wF} \mathcal{A}_2$) when there is a forward simulation (resp., a weak forward simulation) from $\mathcal{A}_1$ to $\mathcal{A}_2$. It is easy to prove that both $\leq_F$ and $\leq_{wF}$ are preorders, that is, reflexive and transitive. Since all simulation relations in this paper are forward simulations, we often omit the word "forward".

Proposition 2.3 *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be nondeterministic automata. Then:*

1. *If $\mathcal{A}_1 \leq_F \mathcal{A}_2$ then $\mathcal{A}_1 \leq_{wF} \mathcal{A}_2$.*
2. *If $H_1 = H_2 = \emptyset$, then $\mathcal{A}_1 \leq_F \mathcal{A}_2$ iff $\mathcal{A}_1 \leq_{wF} \mathcal{A}_2$.*
3. *If $\mathcal{A}_1 \leq_{wF} \mathcal{A}_2$ then $\mathcal{A}_1 \leq_T \mathcal{A}_2$.*

Proof. Standard; for instance, see [26]. □

2.4 Tree-Structured Nondeterministic Automata

A nondeterministic automaton is *tree-structured* if each state is reached via (i.e., occurs as a final state of) a unique execution.

The *unfolding* of nondeterministic automaton $\mathcal{A}$, denoted by *Unfold*($\mathcal{A}$), is the tree-structured nondeterministic automaton $\mathcal{B}$ obtained from $\mathcal{A}$ by unfolding its transition graph into a tree. Formally,

- $Q_{\mathcal{B}} = \mathit{execs}^*(\mathcal{A})$,
- $\bar{q}_{\mathcal{B}} = \bar{q}_{\mathcal{A}}$,

- $E_{\mathcal{B}} = E_{\mathcal{A}}$,
- $H_{\mathcal{B}} = H_{\mathcal{A}}$, and
- $D_{\mathcal{B}} = \{(\alpha, a, \alpha a q) \mid (lstate(\alpha), a, q) \in D_{\mathcal{A}}\}$.

Proposition 2.4 $\mathcal{A} \equiv_F Unfold(\mathcal{A})$.

Proof. See [26]. It is easy to check that the relation R, where $\alpha \; R \; q$ iff $lstate(\alpha) = q$, is a forward simulation from $Unfold(\mathcal{A})$ to $\mathcal{A}$ and that the inverse relation of R is a forward simulation from $\mathcal{A}$ to $Unfold(\mathcal{A})$. □

Proposition 2.5 $\mathcal{A} \equiv_T Unfold(\mathcal{A})$.

Proof. By Proposition 2.4 and Proposition 2.3, Parts 1 and 3. □

3 Definitions and Basic Results for Probabilistic Automata

3.1 Preliminaries and Notation on Measure Theory

We recall a few basic definitions and notation for measure theory that can be retrieved from any standard book on the subject (e.g., [11]).

A *σ-field* over a set X is a set $\mathcal{F} \subseteq 2^X$ that contains the empty set and is closed under complement and countable union. A pair $(X, \mathcal{F})$ where $\mathcal{F}$ is a σ-field over X, is called a *measurable space*. A *measure* on a measurable space $(X, \mathcal{F})$ is a function $\mu : \mathcal{F} \to [0, \infty]$ such that $\mu(\emptyset) = 0$ and μ is countably additive: for each countable family $\{C_i\}_i$ of pairwise disjoint elements of $\mathcal{F}$, $\mu(\cup_i C_i) = \sum_i \mu(C_i)$. A *probability measure* on $(X, \mathcal{F})$ is a measure μ on $(X, \mathcal{F})$ such that $\mu(X) = 1$. A *sub-probability measure* on $(X, \mathcal{F})$ is a measure μ on $(X, \mathcal{F})$ such that $\mu(X) \leq 1$. A *discrete probability measure* on a set X is a probability measure μ on $(X, 2^X)$. A *discrete sub-probability measure* on X is a sub-probability measure μ on $(X, 2^X)$. We denote the set of discrete probability measures and discrete sub-probability measures on X by $Disc(X)$ and $SubDisc(X)$, respectively. We denote the *support* of a discrete measure μ, that is, the set of elements of X that have non-zero measure, by $supp(\mu)$. We let $\delta(q)$ denote the *Dirac measure* for q, the discrete probability measure that assigns probability 1 to $\{q\}$. Finally, if X is nonempty and finite, then $\mathcal{U}(X)$ denotes the *uniform distribution* over X, the discrete measure that assigns probability $1/|X|$ to each element of X. Given two discrete probability measures μ_1, μ_2 on $(X, 2^X)$ and $(Y, 2^Y)$, respectively, we denote by $\mu_1 \times \mu_2$ the *product measure*, that is, the measure on $(X \times Y, 2^{(X \times Y)})$ such that $\mu_1 \times \mu_2((x, y)) = \mu_1(x)\mu_2(y)$ for each $x \in X, y \in Y$.

Sometimes it is useful to know the probability μ of some event C knowing that some other event C' takes place. We call this the measure of C *conditional* on C' and denote it by $\mu(C \mid C')$. Such probability is defined to be 0 if $\mu(C') = 0$ and $\mu(C \cap C')/\mu(C')$ otherwise.

A function $f : X \to Y$ is said to be *measurable* from $(X, \mathcal{F}_X)$ to $(Y, \mathcal{F}_Y)$ if the inverse image of each element of $\mathcal{F}_Y$ is an element of $\mathcal{F}_X$, that is, for each $C \in F_Y$, $f^{-1}(C) \in \mathcal{F}_X$. In such a case, given a measure μ on $(X, \mathcal{F}_X)$, the function $f(\mu)$ defined on $\mathcal{F}_Y$ by $f(\mu)(C) = \mu(f^{-1}(C))$ for each $C \in \mathcal{F}_Y$ is a measure on $(Y, \mathcal{F}_Y)$ and is called the *image measure* of μ under f.

Given a countable collection of measures $\{\mu_i\}_i$ on $(X, \mathcal{F}_X)$ and a countable collection $\{p_i\}_i$ of real numbers in $[0, \infty)$, denote by $\sum_i p_i \mu_i$ a new function μ such that, for each element $C \in \mathcal{F}_X$, $\mu(C) = \sum_i p_i \mu_i(C)$. We state a few elementary properties.

Proposition 3.1 *The following hold.*

1. $\sum_i p_i \mu_i$ *is a measure on* $(X, \mathcal{F}_X)$.
2. *If each* μ_i *is a (sub)-probability measure and* $\sum_i p_i = 1$, *then* $\sum_i p_i \mu_i$ *is a (sub)-probability measure.*
3. *If* f *is a measurable function from* $(X, \mathcal{F}_X)$ *to* $(Y, \mathcal{F}_Y)$, *then* $f(\sum_i p_i \mu_i) = \sum_i p_i f(\mu_i)$.

3.2 Probabilistic Automata, Executions, and Traces

A *probabilistic automaton (PA)* is a tuple $\mathcal{P} = (Q, \bar{q}, E, H, D)$, where all components are exactly as for nondeterministic automata, except that:

- D, the *transition relation*, is a subset of $Q \times (E \cup H) \times Disc(Q)$.

We define A as before. Also, we use the name $\mathcal{P}$ for a generic probabilistic automaton and we refer to its components by writing simply Q, $\bar{q}$, E, H, D, A, and $\xrightarrow{a}$. We extend this convention to allow indices and primes as well; thus, the set of states of a PA $\mathcal{P}'_i$ is denoted by Q'_i. We denote a transition (q, a, μ) by $q \xrightarrow{a} \mu$. We assume finite branching: for each state q the number of pairs (a, μ) such that $q \xrightarrow{a} \mu$ is finite. Given a transition $tr = (q, a, \mu)$ we denote q by $source(tr)$ and μ either by $target(tr)$ or by μ_{tr}.

Thus, a probabilistic automaton differs from a nondeterministic automaton in that a transition leads to a probability measure over states rather than to a single state. A nondeterministic automaton can be viewed as a special case of a probabilistic automaton, where the last component of each transition is a Dirac measure. Conversely, we can associate a nondeterministic automaton with each probabilistic automaton by replacing transition relation D by the relation D' given by

$$(q, a, q') \in D' \quad \Leftrightarrow \quad (\exists \mu)[(q, a, \mu) \in D \wedge \mu(q') > 0].$$

Using this correspondence, notions such as execution fragments and traces carry over from nondeterministic automata to probabilistic automata.[1] For instance, an execution fragment of a PA is simply an execution fragment of its associated nondeterministic automaton. Along the same lines we write $q \xrightarrow{a} q'$ whenever there exists a measure μ such that $q \xrightarrow{a} \mu$ and $q' \in supp(\mu)$.

An execution fragment of a probabilistic automaton is the result of resolving nondeterministic as well as probabilistic choices; however we are interested also in the outcome of the resolution of nondeterministic choices only. We can think of resolving nondeterminism by unfolding the transition relation of a PA and then choosing only one transition at each point. From the formal point of view it is more convenient to define a function, called a *scheduler*, that chooses transitions based on the past history (i.e., the current position in the unfolding of the transition relation).

A *scheduler* for a PA $\mathcal{P}$ is a function $\sigma : frags^*(\mathcal{P}) \rightarrow SubDisc(D)$ such that $tr \in supp(\sigma(\alpha))$ implies $source(tr) = lstate(\alpha)$. A scheduler σ is said to be *deterministic* if for each finite execution fragment α, either $\sigma(\alpha)(D) = 0$ or else $\sigma(\alpha) = \delta(tr)$ (the Dirac measure for tr) for some $tr \in D$. A scheduler σ is *memoryless* if it depends only on the last state of its argument, that is, for each pair α_1, α_2 of finite execution fragments, if $lstate(\alpha_1) = lstate(\alpha_2)$, then $\sigma(\alpha_1) = \sigma(\alpha_2)$.

Informally, $\sigma(\alpha)$ describes the rule for choosing a transition after α has occurred. The rule itself may be randomized. Since $\sigma(\alpha)$ is a sub-probability measure, it is possible that with some non-zero

[1] The correspondence between nondeterministic automata and probabilistic automata is worked out in great detail in [4].

probability no transition is chosen, which corresponds to terminating the computation (with what in nondeterministic automata is called a finite execution fragment). Deterministic schedulers are not allowed to use randomization in their choices, while memoryless schedulers are not allowed to look at the past history in their choices. Deterministic and memoryless schedulers are easier to analyze compared to general schedulers, and several properties (e.g., reachability) can be studied by referring to deterministic memoryless schedulers only. Note that a deterministic memoryless scheduler can be represented alternatively as a partial function from Q to D.

A scheduler σ and a discrete probability measure over states μ induce a measure ϵ on the σ-field generated by cones of execution fragments as follows. If α is a finite execution fragment, then the *cone* of α is defined by $C_\alpha = \{\alpha' \in \mathit{frags}(\mathcal{P}) \mid \alpha \leq \alpha'\}$. The measure ϵ of a cone C_α is defined to be $\mu(q)$ if $\alpha = q$ for some state $q \in Q$, and, if α is of the form $\alpha' a' q'$, it is defined by the recursive equation

$$\epsilon(C_\alpha) \quad = \quad \epsilon(C_{\alpha'}) \sum_{tr \in D(a')} \sigma(\alpha')(tr)\mu_{tr}(q'), \tag{1}$$

where $D(a')$ denotes the set of transitions of D that are labeled by a'. Roughly speaking, the measure of a cone C_α equals the probability of doing α when using σ to resolve nondeterminism. Standard measure theoretical arguments ensure that ϵ is well defined. We call the measure ϵ a *probabilistic execution fragment* of $\mathcal{P}$ and we say that ϵ is *generated* by σ and μ. We also denote by $\epsilon_{\sigma,\mu}$ the probabilistic execution fragment generated by σ and μ.

Proposition 3.2 *Let σ be a scheduler and μ be a discrete probability measure over states. Then $\mathit{fstate}(\epsilon_{\sigma,\mu}) = \mu$.*

Proof. Follows immediately by definition of $\epsilon_{\sigma,\mu}$ after observing that the inverse image under *fstate* of a state q is the set C_q. □

We call the measure $\mathit{fstate}(\epsilon)$ the *first state* of ϵ. If $\mathit{fstate}(\epsilon)$ is the Dirac measure over the start state $\bar{q}$, then ϵ is called a *probabilistic execution*. We often write $\epsilon_{\sigma,q}$ for $\epsilon_{\sigma,\delta(q)}$, and we say that $\epsilon_{\sigma,q}$ is generated by σ and q.

Example 3.1 The cone construction is rich

The cone construction produces a very rich set of measurable events. The event "action a occurs at least once", that is the set of execution fragments where an action a occurs at least once, is measurable since it can be expressed as a union of cones and there are at most countably many cones in a PA. Similarly the event "action a occurs at least n times" is measurable for any natural number n. The event "action a occurs exactly n times" is measurable since it is the intersection of "action a occurs at least n times" with the complement of "action a occurs at least $n+1$ times". Also the event "action a occurs finitely (infinitely) many times" is measurable since it is the countable union (intersection) of "action a occurs exactly (at least) n times". Similar arguments hold for occurrences of states rather than actions.

Any singleton set is measurable since for an infinite execution fragment α the set $\{\alpha\}$ is the intersection of the cones of all its finite prefixes, while for a finite execution fragment α the set $\{\alpha\}$ is the intersection of C_α with the complement of the union of the cones of the extensions of α. Thus, also the set of finite execution fragments is measurable and the set of infinite execution fragments is measurable as well.

Observe that the probability of a finite execution fragment α is the probability that α occurs and then the computation terminates. Thus, the probability of the set of finite execution fragments represents the probability of termination in a probabilistic execution fragment. This leads to the idea that a probabilistic execution fragment should be called *finite* if the probability of the set of finite execution fragments is 1.

We now show how to obtain a probability measure over traces from a probabilistic execution fragment. The measurable space is the pair $(E^* \cup E^\omega, \mathcal{F})$, where $\mathcal{F}$ is the σ-field generated by cones of traces. More precisely, the cone of a finite trace β is defined by $C_\beta = \{\beta' \in E^* \cup E^\omega \mid \beta \leq \beta'\}$, where $\leq$ denotes the prefix ordering on sequences. It is easy to check that the *trace* function is measurable since the inverse image of a cone C_β is a union of cones, specifically those cones C_α such that $\beta \leq \mathit{trace}(\alpha)$, and since there are countably many finite execution fragments in a PA.

Given a probabilistic execution fragment ϵ, we define the *trace distribution* of ϵ, $\mathit{tdist}(\epsilon)$, to be the image measure of ϵ under *trace*. We denote the set of trace distributions of probabilistic executions of a PA $\mathcal{P}$ by $\mathit{tdists}(\mathcal{P})$. We define the *trace distribution preorder* relation on probabilistic automata by: $\mathcal{P}_1 \leq_D \mathcal{P}_2$ iff $E_1 = E_2$ and $\mathit{tdists}(\mathcal{P}_1) \subseteq \mathit{tdists}(\mathcal{P}_2)$.

An example of a measurable set of traces that is used extensively throughout the paper is the set $E^* a (E^* \cup E^\omega)$ of traces in which a specific action a occurs. We denote this set by $\diamond a$. The inverse image under *trace* of $\diamond a$ can be expressed as a disjoint union of cones of executions, namely the cones of the minimal executions with trace in $\diamond a$. Thus, we have the following proposition, whose elementary proof is omitted.

Proposition 3.3 *Let η be the trace distribution of a probabilistic execution ϵ of a probabilistic automaton $\mathcal{P}$, and let Θ_a be the set of finite executions of $\mathcal{P}$ with a single occurrence of action a whose last transition is labeled by a. Then,*

$$\eta(\diamond a) = \sum_{\alpha \in \Theta_a} \epsilon(C_\alpha). \tag{2}$$

3.3 Combined, Weak, and Hyper Transitions

We define three new kinds of transitions that play crucial roles in the paper. Informally, a combined transition is a convex combination of transitions that are labeled by the same action, a weak combined transition abstracts from internal computation and is obtained by performing several, possibly zero, combined transitions, while a hyper-transition is a generalization of combined transitions and weak combined transitions where the starting point is a measure over states rather than a single state.

3.3.1 Combined Transitions

Let $\{q \xrightarrow{a} \mu_i\}_{i \in I}$ be a collection of transitions of a PA $\mathcal{P}$, and let $\{p_i\}_{i \in I}$ be a collection of probabilities such that $\sum_{i \in I} p_i = 1$. Then the triple $(q, a, \sum_{i \in I} p_i \mu_i)$ is called a *combined transition* of $\mathcal{P}$.

3.3.2 Weak Transitions

Consider a probabilistic execution fragment ϵ of a PA $\mathcal{P}$, with first state $\delta(q)$, that assigns probability 1 to the set of all finite execution fragments with trace $\mathit{trace}(\beta)$ for some $\beta \in A^*$. Let μ be the discrete measure on Q defined by $\mu(q') = \epsilon(\{\alpha \mid \mathit{lstate}(\alpha) = q'\})$. Then $q \stackrel{\beta}{\Longrightarrow} \mu$ is a *weak*

combined transition of $\mathcal{P}$. We refer to ϵ as a *representation* of $q \stackrel{\beta}{\Longrightarrow} \mu$. Observe that the measure μ can be seen alternatively as the image measure of ϵ under *lstate*. This is an abuse of notation because *lstate* is not defined for infinite execution fragments; however, since ϵ assigns measure 1 to the set of finite execution fragments, we can extend the definition of *lstate* to infinite execution fragments for this purpose: for instance, we define the last state of any infinite execution fragment to be $\bar{q}$.

The notion of weak combined transition that we have just defined for probabilistic automata is a conservative extension of the corresponding notion defined for nondeterministic automata. Indeed, it is routine to check that whenever $q \stackrel{\beta}{\Longrightarrow} q'$ is a weak transition of a nondeterministic automaton $\mathcal{A}$, then $q \stackrel{\beta}{\Longrightarrow} \delta(q')$ is a weak combined transition of $\mathcal{A}$ viewed as a probabilistic automaton.

Proposition 3.4 *Let $\{tr_i = (q, a, \mu_i)\}_{i \in I}$ be a collection of weak combined transitions of a PA $\mathcal{P}$, and let $\{p_i\}_{i \in I}$ be probabilities such that $\sum_{i \in I} p_i = 1$. Then $(q, a, \sum_{i \in I} p_i \mu_i)$, written $\sum_{i \in I} p_i tr_i$, is a weak combined transition of $\mathcal{P}$.*

Proof. For each $i \in I$, let ϵ_i be a representation of tr_i, and σ_i be a scheduler that, together with state q, induces ϵ_i. We omit the index set I in the rest of the proof. For each finite execution fragment α, let $N(\alpha) \stackrel{\Delta}{=} \sum_i p_i \epsilon_i(C_\alpha)$. Define a new scheduler σ as follows.

$$\sigma(\alpha) = \begin{cases} \displaystyle\sum_i \frac{p_i \epsilon_i(C_\alpha)}{N(\alpha)} \sigma_i(\alpha) & \text{if } N(\alpha) > 0, \\ \text{arbitrarily} & \text{otherwise.} \end{cases}$$

Informally, the weight that $\sigma(\alpha)$ gives to the choice $\sigma_i(\alpha)$ is the normalized probability with which σ_i contributes to the generation of α. Let ϵ be the probabilistic execution fragment induced by σ and q. Let α be a finite execution fragment of $\mathcal{P}$. We first prove by induction on the length of α that $\epsilon(C_\alpha) = N(\alpha)$. The base case is trivial since $\epsilon(C_q) = 1$ and for each i, $\epsilon_i(C_q) = 1$, which implies $N(q) = \sum_i p_i \epsilon_i(C_q) = 1$; similarly, for each state $q' \neq q$, $\epsilon(C_{q'}) = 0$ and for each i, $\epsilon_i(C_{q'}) = 0$. For the inductive step, let $\alpha = \alpha' a' q'$. If $\epsilon(C_{\alpha'}) = 0$, then, by induction, $N(\alpha') = \sum_i p_i \epsilon_i(C_{\alpha'}) = 0$, which implies that for each i, $p_i \epsilon_i(C_{\alpha'}) = 0$. By definition of measure of a cone, Equation (1), $\epsilon(C_\alpha) = 0$. Furthermore, for each i, if $p_i = 0$ then $p_i \epsilon_i(C_\alpha) = 0$ trivially, and if $p_i > 0$, then $\epsilon_i(C_{\alpha'}) = 0$ and by definition of measure of a cone, Equation (1), $\epsilon_i(C_\alpha) = 0$, which implies $p_i \epsilon_i(C_\alpha) = 0$. Thus, $N(\alpha) = 0$ as needed. If $\epsilon(C_{\alpha'}) > 0$, then, by definition of measure of a cone, Equation (1),

$$\epsilon(C_\alpha) = \epsilon(C_{\alpha'}) \sum_{tr \in D(a')} \sigma(\alpha')(tr) \mu_{tr}(q').$$

By induction hypothesis, $N(\alpha') > 0$. Thus, by expanding $\sigma(\alpha')(tr)$ with the definition of σ we obtain

$$\epsilon(C_\alpha) = \epsilon(C_{\alpha'}) \sum_{tr \in D(a')} \left(\sum_i \frac{p_i \epsilon_i(C_{\alpha'})}{N(\alpha')} \sigma_i(\alpha')(tr) \right) \mu_{tr}(q').$$

By standard algebraic manipulations (exchanges of sums and rearrangements of constants) we obtain

$$\epsilon(C_\alpha) = \frac{\epsilon(C_{\alpha'})}{N(\alpha')} \sum_i \sum_{tr \in D(a')} p_i \epsilon_i(C_{\alpha'}) \sigma_i(\alpha')(tr) \mu_{tr}(q').$$

By induction, $\epsilon(C_{\alpha'}) = N(\alpha')$. Thus, by simplifying (removing) the leftmost term and rearranging constants we obtain

$$\epsilon(C_\alpha) = \sum_i p_i \left(\epsilon_i(C_{\alpha'}) \sum_{tr \in D(a')} \sigma_i(\alpha')(tr)\mu_{tr}(q') \right).$$

Finally, by definition of measure of a cone, Equation (1), we get the desired equation

$$\epsilon(C_\alpha) = N(\alpha) = \sum_i p_i \epsilon_i(C_\alpha).$$

Thus, $\epsilon = \sum_i p_i \epsilon_i$. Since each ϵ_i assigns probability 1 to the set of finite execution fragments of $\mathcal{P}$ with trace $trace(a)$, then so does ϵ. Furthermore, by Proposition 3.1, Item 3, $lstate(\epsilon) = \sum_i p_i\, lstate(\epsilon_i)$. That is, ϵ is a representation of a weak combined transition $(q, a, \sum_i p_i\, lstate(\epsilon_i))$, which, since $lstate(\epsilon_i) = \mu_i$, is the triplet $(q, a, \sum_i p_i\, \mu_i)$. Hence, $\sum_i p_i tr_i$ is a weak combined transition of $\mathcal{P}$. □

3.3.3 Hyper-Transitions

Let $\mathcal{P}$ be a PA with $a \in A$, and let $\mu \in Disc(Q)$. For each $q \in supp(\mu)$, suppose $q \xrightarrow{a} \mu_q$ is a combined transition of $\mathcal{P}$. Let μ' be $\sum_{q \in supp(\mu)} \mu(q)\mu_q$. Then $\mu \xrightarrow{a} \mu'$ is called a *hyper-transition* of $\mathcal{P}$. Also, let $\beta \in A^*$, and for each $q \in supp(\mu)$, suppose $q \stackrel{\beta}{\Longrightarrow} \mu_q$ is a weak combined transition of $\mathcal{P}$. Let μ' be $\sum_{q \in supp(\mu)} \mu(q)\mu_q$. Then $\mu \stackrel{\beta}{\Longrightarrow} \mu'$ is called a *weak hyper-transition* of $\mathcal{P}$.

We prove now two technical properties of weak hyper-transitions. The first property gives an alternative definition of weak hyper-transition and is used to prove the second property; the second property states that weak hyper-transitions can be concatenated. It will be used in Section 6.2.

Proposition 3.5 *There is a weak hyper-transition $\mu \stackrel{\beta}{\Longrightarrow} \mu'$ iff there is a scheduler σ such that $\epsilon_{\sigma,\mu}$ assigns probability 1 to the set of finite execution fragments with trace β, and $lstate(\epsilon_{\sigma,\mu}) = \mu'$. We say that $\epsilon_{\sigma,\mu}$ represents $\mu \stackrel{\beta}{\Longrightarrow} \mu'$.*

Proof. Let $\{q_i\}_I$ be an enumeration of the states in $supp(\mu)$. We prove the two implications separately.

⇒ For each i let σ_i be a scheduler such that ϵ_{σ_i,q_i} represents $q_i \stackrel{\beta}{\Longrightarrow} \mu_i$ and $\mu' = \sum_I \mu(q_i)\mu_i$. Let σ be a new scheduler defined as follows.

$$\sigma(\alpha) = \begin{cases} \sigma_i(\alpha) & \text{if } fstate(\alpha) = q_i \text{ for some } i \in I, \\ 0 & \text{otherwise.} \end{cases}$$

We prove that $\epsilon_{\sigma,\mu} = \sum_i \mu(q_i)\epsilon_{\sigma_i,q_i}$ by showing that $\epsilon_{\sigma,\mu}(C_\alpha) = \sum_i \mu(q_i)\epsilon_{\sigma_i,q_i}(C_\alpha)$ for each finite execution fragment α. The proof is by induction on the length of α. For the base case, let $\alpha = q$ for some state q. By definition of measure of a cone, $\epsilon_{\sigma,\mu}(C_\alpha) = \mu(q)$ and, for each i, $\epsilon_{\sigma_i,q_i}(C_\alpha)$ is 1 if $q = q_i$ and 0 otherwise. Thus, $\epsilon_{\sigma,\mu}(C_q) = \sum_i \mu(q_i)\epsilon_{\sigma_i,q_i}(C_q)$ trivially. For the inductive step, let α be $\alpha' aq$. If $fstate(\alpha') \notin supp(\mu)$, then trivially $\epsilon_{\sigma,\mu}(C_\alpha) = 0$ and, for each i, $\epsilon_{\sigma_i,q_i}(C_\alpha) = 0$. Thus, $\epsilon_{\sigma,\mu}(C_\alpha) = \sum_i \mu(q_i)\epsilon_{\sigma_i,q_i}(C_\alpha)$. If $fstate(\alpha') \in supp(\mu)$, then let j be the index of $fstate(\alpha')$. By definition of measure of a cone, $\epsilon_{\sigma,\mu}(C_\alpha) = \epsilon_{\sigma,\mu}(C_{\alpha'}) \sum_{tr \in D(a)} \sigma(\alpha')(tr)\mu_{tr}(q)$. By induction and definition of σ,

$\epsilon_{\sigma,\mu}(C_\alpha) = \sum_i \mu(i)\epsilon_{\sigma_i,q_i}(C_{\alpha'}) \sum_{tr\in D(a)} \sigma_j(\alpha')(tr)\mu_{tr}(q)$. Since only $\epsilon_{\sigma_j,q_j}(C_{\alpha'})$ may be different from 0, we get $\epsilon_{\sigma,\mu}(C_\alpha) = \mu(q_j)\epsilon_{\sigma_j,q_j}(C_{\alpha'}) \sum_{tr\in D(a)} \sigma_j(\alpha')(tr)\mu_{tr}(q) = \mu(q_j)\epsilon_{\sigma_j,q_j}(C_\alpha)$. For the same reason, $\sum_i \mu(q_i)\epsilon_{\sigma_i,q_i}(C_\alpha) = \mu(q_j)\epsilon_{\sigma_j,q_j}(C_\alpha)$. Thus, $\epsilon_{\sigma,\mu}(C_\alpha) = \sum_i \mu(i)\epsilon_{\sigma_i,q_i}(C_\alpha)$ as needed.

Since each ϵ_{σ_i,q_i} assigns probability 1 to the set of finite execution fragments with trace β, then also $\epsilon_{\sigma,\mu}$ assigns probability 1 to the set of finite execution fragments with trace β. Furthermore, by Item 3 of Proposition 3.1, $lstate(\epsilon_{\sigma,\mu}) = \sum_i \mu(q_i) lstate(\epsilon_{\sigma_i,q_i}) = \mu'$. Thus, $lstate(\epsilon_{\sigma,\mu})$ represents $\mu \stackrel{\beta}{\Longrightarrow} \mu'$.

$\Leftarrow$ Let σ be a scheduler that represents $\mu \stackrel{\beta}{\Longrightarrow} \mu'$. For each i, let $\sigma_i = \sigma$. Observe that for each finite execution fragment α, $\epsilon_{\sigma,\mu}(C_\alpha) = \mu(fstate(\alpha))\epsilon_{\sigma,fstate(\alpha)}(C_\alpha)$. Thus, as in the previous case, $\epsilon_{\sigma,\mu} = \sum_i \mu(i)\epsilon_{\sigma_i,q_i}$. Let $q_i \stackrel{\beta}{\Longrightarrow} \mu_i$ be the weak transition represented by ϵ_{σ_i,q_i}. Since $\mu' = lstate(\epsilon_{\sigma,\mu})$ and for each i, $\mu_i = lstate(\epsilon_{\sigma_i,q_i})$, by Item 3 of Proposition 3.1, $\mu' = \sum_i \mu(q_i)\mu_i$. This suffices.

□

Proposition 3.6 *Suppose that $\mu_1 \stackrel{\beta_1}{\Longrightarrow} \mu_2$ and $\mu_2 \stackrel{\beta_2}{\Longrightarrow} \mu_3$ are weak hyper-transitions of a PA $\mathcal{P}$. Then $\mu_1 \stackrel{\beta_1\beta_2}{\Longrightarrow} \mu_3$ is a weak hyper-transition of $\mathcal{P}$.*

Proof. Let ϵ_1 and ϵ_2 be the probabilistic execution fragments that represent $\mu_1 \stackrel{\beta_1}{\Longrightarrow} \mu_2$ and $\mu_2 \stackrel{\beta_2}{\Longrightarrow} \mu_3$, respectively, and let σ_1 and σ_2 be the schedulers that generate ϵ_1 and ϵ_2, respectively. Let $N(\alpha) \stackrel{\Delta}{=} \epsilon_1(C_\alpha) + \sum_{\alpha'<\alpha} \epsilon_1(\alpha')\epsilon_2(C_{\alpha\triangleright\alpha'} \mid C_{\alpha'\triangleright\alpha'})$, where we recall that $\epsilon_2(C_{\alpha\triangleright\alpha'} \mid C_{\alpha'\triangleright\alpha'})$ denotes the probability of $C_{\alpha\triangleright\alpha'}$ conditional on $C_{\alpha'\triangleright\alpha'}$. Define a new scheduler σ as follows.

$$\sigma(\alpha) = \begin{cases} \frac{\epsilon_1(C_\alpha)\sigma_1(\alpha) + \sum_{\alpha'\leq\alpha} \epsilon_1(\alpha')\epsilon_2(C_{\alpha\triangleright\alpha'} | C_{\alpha'\triangleright\alpha'})\sigma_2(\alpha\triangleright\alpha')}{N(\alpha)} & \text{if } N(\alpha) > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Informally, σ is a scheduler that represents a concatenation of ϵ_1 and ϵ_2. A finite execution fragment α can be reached in the concatenation of ϵ_1 and ϵ_2 either because ϵ_1 reaches it, or because ϵ_1 terminates at a prefix of α from which ϵ_2 continues. The scheduler σ from α should behave according to σ_1 whenever α is reached in ϵ_1, and according to σ_2, with the appropriate argument, whenever α is reached in ϵ_2. The schedules from α must be weighted by the probabilities with which ϵ_1 and ϵ_2 lead to α. The term $N(\alpha)$ is a normalization factor whose computation is just technical. In the formal proof we have to show that σ is indeed a scheduler (in other words the value of N is correct), and that σ generates the representation of $\mu_1 \stackrel{\beta_1\beta_2}{\Longrightarrow} \mu_3$. These proofs are mainly detailed algebraic manipulations. An interesting point of the proof is Equation (4) that expresses the probability of a finite execution fragment in the concatenation of ϵ_1 and ϵ_2 in terms of the probabilities of finite execution fragments of ϵ_1 and ϵ_2.

We show that σ is a scheduler. That is, for each finite execution fragment α, $\sigma(\alpha)(D) \leq 1$, or equivalently, $\epsilon_1(C_\alpha)\sigma_1(\alpha)(D) + \sum_{\alpha'\leq\alpha} \epsilon_1(\alpha')\epsilon_2(C_{\alpha\triangleright\alpha'} \mid C_{\alpha'\triangleright\alpha'})\sigma_2(\alpha \triangleright \alpha')(D) \leq N(\alpha)$, where the left term is the numerator of the definition of σ applied to D. Since σ_2 is a scheduler, we know that $\sigma_2(\alpha \triangleright \alpha')(D) \leq 1$. Also, observe that $\epsilon_1(C_\alpha)\sigma_1(\alpha)(D) = \epsilon_1(C_\alpha - \{\alpha\})$. Thus, it suffices to show that $\epsilon_1(C_\alpha - \{\alpha\}) + \sum_{\alpha'\leq\alpha} \epsilon_1(\alpha')\epsilon_2(C_{\alpha\triangleright\alpha'} \mid C_{\alpha'\triangleright\alpha'}) \leq N(\alpha)$. We separate from the sum the term with $\alpha' = \alpha$ and observe that $\epsilon_2(C_{\alpha\triangleright\alpha} \mid C_{\alpha\triangleright\alpha}) \leq 1$. The new inequality, that suffices for our purposes, is $\epsilon_1(C_\alpha - \{\alpha\}) + \epsilon_1(\{\alpha\}) + \sum_{\alpha'<\alpha} \epsilon_1(\alpha')\epsilon_2(C_{\alpha\triangleright\alpha'} \mid C_{\alpha'\triangleright\alpha'}) \leq N(\alpha)$. However, since $\epsilon_1(C_\alpha - \{\alpha\}) + \epsilon_1(\{\alpha\}) = \epsilon_1(C_\alpha)$, the inequality above is $N(\alpha) \leq N(\alpha)$ which is trivially true.

Let ϵ be the probabilistic execution fragment generated by σ and μ_1. We show that ϵ represents $\mu_1 \stackrel{\beta_1\beta_2}{\Longrightarrow} \mu_3$ in three steps. First we show by induction on the length of a finite execution fragment α that $\epsilon(C_\alpha) = N(\alpha)$. The base case is trivial since $N(q) = \epsilon_1(C_q)$ by definition of N, and $\epsilon_1(C_q) = \epsilon(C_q) = \mu_1(q)$ by definition of ϵ_1 and ϵ since both measures are generated by μ_1. For the inductive step, let $\alpha = \alpha' a' q'$. If $\epsilon(C_{\alpha'}) = 0$, then by definition of measure of a cone, Equation (1), $\epsilon(C_\alpha) = 0$. We show that $N(\alpha) = 0$ as well. By induction, since $\epsilon(C_{\alpha'}) = 0$, $N(\alpha') = 0$. By definition of N, $\epsilon_1(C_{\alpha'}) = 0$, and $\epsilon_1(\alpha'')\epsilon_2(C_{\alpha' \triangleright \alpha''} \mid C_{\alpha'' \triangleright \alpha''}) = 0$ for each $\alpha'' < \alpha'$. Then, by definition of conditional measure and of measure of a cone, $\epsilon_1(C_\alpha) = 0$ and $\epsilon_1(\alpha'')\epsilon_2(C_{\alpha \triangleright \alpha''} \mid C_{\alpha'' \triangleright \alpha''}) = 0$ for each $\alpha'' < \alpha'$. By $\epsilon_1(C_{\alpha'}) = 0$, also $\epsilon_1(\alpha') = 0$, and hence $\epsilon_1(\alpha')\epsilon_2(C_{\alpha \triangleright \alpha'} \mid C_{\alpha' \triangleright \alpha'}) = 0$. We have shown that all the terms of the definition of $N(\alpha)$ are 0, and thus $N(\alpha) = 0$ as needed. If $\epsilon(C_{\alpha'}) > 0$, then by expanding σ with its definition in Equation (1), the definition of measure of a cone, we get

$$\epsilon(C_\alpha) = \epsilon(C_{\alpha'}) \sum_{tr \in D(a)} \frac{\epsilon_1(C_{\alpha'})\sigma_1(\alpha')(tr) + \sum_{\alpha'' \leq \alpha'} \epsilon_1(\alpha'')\epsilon_2(C_{\alpha' \triangleright \alpha''} \mid C_{\alpha'' \triangleright \alpha''})\sigma_2(\alpha' \triangleright \alpha'')(tr)}{N(\alpha')} \mu_{tr}(q').$$

By induction, $\epsilon(C_{\alpha'}) = N(\alpha')$, and thus the two terms can be simplified in the equation above. Then, by rearranging terms algebraically, we get

$$\begin{aligned} \epsilon(C_\alpha) \quad = \quad & \sum_{tr \in D(a)} \epsilon_1(C_{\alpha'})\sigma_1(\alpha')(tr)\mu_{tr}(q') + \\ & \sum_{\alpha'' \leq \alpha'} \epsilon_1(\alpha'') \sum_{tr \in D(a)} \epsilon_2(C_{\alpha' \triangleright \alpha''} \mid C_{\alpha'' \triangleright \alpha''})\sigma_2(\alpha' \triangleright \alpha'')(tr)\mu_{tr}(q'). \end{aligned}$$

By definition of measure of a cone, the the first term above is $\epsilon_1(C_\alpha)$. With a similar argument, after applying the definition of conditional measure and distinguishing the cases where $\epsilon_2(C_{\alpha'' \triangleright \alpha''}) = 0$, the second term above is $\sum_{\alpha'' \leq \alpha'} \epsilon_1(\alpha'')\epsilon_2(C_{\alpha \triangleright \alpha''} \mid C_{\alpha'' \triangleright \alpha''})$. Thus, we get

$$\epsilon(C_\alpha) = \epsilon_1(C_\alpha) + \sum_{\alpha'' \leq \alpha'} \epsilon_1(\alpha'')\epsilon_2(C_{\alpha \triangleright \alpha''} \mid C_{\alpha'' \triangleright \alpha''}).$$

Then it is enough to observe that the right-hand side of the equation above is $N(\alpha)$ since $\alpha'' \leq \alpha'$ iff $\alpha'' < \alpha$.

Second we show that for each finite execution fragment α,

$$\epsilon(\alpha) = \sum_{\alpha' \leq \alpha} \epsilon_1(\alpha')\epsilon_2(\alpha \triangleright \alpha' \mid C_{\alpha' \triangleright \alpha'}). \tag{4}$$

Observe that, by definition of the cone σ-field, $\epsilon(\alpha) = \epsilon(C_\alpha) - \sum_{a \in A, q \in Q} \epsilon(C_{\alpha a q})$. By replacing the ϵ measures of cones with the definition of N in the equation above, we get

$$\begin{aligned} \epsilon(\alpha) \quad = \quad & \epsilon_1(C_\alpha) + \sum_{\alpha' < \alpha} \epsilon_1(\alpha')\epsilon_2(C_{\alpha \triangleright \alpha'} \mid C_{\alpha' \triangleright \alpha'}) - \\ & \sum_{a \in A, q \in Q} \left(\epsilon_1(C_{\alpha a q}) + \sum_{\alpha' < \alpha a q} \epsilon_1(\alpha')\epsilon_2(C_{\alpha a q \triangleright \alpha'} \mid C_{\alpha' \triangleright \alpha'}) \right). \end{aligned}$$

We now use the terms $\epsilon_1(C_\alpha)$ and $-\sum_{a \in A, q \in Q} \epsilon_1(C_{\alpha a q})$ in the equation above to derive $\epsilon_1(\alpha)$, and similarly we use part of the other two terms with to derive the following.

$$\epsilon(\alpha) = \epsilon_1(\alpha) + \sum_{\alpha' < \alpha} \epsilon_1(\alpha')\epsilon_2(\alpha \triangleright \alpha' \mid C_{\alpha' \triangleright \alpha'}) - \sum_{a \in A, q \in Q} \epsilon_1(\alpha)\epsilon_2(C_{\alpha a q \triangleright \alpha} \mid C_{\alpha \triangleright \alpha}).$$

Observe that the third term in the equation above is $\epsilon_1(\alpha)\epsilon_2((C_{\alpha\triangleright\alpha} - \{\alpha \triangleright \alpha\}) \mid C_{\alpha\triangleright\alpha})$. Thus, by adding and subtracting the term $\epsilon_1(\alpha)\epsilon_2(\alpha \triangleright \alpha \mid C_{\alpha\triangleright\alpha})$ and rearranging algebraically, we get

$$\epsilon(\alpha) = \sum_{\alpha' \leq \alpha} \epsilon_1(\alpha')\epsilon_2(\alpha \triangleright \alpha' \mid C_{\alpha'\triangleright\alpha'}) + \epsilon_1(\alpha)\left(1 - \epsilon_2(C_{\alpha\triangleright\alpha} \mid C_{\alpha\triangleright\alpha})\right).$$

If $\epsilon_1(\alpha) = 0$, then Equation (4) follows trivially. If $\epsilon_1(\alpha) > 0$, then, since ϵ_1 represents $\mu_1 \stackrel{\beta_1}{\Longrightarrow} \mu_2$, $\mu_2(q) > 0$, where q is $lstate(\alpha)$. Since ϵ_2 is generated by σ_2 and μ_2, $\epsilon_2(C_q) = \mu_2(q) > 0$. By definition of $\triangleright$, $\alpha \triangleright \alpha = q$, and thus $\epsilon_2(C_{\alpha\triangleright\alpha}) > 0$. By definition of conditional measure, $\epsilon_2(C_{\alpha\triangleright\alpha} \mid C_{\alpha\triangleright\alpha}) = 1$, which leads again to Equation (4).

Third we show that ϵ represents $\mu_1 \stackrel{\beta_1\beta_2}{\Longrightarrow} \mu_3$. Let α be such that $\epsilon(\alpha) > 0$. By Equation (4) there exists a prefix α' of α such that $\epsilon_1(\alpha') > 0$ and $\epsilon_2(\alpha' \triangleright \alpha) > 0$. Then $trace(\alpha') = \beta_1$ and $trace(\alpha \triangleright \alpha') = \beta_2$. Thus, $trace(\alpha) = \beta_1\beta_2$. Let μ be $lstate(\epsilon)$. We are left to show that $\mu = \mu_3$. Consider a state q of Q. By definition of image measure and Equation (4), $\mu(q) = \sum_{\alpha | lstate(\alpha)=q} \sum_{\alpha' \leq \alpha} \epsilon_1(\alpha')\epsilon_2(\alpha \triangleright \alpha' \mid C_{\alpha'\triangleright\alpha'})$. The sum above can be restructured as follows: $\mu(q) = \sum_{\alpha'} \sum_{\alpha | fstate(\alpha)=lstate(\alpha'), lstate(\alpha)=q} \epsilon_1(\alpha')\epsilon_2(\alpha \mid C_{\alpha'\triangleright\alpha'})$. We partition further the sums over $lstate(\alpha')$, thus getting $\mu(q) = \sum_{q' \in Q} \sum_{\alpha' | lstate(\alpha')=q'} \sum_{\alpha | fstate(\alpha)=q', lstate(\alpha)=q} \epsilon_1(\alpha')\epsilon_2(\alpha \mid C_{q'})$. Observe that, since $\mu_2 = lstate(\epsilon_1)$, if $\mu_2(q') = 0$, then there is no α' with last state q' such that $\epsilon_1(\alpha') > 0$. Thus we can restrict the first sum to those q' such that $\mu_2(q') > 0$. Also, if $\mu_2(q') > 0$, we have already concluded before that $\epsilon_2(C_{\alpha'\triangleright\alpha'}) = \mu_2(q')$. Thus, we get $\mu(q) = \sum_{q' \in Q | \mu_2(q') > 0} \sum_{\alpha' | lstate(\alpha')=q'} \sum_{\alpha | fstate(\alpha)=q', lstate(\alpha)=q} \epsilon_1(\alpha')\epsilon_2(\alpha)/\mu_2(q')$. Observe that the two inner sums can be exchanged. By definition of μ_2, $\sum_{\alpha' | lstate(\alpha')=q'} \epsilon_1(\alpha') = \mu_2(q')$. Thus, we get $\mu(q) = \sum_{q' \in Q | \mu_2(q') > 0} \sum_{\alpha | fstate(\alpha)=q', lstate(\alpha)=q} \epsilon_2(\alpha)$. Following the same argument that we used to restrict the sum over q', we can remove such restriction, and thus we can remove the most external sum, leading to $\mu(q) = \sum_{\alpha | lstate(\alpha)=q} \epsilon_2(\alpha)$, which is the definition of $\mu_3(q)$. □

3.4 Composition

Two PAs, $\mathcal{P}_1$ and $\mathcal{P}_2$, are *compatible* if $H_1 \cap A_2 = A_1 \cap H_2 = \emptyset$. The *(parallel) composition* of two compatible PAs $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted by $\mathcal{P}_1 \| \mathcal{P}_2$, is the PA $\mathcal{P} = (Q_1 \times Q_2, (\bar{q}_1, \bar{q}_2), E_1 \cup E_2, H_1 \cup H_2, D)$ where D is the set of triples $(q, a, \mu_1 \times \mu_2)$ such that, for $i \in \{1, 2\}$:

$$a \in A_i \Rightarrow (\pi_i(q), a, \mu_i) \in D_i \text{ and } a \notin A_i \Rightarrow \mu_i = \delta(\pi_i(q)).$$

Let ϵ be a probabilistic execution (fragment) of $\mathcal{P}_1 \| \mathcal{P}_2$ and let $i \in \{1, 2\}$. Define $\pi_i(\epsilon)$, the i^{th} projection of ϵ, to be the image measure under π_i of ϵ. It is easy to verify that the projection function is measurable. When convenient, we denote a projection by $\epsilon \lceil \mathcal{P}_i$, where $\mathcal{P}_i$ is the PA that appears in the i^{th} position.

Proposition 3.7 *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be compatible PAs and let ϵ be a probabilistic execution (fragment) of $\mathcal{P}_1 \| \mathcal{P}_2$. Then for each $i \in \{1, 2\}$, $\pi_i(\epsilon)$ is a probabilistic execution (fragment) of $\mathcal{P}_i$.*

Proof. By Propositions 4.3.4 and 4.3.5 of [32]. □

The trace distribution preorder is not preserved by composition [34, 37] as is shown by the following example.

Example 3.2 Failure of compositionality

Consider the two (nondeterministic) automata $\mathcal{P}_1$ and $\mathcal{P}_2$ of Figure 1. The two automata

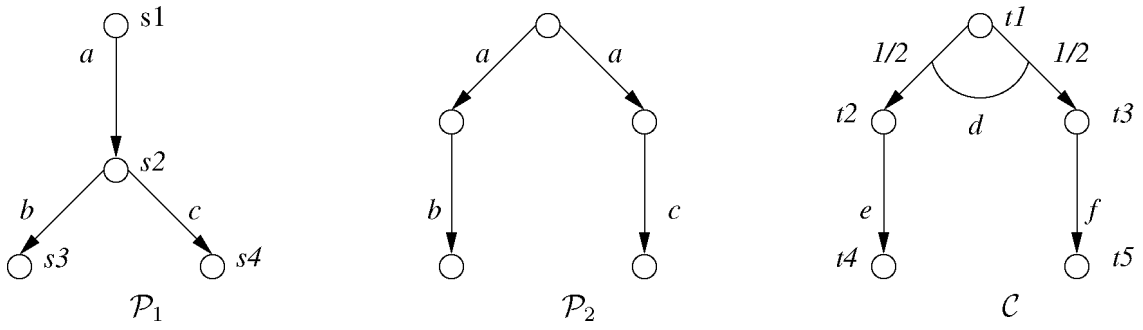

Figure 1: Trace distribution inclusion is not preserved by composition (without communication).

are trace equivalent, and it is easy to see that they are also trace distribution equivalent. Now consider the compositions $\mathcal{P}_1 \| \mathcal{C}$ and $\mathcal{P}_2 \| \mathcal{C}$, where $\mathcal{C}$ is the probabilistic automaton of Figure 1 and we assume that the actions of $\mathcal{C}$ are not shared with $\mathcal{P}_1$ and $\mathcal{P}_2$. It is possible to build a probabilistic execution of $\mathcal{P}_1 \| \mathcal{C}$ as follows: first a is scheduled followed by d; then e or f is scheduled depending on the outcome state of the transition labeled by d; finally, b or c is scheduled depending on whether e or f was scheduled. Formally, we consider the probabilistic execution induced by the deterministic memoryless scheduler specified by the following partial function, where a transition is denoted by the unique action labeling it:

Q_1	$Q_{\mathcal{C}}$	$D_{\mathcal{P}_1 \| \mathcal{C}}$
s_1	t_1	a
s_2	t_1	d
s_2	t_2	e
s_2	t_3	f
s_2	t_4	b
s_2	t_5	c

Thus, in the resulting trace distribution there is a total correlation between e, b and f, c, respectively. The same trace distribution cannot be obtained from $\mathcal{P}_2 \| \mathcal{C}$ because after scheduling the transition labeled by a we are already bound to b or c, and thus the occurrence of b or c cannot be correlated to e or f in this case.

Example 3.2 may appear pathological since, in the probabilistic execution of $\mathcal{P}_1 \| \mathcal{C}$ that correlates the choices between e and f and between b and c, a nondeterministic choice of $\mathcal{P}_1$ is resolved based on information that is not available to $\mathcal{P}_1$. This may lead us to propose a naive solution to the non-preservation of trace distribution inclusion by parallel composition where we require that each probabilistic automaton in a parallel composition can resolve its nondeterministic choices based on local knowledge only. However, a more elaborate example shows that this naive idea also does not work.

Example 3.3 Failure of compositionality

Consider the two automata $\mathcal{P}_1$ and $\mathcal{P}_2$ of Figure 2, which are essentially the automata of Example 3.2 where self-loop transitions labeled by e and f are added to each state. In this case the context $\mathcal{C}$ synchronizes with $\mathcal{P}_1$ and $\mathcal{P}_2$ on actions e and f, and $\mathcal{P}_1$ is

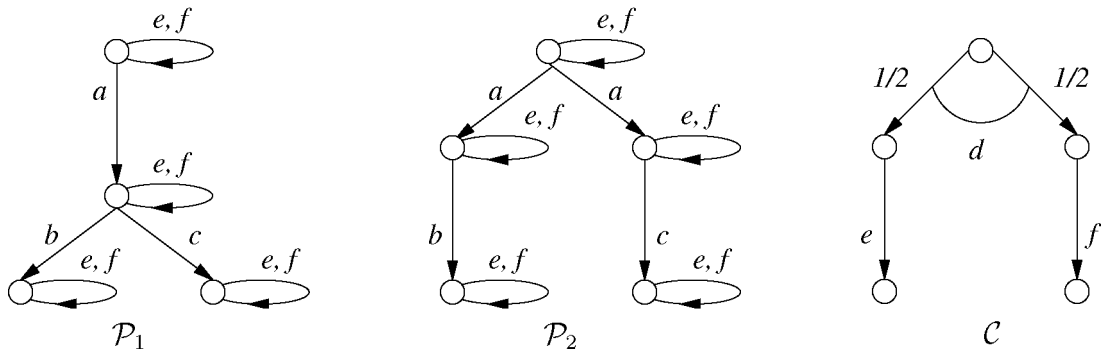

Figure 2: Trace distribution inclusion is not preserved by composition (with communication).

able to learn which of e or f occurs, thus determining the correlation with b and c based on local knowledge only.

The solution of resolving nondeterminism based on local knowledge is adopted in [10] for a probabilistic extension of reactive modules; however the idea of [10] cannot be extended easily to probabilistic automata because of key structural differences in the models: in probabilistic automata there is a total interleaving of the transitions taken by different probabilistic automata in a parallel composition, while in probabilistic reactive modules there are several independent *atoms* that are not forced to interleave. A direct adaptation of the idea of [10] to probabilistic automata would require drastic modifications of the model that go beyond the scope of this paper: transitions would have to be labeled by sets of actions and be structured in such a way that each action affects different parts of the state.

An alternative approach, followed in [32] and adopted in this paper, consists of defining a new *trace distribution precongruence* relation, denoted by $\leq_{DC}$, as the coarsest precongruence (for parallel composition) that is included in the trace distribution preorder $\leq_D$, and finding alternative characterizations of $\leq_{DC}$. It is known from [32] that there exists a simple context, called the *principal context*, that is sufficiently powerful to distinguish all probabilistic automata that are not in the trace distribution precongruence relation; alternatively, a testing scenario is proposed in [33].

In this paper we characterize $\leq_{DC}$ in terms of probabilistic simulation relations. Another simple alternative characterization of $\leq_{DC}$ that is useful for our study is given by the following proposition.

Proposition 3.8 *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be PAs. Then $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ iff for every PA $\mathcal{C}$ that is compatible with both $\mathcal{P}_1$ and $\mathcal{P}_2$, $\mathcal{P}_1 \| \mathcal{C} \leq_D \mathcal{P}_2 \| \mathcal{C}$.*

Proof. Define relation $\sqsubseteq$ such that $\mathcal{P}_1 \sqsubseteq \mathcal{P}_2$ iff for every PA $\mathcal{C}$ that is compatible with both $\mathcal{P}_1$ and $\mathcal{P}_2$, $\mathcal{P}_1 \| \mathcal{C} \leq_D \mathcal{P}_2 \| \mathcal{C}$.

Let $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ and let $\mathcal{C}$ be a PA compatible with both $\mathcal{P}_1$ and $\mathcal{P}_2$. Since $\leq_{DC}$ is a precongruence by definition, then $\mathcal{P}_1 \| \mathcal{C} \leq_{DC} \mathcal{P}_2 \| \mathcal{C}$. Since, again by definition, $\leq_{DC}$ is included in $\leq_D$, then $\mathcal{P}_1 \| \mathcal{C} \leq_D \mathcal{P}_2 \| \mathcal{C}$. Thus, $\mathcal{P}_1 \sqsubseteq \mathcal{P}_2$, which implies that $\leq_{DC}$ is included in $\sqsubseteq$.

Conversely, observe that $\sqsubseteq$ is reflexive and transitive, and thus a preorder relation. Observe also that, by using a trivial context $\mathcal{C}$ with no external actions and no transitions, $\sqsubseteq$ is included in $\leq_D$. Finally, using the associativity of parallel composition, observe that $\sqsubseteq$ is preserved by parallel composition, and thus is a precongruence. This means that $\sqsubseteq$ is a precongruence included in $\leq_D$. Since $\leq_{DC}$ is the coarsest precongruence included in $\leq_D$, we get that $\sqsubseteq$ is included in $\leq_{DC}$. $\square$

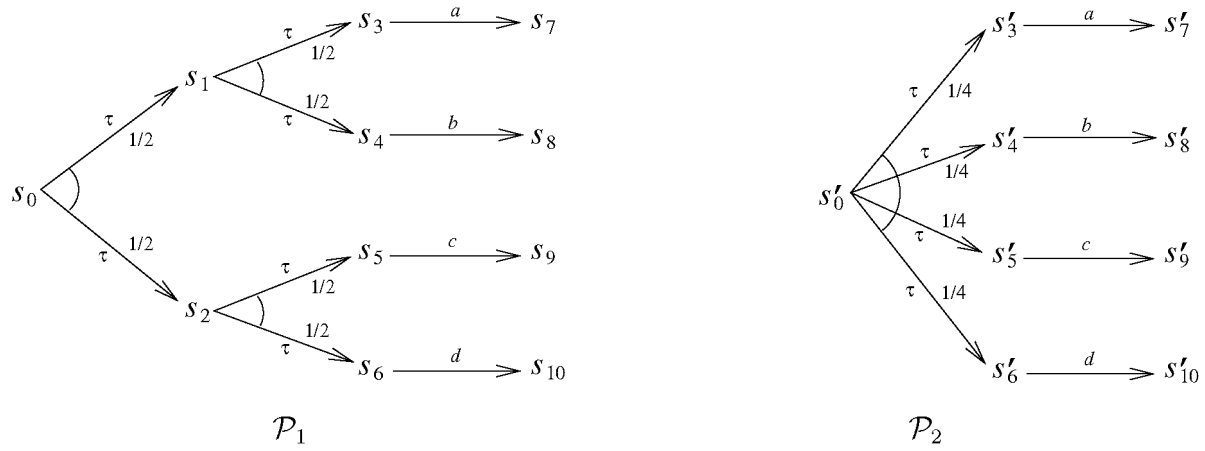

Figure 3: A forward probabilistic simulation between two probabilistic automata.

3.5 Simulation Relations

The definitions of forward simulation and weak forward simulation in Section 2 can be extended naturally to PAs [34]. However, Segala has shown [31] that the resulting simulations are not complete for $\leq_{DC}$, and has defined new candidate simulations. These new simulations relate states to probability measures on states.

In order to define the new simulations formally, we need two new concepts. First we show how to *lift* a relation between sets to a relation between measures over sets [18]. Let $R \subseteq X \times Y$. The *lifting* of R is a relation $R' \subseteq \mathit{Disc}(X) \times \mathit{Disc}(Y)$ such that $\mu_X \; R' \; \mu_Y$ iff there is a function $w : X \times Y \to [0,1]$ that satisfies:

1. If $w(x,y) > 0$ then $x \; R \; y$.
2. For each $x \in X$, $\sum_{y \in Y} w(x,y) = \mu_X(x)$.
3. For each $y \in Y$, $\sum_{x \in X} w(x,y) = \mu_Y(y)$.

We abuse notation slightly and denote the lifting of a relation R by R as well.

Second, we define a *flattening* operation that converts a measure μ in $\mathit{Disc}(\mathit{Disc}(X))$ into a measure $\mathit{flatten}(\mu)$ in $\mathit{Disc}(X)$. Namely, we define $\mathit{flatten}(\mu) = \sum_{\rho \in \mathit{supp}(\mu)} \mu(\rho)\rho$.

We now define simulations for probabilistic automata. A relation $R \subseteq Q_1 \times \mathit{Disc}(Q_2)$ is a *probabilistic forward simulation* (resp., *weak probabilistic forward simulation*) from PA $\mathcal{P}_1$ to PA $\mathcal{P}_2$ iff $E_1 = E_2$ and both of the following hold:

1. $\bar{q}_1 \; R \; \delta(\bar{q}_2)$.
2. For each pair q_1, μ_2 such that $q_1 \; R \; \mu_2$ and each transition $q_1 \xrightarrow{a} \mu'_1$, there exists a measure $\xi'_2 \in \mathit{Disc}(\mathit{Disc}(Q_2))$ such that $\mu'_1 \; R \; \xi'_2$ and such that $\mu_2 \xrightarrow{a} \mathit{flatten}(\xi'_2)$ (resp., $\mu_2 \stackrel{a}{\Longrightarrow} \mathit{flatten}(\xi'_2)$) is a hyper-transition (resp., a weak hyper-transition) of $\mathcal{P}_2$.

We write $\mathcal{P}_1 \leq_{PF} \mathcal{P}_2$ (resp., $\mathcal{P}_1 \leq_{wPF} \mathcal{P}_2$) whenever there is a probabilistic forward simulation (resp., a weak probabilistic forward simulation) from $\mathcal{P}_1$ to $\mathcal{P}_2$.

Example 3.4 Forward probabilistic simulation

Figure 3 gives an example of two probabilistic automata that are in the kernel of probabilistic forward simulation. However, there would be no simulation from $\mathcal{P}_1$ to $\mathcal{P}_2$ if we did not allow states to be related to measures over states. The probabilistic forward simulation R from $\mathcal{P}_1$ to $\mathcal{P}_2$ relates each state of $\mathcal{P}_1$ with the Dirac measure over its primed version of $\mathcal{P}_2$, relates s_1 with the uniform measure over s'_3 and s'_4, and relates s_2 with the uniform measure over s'_5 and s'_6. The transition from s_0 can be simulated from s'_0 by scheduling the only transition enabled. Indeed, the target measure $\mathcal{U}(s'_3, s'_4, s'_5, s'_6)$ is the flattening of $\mathcal{U}(\mathcal{U}(s'_3, s'_4), \mathcal{U}(s'_5, s'_6))$, and it is easy to check that $\mathcal{U}(s_1, s_2)\ R\ \mathcal{U}(\mathcal{U}(s'_3, s'_4), \mathcal{U}(s'_5, s'_6))$.

Note that a forward simulation between nondeterministic automata is a probabilistic forward simulation between the two automata viewed as PAs:

Proposition 3.9 *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be nondeterministic automata. Then:*

1. *$\mathcal{A}_1 \leq_F \mathcal{A}_2$ iff $\mathcal{A}_1 \leq_{PF} \mathcal{A}_2$, and*
2. *$\mathcal{A}_1 \leq_{wF} \mathcal{A}_2$ iff $\mathcal{A}_1 \leq_{wPF} \mathcal{A}_2$.*

Proof. The left-to-right inclusions are easy since, given a (weak) forward simulation R from $\mathcal{A}_1$ to $\mathcal{A}_2$, it is routine to check that the relation $R' \stackrel{\Delta}{=} \{(q_1, \delta(q_2)) \mid q_1\ R\ q_2\}$ is a (weak) probabilistic forward simulation from $\mathcal{A}_1$ to $\mathcal{A}_2$.

For the converse implication, let R be a (weak) probabilistic forward simulation from $\mathcal{A}_1$ to $\mathcal{A}_2$. Define a relation $R' \stackrel{\Delta}{=} \{(q_1, q_2) \mid \exists_\mu q_1\ R\ \mu, q_2 \in \mathit{supp}(\mu)\}$. We show that R' is a (weak) forward simulation from $\mathcal{A}_1$ to $\mathcal{A}_2$.

The start condition is trivial since $\bar{q}_1\ R\ \delta(\bar{q}_2)$, and thus $\bar{q}_1\ R'\ \bar{q}_2$. For the step condition, let $q_1\ R'\ q_2$, and let $q_1 \stackrel{a}{\rightarrow} q'_1$. By definition of R', there exists a measure μ such that $q_1\ R\ \mu$ and $q_2 \in \mathit{supp}(\mu)$. Since R is a (weak) forward simulation, there exists a hyper-transition $\mu \stackrel{a}{\rightarrow} \mu'$ (a weak hyper-transition $\mu \stackrel{a}{\Rightarrow} \mu'$) where μ' is the flattening of some measure μ'' such that $\delta(q'_1)\ R\ \mu''$. By definition of hyper-transition, there is a combined transition $q_2 \stackrel{a}{\rightarrow} \mu_2$ (a weak combined transition $q_2 \stackrel{a}{\Rightarrow} \mu_2$) such that $\mathit{supp}(\mu_2) \subseteq \mathit{supp}(\mu')$. For the strong case, let $q_2 \stackrel{a}{\rightarrow} q'_2$ be one of the transitions of D_2 that are combined in $q_2 \stackrel{a}{\rightarrow} \mu_2$. Then, $q'_2 \in \mathit{supp}(\mu_2)$. For the weak case, consider a scheduler σ that generates $q_2 \stackrel{a}{\Rightarrow} \mu_2$ and build a new scheduler σ' that on input α stops (does not return any transition) if $\sigma(\alpha)$ stops with some non-zero probability, and chooses any transition in $\mathit{supp}(\sigma(\alpha))$ that reduces the distance from a stopping point otherwise. This leads to a weak transition $q_2 \stackrel{a}{\Rightarrow} q'_2$ where $q'_2 \in \mathit{supp}(\mu_2)$. We now show that $q'_1\ R'\ q'_2$, which suffices. Since $q'_2 \in \mathit{supp}(\mu_2)$, and since $\mathit{supp}(\mu_2) \subseteq \mathit{supp}(\mu')$, then $q'_2 \in \mathit{supp}(\mu')$. Since $\mu' = \mathit{flatten}(\mu'')$, then q'_2 is also in the support of some measure $\rho \in \mathit{supp}(\mu'')$. Thus, $q'_1\ R\ \rho$, and, by definition of R', $q'_1\ R'\ q'_2$ as needed. □

Proposition 3.10 *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be PAs. Then:*

1. *If $\mathcal{P}_1 \leq_{PF} \mathcal{P}_2$ then $\mathcal{P}_1 \leq_{wPF} \mathcal{P}_2$.*
2. *If $H_1 = H_2 = \emptyset$ then $\mathcal{P}_1 \leq_{PF} \mathcal{P}_2$ iff $\mathcal{P}_1 \leq_{wPF} \mathcal{P}_2$.*
3. *If $\mathcal{P}_1 \leq_{wPF} \mathcal{P}_2$ then $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$.*

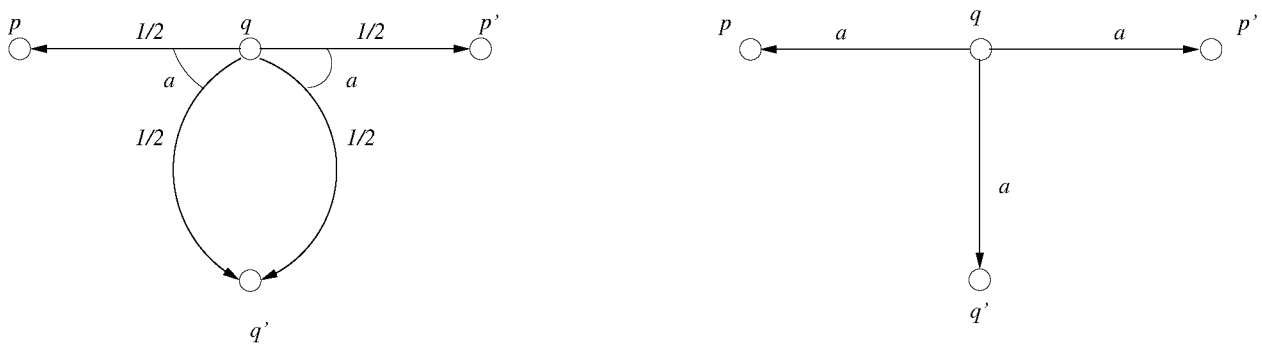

Figure 4: The PA on the left is not tree-structured even though its underlying nondeterministic automaton on the right is.

Proof. The first item follows from the fact that a combined transition is a special case of a weak combined transition; the second item follows from the fact that in the absence of internal actions a weak combined transition is a combined transition. The proof of the third item is quite involved and we refer the reader to Proposition 8.7.1 of [32]. The main idea is to use the weak probabilistic forward simulation from $\mathcal{P}_1$ to $\mathcal{P}_2$ to build, for each probabilistic execution of $\mathcal{P}_1$, a corresponding probabilistic execution of $\mathcal{P}_2$ with the same trace distribution. □

3.6 Tree-Structured Probabilistic Automata

A *path* of a PA $\mathcal{P}$ is a finite sequence $\gamma = q_0 a_1 \mu_1 q_1 a_2 \mu_2 q_2 \dots q_n$ of alternating states, actions and distribution over states, starting with the start state of $\mathcal{P}$ such that for each non-final i, $q_i \stackrel{a_{i+1}}{\rightarrow} \mu_{i+1}$ and $q_{i+1} \in supp(\mu_{i+1})$. We write $lstate(\gamma)$ to denote q_n and $paths(\mathcal{P})$ for the set of all path of $\mathcal{P}$. We say that $\mathcal{P}$ is *tree-structured* if each state is reached via a unique path. Tree-structured probabilistic automata are characterized uniquely by the property that all states are reachable, the start state does not occur in the target of any transition, and each of the other states occurs in the target of exactly one transition. Tree-structured nondeterministic automata are also characterized uniquely by this property, albeit for a different notion of transition.

If a probabilistic automaton is tree-structured then its underlying nondeterministic automaton is also tree-structured. The following example shows that the converse does not hold.

Example 3.5 Non-tree-structured probabilistic automata

Figure 4 shows a probabilistic automaton that is not tree-structured, as state q' can be reached via two different paths. The underlying nondeterministic automaton is tree-structured, however, since the only way to reach state q' is via the execution qaq'.

The *unfolding* of a probabilistic automaton $\mathcal{P}$, denoted by $Unfold(\mathcal{P})$, is the tree-structured probabilistic automaton $\mathcal{Q}$ obtained from $\mathcal{P}$ by unfolding its transition graph into a tree. Formally,

- $Q_{\mathcal{Q}} = paths(\mathcal{P})$,
- $\bar{q}_{\mathcal{Q}} = \bar{q}_{\mathcal{P}}$,
- $E_{\mathcal{Q}} = E_{\mathcal{P}}$,
- $H_{\mathcal{Q}} = H_{\mathcal{P}}$, and

- $D_{\mathcal{Q}} = \{(\gamma, a, \mu) \mid (\exists \mu')[(lstate(\gamma), a, \mu') \in D_{\mathcal{P}} \wedge (\forall q \in supp(\mu'))[\mu'(q) = \mu(\gamma a \mu' q)]]\}$.

Proposition 3.11 $\mathcal{P} \equiv_{PF} Unfold(\mathcal{P})$.

Proof. It is easy to check that the relation R where $\alpha\ R\ \delta(q)$ iff $lstate(\alpha) = q$ is a probabilistic forward simulation from $Unfold(\mathcal{P})$ to $\mathcal{P}$ and that the "inverse" of R, that is, the relation R' such that $q\ R'\ \delta(\alpha)$ iff $lstate(\alpha) = q$, is a probabilistic forward simulation from $\mathcal{P}$ to $Unfold(\mathcal{P})$. □

Proposition 3.12 $\mathcal{P} \equiv_{DC} Unfold(\mathcal{P})$.

Proof. By Proposition 3.11 and Proposition 3.10, Parts 1 and 3. □

3.7 Truncations and Continuations

We now define two simple constructions on probabilistic execution fragments that will be useful for our proofs. Specifically, we define the truncation of a probabilistic execution fragment, which is the result of stopping the computation at some designated points, and the continuation of a probabilistic execution fragment, which represents the rest of a probabilistic execution fragment after some finite execution fragment has occurred.

Let ϵ be a probabilistic execution fragment of a PA $\mathcal{P}$, generated by some scheduler σ, and let Θ be a set of finite execution fragments of $\mathcal{P}$. Define the *truncation of* ϵ *at* Θ to be the same as ϵ except that no transition is scheduled from all the Θ places, that is, the probabilistic execution fragment ϵ', with the same start state as ϵ, generated by a new scheduler σ' such that $\sigma'(\alpha) = \sigma(\alpha)$ if $\alpha \notin \Theta$ and $\sigma'(\alpha)(D) = 0$ if $\alpha \in \Theta$.

Proposition 3.13 *The definition of truncation of a probabilistic execution fragment ϵ is independent of the choice of the inducing scheduler.*

Proof. Let μ be the first state of ϵ and let σ_1, σ_2 be two schedulers that, together with μ, induce ϵ. Let Θ be a set of finite execution fragments of $\mathcal{P}$, and let σ_1', σ_2' be the schedulers built from σ_1, σ_2, respectively, according to the definition of truncation. Let ϵ_1, ϵ_2 be the induced probabilistic execution fragments, and suppose by contradiction that $\epsilon_1 \neq \epsilon_2$. Then there exists a finite execution α such that $\epsilon_1(C_\alpha) \neq \epsilon_2(C_\alpha)$. Consider such a finite execution α of minimum length. Observe that $|\alpha| > 0$ since $\epsilon(C_q) = \epsilon_1(C_q) = \epsilon_2(C_q) = \mu(q)$ for each state $q \in Q$. Thus, $\alpha = \alpha' a' q'$ for some α', a', q', where $\epsilon_1(C_{\alpha'}) = \epsilon_2(C_{\alpha'})$. We distinguish two cases.

If $\alpha' \in \Theta$, then, by definition of σ_1' and σ_2', $\sigma_1'(\alpha')(D) = \sigma_2'(\alpha')(D) = 0$. Thus, $\epsilon_1(C_\alpha) = \epsilon_2(C_\alpha) = 0$, a contradiction.

If $\alpha' \notin \Theta$, then, by definition of σ_1' and σ_2', $\sigma_1'(\alpha') = \sigma_1(\alpha')$ and $\sigma_2'(\alpha') = \sigma_2(\alpha')$. Then,

$$
\begin{array}{lcl}
\epsilon_1(C_\alpha) & = & \text{(by Equation (1))} \\
\epsilon_1(C_{\alpha'}) \sum_{tr \in D(a')} \sigma_1'(\alpha')(tr) \mu_{tr}(q') & = & \text{(by } \sigma_1'(\alpha') = \sigma_1(\alpha') \text{ and } \epsilon_1(C_{\alpha'}) = \epsilon_2(C_{\alpha'})\text{)} \\
\epsilon_2(C_{\alpha'}) \sum_{tr \in D(a')} \sigma_1(\alpha')(tr) \mu_{tr}(q') & = & \text{(by } \sigma_1 \text{ and } \sigma_2 \text{ induce } \epsilon\text{)} \\
\epsilon_2(C_{\alpha'}) \sum_{tr \in D(a')} \sigma_2(\alpha')(tr) \mu_{tr}(q') & = & \text{(by } \sigma_2'(\alpha') = \sigma_2(\alpha')\text{)} \\
\epsilon_2(C_{\alpha'}) \sum_{tr \in D(a')} \sigma_2'(\alpha')(tr) \mu_{tr}(q') & = & \text{(by Equation (1))} \\
\epsilon_2(C_\alpha), & &
\end{array}
$$

again a contradiction. □

Let ϵ be a probabilistic execution fragment of a PA $\mathcal{P}$, generated by a scheduler σ, and let α be a finite execution fragment with $fstate(\alpha) \in supp(fstate(\epsilon))$. Define $\epsilon \triangleright \alpha$, the *continuation of* ϵ *after prefix* α, to be the probabilistic execution fragment generated by the following scheduler σ' from $lstate(\alpha)$:

$$\sigma'(\alpha') = \begin{cases} \sigma(\alpha \frown \alpha') & \text{if } fstate(\alpha') = lstate(\alpha) \\ 0 & \text{otherwise,} \end{cases}$$

where by 0 we denote the identically 0 function.

Proposition 3.14 *The definition of $\epsilon \triangleright \alpha$ is independent of the choice of the inducing scheduler.*

Proof. Similar to the proof of Proposition 3.13. Let μ be the first state of ϵ and let σ_1, σ_2 be two schedulers that, together with μ, induce ϵ. Let q' be $lstate(\alpha)$. Let σ_1', σ_2' be the schedulers built from σ_1, σ_2, respectively, according to the definition of $\epsilon \triangleright \alpha$. Let ϵ_1, ϵ_2 be the induced probabilistic execution fragments from q', and suppose by contradiction that $\epsilon_1 \neq \epsilon_2$. Then there exists a finite execution α' such that $\epsilon_1(C_{\alpha'}) \neq \epsilon_2(C_{\alpha'})$. Consider such a finite execution α' of minimum length. Observe that $|\alpha'| > 0$ since $\epsilon(C_{q'}) = \epsilon_1(C_{q'}) = \epsilon_2(C_{q'}) = 1$ and, for each state $q'' \neq q'$, $\epsilon(C_{q''}) = \epsilon_1(C_{q''}) = \epsilon_2(C_{q''}) = 0$. Thus, $\alpha' = \alpha'' a'' q''$ for some α'', a'', q'', where $\epsilon_1(C_{\alpha''}) = \epsilon_2(C_{\alpha''})$. We distinguish two cases.

If $fstate(\alpha'') \neq q'$, then, by definition of ϵ_1 and ϵ_2, $\epsilon_1(C_{\alpha'}) = \epsilon_2(C_{\alpha'}) = 0$, a contradiction.

If $fstate(\alpha'') = q'$, then, by definition of σ_1' and σ_2', $\sigma_1'(\alpha'') = \sigma_1(\alpha \frown \alpha'')$ and $\sigma_2'(\alpha'') = \sigma_2(\alpha \frown \alpha'')$. Then,

$$\begin{array}{lcl}
\epsilon_1(C_{\alpha'}) & = & \text{(by Equation (1))} \\
\epsilon_1(C_{\alpha''}) \sum_{tr \in D(a'')} \sigma_1'(\alpha'')(tr)\mu_{tr}(q'') & = & \text{(by } \sigma_1'(\alpha'') = \sigma_1(\alpha \frown \alpha'') \text{ and } \epsilon_1(C_{\alpha''}) = \epsilon_2(C_{\alpha''})) \\
\epsilon_2(C_{\alpha''}) \sum_{tr \in D(a'')} \sigma_1(\alpha \frown \alpha'')(tr)\mu_{tr}(q'') & = & \text{(by } \sigma_1 \text{ and } \sigma_2 \text{ induce } \epsilon) \\
\epsilon_2(C_{\alpha''}) \sum_{tr \in D(a'')} \sigma_2(\alpha \frown \alpha'')(tr)\mu_{tr}(q'') & = & \text{(by } \sigma_2'(\alpha'') = \sigma_2(\alpha \frown \alpha'')) \\
\epsilon_2(C_{\alpha''}) \sum_{tr \in D(a'')} \sigma_2'(\alpha'')(tr)\mu_{tr}(q'') & = & \text{(by Equation (1))} \\
\epsilon_2(C_{\alpha'}), & &
\end{array}$$

again a contradiction. □

The following proposition relates the continuation of ϵ after some prefix α with ϵ itself. In practice it states that $\epsilon \triangleright \alpha$ is closely related to $\epsilon \mid C_\alpha$.

Proposition 3.15 *Let ϵ be a probabilistic execution fragment of a PA $\mathcal{P}$, and let α be a finite execution fragment of $\mathcal{P}$. Then, for each finite execution α' with $lstate(\alpha) = fstate(\alpha')$, $\epsilon(C_{\alpha \frown \alpha'}) = \epsilon(C_\alpha) \cdot (\epsilon \triangleright \alpha)(C_{\alpha'})$.*

Proof. Follows easily by induction on the length of α' from the definition of the probability of a cone. □

4 Tester Automata and Observer Schedulers

The proofs of our completeness results rely on a special context for a probabilistic automaton, which we call its *tester probabilistic automaton*. The tester automaton, $tester(\mathcal{P})$, of a PA $\mathcal{P}$ can observe the states $\mathcal{P}$ goes through and the transitions that are scheduled during a probabilistic execution. This information is revealed by means of externally visible transitions of $tester(\mathcal{P})$ with

the help of a specific scheduler, called the *observer*, which synchronizes $\mathcal{P}$ with its tester. In this section we present the constructions of the tester and observer, and prove some results about the resulting trace distributions.

Informally, the tester of a probabilistic automaton $\mathcal{P}$ is a probabilistic automaton $\mathcal{C}$ whose states include a distinguished start state, all the states of $\mathcal{P}$, and all the transitions of $\mathcal{P}$. Automaton $\mathcal{C}$ has a special transition from its own start state, $\bar{q}_{\mathcal{C}}$, to the start state of $\mathcal{P}$, $\bar{q}_{\mathcal{P}}$, labeled by $\bar{q}_{\mathcal{P}}$. Also, from every state q of $\mathcal{P}$, $\mathcal{C}$ has a uniform transition labeled by ch ("choose") to the set of transitions of $\mathcal{P}$ that begin in state q. Finally, for every transition tr of $\mathcal{P}$, and every state q in the support of μ_{tr}, $\mathcal{C}$ has a transition labeled by q from tr to q.

Definition 4.1 *The* tester probabilistic automaton *of a PA $\mathcal{P}$, denoted by tester($\mathcal{P}$), is a PA $\mathcal{C} = (Q_{\mathcal{C}}, \bar{q}_{\mathcal{C}}, E_{\mathcal{C}}, H_{\mathcal{C}}, D_{\mathcal{C}})$ where*

- $Q_{\mathcal{C}} = \{\bar{q}_{\mathcal{C}}\} \cup Q_{\mathcal{P}} \cup D_{\mathcal{P}}$,
- $E_{\mathcal{C}} = Q_{\mathcal{P}} \cup \{ch\}$,
- $H_{\mathcal{C}} = \emptyset$, *and*
- $D_{\mathcal{C}} = \{(\bar{q}_{\mathcal{C}}, \bar{q}_{\mathcal{P}}, \delta(\bar{q}_{\mathcal{P}}))\} \cup$
  $\{(q, ch, \mathcal{U}(\{tr \in D_{\mathcal{P}} \mid source(tr) = q\})) \mid q \in Q_{\mathcal{P}} \wedge q \rightarrow\} \cup$
  $\{(tr, q, \delta(q)) \mid tr \in D_{\mathcal{P}}, q \in supp(\mu_{tr})\}$.

Observe that the tester of an ordinary nondeterministic automaton enables at most one transition from each state, and dually, the tester of an automaton that enables at most one transition from each state is a nondeterministic automaton. This observation, together with the results that we prove later in the paper, imply that a fully probabilistic context is enough to observe the branching structure of a nondeterministic automaton.

Proposition 4.2 *The following hold.*

1. *The tester of a nondeterministic automaton is fully probabilistic, that is, it enables at most one transition from each state.*
2. *The tester of a fully probabilistic automaton is a nondeterministic automaton, that is, it contains only transitions whose target measures are Dirac.*

Proof. For the first item, observe that the only states of *tester*($\mathcal{P}$) that may enable more than one transition are of the form $tr \in D_{\mathcal{P}}$, which enable one transition for each state in $supp(\mu_{tr})$; however, the size of $supp(\mu_{tr})$ is 1 in a nondeterministic automaton.

For the second item, observe that the only states of *tester*($\mathcal{P}$) that may enable non-Dirac transitions are of the form $q \in Q_{\mathcal{P}}$, which may enable a transition labeled by ch to a uniform measure over the set of transitions enabled from q in $\mathcal{P}$; however, there is at most one transition enabled from q in a fully probabilistic automaton. □

We assume without loss of generality that a probabilistic automaton $\mathcal{P}$ and its tester do not have any actions in common (otherwise we can simply rename states of $\mathcal{P}$ to achieve our goal), and thus $\mathcal{P}$ and its tester are compatible.

Since *tester*($\mathcal{P}$) and $\mathcal{P}$ share no actions, merely composing *tester*($\mathcal{P}$) with $\mathcal{P}$ does not ensure that *tester*($\mathcal{P}$) faithfully emulates the behavior of $\mathcal{P}$. However, an appropriate scheduler can synchronize

the two automata and ensure such an emulation, which will be sufficient for our purposes. Given a probabilistic automaton $\mathcal{P}$, we define a specific scheduler σ for $\mathcal{P}\|tester(\mathcal{P})$, called the *observer* of $\mathcal{P}$, that synchronizes the two automata so that the internal structure of $\mathcal{P}$ is visible in the trace. Specifically, the scheduler σ starts by scheduling the transition of $tester(\mathcal{P})$ from the start state of $tester(\mathcal{P})$ to the start state of $\mathcal{P}$, leading to state $(\bar{q}, \bar{q})$, which is of the form (q, q). Then σ repeats the following as long as $q \rightarrow$:

1. Schedule the ch transition of $tester(\mathcal{P})$, thus choosing a transition tr of $\mathcal{P}$.
2. Schedule transition tr of $\mathcal{P}$, leading $\mathcal{P}$ to a new state q'.
3. Schedule the transition of $tester(\mathcal{P})$ labeled by the state q', resulting in the state (q', q'), which is again of the form (q, q).

Definition 4.3 *The* observer *of a probabilistic automaton $\mathcal{P}$, denoted by $observer(\mathcal{P})$, is a deterministic (almost memoryless) scheduler for $\mathcal{P}\|tester(\mathcal{P})$ that bases its decisions on the last state and sometimes the last action of its argument according to the following table. Here, $\mathcal{C}$ denotes $tester(\mathcal{P})$, q is any state such that $q \rightarrow$, tr is any transition in $D_{\mathcal{P}}$, and q' is any state in μ_{tr}.*

$Q_{\mathcal{P}}$	$Q_{tester(\mathcal{P})}$	*last action*	$D_{\mathcal{P}\|tester(\mathcal{P})}$
$\bar{q}_{\mathcal{P}}$	$\bar{q}_{tester(\mathcal{P})}$		$\bar{q}_{\mathcal{P}}$*-labeled transition of* $tester(\mathcal{P})$
q	q		ch*-labeled transition of* $tester(\mathcal{P})$
q	tr	ch	*transition* tr *of* $\mathcal{P}$
q'	tr	*not* ch	q'*-labeled transition of* $tester(\mathcal{P})$

Scheduler $observer(\mathcal{P})$ and start state $(\bar{q}_{\mathcal{P}}, \bar{q}_{tester(\mathcal{P})})$ induce a trace distribution for $\mathcal{P}\|tester(\mathcal{P})$ where all states and external actions of $\mathcal{P}$ appear explicitly.

Definition 4.4 *The* observation *of a probabilistic automaton $\mathcal{P}$, denoted by $observation(\mathcal{P})$, is the trace distribution induced by $observer(\mathcal{P})$ and $(\bar{q}_{\mathcal{P}}, \bar{q}_{tester(\mathcal{P})})$.*

Remark 4.5 *The ch-labeled transitions of a tester are defined to lead to uniform measures over states of $tester(\mathcal{P})$ that represent transitions of $\mathcal{P}$. This is well defined since we have assumed that nondeterministic and probabilistic automata are finite branching. From the technical point of view, however, the proofs of this paper rely on the fact that the transitions labeled by ch assign non-zero probability (non necessarily the same probability) to each one of the options that are available in a nondeterministic choice. Thus, it would be possible to remove the finite branching restriction from the definition of automata and modify the definition of a tester automaton so that, whenever there are countably many transitions from a state q, the corresponding ch-labeled transition of the tester assigns a non-uniform measure to the transitions enabled from q, for example, a Poisson distribution after enumerating all possible transitions enabled from q. We have chosen not to deal with countable branching automata in the paper because it would complicate proofs without adding much insight.*

We state and prove some properties of $observation(\mathcal{P})$. The first property, Equation (5), says that the cone of traces beginning with the start state of $\mathcal{P}$ has probability 1. The second property, Equation (6), says that for any state q of $\mathcal{P}$ from which some transition is enabled and for each finite trace β of $\mathcal{P}\|tester(\mathcal{P})$, the probability of the cone of traces beginning with βq is the same as the probability of the cone beginning with $\beta q\, ch$, that is, once βq occurs, the probability that ch

follows is 1. The third property, Equation (7), says that for any state q of $\mathcal{P}$ and for each finite trace β of $\mathcal{P}\|tester(\mathcal{P})$, the probability of the cone of traces beginning with $\beta q\ ch$ is the same as the sum of the probabilities of the cones beginning with $\beta q\ ch\ \beta'$ where β' represents one single step of $\mathcal{P}$ from q, that is, once ch occurs, one of the transitions of $\mathcal{P}$ that are enabled from q is exposed. The right-hand side of Equation (7) consists of two parts dealing with external and internal transitions, respectively.

Proposition 4.6 *The trace distribution $\eta = observation(\mathcal{P})$ induced by the observer of a probabilistic automaton $\mathcal{P}$ satisfies the following three properties, for all finite traces β of $\mathcal{P}\|tester(\mathcal{P})$ and for all states q of $\mathcal{P}$:*

$$\eta(C_{\bar{q}}) = 1 \tag{5}$$

$$q \rightarrow \quad \Longrightarrow \quad \eta(C_{\beta q}) = \eta(C_{\beta q\ ch}) \tag{6}$$

$$\eta(C_{\beta q\ ch}) = \sum_{(a,q')|a\in E, q\xrightarrow{a}q'} \eta(C_{\beta q\ ch\ a q'}) + \sum_{q'|(\exists a) a\in H, q\xrightarrow{a}q'} \eta(C_{\beta q\ ch\ q'}) \tag{7}$$

Proof. Equation (5) follows from the fact that $observer(\mathcal{P})$ schedules action $\bar{q}$ immediately. Equation (6) follows from the fact that, after scheduling action q, thus leading to a state of the form (q, q), $observer(\mathcal{P})$ immediately schedules action ch if q enables at least one transition. Equation (7) follows from the fact that, after scheduling ch, $observer(\mathcal{P})$ schedules one of the transitions of $\mathcal{P}$ that are enabled from q, say $q \xrightarrow{a} \mu$, followed by a transition of $tester(\mathcal{P})$ labeled by a state in $supp(\mu)$. □

The following technical properties will be needed in the proofs of Section 6. The first property, Equation (8), says that the probability of observing a state q' reachable in a tree-structured probabilistic automaton $\mathcal{P}$ with a single transition, say tr, from another state q is the probability of observing q, divided by the number of transitions enabled from q, and multiplied by the probability of reaching q' in tr, the only transition that may lead to q' since $\mathcal{P}$ is tree-structured. Indeed, q' can be observed only if q is observed (probability of observing q), the transition tr is chosen (factor $1/k$ since transitions are chosen uniformly), and the chosen transition leads to q'. The second property, Equation (9), is similar to the first one, where the probability of observing q and scheduling the transition tr is replaced by the probability of observing any state in the target of tr.

Proposition 4.7 *Let $\mathcal{P}$ be a tree-structured probabilistic automaton, and let η be $observation(\mathcal{P})$. Let $tr = (q, a, \mu)$ be a transition of $\mathcal{P}$. Let k be the number of transitions that are enabled from q in $\mathcal{P}$, and let q' be a state in $supp(\mu)$. Then the following properties hold:*

$$\eta(\diamond q') = \frac{\eta(\diamond q)}{k}\mu(q') \tag{8}$$

$$\eta(\diamond q') = \left(\sum_{q''\in supp(\mu)} \eta(\diamond q'')\right)\mu(q') \tag{9}$$

Proof. Let σ be the observer of $\mathcal{P}$, and let ϵ_σ be the probabilistic execution induced by σ. Since $\mathcal{P}$ is tree-structured, the set Θ_q contains a single execution α. Indeed, by definition of tree-structured, there is only one execution in $\mathcal{P}$ ending with state q, and σ simply interleaves this execution with transitions labeled by ch, by the names of the transitions of $\mathcal{P}$ that are needed to reach q, and by the names of the states that are reached. Similarly, $\Theta_{q'}$ contains a single execution α'.

Once state q is reached, σ schedules action ch, reaching state tr of $tester(\mathcal{P})$ with probability $1/k$. Then, σ schedules transition tr, reaching state q' in $\mathcal{P}$ with probability $\mu(q')$, and finally σ schedules the transition of $tester(\mathcal{P})$ labeled by q'. Thus, $\epsilon_\sigma(C_{\alpha'}) = \epsilon_\sigma(C_\alpha)(1/k)\mu(q')$. Then Equation (8) follows by Equation (2).

By summing over $supp(\mu)$ in Equation (8), we get

$$\sum_{q'' \in supp(\mu)} \eta(\diamond q'') = \frac{\eta(\diamond q)}{k} \sum_{q'' \in supp(\mu)} \mu(q''). \tag{10}$$

Observe that $\sum_{q'' \in supp(\mu)} \mu(q'') = 1$. Hence, Equation (10) simplifies to

$$\sum_{q'' \in supp(\mu)} \eta(\diamond q'') = \frac{\eta(\diamond q)}{k}. \tag{11}$$

Substitution of Equation (11) in Equation (8) gives us Equation (9) as needed. □

5 Characterizations of $\leq_{DC}$ for Nondeterministic Automata

In this section, we present our characterization theorems for $\leq_{DC}$ for nondeterministic automata: Theorem 5.2 characterizes $\leq_{DC}$ in terms of $\leq_F$, for nondeterministic automata without internal actions, and Theorem 5.4 characterizes $\leq_{DC}$ in terms of $\leq_{wF}$, for arbitrary nondeterministic automata. In each case, we prove the result first for tree-structured nondeterministic automata and then extend it to the non-tree-structured case via unfolding. The interesting direction for each of these results is the completeness direction, showing that $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$ implies the existence of a simulation relation from $\mathcal{A}_1$ to $\mathcal{A}_2$.

The strategy that we use to prove our completeness results is also applied in many other full abstraction results, see for example, [5, 14]. By Proposition 3.8, $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$ implies that $\mathcal{A}_1 \leq_D \mathcal{A}_2$ for all contexts $\mathcal{C}$. Thus it suffices to construct a specific context $\mathcal{C}$ with the property that the trace distributions of $\mathcal{A}_1 \| \mathcal{C}$ contain all information about $\mathcal{A}_1$ that is preserved by the simulation preorder. More specifically, we compose $\mathcal{A}_1$ with the context $\mathcal{C} = tester(\mathcal{A}_1)$ and consider just a single trace distribution of the composed system, namely $observation(\mathcal{A}_1)$, the one generated by $observer(\mathcal{A}_1)$. We show, for any other nondeterministic automaton $\mathcal{A}_2$, if the composition $\mathcal{A}_2 \| tester(\mathcal{A}_1)$ generates the trace distribution $observation(\mathcal{A}_1)$, then $\mathcal{A}_2$ actually simulates $\mathcal{A}_1$ in a strong sense. Namely, whenever $\mathcal{A}_1$ reaches some state q_1, $\mathcal{A}_2$ can reach a corresponding state q_2 from which it generates the same trace distribution. The formalities of the proof are intricate, in part because states of $\mathcal{A}_1$ also show up as states of $tester(\mathcal{A}_1)$ and within the trace distribution of $observation(\mathcal{A}_1)$. In the proof we try to be very explicit about the roles of states of $\mathcal{A}_1$, but we also ward the to be alert to this potential source of confusion.

5.1 Nondeterministic Automata Without Internal Actions

We begin by considering nondeterministic automata without internal actions. We first consider tree-structured nondeterministic automata.

Proposition 5.1 *Let $\mathcal{A}_1$, $\mathcal{A}_2$ be nondeterministic automata without internal actions such that $\mathcal{A}_1$ is tree-structured. Then $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$ implies $\mathcal{A}_1 \leq_F \mathcal{A}_2$.*

Proof. Assume that $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$. Let $\mathcal{C}$ be *tester*$(\mathcal{A}_1)$ and η be *observation*$(\mathcal{A}_1)$, that is, the trace distribution of $\mathcal{A}_1 \| \mathcal{C}$ induced by the scheduler *observer*$(\mathcal{A}_1)$. Since $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$ implies $\mathcal{A}_1 \| \mathcal{C} \leq_D \mathcal{A}_2 \| \mathcal{C}$, Proposition 3.8 implies that η is also a trace distribution of $\mathcal{A}_2 \| \mathcal{C}$. That is, there exists a probabilistic execution ϵ of $\mathcal{A}_2 \| \mathcal{C}$, induced by some scheduler σ_2, such that $tdist(\epsilon) = \eta$.

For each state q_1 in Q_1, let Θ_{q_1} be the set of finite executions of $\mathcal{A}_2 \| \mathcal{C}$ whose last transition is labeled by q_1. For each state q_2 of $\mathcal{A}_2$, let Θ_{q_1,q_2} be the set of executions in Θ_{q_1} whose last state is the pair (q_2, q_1).

Define a relation R on $Q_1 \times Q_2$ as follows: $q_1 \; R \; q_2$ if and only if there exists a finite execution α in Θ_{q_1,q_2} such that $\epsilon(C_\alpha) > 0$. We claim that R is a forward simulation from $\mathcal{A}_1$ to $\mathcal{A}_2$.

For the start condition, we must show that $\bar{q}_1 \; R \; \bar{q}_2$. Consider the start state $(\bar{q}_2, \bar{q}_\mathcal{C})$ of $\mathcal{A}_2 \| \mathcal{C}$. Since there are no internal actions in $\mathcal{A}_2$ or $\mathcal{C}$, and since, by Equation (5) from Proposition 4.6, $\eta(C_{\bar{q}_1}) = 1$, the only action that is scheduled initially by σ_2 is $\bar{q}_1$, leading to state $(\bar{q}_2, \bar{q}_1)$. Thus, the finite execution $\alpha = (\bar{q}_2, \bar{q}_\mathcal{C}) \bar{q}_1 (\bar{q}_2, \bar{q}_1)$ is an element of $\Theta_{\bar{q}_1,\bar{q}_2}$ such that $\epsilon(C_\alpha) > 0$, as needed.

For the step condition, assume $q_1 \; R \; q_2$ and let $q_1 \xrightarrow{a}_1 q_1'$ be a transition of $\mathcal{A}_1$, which we denote by tr for convenience. We exhibit a matching transition $q_2 \xrightarrow{a}_2 q_2'$.

By definition of R, there exists a finite execution α in Θ_{q_1,q_2}, such that $\epsilon(C_\alpha) > 0$. Since Θ_{q_1,q_2} is a subset of Θ_{q_1}, by definition of Θ_{q_1}, $trace(\alpha) = \beta q_1$ for some finite trace β. Therefore, $\eta(C_{\beta q_1}) > 0$. Since q_1 enables at least one transition in $\mathcal{A}_1$, specifically transition tr, Equation (6) from Proposition 4.6 implies that $\eta(C_{\beta q_1 \, ch}) = \eta(C_{\beta q_1})$. Then, since $\mathcal{A}_2$ and $\mathcal{C}$ have no internal actions, σ_2 schedules action ch from α with probability 1.

By definition of *tester*$(\mathcal{A}_1)$, the transition labeled by ch that leaves from state q_1 of $\mathcal{C}$ leads to state tr with non-zero probability. Therefore, $\epsilon(C_{\alpha \, ch \, (q_2, tr)}) > 0$. By Equation (7) from Proposition 4.6, where only the first term of the right-hand side is non-zero due to the absence of internal actions, $\eta(C_{\beta q_1 \, ch}) = \sum_{(a,q') | a \in E, q_1 \xrightarrow{a} q'} \eta(C_{\beta q_1 \, ch \, a q'})$. Hence, σ_2 must extend $\alpha \, ch \, (q_2, tr)$ with two steps labeled by an action and a state of $\mathcal{A}_1$, respectively, where the action and the state are compatible with one of the transitions of $\mathcal{A}_1$ that are enabled from q_1. Since state tr of $\mathcal{C}$ enables only action q_1', and since, by the tree-structure of $\mathcal{A}_1$, a is uniquely determined by q_1', the action and state scheduled by σ_2 are a and q_1'. Therefore, there exists a state q_2' of $\mathcal{A}_2$ such that the execution $\alpha' = \alpha \, ch \, (q_2, tr) a (q_2', tr) q_1' (q_2', q_1')$ is an execution in $\Theta_{q_1',q_2'}$ such that $\epsilon(C_{\alpha'}) > 0$. Then $q_1' \; R \; q_2'$ and $q_2 \xrightarrow{a} q_2'$ as needed. □

Now we present our result for general (non-tree-structured) nondeterministic automata without internal actions.

Theorem 5.2 *Let $\mathcal{A}_1$, $\mathcal{A}_2$ be nondeterministic automata without internal actions. Then $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$ if and only if $\mathcal{A}_1 \leq_F \mathcal{A}_2$.*

Proof. First we prove soundness of forward simulations:

$$
\begin{array}{lcl}
\mathcal{A}_1 \leq_F \mathcal{A}_2 & \Rightarrow & \text{(Proposition 3.9, Part 1)} \\
\mathcal{A}_1 \leq_{PF} \mathcal{A}_2 & \Rightarrow & \text{(Proposition 3.10, Part 1)} \\
\mathcal{A}_1 \leq_{wPF} \mathcal{A}_2 & \Rightarrow & \text{(Proposition 3.10, Part 3)} \\
\mathcal{A}_1 \leq_{DC} \mathcal{A}_2 \; . & &
\end{array}
$$

Next we establish completeness:

$$
\begin{array}{lcl}
\mathcal{A}_1 \leq_{DC} \mathcal{A}_2 & \Rightarrow & \text{(Proposition 2.4)} \\
\mathit{Unfold}(\mathcal{A}_1) \leq_F \mathcal{A}_1 \leq_{DC} \mathcal{A}_2 & \Rightarrow & \text{(as in soundness proof)} \\
\mathit{Unfold}(\mathcal{A}_1) \leq_{DC} \mathcal{A}_1 \leq_{DC} \mathcal{A}_2 & \Rightarrow & (\leq_{DC} \text{ is transitive}) \\
\mathit{Unfold}(\mathcal{A}_1) \leq_{DC} \mathcal{A}_2 & \Rightarrow & \text{(Proposition 5.1)} \\
\mathit{Unfold}(\mathcal{A}_1) \leq_F \mathcal{A}_2 & \Rightarrow & \text{(Proposition 2.4)} \\
\mathcal{A}_1 \leq_F \mathit{Unfold}(\mathcal{A}_1) \leq_F \mathcal{A}_2 & \Rightarrow & (\leq_F \text{ is transitive}) \\
\mathcal{A}_1 \leq_F \mathcal{A}_2 \ . & &
\end{array}
$$

□

5.2 Nondeterministic Automata With Internal Actions

Next we extend the results of Section 5.1 to nondeterministic automata that may include internal actions. The proofs are analogous to those in Section 5.1. The difference is that, in several places in the proof of Proposition 5.3, we need to reason about multi-step extensions of executions instead of single-step extensions. Again, we begin with tree-structured nondeterministic automata.

Proposition 5.3 *Let $\mathcal{A}_1$, $\mathcal{A}_2$ be nondeterministic automata such that $\mathcal{A}_1$ is tree-structured. Then $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$ implies $\mathcal{A}_1 \leq_{wF} \mathcal{A}_2$.*

Proof. Assume that $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$. Let $\mathcal{C}$ be $\mathit{tester}(\mathcal{A}_1)$ and η be $\mathit{observation}(\mathcal{A}_1)$, that is, the trace distribution of $\mathcal{A}_1 \| \mathcal{C}$ induced by the scheduler $\mathit{observer}(\mathcal{A}_1)$. Define the scheduler σ_2, the probabilistic execution ϵ, and the Θ sets as in the proof of Proposition 5.1.

The definition of R is slightly different: $q_1 \ R \ q_2$ iff there exists a state q_2' such that $q_2 \Longrightarrow q_2'$ and there exists $\alpha \in \Theta_{q_1,q_2'}$ such that $\epsilon(C_\alpha) > 0$. We claim that R is a weak forward simulation from $\mathcal{A}_1$ to $\mathcal{A}_2$.

For the start condition, we must show that $\bar{q}_1 \ R \ \bar{q}_2$. By Item 1 of Proposition 4.6, $\eta(C_{\bar{q}_1}) = 1$. This means that there exists a finite execution fragment α of $\mathcal{A}_2 \| \mathcal{C}$ with trace $\bar{q}_1$ that ends with action $\bar{q}_1$, such that $\epsilon(C_\alpha) > 0$. By definition of $\mathcal{C}$, the last state of α is $(q_2, \bar{q}_1)$ for some state q_2 satisfying $\bar{q}_2 \Longrightarrow q_2$. By definition of R, $\bar{q}_1 \ R \ \bar{q}_2$ as needed.

For the step condition, assume $q_1 \ R \ q_2$ and let $q_1 \xrightarrow{a}_1 q_1'$ be a transition of $\mathcal{A}_1$, which we denote by tr. We exhibit a matching weak transition $q_2 \stackrel{a}{\Longrightarrow}_2 q_2'$.

By definition of R, there exists a state q_2'' of $\mathcal{A}_2$ such that $q_2 \Longrightarrow q_2''$ and there exists a finite execution α in $\Theta_{q_1,q_2''}$ such that $\epsilon(C_\alpha) > 0$. Since $\Theta_{q_1,q_2''}$ is a subset of Θ_{q_1}, by definition of Θ_{q_1}, $\mathit{trace}(\alpha) = \beta q_1$ for some finite trace β. Therefore, $\eta(C_{\beta q_1}) > 0$. Since q_1 enables at least one transition in $\mathcal{A}_1$, specifically transition tr, Equation (6) from Proposition 4.6 implies that $\eta(C_{\beta q_1 \, ch}) = \eta(C_{\beta q_1})$. Thus, there exists an execution fragment α' of $\mathcal{A}_2 \| \mathcal{C}$ with trace ch such that $\epsilon(C_{\alpha^\frown \alpha'}) > 0$. Furthermore, since, by definition of $\mathcal{C} = \mathit{tester}(\mathcal{A}_1)$, the transition of $\mathcal{C}$ labeled by ch that leaves from state q_1 leads to state tr with non-zero probability, we can assume that the last state of α' is of the form (q', tr) for some state q' of $\mathcal{A}_2$.

Recall from above that $\eta(C_{\beta q_1 \, ch}) > 0$. By Equation (7) from Proposition 4.6, $\eta(C_{\beta q_1 \, ch}) = \sum_{(a,q') | a \in E, q \xrightarrow{a} q'} \eta(C_{\beta q_1 \, ch \, a q'}) + \sum_{q' | (\exists a) a \in H, q_1 \xrightarrow{a} q'} \eta(C_{\beta q_1 \, ch \, q'})$. Hence, σ_2 must extend $\alpha^\frown \alpha'$ in such a way that the first or the first two external actions are compatible with one of the transitions of $\mathcal{A}_1$ that are enabled from q_1. (The number of external actions depends on whether the compatible transition of $\mathcal{A}_1$ is labeled by an internal or external action.) Since state tr of $\mathcal{C}$ enables only action q_1', and since, by the tree-structure of $\mathcal{A}_1$, a is uniquely determined by q_1', the first or first two external actions of $\mathcal{A}_2 \| \mathcal{C}$ scheduled by σ_2 are either q_1' or $a q_1'$ depending on whether a is internal

or external. Thus, there exists an execution fragment α'' of $\mathcal{A}_2 \| \mathcal{C}$, with trace $trace(aq_1')$, such that $\epsilon(C_{\alpha \frown \alpha' \frown \alpha''}) > 0$. Furthermore, we can assume that the last transition of α'' is labeled by q_1' (simply truncate α'' otherwise).

Let (q_2', q_1') be the last state of α''. Then, $\alpha \frown \alpha' \frown \alpha'' \in \Theta_{q_1', q_2'}$, thus showing that $q_1' \ R \ q_2'$. It remains to show that $q_2 \stackrel{a}{\Longrightarrow} q_2'$. For this, it suffices to recall that $q_2 \Longrightarrow q_2''$ and observe that $q_2'' \stackrel{a}{\Longrightarrow} q_2'$ since the execution fragment $(\alpha' \frown \alpha'') \lceil \mathcal{A}_2$ has trace $trace(a)$, first state q_2'', and last state q_2'. □

Using the same approach as before, we may eliminate the assumption in Proposition 5.3 that $\mathcal{A}_1$ is tree-structured.

Theorem 5.4 *Let $\mathcal{A}_1$, $\mathcal{A}_2$ be nondeterministic automata. Then $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$ if and only if $\mathcal{A}_1 \leq_{wF} \mathcal{A}_2$.*

Proof. Analogous to the proof of Theorem 5.2. First we prove soundness of weak forward simulations:

$$\begin{array}{lll} \mathcal{A}_1 \leq_{wF} \mathcal{A}_2 & \Rightarrow & \text{(Proposition 3.9, Part 2)} \\ \mathcal{A}_1 \leq_{wPF} \mathcal{A}_2 & \Rightarrow & \text{(Proposition 3.10, Part 3)} \\ \mathcal{A}_1 \leq_{DC} \mathcal{A}_2 \ . & & \end{array}$$

Now we prove completeness:

$$\begin{array}{lll} \mathcal{A}_1 \leq_{DC} \mathcal{A}_2 & \Rightarrow & \text{(Proposition 2.4)} \\ \mathit{Unfold}(\mathcal{A}_1) \leq_F \mathcal{A}_1 \leq_{DC} \mathcal{A}_2 & \Rightarrow & \text{(as in proof Theorem 5.2, soundness part)} \\ \mathit{Unfold}(\mathcal{A}_1) \leq_{DC} \mathcal{A}_1 \leq_{DC} \mathcal{A}_2 & \Rightarrow & (\leq_{DC} \text{ is transitive}) \\ \mathit{Unfold}(\mathcal{A}_1) \leq_{DC} \mathcal{A}_2 & \Rightarrow & \text{(Proposition 5.3)} \\ \mathit{Unfold}(\mathcal{A}_1) \leq_{wF} \mathcal{A}_2 & \Rightarrow & \text{(Proposition 2.4)} \\ \mathcal{A}_1 \leq_F \mathit{Unfold}(\mathcal{A}_1) \leq_{wF} \mathcal{A}_2 & \Rightarrow & \text{(Proposition 2.3, Part 1)} \\ \mathcal{A}_1 \leq_{wF} \mathit{Unfold}(\mathcal{A}_1) \leq_{wF} \mathcal{A}_2 & \Rightarrow & (\leq_{wF} \text{ is transitive}) \\ \mathcal{A}_1 \leq_{wF} \mathcal{A}_2 \ . & & \end{array}$$

□

6 Characterizations of $\leq_{DC}$ for Probabilistic Automata

Now we present our characterization theorems for $\leq_{DC}$ for probabilistic automata: Theorem 6.3 characterizes $\leq_{DC}$ in terms of $\leq_{PF}$, for PAs without internal actions, and Theorem 6.5 characterizes $\leq_{DC}$ in terms of $\leq_{wPF}$, for arbitrary probabilistic automata. Again, we give the results first for tree-structured probabilistic automata and extend them by unfolding. As before, the interesting direction is the completeness direction, showing that $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ implies the existence of a simulation relation from $\mathcal{P}_1$ to $\mathcal{P}_2$. Our proofs of completeness for PAs are analogous to those for nondeterministic automata.

6.1 Probabilistic Automata Without Internal Actions

We first consider tree-structured probabilistic automata.

Proposition 6.1 *Let $\mathcal{P}_1$, $\mathcal{P}_2$ be probabilistic automata without internal actions such that $\mathcal{P}_1$ is tree-structured. Then $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ implies $\mathcal{P}_1 \leq_{PF} \mathcal{P}_2$.*

Proof. Assume that $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$. Let $\mathcal{C}$ be $tester(\mathcal{P}_1)$ and η be $observation(\mathcal{P}_1)$, that is, the trace distribution of $\mathcal{P}_1 \| \mathcal{C}$ induced by the scheduler $observer(\mathcal{P}_1)$. Define the scheduler σ_2, the probabilistic execution ϵ, and the Θ sets as in the proof of Proposition 5.1.

Define a relation R as follows: $q_1 \; R \; \mu_2$ if and only if $\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha) > 0$ and for each state $q_2 \in Q_2$,

$$\mu_2(q_2) = \frac{\sum_{\alpha \in \Theta_{q_1,q_2}} \epsilon(C_\alpha)}{\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha)}. \tag{12}$$

That is, the measure μ_2 describes probabilities of the various Θ_{q_1,q_2}'s relative to Θ_{q_1}. Note that the equation above is well defined since, by the tree-structure of $\mathcal{P}_1$, all the cones represented by Θ_{q_1} are disjoint, and thus $\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha) \leq 1$. We claim that R is a probabilistic forward simulation from $\mathcal{P}_1$ to $\mathcal{P}_2$.

Before proving that R is a probabilistic forward simulation we make several observations.

1. Relation R is a function from Q_1 to $Disc(Q_2)$.

   Indeed, if $\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha) > 0$, then there exists exactly one measure that satisfies Equation (12). Furthermore, given the construction of η and the fact that $\mathcal{P}_1$ is tree-structured (i.e., all states are reachable), every state q_1 of Q_1 occurs with some positive probability in η. Thus, since η is induced by ϵ, $\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha) > 0$ for all states q_1 of Q_1.

2. If $q_1 \; R \; \mu_2$, then, for each state $q_2 \in Q_2$ and each execution $\alpha \in \Theta_{q_1,q_2}$,

   $$\epsilon(C_\alpha) > 0 \quad \Rightarrow \quad q_2 \in supp(\mu_2). \tag{13}$$

   That is, the execution α occurs with non-zero probability in ϵ only if μ_2 assigns non-zero probability to q_2. This property is a direct consequence of Equation (12).

3. For each transition $q_1 \xrightarrow{a} \mu_1'$ of $\mathcal{P}_1$, the following equation holds:

   $$\mu_1'(q_1') = \frac{\sum_{\alpha \in \Theta_{q_1'}} \epsilon(C_\alpha)}{\sum_{q \in supp(\mu_1'), \alpha \in \Theta_q} \epsilon(C_\alpha)}. \tag{14}$$

   That is, the relative probabilities of the states of $supp(\mu_1')$ in ϵ are given by μ_1'. This result follows by instantiating Equation (9) from Proposition 4.7 with $q_1 \xrightarrow{a} \mu_1'$ to derive the probability of a state q_1' in the support of μ_1', and by replacing the diamond expressions according to Equation (2) from Proposition 3.3.

4. For each transition $tr = q_1 \xrightarrow{a} \mu_1'$ of $\mathcal{P}_1$, the following equation holds:

   $$\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha) = k \sum_{q \in supp(\mu_1'), \alpha \in \Theta_q} \epsilon(C_\alpha), \tag{15}$$

   where k is the number of transitions of $\mathcal{P}_1$ enabled from q_1. That is, the probability of reaching q_1 in ϵ is k times the probability of reaching q_1 and scheduling tr. Informally, transition tr is scheduled only if state q_1 is reached and the outcome of the following transition labeled by ch is tr, which happens with probability $1/k$. The reason why $\sum_{q \in supp(\mu_1'), \alpha \in \Theta_q} \epsilon(C_\alpha)$ is the probability of reaching q_1 and scheduling tr is that states from $supp(\mu_1')$ can occur only after q_1 has occurred and tr is reached (see the definition of tester automaton and of observer of a

tester automaton) and furthermore states from $supp(\mu_1')$ occur with probability 1 once tr is reached (see Equation (7) from Proposition 4.6).

This result follows by instantiating Equation (8) from Proposition 4.7 with tr, replacing the diamond expressions according to Equation (2) from Proposition 3.3, summing over $supp(\mu_1')$, observing that $\sum_{q_1' \in supp(\mu_1')} \mu_1'(q_1') = 1$, and deriving $\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha)$ from the resulting equation.

We are now ready to show that R is a probabilistic forward simulation. For the start condition, we must show that $\bar{q}_1 \; R \; \delta(\bar{q}_2)$.

Consider the start state $(\bar{q}_2, \bar{q}_\mathcal{C})$ of $\mathcal{P}_2 \| \mathcal{C}$. Since there are no internal actions in $\mathcal{P}_2$ or $\mathcal{C}$, and since, by Equation (5) from Proposition 4.6, $\eta(C_{\bar{q}_1}) = 1$, the only action that is scheduled initially by σ_2 is $\bar{q}_1$, leading to state $(\bar{q}_2, \bar{q}_1)$ with probability 1. Thus, the finite execution $\alpha = (\bar{q}_2, \bar{q}_\mathcal{C})\bar{q}_1(\bar{q}_2, \bar{q}_1)$ is an element of $\Theta_{\bar{q}_1, \bar{q}_2}$ such that $\epsilon(C_\alpha) = 1$, and, by definition of R, $\bar{q}_1 \; R \; \delta(\bar{q}_2)$ as needed.

For the step condition, assume that $q_1 \; R \; \mu_2$ and let $q_1 \xrightarrow{a}_1 \mu_1'$ be a transition of $\mathcal{P}_1$, which we denote by tr. We must exhibit a probability measure $\xi_2' \in Disc(Disc(Q_2))$ and a hyper-transition $\mu_2 \xrightarrow{a}_2 \mu_2''$, matching the given transition, where $\mu_2'' = flatten(\xi_2')$ and $\mu_1' \; R \; \xi_2'$. We do this by deriving a transition tr_α of $\mathcal{P}_2$ for each execution α of Θ_{q_1} and by combining the tr_α's appropriately into transitions tr_q, for each state $q \in supp(\mu_2)$, that are the basis for the required hyper-transition. The tr_α transitions are derived from η; the construction considers only those α's for which $\epsilon(C_\alpha) > 0$. The other α's can be treated arbitrarily.

Consider an execution α of Θ_{q_1} such that $\epsilon(C_\alpha) > 0$. By Property (13), $\alpha \in \Theta_{q_1, q_2}$ for some state q_2 in $supp(\mu_2)$. Since Θ_{q_1,q_2} is a subset of Θ_{q_1}, by definition of Θ_{q_1}, $trace(\alpha) = \beta q_1$ for some finite trace β. Therefore, $\eta(C_{\beta q_1}) > 0$. Since q_1 enables at least one transition in $\mathcal{P}_1$, specifically transition tr, Equation (6) from Proposition 4.6 implies that $\eta(C_{\beta q_1\, ch}) = \eta(C_{\beta q_1})$. Then, since $\mathcal{P}_2$ and $\mathcal{C}$ have no internal actions, σ_2 schedules action ch from α with probability 1.

By definition of $\mathcal{C} = tester(\mathcal{P}_1)$, the transition labeled by ch that leaves from state q_1 of $\mathcal{C}$ leads to state tr with non-zero probability. Therefore, $\epsilon(C_{\alpha\, ch\, (q_2, tr)}) > 0$. By Equation (7) from Proposition 4.6, where only the first term of the right-hand side is non-zero due to the absence of internal actions, $\eta(C_{\beta q_1\, ch}) = \sum_{(a,q') | a \in E, q_1 \xrightarrow{a} q'} \eta(C_{\beta q_1\, ch\, a q'})$. Hence, σ_2 must extend $\alpha\, ch\, (q_2, tr)$ with two steps labeled by an action and a state of $\mathcal{P}_1$, respectively, where the action and the state are compatible with one of the transitions of $\mathcal{P}_1$ that are enabled from q_1. Since state tr of $\mathcal{C}$ enables only actions in $supp(\mu_1')$, and since, by the tree-structure of $\mathcal{P}_1$, a is uniquely determined by μ_1', the action that is scheduled is a and the state that is scheduled is a state in $supp(\mu_1')$. Thus, $\sigma_2(\alpha\, ch\, (q_2, tr))$ returns a probability measure over transitions labeled by a. This measure identifies a combined transition of $\mathcal{P}_2$ labeled by a that leaves from q_2, which we denote by tr_α.

Now, using the tr_α transitions, we define a combined transition from each state in the support of μ_2. Namely, for each state $q \in supp(\mu_2)$, let tr_q be the combined transition of $\mathcal{P}_2$ defined by:

$$tr_q \stackrel{\Delta}{=} \sum_{\alpha \in \Theta_{q_1,q}} \frac{\epsilon(C_\alpha)}{\sum_{\alpha' \in \Theta_{q_1,q}} \epsilon(C_{\alpha'})} tr_\alpha. \tag{16}$$

Informally, each element of $\Theta_{q_1,q}$ is an execution that contributes to the emulation of transition $q_1 \xrightarrow{a}_1 \mu_1'$ from q. Equation (16) computes tr_q, the overall contribution to the emulation from q, by averaging over all elements of $\Theta_{q_1,q}$. We could prove that $\Theta_{q_1,q}$ contains only one element α' such that $\epsilon(C_{\alpha'}) > 0$ and simplify Equation (16) accordingly. However, this simplification is not

necessary for the proof. Now we define the measure $\mu_2'' \in Disc(Q_2)$:

$$\mu_2'' \stackrel{\Delta}{=} \sum_{q \in supp(\mu_2)} \mu_2(q)\mu_{tr_q}. \tag{17}$$

Then, by construction, $\mu_2 \xrightarrow{a} \mu_2''$ is a hyper-transition of $\mathcal{P}_2$.

It remains to define a probability measure $\xi_2' \in Disc(Disc(Q_2))$ such that $\mu_2'' = \mathit{flatten}(\xi_2')$ and $\mu_1' \; R \; \xi_2'$.

For each $q \in supp(\mu_1')$, let μ_q be the unique measure such that $q \; R \; \mu_q$. We can identify μ_q because R is a function. Define $\xi_2' \in Disc(Disc(Q_2))$ such that, for each $q \in supp(\mu_1')$, $\xi_2'(\mu_q) = \sum_{q' \in supp(\mu_1') | \mu_{q'} = \mu_q} \mu_1'(q')$. Then $\mu_1' \; R \; \xi_2'$ by definition of ξ_2'.

It remains to show that $\mu_2'' = \mathit{flatten}(\xi_2')$, that is, that $\mu_2'' = \sum_{\rho \in supp(\xi_2')} \xi_2'(\rho)\rho$. From the definition of ξ_2' and of the flatten operator, it suffices to show that for every $q_2 \in Q_2$,

$$\mu_2''(q_2) = \sum_{q \in supp(\mu_1')} \mu_1'(q)\mu_q(q_2). \tag{18}$$

To prove Equation (18) we first claim that the following equation is valid for each pair of states q_1, q_2 of $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, if k denotes the number of transitions of $\mathcal{P}_1$ that are enabled from q_1:

$$\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha)\mu_{tr_\alpha}(q_2) = k \sum_{q \in supp(\mu_1'), \alpha \in \Theta_{q,q_2}} \epsilon(C_\alpha). \tag{19}$$

Informally, the left-hand side of Equation (19) represents the probability of scheduling q_1 and then reaching q_2 according to the transition tr_α, without considering the outcome of the transition labeled by ch. The right-hand side, on the other hand, computes the probability of scheduling q_1, scheduling ch and reaching μ_1', and then scheduling tr_α and reaching q_2. State μ_1' is reached by ch with probability $1/k$, which justifies the k factor in the right-hand side.

To prove Equation (19), consider an execution $\alpha \in \Theta_{q,q_2}$ where $q \in supp(\mu_1')$. Since q occurs always after q_1, execution α can be split into $\alpha' \frown \alpha''$ where $\alpha' \in \Theta_{q_1}$. Furthermore, $trace(\alpha'') = ch\, aq$, and since there are no internal actions in $\mathcal{P}_2$ and $\mathcal{C}$, α is the unique extension of α' that is in Θ_{q,q_2}. In particular, $\alpha'' = (q', q_1)\, ch\, (q', tr)a(q_2, tr)q(q_2, q)$ for some state q' of $\mathcal{P}_2$, and $\epsilon(C_\alpha) = \epsilon(C_{\alpha'})(1/k)\mu_{tr_{\alpha'}}(q_2)$. Thus, each summand in the right-hand side of Equation (19) has a corresponding summand in the left-hand side that differs by a factor of k, and the correspondence relation is an injection. If the correspondence is not a bijection, then the α terms that are left out on the left-hand side are such that $\mu_{tr_\alpha}(q_2) = 0$ (otherwise an extension in Θ_{q,q_2} for some q exists). This suffices.

We now consider the left-hand side of Equation (18). Consider the definition of μ_2'' given by Equation (17). By expanding $\mu_2(q)$ according to the definition of μ_2 given by Equation (12), and expanding $\mu_{tr}(q_2)$ according to the definition of μ_{tr} given by Equation (16), we obtain

$$\mu_2''(q_2) = \sum_{q \in supp(\mu_2)} \frac{\sum_{\alpha \in \Theta_{q_1,q}} \epsilon(C_\alpha)}{\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha)} \frac{\sum_{\alpha \in \Theta_{q_1,q}} \epsilon(C_\alpha)\mu_{tr_\alpha}(q_2)}{\sum_{\alpha \in \Theta_{q_1,q}} \epsilon(C_\alpha)}.$$

By cross simplifying the top leftmost and bottom rightmost factors, and by factoring the left denominator out of the sum, we obtain

$$\mu_2''(q_2) = \frac{\sum_{q \in supp(\mu_2)} \sum_{\alpha \in \Theta_{q_1,q}} \epsilon(C_\alpha)\mu_{tr_\alpha}(q_2)}{\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha)}.$$

By Property (13), we can rewrite the numerator as follows:

$$\mu_2''(q_2) = \frac{\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha) \mu_{tr_\alpha}(q_2)}{\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha)}.$$

By multiplying numerator and denominator by k, applying Equation (19) to the numerator, and applying Equation (15) to the denominator, we obtain

$$\mu_2''(q_2) = \frac{\sum_{q \in supp(\mu_1'),\, \alpha \in \Theta_{q,q_2}} \epsilon(C_\alpha)}{\sum_{q \in supp(\mu_1'),\, \alpha \in \Theta_q} \epsilon(C_\alpha)}. \tag{20}$$

We now consider the right-hand side of Equation (18). By applying Equations (14) and (12) to the two factors of the right-hand side of Equation (18), and by simplifying common factors algebraically, we obtain

$$\sum_{q \in supp(\mu_1')} \mu_1'(q) \mu_q(q_2) = \frac{\sum_{q \in supp(\mu_1'),\, \alpha \in \Theta_{q,q_2}} \epsilon(C_\alpha)}{\sum_{q \in supp(\mu_1'),\, \alpha \in \Theta_q} \epsilon(C_\alpha)}. \tag{21}$$

Now Equation (18) follows by direct combination of Equations (20) and (21). □

Interestingly, the probabilistic forward simulation that we constructed in the above proof is functional. Functional simulations are usually called *refinement mappings* [21, 26]. Write $\mathcal{P}_1 \leq_{PR} \mathcal{P}_2$ if there exists a functional probabilistic forward simulations from $\mathcal{P}_1$ to $\mathcal{P}_2$. Then we can state the following new proposition, which is a probabilistic version of Proposition 3.12 in [26]:

Proposition 6.2 *Let $\mathcal{P}_1$, $\mathcal{P}_2$ be probabilistic automata without internal actions such that $\mathcal{P}_1$ is tree-structured. Then $\mathcal{P}_1 \leq_{PF} \mathcal{P}_2$ iff $\mathcal{P}_1 \leq_{PR} \mathcal{P}_2$.*

Proof. It is enough to observe that each state q_1 of $\mathcal{P}_1$ occurs with some positive probability in the trace distribution η of the proof of Proposition 6.1. □

As usual, we may eliminate the assumption that $\mathcal{P}$ is tree-structured.

Theorem 6.3 *Let $\mathcal{P}_1$, $\mathcal{P}_2$ be probabilistic automata without internal actions. Then $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ if and only if $\mathcal{P}_1 \leq_{PF} \mathcal{P}_2$.*

Proof. First we prove soundness of probabilistic forward simulations:

$$\begin{array}{lcl}
\mathcal{P}_1 \leq_{PF} \mathcal{P}_2 & \Rightarrow & \text{(Proposition 3.10, Part 1)} \\
\mathcal{P}_1 \leq_{wPF} \mathcal{P}_2 & \Rightarrow & \text{(Proposition 3.10, Part 3)} \\
\mathcal{P}_1 \leq_{DC} \mathcal{P}_2 \ . &&
\end{array}$$

Now we prove completeness:

$$\begin{array}{lcl}
\mathcal{P}_1 \leq_{DC} \mathcal{P}_2 & \Rightarrow & \text{(Proposition 3.12)} \\
\mathit{Unfold}(\mathcal{P}_1) \leq_{DC} \mathcal{P}_1 \leq_{DC} \mathcal{P}_2 & \Rightarrow & (\leq_{DC} \text{ is transitive}) \\
\mathit{Unfold}(\mathcal{P}_1) \leq_{DC} \mathcal{P}_2 & \Rightarrow & \text{(Proposition 6.1)} \\
\mathit{Unfold}(\mathcal{P}_1) \leq_{PF} \mathcal{P}_2 & \Rightarrow & \text{(Proposition 3.11)} \\
\mathcal{P}_1 \leq_{PF} \mathit{Unfold}(\mathcal{P}_1) \leq_{PF} \mathcal{P}_2 & \Rightarrow & (\leq_{PF} \text{ is transitive}) \\
\mathcal{P}_1 \leq_{PF} \mathcal{P}_2 \ . &&
\end{array}$$

□

6.2 Probabilistic Automata With Internal Actions

Again, we start with tree-structured PAs.

Proposition 6.4 *Let $\mathcal{P}_1$, $\mathcal{P}_2$ be probabilistic automata with $\mathcal{P}_1$ tree-structured. Then $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ implies $\mathcal{P}_1 \leq_{wPF} \mathcal{P}_2$.*

Proof. Assume that $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$. Define the tester probabilistic automaton $\mathcal{C}$ of $\mathcal{P}_1$, the observer σ_1, the trace distribution η, the scheduler σ_2, the probabilistic execution ϵ, and the Θ sets as in the proof of Proposition 5.1. Define relation R according to Equation (12) as in the proof of Proposition 6.1. Observe that Property (13) and Equations (14) and (15) hold for the same reasons as before. Define a new relation R' as follows: $q_1\ R'\ \mu_2$ iff there exists a measure μ_2' such that $\mu_1 \Longrightarrow \mu_2'$ and $q_1\ R\ \mu_2'$. Observe that trivially $R \subseteq R'$. We show that R' is a weak probabilistic forward simulation from $\mathcal{P}_1$ to $\mathcal{P}_2$.

For the start condition, we must show that $\bar{q}_1\ R'\ \delta(\bar{q}_2)$. By Item 1 of Proposition 4.6, $\eta(C_{\bar{q}_1}) = 1$. This means that $\sum_{\alpha \in \Theta_{\bar{q}_1} | trace(\alpha) = \bar{q}_1} \epsilon(C_\alpha) = 1$. Let $\Theta'_{\bar{q}_1}$ be the set of elements of $\Theta_{\bar{q}_1}$ with trace $\bar{q}_1$, and let ϵ' be the truncation of ϵ to $\Theta'_{\bar{q}_1}$. Then ϵ' assigns probability 1 to the set of finite execution fragments with trace $\bar{q}_1$. Furthermore, observing that all elements of $\Theta'_{\bar{q}_1}$ are not prefixes of each other, we derive $\epsilon'(C_\alpha) = \epsilon'(\{\alpha\})$ for each $\alpha \in \Theta'_{\bar{q}_1}$. Finally, observing that each element of $\Theta_{\bar{q}_1} - \Theta'_{\bar{q}_1}$ is not a prefix of any element of $\Theta'_{\bar{q}_1}$, we derive $\epsilon'(C_\alpha) = \epsilon'(\{\alpha\}) = 0$ for each $\alpha \in \Theta_{\bar{q}_1} - \Theta'_{\bar{q}_1}$. Let μ_2' be $lstate(\epsilon')$. By definition of weak hyper-transition, $\delta(\bar{q}_2) \Longrightarrow \mu_2'$. We show that $\bar{q}_1\ R\ \mu_2'$, which suffices. Consider a state q_2 of $\mathcal{P}_2$. By definition of μ_2', definition of $\Theta_{\bar{q}_1, q_2}$, and the fact that $supp(\epsilon') \subseteq \Theta_{\bar{q}_1}$, $\mu_2'(q_2) = \sum_{\alpha \in \Theta_{\bar{q}_1, q_2}} \epsilon'(\{\alpha\})$. Then the result follows immediately by observing that this equation corresponds to Equation (12) since $\epsilon'(\{\alpha\}) = \epsilon'(C_\alpha)$ when $\alpha \in \Theta_{\bar{q}_1}$ and since $\sum_{\alpha \in \Theta_{\bar{q}_1}} \epsilon(C_\alpha) = 1$.

For the step condition, assume that $q_1\ R'\ \mu_2$ and let $q_1 \xrightarrow{a}_1 \mu_1'$ be a transition of $\mathcal{P}_1$, which we denote by tr. By definition of R', there exists a measure μ_2' such that $\mu_2 \Longrightarrow \mu_2'$ and $q_1\ R\ \mu_2'$. We now show that there exists a measure ξ_2' and a measure $\mu_2'' = flatten(\xi_2')$ such that $\mu_1\ R\ \xi_2'$ and $\mu_2' \overset{a}{\Longrightarrow} \mu_2''$. Then $\mu_1\ R'\ \xi_2'$, and by Proposition 3.6, $\mu_2 \overset{a}{\Longrightarrow} \mu_2''$.

The proof of existence of ξ_2' and μ_2'' proceeds exactly as in the case of Proposition 6.1 except for the definition of the tr_α transitions. Thus, in the rest of the proof we construct the tr_α's and prove that Equation (19) still holds.

We introduce a special *conditional* construction that is needed for the definition of the tr_α's. Let $\mathcal{C}_{tr}$ be the same as $\mathcal{C}$ except that the transition $q_1 \xrightarrow{ch} \mu$, where μ is uniquely determined by q_1, is replaced by $q_1 \xrightarrow{ch} \delta(tr)$. Given a scheduler σ for $\mathcal{P}_2 \| \mathcal{C}$, define the scheduler $\sigma \mid tr$ for $\mathcal{P}_2 \| \mathcal{C}_{tr}$ that is the same as σ except that transition $q_1 \xrightarrow{ch} \delta(tr)$ of $\mathcal{C}_{tr}$ is chosen whenever σ chooses $q_1 \xrightarrow{ch} \mu$. Given a probabilistic execution fragment ϵ' of $\mathcal{P}_2 \| \mathcal{C}$, generated by some scheduler σ, define $\epsilon' \mid tr$ to be the result of $\sigma \mid tr$ applied to $\mathcal{P} \| \mathcal{C}_{tr}$ from the start state of ϵ'. The intuition behind $\epsilon' \mid tr$ is that we study ϵ' under the condition that tr is the outcoming state of $\mathcal{C}$ whenever $q_1 \xrightarrow{ch} \mu$ is scheduled. Then, the following two properties are valid.

1. $(\epsilon' \mid tr) \lceil \mathcal{P}_2$ is a probabilistic execution fragment of $\mathcal{P}_2$.
2. For each finite execution fragment α of $\mathcal{P}_2 \| \mathcal{C}$ where state tr occurs and such that $fstate(\alpha)$ is not of the form $(\cdot, tr)$, $(\epsilon' \mid tr)(C_\alpha) = k\epsilon(C_\alpha)$, where k is the size of $supp(\mu)$.

The first item follows immediately from Proposition 3.7 given that $\epsilon' \mid tr$ is a probabilistic execution fragment of $\mathcal{P}_2 \| \mathcal{C}_{tr}$. The second item follows directly from the definition of probability of a cone

since in ϵ' the probability associated with the edge $q\ ch\ (\cdot, tr)$ is $1/k$ while in $\epsilon' \mid tr$ the probability of the same edge is 1.

We now define the tr_α's. Consider an execution α of Θ_{q_1} such that $\epsilon(C_\alpha) > 0$. Let ϵ^1 be the truncation of ϵ at all the points in $\cup_{q \in supp(\mu_1')} \Theta_q$, which is a probabilistic execution of $\mathcal{P}_2 \| \mathcal{C}$ by definition. Let ϵ^1_α be $\epsilon^1 \triangleright \alpha$, which is a probabilistic execution fragment of $\mathcal{P}_2 \| \mathcal{C}$ by definition Finally, let ϵ^2_α be $(\epsilon^1_\alpha \mid tr) \lceil \mathcal{P}_2$, which is a probabilistic execution fragment of $\mathcal{P}_2$ by Property 1.

By definition of Θ_{q_1}, $trace(\alpha) = \beta q_1$ for some finite trace β. Therefore, $\eta(C_{\beta q_1}) > 0$. Since q_1 enables at least one transition in $\mathcal{P}_1$, specifically transition tr, Equation (6) from Proposition 4.6 implies that $\eta(C_{\beta q_1\ ch}) = \eta(C_{\beta q_1})$. Thus, action ch occurs as the first external action with probability 1 in μ^1_α.

By Equation (7) from Proposition 4.6, if the occurrence of action ch leads $\mathcal{C}$ to state tr, then an action in $supp(\mu_1')$ occurs eventually in ϵ with probability 1, leading $\mathcal{C}$ to a state in $supp(\mu_1')$, which is a truncation point according to the definition of ϵ^1. Thus, the probability of termination in $\epsilon^1_\alpha \mid tr$ is 1, as well as the probability of termination in ϵ^2_α, that is, ϵ^2_α assigns probability 1 to the set of finite executions. Furthermore, given that action a is uniquely determined by μ_1' ($\mathcal{P}_1$ is tree-structured), again by Equation (7) from Proposition 4.6 all finite executions α' with $\epsilon^2_\alpha(\alpha') > 0$ have trace $trace(a)$. Thus, ϵ^2_α is a representation of a weak combined transition labeled by a from $lstate(\alpha) \lceil \mathcal{P}_2$. Denote such transition by tr_α.

We are left to show that Equation (19) still holds. That is,

$$\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha) \mu_{tr_\alpha}(q_2) = k \sum_{q \in supp(\mu_1')\,,\, \alpha \in \Theta_{q,q_2}} \epsilon(C_\alpha).$$

We consider first the term $\mu_{tr_\alpha}(q_2)$. From the definition of tr_α and of weak combined transition we get

$$\mu_{tr_\alpha}(q_2) = \sum_{\alpha' \mid lstate(\alpha') = q_2} \epsilon^2_\alpha(\alpha').$$

By applying the definition of projection, and using the fact that $\epsilon^1_\alpha \mid tr$ assigns probability 1 to the set of finite executions, we get

$$\mu_{tr_\alpha}(q_2) = \sum_{\alpha' \mid lstate(\alpha' \lceil \mathcal{P}_2) = q_2} (\epsilon^1_\alpha \mid tr)(\alpha').$$

Given that the truncation points of ϵ^1 are all at the $\cup_{q \in supp(\mu_1')} \Theta_q$ points, the only finite executions α' that have non-zero probability are such that $\alpha \frown \alpha'$ is in some set Θ_q. Furthermore, given that no execution in $\cup_{q \in supp(\mu_1')} \Theta_q$ is a prefix of another (our PAs are tree-structured and all actions in $supp(\mu_1')$ occur in different branches), the probabilities of the finite executions can be replaced by the probabilities of their cones, thus getting

$$\mu_{tr_\alpha}(q_2) = \sum_{q \in supp(\mu_1')} \sum_{\alpha' \mid \alpha \frown \alpha' \in \Theta_{q,q_2}} (\epsilon^1_\alpha \mid tr)(C_{\alpha'}).$$

By Property 2 we can get rid of the conditional on tr by introducing a k factor, thus getting

$$\mu_{tr_\alpha}(q_2) = \sum_{q \in supp(\mu_1')} \sum_{\alpha' \mid \alpha \frown \alpha' \in \Theta_{q,q_2}} k \epsilon^1_\alpha(C_{\alpha'}). \tag{22}$$

By replacing $\mu_{tr_\alpha}(q_2)$ according to Equation (22) in the left-hand side of Equation (19), and by rearranging terms algebraically, we obtain

$$\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha)\mu_{tr_\alpha}(q_2) = k \sum_{q \in supp(\mu_1')} \sum_{\alpha \in \Theta_{q_1}} \sum_{\alpha' | \alpha^\frown \alpha' \in \Theta_{q,q_2}} \epsilon(C_\alpha)\epsilon^1_\alpha(C_{\alpha'}).$$

By using that $\epsilon^1_\alpha = \epsilon^1 \triangleright \alpha$ (by definition) and Proposition 3.15, the two probabilities in the equation above can be grouped into $\epsilon(C_{\alpha^\frown\alpha'})$. By observing that all elements in Θ_{q,q_2}, with $q \in supp(\mu_1')$, have a prefix in Θ_{q_1}, the intermediate sum can be removed, thus getting

$$\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha)\mu_{tr_\alpha}(q_2) = k \sum_{q \in supp(\mu_1')} \sum_{\alpha \in \Theta_{q,q_2}} \epsilon(C_\alpha),$$

which is Equation (19) as needed. □

Theorem 6.5 *Let $\mathcal{P}_1$, $\mathcal{P}_2$ be probabilistic automata. Then $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ if and only if $\mathcal{P}_1 \leq_{wPF} \mathcal{P}_2$.*

Proof. Soundness of weak probabilistic forward simulations follows immediately from Proposition 3.10. Completeness is established by:

$$\begin{array}{lcl}
\mathcal{P}_1 \leq_{DC} \mathcal{P}_2 & \Rightarrow & \text{(Proposition 3.12)} \\
\mathit{Unfold}(\mathcal{P}_1) \leq_{DC} \mathcal{P}_1 \leq_{DC} \mathcal{P}_2 & \Rightarrow & (\leq_{DC} \text{ is transitive}) \\
\mathit{Unfold}(\mathcal{P}_1) \leq_{DC} \mathcal{P}_2 & \Rightarrow & \text{(Proposition 6.4)} \\
\mathit{Unfold}(\mathcal{P}_1) \leq_{wPF} \mathcal{P}_2 & \Rightarrow & \text{(Proposition 3.11)} \\
\mathcal{P}_1 \leq_{PF} \mathit{Unfold}(\mathcal{P}_1) \leq_{wPF} \mathcal{P}_2 & \Rightarrow & \text{(Proposition 3.10)} \\
\mathcal{P}_1 \leq_{wPF} \mathit{Unfold}(\mathcal{P}_1) \leq_{wPF} \mathcal{P}_2 & \Rightarrow & (\leq_{wPF} \text{ is transitive}) \\
\mathcal{P}_1 \leq_{wPF} \mathcal{P}_2 \ . & &
\end{array}$$

□

7 Concluding Remarks

We have characterized the trace distribution precongruence for nondeterministic and probabilistic automata, with and without internal actions, in terms of four kinds of simulation relations, $\leq_F$, $\leq_{wF}$, $\leq_{PF}$, and $\leq_{wPF}$. In particular, this shows that probabilistic contexts are capable of observing all the distinctions that can be expressed using these simulation relations. Our main technical contribution is the definition of special contexts, called testers, that, under the action of an appropriate scheduler, can reveal the branching structure of a probabilistic automaton via a trace distribution. Some technical improvements are possible. For example, our finite branching restriction can be relaxed to countable branching, simply by replacing uniform distributions in the tester automata by other distributions such as exponential distributions. Calculations become more complicated, however. We have also considered nondeterministic and probabilistic automata with countably many states and actions. Again, this restriction can be relaxed at the cost of complicating the definition of the σ-field of execution fragments: the generators would be arbitrary unions of cones, and the measure of a union of cones would be just the sum of the measures of each single cone. Indeed, discrete transitions and discrete schedulers ensure that there are at most countably many cones with non-zero measure.

Although in this paper we reach a point where we have a full understanding of trace distribution precongruence as a branching relation, a natural question is whether it is possible to define

linear probabilistic extensions of language inclusion. A potential approach is to consider ordinary traces paired with their maximal or minimal probabilities under all schedulers, but the induced preorder relations do not appear to be interesting. Another approach, followed in [19] is to extend classical testing preorders by considering the maximal and minimal probabilities of success of a test; however, even in such case the resulting precongruence is characterized in terms of simulation relations that, although weaker than the relations studied in this paper, are still branching relations. Other approaches ensure compositionality of trace distribution inclusion by restricting parallel composition so that the nondeterminism of each component is resolved based only on externally-visible behavior of the other components. This approach is investigated in [10] in a synchronous model. In [8, 7], an asynchronous switched probabilistic Input/Output automaton model (PIOA) is presented, which uses a token structure to eliminate global nondeterministic choices. This token structure ensures that, at any point in time, there is at most one active component in a system and this unique component determines the next active component. Thus, global scheduling is performed jointly by all local schedulers, which have access to local information only. A notion of switched probabilistic systems is defined, which are switched PIOAs paired with sets of *acceptable* I/O schedulers. A trace-style semantics for switched probabilistic systems is given, using the notion of likelihood assignments. This semantics is shown to be compositional with respect to a parallel operator that combines local I/O schedulers into a joint I/O scheduler. Thus, the approach of [8, 7] can be characterized as *schedule-and-compose*, where local nondeterministic choices are resolved before the components are placed in parallel. In [7] also a similar strategy is pursued, but without the token structure. Instead, several axioms are imposed on the reactive and generative transition structures, so that branching only occurs when it is meant to be globally visible (i.e., the branches carry different visible action labels). These axioms capture a local-oblivious assumption on adversaries, which is well-known in the area of randomized consensus [9, 3]. The model is proven to be compositional with respect to a *schedule-and-compose* operator similar to that in [8].

Acknowledgments. We are especially grateful to one of the anonymous reviewers for numerous detailed observations and suggestions for improvements.

References

[1] S. Aggarwal. Time optimal self-stabilizing spanning tree algorithms. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, May 1994. Available as Technical Report MIT/LCS/TR-632.

[2] S. Andova and T. Willemse. Branching bisimulation for probabilistic systems: characteristics and decidability. *Theoretical Computer Science*, 356(3):325–355, 2006.

[3] Y. Aumann and M.A. Bender. Efficient low-contention asynchronous consensus with the value-oblivious adversary scheduler. *Distributed Computing*, 17(3):191–207, 2005.

[4] F. Bartels, A. Sokolova, and E. de Vink. A hierarchy of probabilistic system types. *Theoretical Computer Science*, 327(1-2):3–22, 2004.

[5] J.A. Bergstra, J.W. Klop, and E.-R. Olderog. Readies and failures in the algebra of communicating processes. *SIAM Journal on Computing*, 17(6):1134–1177, 1988.

[6] L. Cheung. Randomized wait-free consensus using an atomicity assumption. In J.H. Anderson and R. Wattenhofer, editors, *Proceedings OPODIS 2005, Pisa, Italy*, pages 36–45, 2005. Pre-event proceedings. Final proceedings to appear as LNCS.

[7] L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices.* PhD thesis, Radboud University Nijmegen, September 2006.

[8] L. Cheung, N.A. Lynch, R. Segala, and F.W. Vaandrager. Switched PIOA: Parallel composition via distributed scheduling. *Theoretical Computer Science*, 2006. To appear.

[9] B. Chor, A. Israeli, and M. Li. Wait-free consensus using asynchronous hardware. *SIAM Journal on Computing*, 23(4):701–712, 1994.

[10] L. de Alfaro, T.A. Henzinger, and R. Jhala. Compositional methods for probabilistic systems. In K.G. Larsen and M. Nielsen, editors, *Proceedings CONCUR 01,* Aalborg, Denmark, August 20-25, 2001, volume 2154 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2001.

[11] W. Feller. *An Introduction to Probability Theory and its Applications. Volume 1.* John Wiley & Sons, Inc., 1950.

[12] R.J. van Glabbeek. The linear time — branching time spectrum I. The semantics of concrete, sequential processes. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, pages 3–99. North-Holland, 2001.

[13] R.J. van Glabbeek, S.A. Smolka, and B. Steffen. Reactive, generative, and stratified models of probabilistic processes. *Information and Control*, 121(1):59–80, 1995.

[14] J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, October 1992.

[15] H.A. Hansson. *Time and Probability in Formal Design of Distributed Systems*, volume 1 of *Real-Time Safety Critical Systems.* Elsevier, 1994.

[16] A. Hinton, M.Z. Kwiatkowska, G. Norman, and D. Parker. Prism: A tool for automatic verification of probabilistic systems. In H. Hermanns and J. Palsberg, editors, *TACAS*, volume 3920 of *Lecture Notes in Computer Science*, pages 441–444. Springer-Verlag, 2006.

[17] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, Englewood Cliffs, 1985.

[18] B. Jonsson and K.G. Larsen. Specification and refinement of probabilistic processes. In *Proceedings* 6^{th} *Annual Symposium on Logic in Computer Science,* Amsterdam, pages 266–277. IEEE Press, 1991.

[19] B. Jonsson and W. Yi. Testing preorders for probabilistic processes can be characterized by simulations. *Theoretical Computer Science*, 282(1):33–51, 2002.

[20] M.Z. Kwiatkowska and G. Norman. Verifying randomized byzantine agreement. In D. Peled and M.Y. Vardi, editors, *FORTE*, volume 2529 of *Lecture Notes in Computer Science*, pages 194–209. Springer-Verlag, 2002.

[21] L. Lamport. Specifying concurrent program modules. *ACM Transactions on Programming Languages and Systems*, 5(2):190–222, 1983.

[22] K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94, 1991.

[23] N.A. Lynch, I. Saias, and R. Segala. Proving time bounds for randomized distributed algorithms. In *Proceedings of the 13th Annual ACM Symposium on the Principles of Distributed Computing*, pages 314–323, Los Angeles, CA, August 1994.

[24] N.A. Lynch, R. Segala, and F.W. Vaandrager. Compositionality for probabilistic automata. In R. Amadio and D. Lugiez, editors, *Proceedings 14th International Conference on Concurrency Theory (CONCUR 2003),* Marseille, France, volume 2761 of *Lecture Notes in Computer Science*, pages 208–221. Springer-Verlag, September 2003.

[25] N.A. Lynch and M.R. Tuttle. An introduction to input/output automata. *CWI Quarterly*, 2(3):219–246, September 1989.

[26] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations, I: Untimed systems. *Information and Computation*, 121(2):214–233, September 1995.

[27] R. Milner. *Communication and Concurrency.* Prentice-Hall International, Englewood Cliffs, 1989.

[28] G. Norman. Analysing randomized distributed algorithms. In C. Baier, B.R. Haverkort, H. Hermanns, J.-P. Katoen, and M. Siegle, editors, *Validation of Stochastic Systems*, volume 2925 of *Lecture Notes in Computer Science*, pages 384–418. Springer-Verlag, 2004.

[29] A. Philippou, I. Lee, and O. Sokolsky. Weak bisimulation for probabilistic systems. In C. Palamidessi, editor, *Proceedings of CONCUR 2000,* University Park, PA, USA, volume 1877 of *Lecture Notes in Computer Science*, pages 334–349. Springer-Verlag, 2000.

[30] A. Pogosyants, R. Segala, and N.A. Lynch. Verification of the randomized consensus algorithm of Aspnes and Herlihy: a case study. *Distributed Computing*, 13(3):155–186, 2000.

[31] R. Segala. Compositional trace–based semantics for probabilistic automata. In I. Lee and S.A. Smolka, editors, *Proceedings CONCUR 95,* Philadelphia, PA, USA, volume 962 of *Lecture Notes in Computer Science*, pages 234–248. Springer-Verlag, 1995.

[32] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1995. Available as Technical Report MIT/LCS/TR-676.

[33] R. Segala. Testing probabilistic automata. In U. Montanari and V. Sassone, editors, *Proceedings CONCUR 96,* Pisa, Italy, August 26-29, 1996, volume 1119 of *Lecture Notes in Computer Science*, pages 299–314. Springer, 1996.

[34] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

[35] R. Segala and A. Turrini. Comparative analysis of bisimulation relations on alternating and non-alternating probabilistic models. In *QEST*, pages 44–53. IEEE Computer Society, 2005.

[36] A. Sokolova and E.P. de Vink. Probabilistic automata: system types, parallel composition and comparison. In C. Baier et al., editor, *Validation of Stochastic Systems*, volume 2925 of *Lecture Notes in Computer Science*, pages 1–43. Springer-Verlag, 2004.

[37] M.I.A. Stoelinga. *Alea Jacta Est: Verification of Probabilistic, Real-Time and Parametric Systems*. PhD thesis, University of Nijmegen, April 2002.

[38] M.I.A. Stoelinga. An introduction to probabilistic automata. *Bulletin of the European Association for Theoretical Computer Science*, 78:176–198, October 2002.

[39] M.I.A. Stoelinga and F.W. Vaandrager. Root contention in IEEE 1394. In J.-P. Katoen, editor, *Proceedings 5th International AMAST Workshop on Formal Methods for Real-Time and Probabilistic Systems,* Bamberg, Germany, volume 1601 of *Lecture Notes in Computer Science*, pages 53–74. Springer-Verlag, 1999.

[40] 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, Portland, OR, 1985.