# Probability and Nondeterminism in Operational Models of Concurrency

Roberto Segala[⋆]

Dipartimento di Informatica, Università di Verona, Italy

**Abstract.** We give a brief overview of operational models for concurrent systems that exhibit probabilistic behavior, focussing on the interplay between probability and nondeterminism. Our survey is carried out from the perspective of probabilistic automata, a model originally developed for the analysis of randomized distributed algorithms.

## 1 Introduction

The study of randomization in concurrency theory started almost two decades ago, leading to the proposal of several formalisms. In this paper we focus on operational nondeterministic models with discrete probabilities, and we analyze them from the perspective of Probabilistic Automata.

After giving the formal definition of probabilistic automata, we describe other existing proposals as extensions or restrictions of probabilistic automata, thus surveying the existing literature from a uniform point of view. We then turn to the definition of simulation and bisimulation relations. These relations are studied extensively for their mathematical simplicity; yet, several existing definitions appear incomparable. We show how to view the existing definitions based on the definitional style of probabilistic automata. We give several references along the way, including references to other relevant topics that we do not cover explicitly.

## 2 Preliminaries on Measure Theory

We start with some preliminary notions from measure theory. Although we define all the necessary concepts, some familiarity is useful. We refer the reader to any textbook on measure theory in case the use of some concepts is hard to grasp.

A *$\sigma$-field* over a set $\Omega$ is a subset $\mathcal{F}$ of $2^{\Omega}$ that includes the empty set and is closed under complement and countable union. We call the pair $(\Omega, \mathcal{F})$ a *measurable space*. A special $\sigma$-field is the set $2^{\Omega}$, which we call the *discrete* $\sigma$-field over $\Omega$. Given a subset $\mathcal{C}$ of $2^{\Omega}$, we denote by $\sigma(\mathcal{C})$ the smallest $\sigma$-field that includes $\mathcal{C}$, and we call it the $\sigma$-field *generated* by $\mathcal{C}$.

A *measure* over a measurable space $(\Omega, \mathcal{F})$ is a function $\mu : \mathcal{F} \to \mathcal{R}^{\geq 0}$ such that $\mu(\emptyset) = 0$ and, for each countable family $\{X_i\}_I$ of pairwise disjoint elements of $\mathcal{F}$, $\mu(\cup_I X_i) = \sum_I \mu(X_i)$. If $\mu(\Omega) \leq 1$, then we say that $\mu$ is a *sub-probability*

[⋆] Supported by MIUR project AIDA and INRIA project ProNoBiS.

*measure*, and if $\mu(\Omega) = 1$, then we say that $\mu$ is a *probability measure*. If $\mathcal{F}$ is the discrete $\sigma$-field over $\Omega$, then we say that $\mu$ is a *discrete* measure over $\Omega$. In such case, for each set $X \subseteq \Omega$, $\mu(X) = \sum_{x \in X} \mu(\{x\})$. We drop brackets from singletons whenever this does not cause any confusion. We denote by $\mathsf{Disc}(\Omega)$ the set of discrete probability measures over $\Omega$ and by $\mathsf{SubDisc}(\Omega)$ the set of discrete sub-probability measures over $\Omega$. We say that a set $X \subseteq \Omega$ is a *support* of a measure $\mu$ if $\mu(\Omega - X) = 0$. If $\mu$ is a discrete measure, then there is a minimum support of $\mu$ consisting of those elements $x \in \Omega$ such that $\mu(x) > 0$.

A function $f : \Omega_1 \to \Omega_2$ is said to be a *measurable function* from $(\Omega_1, \mathcal{F}_1)$ to $(\Omega_2, \mathcal{F}_2)$ if the inverse image under $f$ of any element of $\mathcal{F}_2$ is an element of $\mathcal{F}_1$. In this case, given a measure $\mu$ on $(\Omega_1, \mathcal{F}_1)$ it is possible to define a measure on $(\Omega_2, \mathcal{F}_2)$ via $f$, called the *image measure* of $\mu$ under $f$ and denoted by $f(\mu)$, as follows: for each $X \in \mathcal{F}_2$, $f(\mu)(X) = \mu(f^{-1}(X))$. In other words, the measure of $X$ in $\mathcal{F}_2$ is the measure in $\mathcal{F}_1$ of those elements whose $f$-image is in $X$. The measurability of $f$ ensures that $f(\mu)$ is indeed a well defined measure.

## 3 Probabilistic Automata

In this section we define probabilistic automata and we relate them to several other existing models.

### 3.1 Probabilistic Automata

The main idea behind probabilistic automata is that the target of a transition is not just a single state, but rather is determined by a probability measure. Thus, if a transition describes the act of flipping a fair coin, then the target state corresponds to head with probability 1/2 and tail with probability 1/2. However, in contraposition to ordinary Markov processes, for each state there may be several possible transitions.

A *probabilistic automaton* (PA) is a tuple $(Q, \bar{q}, A, D)$, where $Q$ is a countable set of *states*, $\bar{q} \in Q$ is a *start* state, $A$ is a countable set of *actions*, and $D \subseteq Q \times A \times \mathsf{Disc}(Q)$ is a *transition relation*. The set of actions $A$ is further partitioned into two sets $E, H$ of *external* and *internal* (hidden) actions, respectively.

The only difference with respect to ordinary automata is in the third element of the transition relation $D$, which is not a single state but rather a discrete probability measures over states. Indeed, an ordinary automaton can be seen as a special case of a probabilistic automaton where all transitions lead to *Dirac* measures, i.e., measures that assign probability 1 to a single state.

In the sequel we use $\mathcal{A}$ to denote a probabilistic automaton and we refer to the elements of $\mathcal{A}$ by $Q, \bar{q}, A, D$, propagating indices and primes as well. Thus, e.g., $H'_i$ is the set of internal actions of a PA $\mathcal{A}'_i$.

*Remark 1.* The definition of probabilistic automaton given here includes a single start state; however, nothing prevents us from defining PAs with multiple start states or where start states are replaced by start probability measures. Such

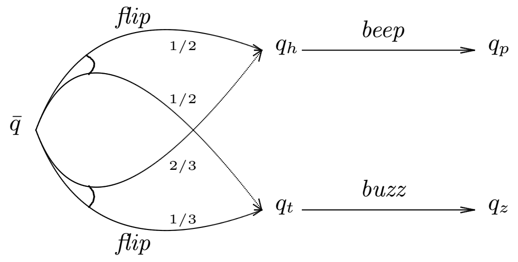

**Fig. 1.** A probabilistic automaton that flips a coin.

extensions do not provide much additional insight; however, they impose simple cosmetic adjustments in several definitions that we prefer to avoid in favor of clarity. Similar reasons motivated the restrictions on the cardinality of $Q$ and $A$.

*Example 1.* Figure 1 gives a graphical representation of a PA that performs an action flip and then beeps if the result of the coin flip is head and buzzes otherwise. The PA is nondeterministic since the coin to be flipped can be either fair or unfair, leading to head with probability 2/3 in the unfair case.

An *execution fragment* of a PA $\mathcal{A}$ is a sequence of alternating states and actions, $\alpha = q_0 a_1 q_1 \cdots$, starting with a state and, if the sequence is finite, ending with a state, such that, for each non final index $i$, there exists a transition $(q_i, a_{i+1}, \mu_{i+1})$ in $D$ with $\mu_{i+1}(q_{i+1}) > 0$. We denote by $\mathsf{fstate}(\alpha)$ the first state $q_0$ of $\alpha$, and, if the sequence is finite, we denote by $\mathsf{lstate}(\alpha)$ the last state of $\alpha$. An *execution* of a PA $\mathcal{A}$ is an execution fragment of $\mathcal{A}$ whose first state if $\bar{q}$. We denote by $\mathsf{Frags}^*(\mathcal{A})$ the set of finite execution fragments of a PA $\mathcal{A}$.

An execution fragment is the result of resolving nondeterminism and fixing the outcomes of the probabilistic experiments. However, resolving nondeterministic choices only leads to more complex structures that should be studied.

*Example 2.* Consider the coin flipper of Example 1 and suppose we want to compute the probability that it beeps. We should note first that such probability depends on the coin that is flipped. Indeed, the coin flipper beeps with probability 1/2 if the fair coin is flipped, and with probability 2/3 if the unfair coin is flipped. Therefore, in order to answer our question, we should first fix the coin to be flipped, and then study probabilities on the structure that we get.

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, which we call *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 $\mathcal{A}$ is a function $\sigma : \mathsf{Frags}^*(\mathcal{A}) \to \mathsf{SubDisc}(D)$ such that, for each finite execution fragment $\alpha$ and each transition $tr$ with $\sigma(\alpha)(tr) > 0$, the source state of $tr$ is $\mathsf{lstate}(\alpha)$. A scheduler $\sigma$ is *deterministic* if, for each finite execution fragment $\alpha$, either $\sigma(\alpha)$ assigns probability 1 to a single transition or assigns probability 0 to all transitions. A scheduler $\sigma$ is *memoryless* if it depends

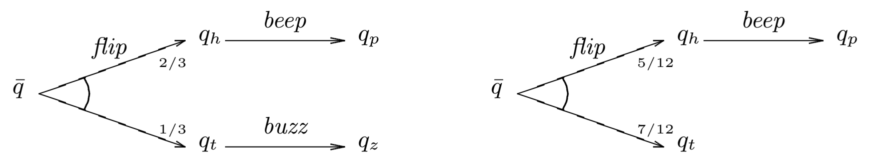

**Fig. 2.** Probabilistic executions

only on the last state of its argument, i.e., for each pair $\alpha_1, \alpha_2$ of finite execution fragments, if $\mathsf{lstate}(\alpha_1) = \mathsf{lstate}(\alpha_2)$, then $\sigma(\alpha_1) = \sigma(\alpha_2)$.

Informally, $\sigma(\alpha)$ describes the rule for choosing a transition after $\alpha$ has occurred. The rule itself may be randomized. Since $\sigma(\alpha)$ is a sub-probability measure, it is possible that with some non-zero probability no transition is chosen, which corresponds to terminating the computation (what in the purely nondeterministic case 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.

*Remark 2.* Terminology may be confusing at this point. In the original definition of PAs [35] a scheduler is called *adversary* since it is seen as a hostile entity that degrades performance as much as possible. In the field of Markov Decision Processes [15], a scheduler is called *policy* since it is seen as an entity that optimizes some cost function. In practice the three terms may be used interchangeably.

*Example 3.* Figure 2 gives two examples of probabilistic executions of the coin flipper of Example 1. In the left case the unfair coin is flipped, while in the right case each coin is flipped with probability 1/2 and the *buzz* transition is never scheduled. In general, considering that a scheduler may also terminate executions, we can say that the coin flipper beeps with probability at most 2/3. Furthermore, if we assume that transitions are scheduled whenever possible, then we can say that the coin flipper beeps with a probability between 1/2 and 2/3.

We now describe formally how to associate a probability measure to execution fragments once nondeterminism is resolved. Interestingly, we do not need to build explicitly the structures depicted in Figure 2. First we need to define the set of measurable events (when we talk about probabilities an element of the $\sigma$-field is called an event); then we associate probabilities to events. As basic measurable events we consider the set of *cones* of finite execution fragments, where the cone of a finite execution fragment $\alpha$ consists of all possible extensions of $\alpha$ and denotes the occurrence of $\alpha$ possibly followed by some other behavior. Formally, the cone of $\alpha$, denoted by $C_\alpha$, is the set $\{\alpha' \in \mathsf{Frags}(\mathcal{A}) \mid \alpha \leq \alpha'\}$, where $\leq$ is the standard prefix preorder on sequences. Informally, referring to Figure 2, the probability of a cone $C_\alpha$ is the product of the probabilities of all the edges of the path $\alpha$. Formally, we give a recursive definition. Fixed a scheduler $\sigma$ and a state $s$, we define a measure $\epsilon_{\sigma,s}$ on cones as follows.

$$\epsilon_{\sigma,s}(C_\alpha) = \begin{cases} 0 & \text{if } \alpha = q \text{ for some state } q \neq s, \\ 1 & \text{if } \alpha = s, \\ \epsilon_{\sigma,s}(C_{\alpha'}) \sum_{tr \in D(a)} \sigma(\alpha)(tr)\mu_{tr}(q) & \text{if } \alpha = \alpha' a q, \end{cases}$$

where $D(a)$ denotes the set of transitions of $D$ with label $a$. Standard measure theoretical arguments ensure that $\epsilon_{\sigma,s}$ extends uniquely to the $\sigma$-field generated by cones. We call the measure $\epsilon_{\sigma,s}$ a *probabilistic execution fragment* of $\mathcal{A}$ and we say that it is generated by $\sigma$ from $s$. If $s$ is the start state of $\mathcal{A}$, then we say that $\epsilon_{\sigma,s}$ is a *probabilistic execution.*

The cone-based definition of $\sigma$-field is quite general, and indeed typical properties of interest are measurable. The occurrence of an action $a$ (of a state $s$) is a union of cones, and thus measurable since there are countably many cones. Similarly, we retain measurability if we require $n$ occurrences of an action or state. Also infinitely many occurrences of an action are measurable, since they can be expressed as the countable intersection, over all naturals $n$, of $n$ occurrences. It is also known that any $\omega$-regular language is measurable [39] and that the properties expressed by existing probabilistic temporal logics are measurable.

We conclude this section with the definition of a parallel composition operator. In our definition we synchronize two probabilistic automata on their common actions; however, many other synchronization styles are possible.

Two probabilistic automata $\mathcal{A}_1, \mathcal{A}_2$ are *compatible* if $H_1 \cap A_2 = A_1 \cap H_2 = \emptyset$. The composition of two compatible probabilistic automata $\mathcal{A}_1, \mathcal{A}_2$, denoted by $\mathcal{A}_1 \| \mathcal{A}_2$, is a probabilistic automaton $\mathcal{A}$ where $Q = Q_1 \times Q_2$, $\bar{q} = (\bar{q}_1, \bar{q}_2)$, $E = E_1 \cup E_2$, $H = H_1 \cup H_2$, and $D$ is defined as follows: $((q_1, q_2), a, \mu_1 \times \mu_2) \in D$ iff, for each $i \in \{1, 2\}$, either $a \in A_i$ and $(q_i, a, \mu_i) \in D_i$, or $a \notin A_i$ and $\mu_i = \delta(q_i)$, where $\delta(q_i)$ denotes the probability measure that assigns probability 1 to $q_i$ and $\mu_1 \times \mu_2((q'_1, q'_2))$ is defined to be $\mu_1(q'_1)\mu_2(q'_2)$.

## 3.2 Reactive, Generative, and Stratified Models

In [19] probabilistic models are classified into reactive, generative, and stratified. The paper was first written in 1990 in the context of concurrency theory, where the trend was to replace nondeterministic choices with probabilistic choices. The main driving idea was that the presence of probabilities does not hide the underlying nondeterminism, but rather gives more information.

A *reactive* system is a labeled transition system whose arcs are equipped with probabilities. Furthermore, for each state $q$ and each action $a$, either there is no transition labeled by $a$ from $q$, or the probabilities of all transition labeled by $a$ from $q$ add to 1. In other words, a reactive system does not provide any information about the way an action is chosen, but provides information about the way a transition is chosen once the action is fixed. The information about the underlying nondeterminism for an action $a$ can be retrieved via an appropriate projection operation that removes all probabilities from the arcs.

A *generative* system is similar to a reactive system; however, this time the requirement is that for each state $q$ either there is no transition from $q$, or the probabilities of all transition from $q$ add to 1. In other words a generative system adds information about the way actions are chosen. The information about the actions available (i.e., the underlying reactive system) can be retrieved by an appropriate projection operation that renormalizes the probabilities of the transitions labeled by the same action.

A *stratified* system adds more information to a generative system in the sense that a measure over visible transitions is obtained via several non-visible transitions that reveal some hierarchy. This model has not received much attention in the literature and therefore we refer the interested reader to [19].

The view of [19] is that stratified is more general than generative and that generative is more general than stratified with the justification that the projection operators preserve bisimilarity. Our definition of probabilistic automata departs considerably from such view. Indeed, there is no way to encode the nondeterminism of the probabilistic automaton of Example 1 within a reactive or generative system. In contraposition to the underlying idea of [19], probabilistic automata keep explicitly both nondeterministic and probabilistic choices.

Observe that a reactive system can be seen also as a *deterministic* PA, i.e., a PA that from each state enables at most one transition for each action. However, this implies abandoning the idea of [19] that the underlying nondeterminism can be retrieved by removing probabilities.

## 3.3 Markov Decision Processes

Another well known model of probabilistic and nondeterministic systems is Markov Decision Processes (MDPs) [15], which in practice correspond to deterministic probabilistic automata. MDPs were studied originally within operational research: a process evolves probabilistically according to measures that depend only on the current states (Markovian property); however, from each state there are several possible actions available, each one leading to different evolutions. The objective is to choose actions from each state (choose a policy) so that some cost function is optimized. States may be associated with rewards that are used to compute the cost function. MDPs are deterministic in the sense that from each state each action identifies a unique evolution.

Another related model which is worth mentioning here are the probabilistic automata of Rabin [34]. Again, these correspond to deterministic probabilistic automata. They were studied originally in the context of language theory to show that finite-state probabilistic automata accept a class of languages which is strictly larger than regular languages.

## 3.4 Alternating Models

In [39] Vardi studies model checking algorithms for Markov processes in the presence of nondeterminism. For the purpose he distinguishes *probabilistic* states,

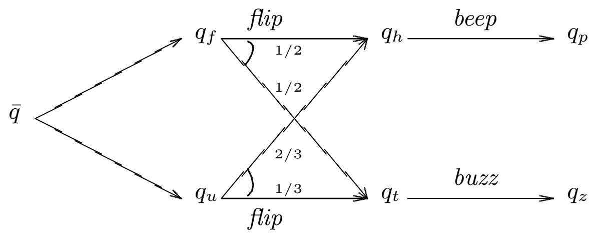

**Fig. 3.** An alternating representation of the coin flipper of Figure 1.

where the next state is determined by a probability measure, from *nondeterministic* states, where several ordinary transitions may occur. The same idea is followed in [22] for the definition of Labeled Concurrent Markov Chains. The objective of [22] was to give a semantics to a probabilistic process algebra, where probability measures are expressed by appropriate expressions. Thus, a distinction between nondeterministic and probabilistic states follows naturally.

The models of [39, 22] are currently referred to as *alternating* models, in contraposition to probabilistic automata which are *non-alternating*. The model of [22] is further called *strictly* alternating since it imposes a strict alternation between nondeterministic and probabilistic states, whereas the model of [39] permits transitions between nondeterministic states.

Since an ordinary transition is a special case of a probabilistic transition, the alternating models can be seen as special cases of probabilistic automata, where restrictions are imposed on the transition relations. This idea is formalized in [37] via appropriate embedding functions. The interesting aspect of viewing the alternating models as special cases of probabilistic automata is that several concepts that in the literature are defined on the models in very different styles turn out to be equivalent [37]. This is reassuring since it means that we can grasp the whole theory by understanding fewer concepts.

*Example 4.* Figure 3 represents the coin flipper of Figure 1 as an alternating probabilistic automaton. The main difference is that the transitions labeled by *flip* are split into two transitions, one from a nondeterministic state to a probabilistic state, and one from the probabilistic state to a probability measure over nondeterministic states. There is indeed a folklore idea, formalized in [37], on how to transform a non-alternating model into an alternating model by splitting transitions and vice-versa how to transform an alternating model into a non-alternating model by collapsing transitions. Observe also that the coin flipper of Figure 3 is a PA as well if we add dummy internal labels to the unlabeled arcs. This is the main idea behind the embedding functions of [37].

## 3.5 Generative Probabilistic Automata

The original definition of probabilistic automaton [35] was based on a more general notion of transition where not only the target state is determined by a probability measure, but also the action to be performed. In addition, it is also possible for a transition to deadlock with some probability. Although several concepts can be adapted to this general definition of probabilistic automaton, the main problem is that we are not aware yet of any meaningful definitions of composition that are conservative extensions of existing definitions in non probabilistic models. It is shown in [35] that several natural attempts break associativity of composition. A successful attempt is reported in [12] with the introduction of the *bundle* model. In this model a transition leads to a probability measure over sets of ordinary transitions and thus parallel composition can be defined easily. However, it is arguable whether this is really a conservative extension of ordinary automata or a faithful representation of generative systems. For example, in [5] it is argued that the bundle model is more expressive than generative probabilistic automata according to a classification that we describe in Section 5; however, we may also see a bundle system just as a special case of a probabilistic automaton in a similar way as we see the alternating models as special cases of probabilistic automata via embedding. This point of view is not investigated yet.

For the above reasons, as we do in this paper, it is now typical to use the term probabilistic automaton to refer to what in [35] is called a simple probabilistic automaton. Following the classification of [19] we could call the probabilistic automata of [35] probabilistic automata with *generative transitions* or generative probabilistic automata and the probabilistic automata of this paper, i.e., the simple probabilistic automata of [35], probabilistic automata with *reactive transitions* or reactive probabilistic automata.

### 3.6 Probabilistic I/O Automata

Following the style of [27], where the external actions of ordinary automata are partitioned into input and output actions, it is possible to introduce the input/output distinction on probabilistic automata as well. The advantage of this approach is that we can recover all the techniques used within I/O automata, including the task mechanisms used to describe fairness properties and the ability to use language inclusion to preserve fairness properties as well.

A probabilistic I/O automaton (PIOA) [10] is a probabilistic automaton whose external actions are partitioned into *input* and *output* actions such that for each state $q$ and each input action $a$ there is at least one transition labeled by $a$ enabled from $q$ (*input enabling* property). Output and internal actions are called *locally controlled* actions. Two PIOAs can be composed only if their locally controlled actions are disjoint. As a consequence, the external environment can never block any locally controlled action of a PIOA (the other automata may have the same action only as an input, which is always enabled), or, in other words, every PIOA is in full control of its locally controlled actions. We can say alternatively that each action is under the control of at most one component.

Another advantage of the input/output distinction is that it is possible to consider PIOAs with generative locally controlled transitions, and yet define a meaningful composition operator [40, 35]. Indeed, the transitions of a composition can be obtained either by synchronizing input transitions, or by synchronizing a locally controlled transition of one component with appropriate input transitions of the other component.

The definition of PIOA of [40] does not include nondeterminism: each state enables exactly one reactive transition for each input action and possibly one generative transition with locally controlled actions. The nondeterminism that arises in a composition is resolved by assigning weights to each component and using relative weights as probabilities to solve conflicts determined by locally controlled transitions. It was observed later that this amounts to assuming that the locally controlled transitions of a component PIOA are performed with time delays governed by an exponential distribution whose delay parameter is the weight of the component. Thus, the PIOAs of [40] are special instances of the pure probabilistic models described later in Section 3.8.

## 3.7 Unlabeled Models

Probabilistic automata include labels; however, especially in the context of model checking, it is typical to consider unlabeled models and add structure to the states to define properties. For example, we could easily define probabilistic Kripke structures by removing labels from the transition relation of a PA and adding a labeling function that associates propositional symbols with states.

If we consider composition with synchronization between components, then synchronization occurs typically via shared variables. However, problems similar to those encountered with generative probabilistic automata arise if we do not impose any control structure on the values of the shared variables (e.g., the values of some variables are under the control of a single component). A successful attempt to solve the synchronization problem in unlabeled models appears in [13], where a probabilistic extension of reactive modules [1] is studied.

*Example 5.* Consider two unlabeled probabilistic automata $\mathcal{A}_1, \mathcal{A}_2$ whose states include a variable $X$. Suppose that $\mathcal{A}_1$ from its initial state flips a fair coin to set $X$ either to 0 or 1, while $\mathcal{A}_2$ from its initial state flips a fair coin to set $X$ either to 0 or 2. What transition should appear from the initial state of $\mathcal{A}_1 \| \mathcal{A}_2$? We have at least two choices: either we deadlock whenever the two automata set $X$ in an incompatible way, and thus $X$ is set to 0 with probability $1/4$ and the system deadlocks with probability $3/4$, or we consider only compatible choices and renormalize probabilities, and thus $X$ is set to 0 with probability 1.

## 3.8 Pure Probabilistic Models

If we remove all nondeterminism and keep only probabilistic choices, then in the unlabeled case we obtain Markov processes, while in the labeled generative case we obtain Markov processes with actions. These models are used mainly for performance evaluation and are studied in the context of *stochastic process algebras* [20, 24, 6]. The underlying idea is that actions describe resources that are available with exponentially distributed delays, which means that there is a close correspondence between the delay parameter of the actions and their probability to occur. Each model manages actions in a slightly different way, but overall actions are partitioned into two sets. Actions from the first set occur according to some probability measures and describe the resources available, while actions from the second set are passive, and simply synchronize with actions from the first set. Passive actions describe the consumers of the resources. Thus some input/output distinction is present. Composition amounts to adding resources and users. If a resource cannot be used, then probabilities are renormalized.

A complete description of stochastic process algebras goes beyond the scope of this paper. Here we observe that composition of stochastic process algebras is not a conservative extensions of composition of ordinary automata and is not comparable with composition of probabilistic automata. A good understanding of the relationship between these models is still open.

We mention also the interesting approach to performance evaluation of Interactive Markov Chains [23]. In this case actions are immediate and time is described by explicit transitions with exponential delays. The advantage of this approach is that it is possible to keep nondeterminism in the model.

### 3.9 Models with Time

There is a vast literature on timed extension of probabilistic models. We have described before some ways to associate delays with actions and we refer the interested reader to a survey that appears in [7]. In the context of probabilistic automata one possibility to deal with time is by adding explicit time-passage transitions to the model and keep the underlying theory unchanged. This is done already in the work of Hansson and Jonsson [22] by discretizing time and representing the passage of a quantum of time via a "tick" action. Segala [35] considers a dense time domain and adds to probabilistic automata time-passage transitions labeled by the amount of time elapsed. However, in order to reuse the theory of probabilistic automata, schedulers can only be discrete.

A treatment of real-time with non-discrete measures poses non-trivial measurability problems that go beyond the scope of this paper. We refer the reader to [30, 18] for an understanding of the problem on deterministic models and to [9] for an understanding of the problem in the presence of nondeterminism.

## 4 Simulations and Bisimulations

Simulation and bisimulation relations are attractive for their mathematical simplicity. They have been studied extensively in the context of probabilistic systems, including reactive systems [26], alternating models [21, 32, 2], and non alternating models [36, 4]. The existing definitions are very different in style; however, as shown in [37], all the proposals end up being equivalent once we see the alternating models as special instances of probabilistic automata.

## 4.1 Lifting Relations

We start by lifting a relation on a set $X$ to a relation on probability measures over $X$. This is useful since the target of a transition in a PA is a probability measure. Let $R$ be a relation on a set $X$. The *lifting* of $R$, denoted by $\mathcal{L}(R)$, is a relation on $\mathsf{Disc}(X)$ such that, $\mu_1 \mathcal{L}(R) \mu_2$ iff for each upper closed set $C \subset X$, $\mu_1(C) \leq \mu_2(C)$, where the upper closure of a set $C$ is the set $\{x \in X \mid \exists_{c \in C}, c\ R\ x\}$. This definition of lifting was first proposed in [16] in the context of non-discrete systems and is equivalent to an earlier proposal of [25, 36] for discrete systems stating that $\mu_1 \mathcal{L}(R) \mu_1'$ iff there exists a *weighting function* $w : Q \times Q \to [0, 1]$ such that (1) $w(x_1, x_2) > 0$ implies $x_1\ R\ x_2$, (2) $\sum_{x_1} w(x_1, x_2) = \mu_2(x_2)$, and (3) $\sum_{x_2} w(x_1, x_2) = \mu_1(x_1)$. Informally, $w$ redistributes probabilities between $\mu_1$ and $\mu_2$ respecting $R$. An important observation is that if $R$ is an equivalence relation, then $\mu_1 \mathcal{L}(R) \mu_2$ iff, for each equivalence class $C$ of $R$, $\mu_1(C) = \mu_2(C)$.

## 4.2 Strong Simulations and Bisimulations

A *strong simulation* on a PA $\mathcal{A}$ is a relation $R$ on $Q$ such that, for each pair of states $(q_1, q_2) \in R$ and each transition $(q_1, a, \mu_1)$ of $\mathcal{A}$ there exists a transition $(q_2, a, \mu_2)$ of $\mathcal{A}$ such that $\mu_1 \mathcal{L}(R) \mu_2$. If $R$ is an equivalence relation, then we say that $R$ is a *strong bisimulation.*

Sometimes it is more convenient to talk about simulation and bisimulation relations between two PAs $\mathcal{A}_1$ and $\mathcal{A}_2$. These definitions can be recovered from the definition above by considering the disjoint union of the states $Q_1 \uplus Q_2$, the union of the transition relations $D_1, D_2$, and requiring start states to be related.

If we apply the definition above to deterministic PAs, then we obtain the definition of bisimulation of [26], and similarly we obtain the definition of bisimulation of [21] if we consider strictly alternating PAs. There is also a definition of bisimulation for alternating PAs proposed in [32]. This definition, however, coincides with our definition above only if we transform an alternating automaton into a PA according to the construction of Example 4. Indeed, the definition of [32] was given by viewing alternation just as a formal artifact to describe PAs.

## 4.3 Strong Probabilistic Simulations and Bisimulations

Consider the two PAs of Figure 4. The two PAs are not bisimilar since the middle transition of $\mathcal{A}_2$ cannot be simulated by $\mathcal{A}_1$. On the other hand, the middle transition of $\mathcal{A}_2$ is just a convex combination of the other two transitions. If we are just interest in bounds to the probabilities of satisfying a property (say performing action *beep*), there should be no reason to distinguish $\mathcal{A}_1$ from $\mathcal{A}_2$. In [36] it is shown that $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the same formulas of PCTL, a probabilistic temporal logic which indeed observes only bounds on probabilities, and thus it is argued that $\mathcal{A}_1$ and $\mathcal{A}_2$ should not be distinguished. This lead to the formulation of a *probabilistic* version of simulation and bisimulation relations, where transitions can be simulated by convex combinations of other transitions.

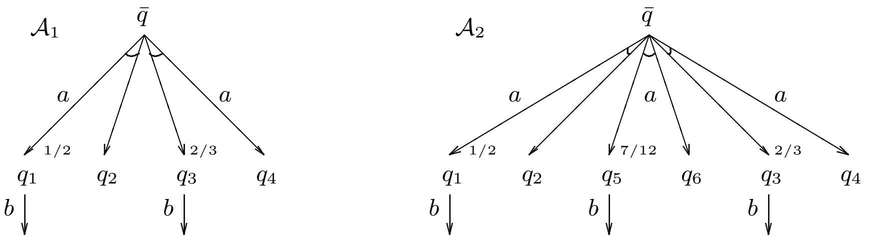

**Fig. 4.** Two PAs that are not strongly bisimilar.

Let $\mathcal{A}$ be a probabilistic automaton, and let $\{(q, a, \mu_i)\}_I$ be a countable family of transitions of $\mathcal{A}$. Let $\{p_i\}_I$ be a family of probabilities such that $\sum_{i\in I} p_i = 1$. Then the triplet $(q, a, \sum_{i\in I} p_i\mu_i)$ is called a *combined transition* of $\mathcal{A}$.

A *strong probabilistic simulation* on a PA $\mathcal{A}$ is a relation $R$ on $Q$ such that, for each pair of states $(q_1, q_2) \in R$ and each transition $(q_1, a, \mu_1)$ of $\mathcal{A}$ there exists a combined transition $(q_2, a, \mu_2)$ of $\mathcal{A}$ such that $\mu_1\ \mathcal{L}(R)\ \mu_2$. If $R$ is an equivalence relation, then we say that $R$ is a *strong probabilistic bisimulation.*

It turns out that in the alternating models strong bisimulation and strong probabilistic bisimulation coincide [37]. Thus, the distinction between the two kinds of bisimulation is relevant only for PAs.

*Example 6.* Observe that the ability to simulate a transition by convex combinations of transitions is lost in the alternating model, which is why strong and strong probabilistic bisimulation coincide. Indeed, if we transform the probabilistic automata of Figure 4 by splitting transitions, then they would not be bisimilar any more since the intermediate probabilistic state reached by the middle transition of $\mathcal{A}_2$ cannot be related to any state of the transformation of $\mathcal{A}_1$.

## 4.4 Weak Probabilistic Simulations and Bisimulations

The next step is to abstract from internal computation and extend weak simulations and bisimulations to PAs. The only interesting aspect is how to define a weak transition in the probabilistic case. On ordinary automata a weak transition is represented by a finite execution fragment whose trace consists of at most one external action. Here the trace of an execution fragment $\alpha$, denoted by $\mathsf{trace}(\alpha)$, is the subsequence of external actions that occur in $\alpha$. In the probabilistic case a weak transition is represented by a probabilistic execution fragment.

Let $\mathcal{A}$ be a probabilistic automaton, and let $\epsilon_{\sigma,q}$ be a probabilistic execution fragment of $\mathcal{A}$ generated by $\sigma$ from state $q$. If $\epsilon_{\sigma,q}(\mathsf{Frags}^*(\mathcal{A})) = 1$ and there exists an action $a$ such that $\mathsf{trace}(\epsilon_{\sigma,q}) = \mathsf{trace}(a)$, then we say that $(q, a, \mathsf{lstate}(\epsilon_{\sigma,q}))$ is a weak combined transition of $\mathcal{A}$. Here we have used implicitly the fact that functions $\mathsf{trace}$ and $\mathsf{lstate}$ are measurable.

A *weak probabilistic simulation* on a PA $\mathcal{A}$ is a relation $R$ on $Q$ such that, for each pair of states $(q_1, q_2) \in R$ and each transition $(q_1, a, \mu_1)$ of $\mathcal{A}$ there exists a weak combined transition $(q_2, a, \mu_2)$ of $\mathcal{A}$ such that $\mu_1\ \mathcal{L}(R)\ \mu_2$. If $R$ is an equivalence relation, then we say that $R$ is a *weak probabilistic bisimulation.*

Since we have imposed no restrictions on the schedulers that generate weak transitions, we have defined directly the probabilistic versions of the weak relations. Indeed, it turns out that the non-probabilistic versions of the weak relations are not transitive [14]. A definition of weak bisimulation is proposed also for the alternating model [33] using deterministic schedulers. The definition uses conditional probability measures and is given in a very different style compared to this paper; however, our definition of weak probabilistic bisimulation restricted to the alternating model coincides with the definition of [32].

### 4.5 Other kinds of Bisimulations

Bisimulation relations are studied extensively also in pure probabilistic models. The definitional style is very similar to ours and we refer the reader to [3] for an extensive comparative study. In particular in the absence of nondeterminism several relations coincide. There are also several variations of simulation and bisimulation relations in the literature, often proposed with specific applications in mind. Here we cite two important proposals.

In [36] there is a proposal for a probabilistic version of *branching bisimulation*, which is shown to preserve a weak version of PCTL. This definition is given in the same style of the previous subsections, except that some restrictions are imposed on the intermediate states of a weak transition, exactly as in the definition of branching bisimulation. A definition of branching bisimulation is also proposed in [2] for the alternating model. Once again, the definition of [36] coincides with the definition of [2] once restricted to the alternating model.

In [4] there is a proposal for a probabilistic version of *normed bisimulation* which has the advantage of being decidable efficiently. Indeed, strong bisimulations are decidable in polynomial time [8], while weak bisimulations are decidable in exponential time on probabilistic automata [8]. Interestingly, though, weak bisimulations are decidable in polynomial time in the alternating model [32] since the alternating structure ensures that two states are bisimilar iff for each action and each equivalence class the maximum probabilities of reaching the given class with the given action coincide. See [8] for more details.

## 5 Concluding Remarks

In this paper we have given an overview of the main operational models for probabilistic and nondeterministic systems. In doing so we have been forced to leave out other important approaches that the reader may want to investigate further. Within the field of domain theory and denotational semantics, several models have been proposed that combine probability and nondeterminism. The interested reader may start from [38, 11] for more details. There has been also extensive research on probabilistic extensions of guarded command languages and their applications. The interested reader is referred to [28].

Our overview is given by taking probabilistic automata as reference model and viewing the others as special cases or generalizations. There are also other ways to classify models. In particular [5] proposes a hierarchy where a model is more expressive than another one if it is possible to transform objects of the least expressive model into objects of the other model so that bisimilarity is preserved and reflected. The transformations should preserve states; thus, for example, the transformations of [37] are not acceptable since they add or remove probabilistic states. Indeed, the alternating and non-alternating models are incomparable according to [5].

We have omitted here any reference to process algebras, axiomatizations and logical characterizations for probabilistic models. For process algebras the reader may look at [31] and references therein, while for logical characterizations the reader may look at [26, 17]. The work in [17] is carried out in the context of alternating non-discrete systems and improves the results of [26].

## References

1. R. Alur and T. Henzinger. Reactive modules. *Formal Methods in System Design* 15(1):7–48, 1999.
2. S. Andova, T. Willemse. Branching bisimulation for probabilistic systems: characteristics and decidability. *Theoretical Computer Science*, 356(3):325–355, 2006.
3. C. Baier, H. Hermanns, J-P.Katoen, and V. Wolf. Comparative branching-time semantics for markov chains. In *Proceedings of CONCUR 2003*, LNCS 2761, 2003.
4. C. Baier and M. Stoelinga. Norm functions for bisimulations with delays. In *Proceedings of FOSSACS*, LNCS 1784, pages 407–418, 2000.
5. F. Bartels, A. Sokolova, and E. de Vink. A hierarchy of probabilistic system types. *Theoretical Computer Science*, 327(1-2):3–22, 2004.
6. M. Bernardo and R. Gorrieri. Extended markovian process algebra. In [29].
7. M. Bravetti and P.R. D'Argenio. Tutte le algebre insieme: concepts, discussions and relations of stochastic process algebras with general distributions. In *Validation of Stochastic Systems*, LNCS 2925, pages 44–88, 2004.
8. S. Cattani and R. Segala. Decision algorithms for probabilistic bisimulation. In *Proceedings of CONCUR*, LNCS 2421, pages 371–385, 2002.
9. S. Cattani, R. Segala, M. Kwiatkowska, and G. Norman. Stochastic transition systems for continuous state spaces and non-determinism. In *Proceedings of FOSSACS*, LNCS 3441, pages 125–139, 2005.
10. L. Cheung, N. Lynch, R. Segala, and F. Vaandrager. Switched probabilistic I/O automata. In *Proceedings of ICTAC*, LNCS 3407, pages 494–510, 2004.
11. V. Daniele and G. Winskel. Distributing probability over non-determinism. *Mathematical Structures in Computer Science*, 16:87–113, 2006.
12. P. D'Argenio, H. Hermanns, and J.P. Katoen. On generative parallel composition. In *Proceedings of PROBMIV'98*, ENTCS 22, 1999.
13. L. de Alfaro, T.A. Henzinger, and R. Jhala. Compositional methods for probabilistic systems. In *Proceedings of CONCUR*, LNCS 2154, 2001.
14. Y. Deng. *Axiomatisations and types for probabilistic and mobile processes.* PhD thesis, Ecole de Mines, 2005.
15. C. Derman. *Finite State Markovian Decision Processes.* Academic Press, 1970.
16. J. Desharnais. *Labelled Markov Processes.* PhD thesis, McGill University, 1999.

17. J. Desharnais, A. Edalat, and P. Panangaden. A logical characterization of bisimulation for labelled Markov processes. In *Proceedings of LICS*, 1998.
18. J. Desharnais, A. Edalat, and P. Panangaden. Bisimulation for labelled Markov processes. *Information and Computation*, 179(2):163–193, 2002.
19. R. van Glabbeek, S. Smolka, B. Steffen. Reactive, generative, and stratified models of probabilistic processes. *Information and Computation*, 121(1):59–80, 1996.
20. N. Götz, U. Herzog, and M. Rettelbach. Multiprocessor and distributed system design: the integration of functional specification and performance analysis using stochastic process algebras. In *Performance Evaluation of Computer and Communication Systems. Joint Tutorial Papers of Performance '93 and Sigmetrics '93*, LNCS 729, pages 121–146, 1993.
21. H. Hansson. *Time and Probability in Formal Design of Distributed Systems*, volume 1 of *Real-Time Safety Critical Systems*. Elsevier, 1994.
22. H. Hansson and B. Jonsson. A calculus for communicating systems with time and probabilities. In *Proceedings of RTSS*, 1990.
23. H. Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*, LNCS 2428, 2002.
24. J. Hillston. *A Compositional Approach to Performance Modeling*. PhD thesis, Department of Computer Science, University of Edimburgh (UK), 1994.
25. B. Jonsson and K.G. Larsen. Specification and refinement of probabilistic processes. In *Proceedings of LICS*, pages 266–277, July 1991.
26. N.A. Lynch and E.W. Stark. A proof of the Kahn principle for Input/Output automata. *Information and Computation*, 82(1):81–92, 1989.
27. N.A. Lynch and M.R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of PoDC*, pages 137–151, 1987.
28. A. McIver and C. Morgan. *Abstraction, refinement, and proof for probabilistic systems*. Springer-Verlag, 2005.
29. U. Montanari and V. Sassone, editors. *Proceedings of CONCUR*, LNCS 1119, 1996.
30. P. Panangaden. Measure and probability for concurrency theorists. *Theoretical Computer Science*, 253(2):287–309, 2001.
31. A. Parma and R. Segala. Axiomatization of trace semantics for stochastic nondeterministic processes. In *Proceedings of QEST*, pages 294–303, 2004.
32. A. Philippou, I. Lee, and O. Sokolsky. Weak bisimulation for probabilistic systems. In *Proceedings of CONCUR 2000*, LNCS 1877, pages 334–349, 2000.
33. A. Pogosyants, R. Segala, and N. Lynch. Verification of the randomized consensus algorithm of Aspnes and Herlihy: a case study. *Distrib. Comp.* 13:155–186, 2000.
34. M.O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.
35. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT, 1995.
36. R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.
37. R. Segala and A. Turrini. Comparative analysis of bisimulation relations on alternating and non-alternating probabilistic models. In *Proceedings of QEST*, 2005.
38. R. Tix, K. Keimel, and G. Plotkin. Semantic domains for combining probability and non-determinism. ENTCS 129:1–104, 2005.
39. M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings of FoCS*, pages 327–338, 1985.
40. S.H. Wu, S. Smolka, and E.W. Stark. Composition and behaviors of probabilistic I/O automata. *Theoretical Computer Science*, 176(1-2):1–38, 1999.