Probabilistic Simulations for Probabilistic Processes [⋆]

Roberto Segala and Nancy Lynch

MIT Laboratory for Computer Science
Cambridge, MA 02139

Abstract. Several probabilistic simulation relations for probabilistic systems are defined and evaluated according to two criteria: compositionality and preservation of "interesting" properties. Here, the interesting properties of a system are identified with those that are expressible in an untimed version of the Timed Probabilistic concurrent Computation Tree Logic (TPCTL) of Hansson. The definitions are made, and the evaluations carried out, in terms of a general labeled transition system model for concurrent probabilistic computation. The results cover weak simulations, which abstract from internal computation, as well as strong simulations, which do not.

1 Introduction

Randomization has been shown to be a useful tool for the solution of problems in distributed systems [1, 2, 12]. In order to support reasoning about probabilistic distributed systems, many researchers have recently focused on the study of models and methods for the analysis of such systems [3, 5, 7, 19–21]. The general approach that is taken is to extend to the probabilistic setting those models and methods that have already proved successful for non-probabilistic distributed systems.

In the non-probabilistic setting, labeled transition systems have become well accepted as a basis for formal specification and verification of concurrent and distributed systems. (See, e.g., [16, 17].) A transition system is an abstract machine that represents either an implementation (i.e., a physical device or software system), or a specification (i.e., a description of the required properties of an implementation). In order to extend labeled transition systems to the probabilistic setting, the main addition that is needed is some mechanism for representing probabilistic choices as well as nondeterministic choices [7, 19, 21].

In the non-probabilistic setting, there are two principal methods that are used for analyzing labeled transition systems: temporal logic (e.g. [18]), which is used to establish that a system satisfies certain properties, and equivalence or preorder relations (e.g., [8, 16, 17]), which are used to establish that one system "implements" another, according to some notion of implementation. Each equivalence or preorder preserves some of the properties of a system, and thus the use of a relation as a notion of implementation means that we are interested only in the properties that such a relation preserves.

Among the equivalences and preorders that have proved most useful are the class of *simulation* relations, which establish step-by-step correspondences between two systems. Bisimulation relations are two-directional relations that have proved fundamental in the process algebraic setting. Unidirectional simulations, such as refinement mappings and forward simulations, have turned out to be quite successful in formal verification of non-probabilistic distributed systems [10, 15, 16]. Thus, it is highly desirable to extend the use of simulations to the probabilistic setting.

[⋆] Supported by NSF grant CCR-89-15206, and CCR-92-25124, by DARPA contracts N00014-89-J-1988 and N00014-92-J-4033, and by ONR contract N00014-91-J-1046.

In this paper, we define several extensions of the classical bisimulation and simulation relations (both in their strong and weak versions), to the probabilistic setting. There are many possible extensions that could be made; it is important to evaluate the various possibilities according to objective criteria. We use two criteria: compositionality and preservation of "interesting" properties. The first requirement, compositionality, is widely accepted since it forms the basis of many modular verification techniques.

To make sense of the second requirement, it is necessary to be specific about what is meant by an "interesting" property. Here, we identify the interesting properties of a system with those that are expressible in an untimed version (PCTL) of the Timed Probabilistic concurrent Computation Tree Logic (TPCTL) of Hansson [7]; as discussed in [7], this logic is sufficiently powerful to represent most of the properties of practical interest. Thus, our second evaluation criterion is based on the types of PCTL formulas that a relation preserves. For the weak relations, i.e., the ones that abstract from internal computation, we use a new version of PCTL, called WPCTL, which abstracts from internal computation as well.

We define and evaluate our simulation relations in terms of a new general labeled transition system model for concurrent probabilistic computation, which borrows ideas from [7, 21]. The model distinguishes between probabilistic and nondeterministic choices but, unlike the Concurrent Markov Chains of [7, 21], does not distinguish between probabilistic and nondeterministic states. A *probabilistic automaton* is a labeled transition system whose transition relation is a set of pairs $(s, (\Omega, \mathcal{F}, P))$, where $(\Omega, \mathcal{F}, P)$ is a discrete[2] probability distribution over (action,state) pairs and a special symbol δ, representing deadlock. If δ is not an element of Ω and all the pairs of Ω have the same action, then a step is called *simple* and can be denoted by $s \xrightarrow{a} (\Omega', \mathcal{F}', P')$, where $(\Omega', \mathcal{F}', P')$ is a discrete probability distribution over states. The separation between nondeterministic and probabilistic behavior is achieved by means of *adversaries* (or schedulers), that, similar to [7, 19, 21], choose a next step to schedule based on the past history of the automaton. In our case, differently from [7, 19, 21], we allow an adversary to choose the next step randomly. Indeed, an external environment that provides some input essentially behaves like a randomized adversary.

Our first major result is that randomized adversaries do not change the distinguishing power of PCTL and WPCTL. Intuitively, the main reason for this result is that PCTL and WPCTL are concerned with probability bounds rather than exact probabilities.

We then redefine the *strong bisimulation* relation of [7, 13] in terms of our model, and also define a *strong simulation* relation that generalizes the simulation relation of [11], strengthening it a bit so that some liveness is preserved. We show that strong simulation preserves PCTL formulas without negation and existential quantification, and we show that the kernel of strong simulation preserves PCTL formulas without existential quantification. Next, we generalize the strong relations by making them insensitive to probabilistic combination of steps, i.e., by allowing probabilistic combination of several transitions in order to simulate a single transition. The motivation for this generalization is that the combination of transitions corresponds to the ability of an adversary to choose the next step probabilistically. Our second main result is that the new relations, called *strong probabilistic bisimulation* and *strong probabilistic simulation*, are still compositional and preserve PCTL formulas and PCTL formulas without negation and existential quantification, respectively.

Similar to the strong case, we define new relations that abstract from internal computation and we show that they preserve WPCTL. However, the straightforward generalization of the strong probabilistic relations, although compositional, does not guarantee that

[2] Discreteness is needed because of measurability issues.

WPCTL is preserved. For this reason we introduce other two relations, called *branching probabilistic bisimulation* and *branching probabilistic simulation*, which impose new restrictions similar to those of branching bisimulation [6]. Our third main result is that branching probabilistic bisimulation and branching probabilistic simulation are compositional and preserve PCTL formulas and PCTL formulas without negation and existential quantification, respectively, up to a condition about divergences.

We conclude with a discussion about some related work in [11]. In particular we show how the idea of *refinement* of [11] applies to our framework. We define a refinement preorder in the style of [11] for each simulation relation of this paper, and, surprisingly, we show that none of the new refinements is compositional. However, the counterexample that we present gives some insight for possible solutions to the problem.

The rest of the paper is organized as follows. Section 2 defines the standard automata of non-probabilistic systems; Section 3 introduces our probabilistic model; Section 4 introduces PCTL, defines its semantics in terms of our model, and shows that the distinguishing power of PCTL does not change by using randomized adversaries; Sections 5 and 6 study the strong and weak relations, respectively, on our probabilistic model, and show how they preserve PCTL formulas; Section 7 contains some concluding remarks concerning the refinement-based preorders of [11] and further work.

2 Automata

An *automaton* A consists of four components: a set $states(A)$ of states, a nonempty set $start(A) \subseteq states(A)$ of start states, an action signature $sig(A) = (ext(A), int(A))$ where $ext(A)$ and $int(A)$ are disjoint sets of external and internal actions, respectively, and a transition relation $steps(A) \subseteq states(A) \times acts(A) \times states(A)$, where $acts(A)$ denotes the set $ext(A) \cup int(A)$ of actions. Thus, an automaton is a state machine with labeled steps (also called transitions). Its action signature describes the interface with the external environment by specifying which actions model events that are visible from the external environment and which ones model internal events.

An *execution fragment* α of an automaton A is a (finite or infinite) sequence of alternating states and actions starting with a state and, if the execution fragment is finite, ending in a state, $\alpha = s_0 a_1 s_1 a_2 s_2 \cdots$, where each $(s_i, a_{i+1}, s_{i+1}) \in steps(A)$. Denote by $fstate(\alpha)$ the first state of α and, if α is finite, denote by $lstate(\alpha)$ the last state of α. Furthermore, denote by $frag^*(A)$ and $frag(A)$ the sets of finite and all execution fragments of A, respectively. An *execution* is an execution fragment whose first state is a start state. Denote by $exec^*(A)$ and $exec(A)$ the sets of finite and all execution of A, respectively. A state s of A is *reachable* if there exists a finite execution that ends in s. A finite execution fragment $\alpha_1 = s_0 a_1 s_1 \cdots a_n s_n$ of A and an execution fragment $\alpha_2 = s_n a_{n+1} s_{n+1} \cdots$ of A can be *concatenated*. In this case the concatenation, written $\alpha_1 \frown \alpha_2$, is the execution fragment $s_0 a_1 s_1 \cdots a_n s_n a_{n+1} s_{n+1} \cdots$. An execution fragment α_1 of A is a *prefix* of an execution fragment α_2 of A, written $\alpha_1 \leq \alpha_2$, if either $\alpha_1 = \alpha_2$ or α_1 is finite and there exists an execution fragment α_1' of A such that $\alpha_2 = \alpha_1 \frown \alpha_1'$.

3 The Basic Probabilistic Model

3.1 Probabilistic Automata

Definition 1. A *probability space* is a triplet $(\Omega, \mathcal{F}, P)$ where Ω is a set, $\mathcal{F}$ is a collection of subsets of Ω that is closed under complement and countable union and such that

$\Omega \in \mathcal{F}$, and P is a function from $\mathcal{F}$ to $[0,1]$ such that $P[\Omega] = 1$ and for any collection $\{C_i\}_i$ of at most countably many pairwise disjoint elements of $\mathcal{F}$, $P[\cup_i C_i] = \sum_i P[C_i]$.

A probability space $(\Omega, \mathcal{F}, P)$ is *discrete*[3] if $\mathcal{F} = 2^\Omega$ and for each $C \subseteq \Omega$, $P[C] = \sum_{x \in C} P[\{x\}]$. It is immediate to verify that for every discrete probability space there are at most countably many points with a positive probability measure.

The Dirac distribution over an element x, denoted by $\mathcal{D}(x)$, is the probability space with a unique element x.

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

Definition 2. A *probabilistic automaton* M is an automaton whose transition relation $\mathit{steps}(M)$ is a subset of $\mathit{states}(M) \times \mathit{Probs}((\mathit{acts}(M) \times \mathit{states}(M)) \cup \{\delta\})$, where $\mathit{Probs}(X)$ is the set of discrete probability spaces $(\Omega, \mathcal{F}, P)$ where $\Omega \subseteq X$.

A probabilistic automaton M is *simple* if for each step $(s, (\Omega, \mathcal{F}, P)) \in \mathit{steps}(M)$ there is an action $a \in \mathit{acts}(M)$ such that $\Omega \subseteq \{a\} \times \mathit{states}(M)$. In such a case a step can alternatively be represented as $(s, a, (\Omega, \mathcal{F}, P))$ where $(\Omega, \mathcal{F}, P) \in \mathit{Probs}(\mathit{states}(M))$, and it is called a *simple step with action a*.

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

Thus a probabilistic automaton differs from an automaton in that the action and the next state of a given transition are chosen probabilistically. The symbol δ that can appear in the sample space of each transition represents those situations where a system deadlocks. Thus, for example, it is possible that from a state s a probabilistic automaton performs some action with probability p and deadlocks with probability $1 - p$.

A simple probabilistic automaton does not allow any kind of probabilistic choice on actions. Once a step is chosen, then the next action is determined and the next state is given by a random distribution. Several systems in practice can be described as simple probabilistic automata; indeed our analysis will focus on simple probabilistic automata and we will use general probabilistic automata only for the analysis of probabilistic schedulers.

A fully probabilistic automaton is a probabilistic automaton without nondeterminism; at each point only one step can be chosen..

The generative model of probabilistic processes of [5] is a special case of a fully probabilistic automaton; simple probabilistic automata are partially captured by the reactive model of [5] in the sense that the reactive model assumes some form of nondeterminism between different actions. However, the reactive model does not allow nondeterministic choices between steps involving the same action. By restricting simple probabilistic automata to have finitely many states, we obtain objects with a structure similar to that of the Concurrent Labeled Markov Chains of [7]; however, in our model we do not need to distinguish between nondeterministic and probabilistic states. In our model nondeterminism is obtained by means of the structure of the transition relation. This allows us to retain most of the traditional notation that is used for automata.

Definition 3. Given a probabilistic automaton M, its *nondeterministic reduction* $\mathcal{N}(M)$ is the automaton A obtained from M by transforming each transition $(s, (\Omega, \mathcal{F}, P))$ into the set of transitions (s, a, s') where $(a, s') \in \Omega$. In other words $\mathcal{N}(M)$ is obtained from M by transforming all the probabilistic behavior into nondeterministic behavior. □

[3] If we accept the Axiom of Choice, then the requirement $\mathcal{F} = 2^\Omega$ is sufficient.

The execution fragments and executions of a probabilistic automaton M are the execution fragments and executions of its nondeterministic reduction $\mathcal{N}(M)$. However, for the study of the probabilistic behavior of a probabilistic automaton, some more detailed structure is needed. Such a structure, which we call an *execution automaton*, is introduced in Section 3.2.

The next definition shows how it is possible to combine several steps of a probabilistic automaton into a new one. It plays a fundamental role for the definition of probabilistic adversaries and the definition of our probabilistic simulations.

Definition 4. Given a probabilistic automaton M, a finite or countable set $\{(\Omega_i, \mathcal{F}_i, P_i)\}_i$ of probability distributions of $\mathit{Probs}((\mathit{acts}(M) \times \mathit{states}(M)) \cup \{\delta\})$, and a positive weight p_i for each i such that $\sum_i p_i \leq 1$, the combination $\sum_i p_i(\Omega_i, \mathcal{F}_i, P_i)$ of the distributions $\{(\Omega_i, \mathcal{F}_i, P_i)\}$ is the probability space $(\Omega, \mathcal{F}, P)$ such that

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

A pair $(s, (\Omega, \mathcal{F}, P))$ is a *combined step* of M if there exists a finite or countable family of steps $\{(s, (\Omega_i, \mathcal{F}_i, P_i))\}_i$ and a set of positive weights $\{p_i\}_i$ with $\sum_i p_i \leq 1$, such that $(\Omega, \mathcal{F}, P) = \sum_i p_i(\Omega_i, \mathcal{F}_i, P_i)$ □

For notational convenience we write $s \xrightarrow{a} (\Omega, \mathcal{F}, P)$ whenever there is a simple step $(s, a, (\Omega, \mathcal{F}, P))$ in M, and we write $s \xrightarrow{a}_{\mathrm{P}} (\Omega, \mathcal{F}, P)$ whenever there is a simple combined step $(s, a, (\Omega, \mathcal{F}, P))$ in M. We extend the arrow notation to weak arrows ($\stackrel{a}{\Longrightarrow}$ and $\stackrel{a}{\Longrightarrow}_{\mathrm{P}}$) to state that $(\Omega, \mathcal{F}, P)$ is reached through a sequence of steps, some of which are internal. Formally, $s \stackrel{a}{\Longrightarrow} (\Omega, \mathcal{F}, P)$ ($s \stackrel{a}{\Longrightarrow}_{\mathrm{P}} (\Omega, \mathcal{F}, P)$) iff there exists a (combined) step $(s, (\Omega', \mathcal{F}', P'))$ such that $(\Omega, \mathcal{F}, P) = \sum_{(b,s') \in \Omega'} P'[(b, s')](\Omega_{(b,s')}, \mathcal{F}_{(b,s')}, P_{(b,s')})$, where, for each $(b, s') \in \Omega'$, if $b = a$ then $s' \Longrightarrow (\Omega_{(b,s')}, \mathcal{F}_{(b,s')}, P_{(b,s')})$ ($s' \Longrightarrow_{\mathrm{P}} (\Omega_{(b,s')}, \mathcal{F}_{(b,s')}, P_{(b,s')})$), and if $b \neq a$ then b is internal and $s' \stackrel{a}{\Longrightarrow} (\Omega_{(b,s')}, \mathcal{F}_{(b,s')}, P_{(b,s')})$ ($s' \stackrel{a}{\Longrightarrow}_{\mathrm{P}} (\Omega_{(b,s')}, \mathcal{F}_{(b,s')}, P_{(b,s')})$). The relation $\Longrightarrow$ ($\Longrightarrow_{\mathrm{P}}$) differs from $\stackrel{a}{\Longrightarrow}$ ($\stackrel{a}{\Longrightarrow}_{\mathrm{P}}$) in that it is also possible not to move from s, i.e., it is possible that $s \Longrightarrow \mathcal{D}(s)$ ($s \Longrightarrow_{\mathrm{P}} \mathcal{D}(s)$).

We now turn to the parallel composition operator for simple probabilistic automata, which is defined in the CSP style [9]. As outlined in [7], the definition of a parallel composition operator for general probabilistic automata is problematic. We will address the issue of a general parallel composition operator in further work.

Definition 5. Two simple probabilistic automata M_1, M_2 are *compatible* if

1. $\mathit{int}(M_1) \cap \mathit{acts}(M_2) = \emptyset$, and
2. $\mathit{int}(M_2) \cap \mathit{acts}(M_1) = \emptyset$.

The *parallel composition* $M_1 \| M_2$ of compatible simple probabilistic automata M_1, M_2 is the simple probabilistic automaton M such that

1. $\mathit{states}(M) = \mathit{states}(M_1) \times \mathit{states}(M_2)$
2. $\mathit{start}(M) = \mathit{start}(M_1) \times \mathit{start}(M_2)$
3. $\mathit{ext}(M) = \mathit{ext}(M_1) \cup \mathit{ext}(M_2)$
4. $\mathit{int}(M) = \mathit{int}(M_1) \cup \mathit{int}(M_2)$

5. $((s_1, s_2), a, (\Omega, \mathcal{F}, P)) \in \mathit{steps}(M)$ iff $(\Omega, \mathcal{F}, P) = (\Omega_1, \mathcal{F}_1, P_1) \otimes (\Omega_2, \mathcal{F}_2, P_2)$, where $\otimes$ denotes the product of probability spaces, such that
   (a) if $a \in \mathit{acts}(M_1)$ then $(s_1, a, (\Omega_1, \mathcal{F}_1, P_1)) \in \mathit{steps}(M_1)$, else $(\Omega_1, \mathcal{F}_1, P_1) = \mathcal{D}(s_1)$, and
   (b) if $a \in \mathit{acts}(M_2)$ then $(s_2, a, (\Omega_2, \mathcal{F}_2, P_2)) \in \mathit{steps}(M_2)$, else $(\Omega_2, \mathcal{F}_2, P_2) = \mathcal{D}(s_2)$. □

3.2 Schedulers and Adversaries

Several papers in the literature use schedulers, sometimes viewed as adversarial entities, to resolve the nondeterminism in probabilistic systems [4, 7, 14, 21]. An adversary is an object that schedules the next step based on the past history of a probabilistic automaton.

Definition 6. An *adversary* for a probabilistic automaton M is a function $\mathcal{A}$ taking a finite execution fragment α of M and returning a probability distribution over $\perp$ and a subset of the steps enabled from $\mathit{lstate}(\alpha)$. Formally, $\mathcal{A} : \mathit{frag}^*(M) \rightarrow \mathit{Probs}(\mathit{steps}(M) \cup \{\perp\})$, such that if $\mathcal{A}(\alpha) = (\Omega, \mathcal{F}, P)$ and $(s, (\Omega', \mathcal{F}', P')) \in \Omega$, then $s = \mathit{lstate}(\alpha)$. An adversary is *deterministic* if it returns only Dirac distributions, i.e., the next step is chosen deterministically. Denote the set of adversaries and deterministic adversaries for a probabilistic automaton M by $\mathit{Advs}(M)$ and $\mathit{DAdvs}(M)$, respectively. □

The symbol $\perp$ in Definition 6 is used to express the fact that an adversary is allowed not to schedule anyone at any point. Such an option is useful when some specific actions are meant to model input from the external environment.

Definition 7. An *adversary schema* for a probabilistic automaton M, denoted by Advs, is a subset of $\mathit{Advs}(M)$. If Advs is a proper subset of $\mathit{Advs}(M)$ then Advs is a *restricted adversary schema*, otherwise Advs is a *full adversary schema*. □

Adversary schemas are used to reduce the power of a class of adversaries. Note, for example, that the set of deterministic adversaries $\mathit{DAdvs}(M)$ is an example of a restricted adversary schema whenever M is not fully probabilistic. Throughout the rest of this paper we denote by $\mathit{Probabilistic}(M)$ the adversary schema where each adversary can choose $\perp$ on input α iff there is no step enabled in M from $\mathit{lstate}(\alpha)$, and we denote by $\mathit{Deterministic}(M)$ the set of deterministic adversaries of $\mathit{Probabilistic}(M)$.

The next step is to define what it means for a probabilistic automaton to run under the control of an adversary. Namely, suppose that M has already performed some execution fragment α and that an adversary $\mathcal{A}$ starts resolving the nondeterminism at that point. The result of the interaction between M and $\mathcal{A}$ is a fully probabilistic automaton, called an *execution automaton*, where at each point the only step enabled is the step due to the choice of $\mathcal{A}$. A similar construction appears in [21]. Unfortunately, the definition of an execution automaton is not simple since each state contains the past history of M.

Definition 8. An *execution automaton* H of a probabilistic automaton M is a fully probabilistic automaton such that

1. $\mathit{states}(H) \subseteq \mathit{frag}^*(M)$.
2. for each step $(\alpha, (\Omega, \mathcal{F}, P))$ of H there is a combined step $(\mathit{lstate}(\alpha), (\Omega', \mathcal{F}', P'))$ of M, called the *corresponding combined step*, such that $\Omega' = \{(a, s) | (a, \alpha a s) \in \Omega\}$, $\mathcal{F}' = 2^{\Omega'}$, and $P'[(a, s)] = P[(a, \alpha a s)]$ for each $(a, s) \in \Omega'$. If $q = \mathit{lstate}(\alpha)$, then denote $(\Omega, \mathcal{F}, P)$ by $(\Omega_q, \mathcal{F}_q, P_q)$.

3. each state of H is reachable, i.e., for each $\alpha \in \mathit{states}(H)$ there exists an execution of $\mathcal{N}(H)$ leading to state α. □

Now we can define formally what it means for a probabilistic automaton M to run under the control of an adversary $\mathcal{A}$.

Definition 9. Given a probabilistic automaton M, an adversary $\mathcal{A} \in \mathit{Advs}(M)$, and an execution fragment $\alpha \in \mathit{frag}^*(M)$, the execution $H(M, \mathcal{A}, \alpha)$ of M under adversary $\mathcal{A}$ with starting fragment α is the execution automaton of M whose start state is α and such that for each state q there is a step $(q, (\Omega, \mathcal{F}, P)) \in \mathit{steps}(H(M, \mathcal{A}, \alpha))$ iff $\mathcal{A}(q) \neq \mathcal{D}(\perp)$ and the corresponding combined step of $(q, (\Omega, \mathcal{F}, P))$ is obtained from $\mathcal{A}(q)$. □

3.3 Events

We define a probability space $(\Omega_H, \mathcal{F}_H, P_H)$ for each execution automaton H, so that it is possible to analyze the probabilistic behavior of an automaton once the nondeterminism is removed. The sample space Ω_H is the set of maximal executions of H, where a maximal execution of H is either infinite or finite and not extendible. Specific kinds of not extendible executions are finite executions α whose last state enables a step where δ has a positive probability. Those executions are denoted by $\alpha\delta$. Note that an execution of H can be uniquely denoted by the corresponding execution fragment of M. Thus, to ease the notation, we define an operator $\alpha\!\uparrow$ that takes an execution fragment of M and gives back the corresponding execution of H, and $\alpha\!\downarrow$ that takes an execution of H and gives back the corresponding execution fragment of M.

For each finite execution α of H, possibly extended with δ, let R_α, the rectangle with prefix α, be the set $\{\alpha' \in \Omega_H \mid \alpha \leq \alpha'\}$, and let $\mathcal{R}_H$ be the class of rectangles for H. The probability $\mu_H(R_\alpha)$ of the rectangle R_α is the product of the probabilities associated with each edge that generates α in H. This is well defined since the steps of H are described by discrete probability distributions. Formally, if $\alpha = q_0 a_1 q_1 \cdots q_{n-1} a_n q_n$, where each q_i is an execution fragment of M, then $\mu_H(R_\alpha) \triangleq P_{q_0}[(a_1, q_1)] \cdots P_{q_{n-1}}[(a_n, q_n)]$. If $\alpha = q_0 a_1 q_1 \cdots q_{n-1} a_n q_n \delta$, then $\mu_H(R_\alpha) \triangleq P_{q_0}[(a_1, q_1)] \cdots P_{q_{n-1}}[(a_n, q_n)] P_{q_n}[\delta]$. Standard measure theory results assert that there is a unique measure $\bar{\mu}_H$ that extends μ_H to the σ-algebra $\sigma(\mathcal{R}_H)$ generated by $\mathcal{R}_H$. $\mathcal{F}_H$ is then obtained from $\sigma(\mathcal{R}_H)$ by extending each event with any set of executions taken from 0-probability rectangles, and P_H is obtained by extending $\bar{\mu}_H$ to $\mathcal{F}_H$ in the obvious way. With this definition it is possible to show that any union of rectangles (even uncountable) is measurable. In fact, at most countably many rectangles have a positive measure.

In our analysis of probabilistic automata we are not interested in events for single execution automata. Whenever we want to express a property, we want to express it relative to any execution automaton. This is the purpose of event schemas.

Definition 10. An *event schema* e for a probabilistic automaton M is a function that associates an event of $\mathcal{F}_H$ with each execution automaton H of M. □

4 Probabilistic Computation Tree Logic

In this section we present the logic that is used for our analysis, and we give it a semantics based on our model. It is a simplification of the *Timed Probabilistic concurrent Computation Tree Logic* (TPCTL) of [7], where we do not consider time issues. Then, we show that randomized adversaries do not change the distinguishing power of the logic.

Consider a set of actions ranged over by a. The syntax of PCTL formulas is defined as follows:

$$f ::= a \mid \neg f \mid f_1 \wedge f_2 \mid \mathcal{J}\mathcal{A} f \mid f_1\ EU_{\geq p}\ f_2 \mid f_1\ AU_{\geq p}\ f_2 \mid f_1\ EU_{>p}\ f_2 \mid f_1\ AU_{>p}\ f_2$$

Informally, the atomic formula a means that action a is the only one that can occur during the first step of a probabilistic automaton; the formula $\mathcal{J}\mathcal{A} f$ means that f is valid for a probabilistic automaton M after making the first transition invisible; the formula $f_1\ EU_{\geq p}\ f_2$ means that there exists an adversary such that the probability of f_2 eventually holding and f_1 holding till f_2 holds is at least p; the formula $f_1\ AU_{\geq p}\ f_2$ means that the same property as above is valid for each adversary. For the formal semantics of PCTL we need two auxiliary operators on probabilistic automata.

Let M be a probabilistic automaton, a an action of M, and s a state of M. Then $M[(a,s)]$ is a probabilistic automaton obtained from M by adding a new state s', adding a new step $(s', a, \mathcal{D}(s))$, and making s' into the unique start state. In other words $M[(a,s)]$ forces M to start with action a and then reach state s.

Let M be a probabilistic automaton. Then $\overrightarrow{M}$ is obtained from M by adding a duplicate of each start state, by making the duplicate states into the new start states, and, for each step $s \xrightarrow{a} (\Omega, \mathcal{F}, P)$ of M, by adding a step $s' \xrightarrow{\tau} (\Omega, \mathcal{F}, P)$ from the duplicate s' of s, where τ is an internal action that cannot occur in any PCTL formula. In other words $\overrightarrow{M}$ makes sure that the first step of M is invisible.

Let M be a probabilistic automaton, and let α be an execution of M. Let $\sqsupseteq$ denote either $\geq$ or $>$. Then we define the satisfaction relations $M \models f$ and $\alpha \models_M g$ as follows

$M \models a$ iff each step leaving from a start state is a simple step with action a,
$M \models \neg f$ iff not $M \models f$,
$M \models f_1 \wedge f_2$ iff $M \models f_1$ and $M \models f_2$,
$\alpha \models_M f_1\ U\ f_2$ iff there exists $n > 0$ such that $\alpha = s_0 a_1 s_1 \cdots a_n s_n \frown \alpha'$, $M[(a_n, s_n)] \models f_2$, and for each $i, 1 \leq i < n$, $M[(a_i, s_i)] \models f_1$,
$M \models \mathcal{J}\mathcal{A} f$ iff $\overrightarrow{M} \models f$,
$M \models f_1\ EU_{\sqsupseteq p}\ f_2$ iff there exists an adversary $\mathcal{A}$ and a start state s_0 such that $P_H[e_{f_1 U f_2}(H)] \sqsupseteq p$, where $H = H(M, \mathcal{A}, s_0)$, and $e_{f_1 U f_2}(H)$ is the set of executions α' of Ω_H such that $\alpha'{\downarrow} \models_M f_1\ U\ f_2$,
$M \models f_1\ AU_{\sqsupseteq p}\ f_2$ iff for each adversary $\mathcal{A}$ and each start state s_0, $P_H[e_{f_1 U f_2}(H)] \sqsupseteq p$, where $H = H(M, \mathcal{A}, s_0)$, and $e_{f_1 U f_2}(H)$ is the set of executions α' of Ω_H such that $\alpha'{\downarrow} \models_M f_1\ U\ f_2$.

Note that for each execution automaton H the set $e_{f_1 U f_2}(H)$ can be expressed as a union of rectangles, and thus it is an element of $\mathcal{F}_H$. This guarantees that the semantics of PCTL is well defined. In the definition above we did not mention explicitly what kind of adversaries to consider for the validity of a formula. In [7] the adversaries are assumed to be deterministic. However, the semantics does not change by adding randomization to the adversaries. The intuitive justification of this claim is that if we are just interested in upper and lower bounds to the probability of some event to happen, then any probabilistic combination of events stays within the bounds. Moreover, deterministic adversaries are sufficient to observe the bounds.

Theorem 11. *For each probabilistic automaton M and each PCTL formula f, $M \models f$ relative to Deterministic(M) iff $M \models f$ relative to Probabilistic(M).*

Proof sketch. The proof is by induction on the structure of the formula f, and most of it is simple routine checking. Two critical points are the following: if $M \models f_1 \; EU_{\sqsupseteq p} \; f_2$ relative to randomized adversaries, then we need to make sure that there exists at least a deterministic adversary that can be used to satisfy $f_1 \; EU_{\sqsupseteq p} \; f_2$; if $M \models f_1 \; AU_{\sqsupseteq p} \; f_2$ relative to deterministic adversaries, then we need to make sure that no probabilistic adversary would lead to a violation of $f_1 \; AU_{\sqsupseteq p} \; f_2$. In both cases the idea is to convert a probabilistic adversary $\mathcal{A}$ for a probabilistic automaton M into a deterministic one such that the probability of $e_{f_1 U f_2}$ is increased (first case) or decreased (second case). □

We now show how to change the syntax and semantics of PCTL to abstract away from internal computation. The new logic is denoted by WPCTL. The syntax of WPCTL is the same as that of PCTL with the additional requirement that no internal action can occur in a formula. For the semantics of WPCTL, there are three main changes.

$M \models a$ iff each weak step leaving from a start state is labeled with action a,
$\alpha \models_M f_1 \; U \; f_2$ iff there exists $n > 0$ such that $\alpha = s_0 a_1 s_1 \cdots a_n s_n \frown \alpha'$,
a_n is external, $M[(a_n, s_n)] \models f_2$, and for each $i, 1 \leq i < n$,
if a_i is external, then $M[(a_i, s_i)] \models f_1$,
$M \models \mathcal{J}\mathcal{A} f$ iff $\overset{\Rightarrow}{M} \models f$,

where $\overset{\Rightarrow}{M}$ hides the first external steps of M, i.e., it is obtained from M by duplicating all its states (and then removing the non-reachable ones at the end), by making the duplicates of the old start states into the new start states, by reproducing all the internal transitions in the duplicated states, and, for each external step $(s, a, (\Omega, \mathcal{F}, P))$ of M, by adding an internal step $(s', \tau, (\Omega, \mathcal{F}, P))$ from the duplicate s' of s, where τ is a new internal action. Note that the satisfaction relation for an execution is defined solely in terms of its external steps.

Theorem 12. *For each probabilistic automaton M and each WPCTL formula f, $M \models f$ relative to Deterministic(M) iff $M \models f$ relative to Probabilistic(M).* □

5 Strong Relations

In this section we analyze relations that are sensitive to internal computation. We formalize the bisimulations of [7] (strong bisimulation) and the simulations of [11,13] (strong simulation) in our model, and we show that the kernel of strong simulation, which is coarser than strong bisimulation, preserves PCTL formulas that do not contain $EU_{\sqsupseteq p}$. We then introduce other two coarser relations that allow probabilistic combination of steps and continue to preserve PCTL formulas without $EU_{\sqsupseteq p}$. For convenience, throughout the rest of this paper we assume that no pair of probabilistic automata has any state in common.

Definition 13. Let $\mathcal{R}$ be an equivalence relation over a set X. Two probability spaces $(\Omega_1, \mathcal{F}_1, P_1)$ and $(\Omega_2, \mathcal{F}_2, P_2)$ of *Probs*(X) are $\mathcal{R}$-equivalent, written $(\Omega_1, \mathcal{F}_1, P_1) \equiv_{\mathcal{R}} (\Omega_2, \mathcal{F}_2, P_2)$, iff for each $[x_1]_{\mathcal{R}} \in \Omega_1/\mathcal{R}$ there exists an $[x_2]_{\mathcal{R}} \in \Omega_2/\mathcal{R}$ such that $x_1 \; \mathcal{R} \; x_2$, for each $[x_2]_{\mathcal{R}} \in \Omega_2/\mathcal{R}$ there exists an $[x_1]_{\mathcal{R}} \in \Omega_1/\mathcal{R}$ such that $x_2 \; \mathcal{R} \; x_1$, and for each $[x_1]_{\mathcal{R}} \in \Omega_1/\mathcal{R}$, $[x_2]_{\mathcal{R}} \in \Omega_2/\mathcal{R}$ such that $x_1 \; \mathcal{R} \; x_2$, $\sum_{x \in \Omega_1 \cap [x_1]_{\mathcal{R}}} P[x] = \sum_{x \in \Omega_2 \cap [x_2]_{\mathcal{R}}} P[x]$. In other words $(\Omega_1, \mathcal{F}_1, P_1)$ and $(\Omega_2, \mathcal{F}_2, P_2)$ are $\mathcal{R}$-equivalent if they assign the same probability measure to each equivalence class of $\mathcal{R}$. □

Definition 14. A *strong bisimulation* between two simple probabilistic automata M_1, M_2 is an equivalence relation $\mathcal{R}$ over $\mathit{states}(M_1) \cup \mathit{states}(M_2)$ such that

1. each start state of M_1 is related to at least one start state of M_2, and vice versa;
2. for each $s_1 \ \mathcal{R} \ s_2$ and each step $s_1 \xrightarrow{a} (\Omega_1, \mathcal{F}_1, P_1)$ of either M_1, M_2, there exists a step $s_2 \xrightarrow{a} (\Omega_2, \mathcal{F}_2, P_2)$ of either M_1, M_2 such that $(\Omega_1, \mathcal{F}_1, P_1) \equiv_{\mathcal{R}} (\Omega_2, \mathcal{F}_2, P_2)$.

We write $M_1 \simeq M_2$ whenever $\mathit{acts}(M_1) = \mathit{acts}(M_2)$ and there is a strong bisimulation between M_1 and M_2. □

Condition 2 of Definition 14 is stated in [7, 13] in a different but equivalent way, i.e., for each equivalence class $[x]$ of $\mathcal{R}$, the probabilities of reaching $[x]$ from s_1 and s_2 are the same. The next definition is used to introduce strong simulations. It appears in a similar form in [11]. Informally, $(\Omega_1, \mathcal{F}_1, P_1) \sqsubseteq_{\mathcal{R}} (\Omega_2, \mathcal{F}_2, P_2)$ means that there is a way to split the probabilities of the states of Ω_1 between the states of Ω_2 and vice versa, expressed by a weight function w, so that the relation $\mathcal{R}$ is preserved. In other words the left probability space can be embedded into the right one up to $\mathcal{R}$.

Definition 15. Let $\mathcal{R} \subseteq X \times Y$ be a relation between two set X, Y, and let $(\Omega_1, \mathcal{F}_1, P_1)$ and $(\Omega_2, \mathcal{F}_2, P_2)$ be two probability spaces of $\mathit{Probs}(X)$ and $\mathit{Probs}(Y)$, respectively. Then $(\Omega_1, \mathcal{F}_1, P_1)$ and $(\Omega_2, \mathcal{F}_2, P_2)$ are in relation $\sqsubseteq_{\mathcal{R}}$, written $(\Omega_1, \mathcal{F}_1, P_1) \sqsubseteq_{\mathcal{R}} (\Omega_2, \mathcal{F}_2, P_2)$, iff there exists a weight function $w : X \times Y \to [0, 1]$ such that

1. for each $x \in X$, $\sum_{y \in Y} w(x, y) = P_1[x]$,
2. for each $y \in Y$, $\sum_{x \in X} w(x, y) = P_2[y]$,
3. for each $(x, y) \in X \times Y$, if $w(x, y) > 0$ then $x \ \mathcal{R} \ y$. □

Definition 16. A *strong simulation* between two simple probabilistic automata M_1, M_2 is a relation $\mathcal{R} \subseteq \mathit{states}(M_1) \times \mathit{states}(M_2)$ such that

1. each start state of M_1 is related to at least one start state of M_2;
2. for each $s_1 \ \mathcal{R} \ s_2$ and each step $s_1 \xrightarrow{a} (\Omega_1, \mathcal{F}_1, P_1)$ of M_1, there exists a step $s_2 \xrightarrow{a} (\Omega_2, \mathcal{F}_2, P_2)$ of M_2 such that $(\Omega_1, \mathcal{F}_1, P_1) \sqsubseteq_{\mathcal{R}} (\Omega_2, \mathcal{F}_2, P_2)$.
3. for each $s_1 \ \mathcal{R} \ s_2$, if $s_2 \xrightarrow{a}$, then $s_1 \xrightarrow{a}$.

We write $M_1 \sqsubseteq_{\mathrm{SS}} M_2$ whenever $\mathit{acts}(M_1) = \mathit{acts}(M_2)$ and there is a strong simulation between M_1 and M_2. The kernel of strong simulation is denoted by $\equiv_{\mathrm{SS}}$. □

The third requirement in the definition of a strong simulation is used to guarantee some minimum liveness requirements. It is fundamental for the preservation of PCTL formulas; however it can be relaxed by requiring s_1 to enable some step whenever s_2 enables some step.

Proposition 17. *$\simeq$ and $\sqsubseteq_{\mathrm{SS}}$ are compositional That is, for each M_1, M_2 such that $\mathit{acts}(M_1) = \mathit{acts}(M_2)$, and for each M_3 compatible with both M_1 and M_2, if $M_1 \simeq M_2$, then $M_1 \| M_3 \simeq M_2 \| M_3$, and if $M_1 \sqsubseteq_{\mathrm{SS}} M_2$, then $M_1 \| M_3 \sqsubseteq_{\mathrm{SS}} M_2 \| M_3$.* □

Lemma 18. *Let X, Y be two disjoint sets, $\mathcal{R}$ be an equivalence relation on $X \cup Y$, and let $(\Omega_1, \mathcal{F}_1, P_1)$ and $(\Omega_2, \mathcal{F}_2, P_2)$ be probability spaces of $\mathit{Probs}(X)$ and $\mathit{Probs}(Y)$, respectively, such that $(\Omega_1, \mathcal{F}_1, P_1) \equiv_{\mathcal{R}} (\Omega_2, \mathcal{F}_2, P_2)$. Then $(\Omega_1, \mathcal{F}_1, P_1) \sqsubseteq_{\mathcal{R}'} (\Omega_2, \mathcal{F}_2, P_2)$, where $\mathcal{R}' = \mathcal{R} \cap X \times Y$.* □

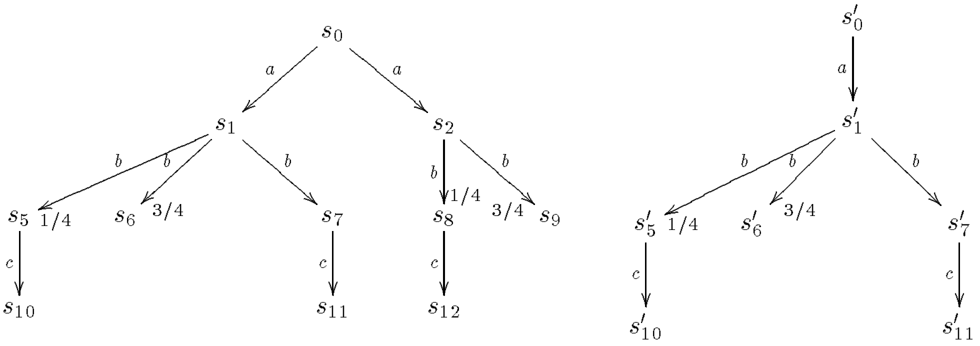
Lemma 18 can be used to prove directly that bisimulation is finer than simulation. The same observation applies to all the other pairs of relations that we define in this paper.

Theorem 19. *Let M_1 and M_2 be two simple probabilistic automata, and let f be a PCTL formula.*

1. *If $M_1 \simeq M_2$, then $M_1 \models f$ iff $M_2 \models f$.*
2. *If $M_1 \sqsubseteq_{\mathrm{SS}} M_2$ and f does not contain any occurrence of $\neg$ and $EU_{\sqsupseteq p}$, then $M_2 \models f$ implies $M_1 \models f$.*
3. *If $M_1 \equiv_{\mathrm{SS}} M_2$ and f does not contain any occurrence of $EU_{\sqsupseteq p}$, then $M_1 \models f$ iff $M_2 \models f$.*

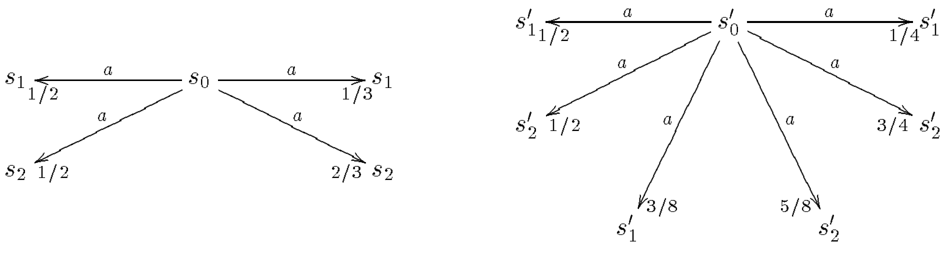
Proof sketch. The proofs are by induction on the structure of f, where the nontrivial step is the analysis of $f_1\ AU_{\sqsupseteq p}\ f_2$. In this case it is enough to show that for each execution automaton H_1 of M_1 there exists an execution automaton H_2 of M_2 such that $P_{H_2}[e_{f_1Uf_2}(H_2)] \leq P_{H_1}[e_{f_1Uf_2}(H_1)]$. The execution automaton H_2 is built by reproducing the structure of H_1 via $\mathcal{R}$. We also need to ensure that H_2 is obtainable from some adversary, and for this part we need Condition 3 of Definition 16. We do not need to show that H_2 can be generated by a deterministic adversary (indeed this is false in general) because of Theorem 11. □

Example 1. PCTL formulas with occurrences of $EU_{\sqsupseteq p}$ are not preserved in general by $\equiv_{\mathrm{SS}}$. Consider the two simple probabilistic automata below.



The two automata are strong simulation equivalent by matching each s_i with s'_i and by matching s_2, s_8, s_9, s_{12} to $s'_1, s'_5, s'_6, s'_{10}$, respectively. However, the right automaton satisfies $\mathit{true}\ AU_{\geq 1}\ (a \wedge (\mathit{true}\ EU_{\geq 1/2}\ c))$, whereas the left automaton does not. □

Example 2. Consider the two probabilistic automata



where s_0, s'_0 are the start states, s_1, s'_1 enable some step with action b, and s_2, s'_2 enable some step with action c. The difference between the left and right automata is that the right automaton enables an additional step which is obtained by combining the two steps of the left automaton. Thus, the two automata satisfy the same PCTL formulas; however, there is no simulation from the right automaton to the left one since the middle step cannot be reproduced. □

Example 2 suggests two coarser relations where it is possible to combine several steps into a unique one. Note that the only difference between the new preorders and the old ones is the use of $\xrightarrow{a}_{\mathrm{P}}$ (combined steps) instead of $\xrightarrow{a}$ (regular steps) in Condition 2.

Definition 20. A *strong probabilistic bisimulation* between two simple probabilistic automata M_1, M_2 is an equivalence relation $\mathcal{R}$ over $states(M_1) \cup states(M_2)$ such that

1. each start state of M_1 is related to at least one start state of M_2, and vice versa;
2. for each $s_1\ \mathcal{R}\ s_2$ and each step $s_1 \xrightarrow{a} (\Omega_1, \mathcal{F}_1, P_1)$ of either M_1, M_2, there exists a combined step $s_2 \xrightarrow{a}_{\mathrm{P}} (\Omega_2, \mathcal{F}_2, P_2)$ of either M_1, M_2 such that $(\Omega_1, \mathcal{F}_1, P_1) \equiv_{\mathcal{R}} (\Omega_2, \mathcal{F}_2, P_2)$.

We write $M_1 \simeq_{\mathrm{P}} M_2$ whenever $acts(M_1) = acts(M_2)$ and there is a strong probabilistic bisimulation between M_1 and M_2. □

Definition 21. A *strong probabilistic simulation* between two simple probabilistic automata M_1, M_2 is a relation $\mathcal{R} \subseteq states(M_1) \times states(M_2)$ such that

1. each start state of M_1 is related to at least one start state of M_2;
2. for each $s_1\ \mathcal{R}\ s_2$ and each step $s_1 \xrightarrow{a} (\Omega_1, \mathcal{F}_1, P_1)$ of M_1, there exists a combined step $s_2 \xrightarrow{a}_{\mathrm{P}} (\Omega_2, \mathcal{F}_2, P_2)$ of M_2 such that $(\Omega_1, \mathcal{F}_1, P_1) \sqsubseteq_{\mathcal{R}} (\Omega_2, \mathcal{F}_2, P_2)$.
3. for each $s_1\ \mathcal{R}\ s_2$, if $s_2 \xrightarrow{a}$, then $s_1 \xrightarrow{a}$.

We write $M_1 \sqsubseteq_{\mathrm{SPS}} M_2$ whenever $acts(M_1) = acts(M_2)$ and there is a strong probabilistic simulation between M_1 and M_2. The kernel of strong probabilistic simulation is denoted by $\equiv_{\mathrm{SPS}}$. □

Proposition 22. $\simeq_{\mathrm{P}}$ and $\sqsubseteq_{\mathrm{SPS}}$ are compositional. □

Theorem 23. *Let M_1 and M_2 be two simple probabilistic automata, and let f be a PCTL formula.*

1. *If $M_1 \simeq_{\mathrm{P}} M_2$, then $M_1 \models f$ iff $M_2 \models f$.*
2. *If $M_1 \sqsubseteq_{\mathrm{SPS}} M_2$ and f does not contain any occurrence of $\neg$ and $EU_{\sqsupseteq p}$, then $M_2 \models f$ implies $M_1 \models f$.*
3. *If $M_1 \equiv_{\mathrm{SPS}} M_2$ and f does not contain any occurrence of $EU_{\sqsupseteq p}$, then $M_1 \models f$ iff $M_2 \models f$.* □

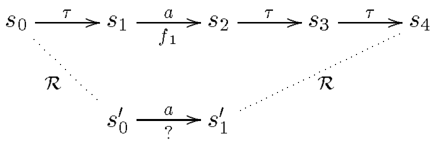
Remark. Our strong probabilistic simulations provides us with a simple way to represent the closed interval specification systems of [11]. A *probabilistic specification system* of [11] is a state machine where each state is associated with a set of probability distributions over the next state. The set of probability distributions for a state s is specified by associating each other state s' with a set of probabilities that can be used from s. In our framework a specification structure can be represented as a probabilistic automaton that, from each state, enables one step for each valid probability distribution over the next states. A *probabilistic process system* is a "fully probabilistic" (in our terms) probabilistic specification system. A probabilistic process system P is said to satisfy a probabilistic specification system S if there exists a strong simulation from P to S.

A *closed interval specification system* is a specification system whose set of probability distributions are described by means of a lower bound and an upper bound, for each pair (s, s'), on the probability of reaching s' from s. Thus, the set of probability distributions that are allowed from any state form a polytope. By using our strong probabilistic simulation as satisfaction relation, it is possible to represent each polytope by means of its corners only. Any point within the polytope is then given by a combination of the corners. □

6 Weak Relations

The relations of Section 5 do not abstract from internal computation, whereas in practice a notion of implementation should ignore the internal steps of a system as much as possible. In this section we study the weak versions of the relations of Section 5, and we show how they relate to WPCTL. We introduce only the probabilistic version of each relation, since the others can be derived subsequently in a straightforward way. We start by presenting the natural extension of the probabilistic relations of Section 5; then, in order to preserve WPCTL, we introduce a branching version of the new relations using the basic idea of branching bisimulation [6].

Weak probabilistic bisimulations and weak probabilistic simulations can be defined in a straightforward manner by changing Condition 2 of Definitions 20 and 21 so that each step $s_1 \xrightarrow{a} (\Omega_1, \mathcal{F}_1, P_1)$ of an automaton can be simulated by a weak combined step $s_2 \stackrel{a\lceil ext(M_2)}{\Longrightarrow}_{\mathrm{P}} (\Omega_2, \mathcal{F}_2, P_2)$ of the other automaton, and by using weak steps in Condition 3. However, although the two weak relations are compositional, WPCTL formulas are not preserved by weak bisimulations and weak simulations. The key problem is that weakly bisimilar executions do not satisfy the same formulas. Consider the diagram below.



Since s'_1 and s_2 are not necessarily related, it is not possible to deduce $M[(a, s'_1)] \models f_1$ from $M[(a, s_2)] \models f_1$. To solve the problem we need to make sure that s'_1 and s_2 are related, and thus we introduce the branching versions of our weak relations.

Definition 24. A *branching probabilistic bisimulation* between two simple probabilistic automata M_1, M_2 is an equivalence relation $\mathcal{R}$ over $states(M_1) \cup states(M_2)$ such that

1. each start state of M_1 is related to at least one start state of M_2, and vice versa;
2. for each $s_1 \;\mathcal{R}\; s_2$ and each step $s_1 \xrightarrow{a} (\Omega_1, \mathcal{F}_1, P_1)$ of either M_1, M_2, there exists a weak combined step $s_2 \stackrel{a\lceil ext(M_2)}{\Longrightarrow}_{\mathrm{P}} (\Omega_2, \mathcal{F}_2, P_2)$ of either M_1, M_2 such that $(\Omega_1, \mathcal{F}_1, P_1) \equiv_{\mathcal{R}} (\Omega_2, \mathcal{F}_2, P_2)$ and $s_2 \stackrel{a\lceil ext(M_2)}{\Longrightarrow}_{\mathrm{P}} (\Omega_2, \mathcal{F}_2, P_2)$ satisfies the branching condition, i.e., for each path α in the step $s_2 \stackrel{a\lceil ext(M_2)}{\Longrightarrow}_{\mathrm{P}} (\Omega_2, \mathcal{F}_2, P_2)$, and each state s that occurs in α, either $s_1 \;\mathcal{R}\; s$, $a\lceil ext(M_2)$ has not occurred yet, and each state s' preceding s in α satisfies $s_1 \;\mathcal{R}\; s'$, or for each $s'_1 \in \Omega_1$ such that $s'_1 \;\mathcal{R}\; lstate(\alpha)$, $s'_1 \;\mathcal{R}\; s$.

We write $M_1 \simeq_{\mathrm{P}} M_2$ whenever $ext(M_1) = ext(M_2)$ and there is a branching probabilistic bisimulation between M_1 and M_2. □

Let a be an external action. Informally, the weak step $s_2 \stackrel{a\lceil ext(M_2)}{\Longrightarrow}_{\mathrm{P}} (\Omega_2, \mathcal{F}_2, P_2)$ in Definition 24 is obtained by concatenating several combined steps of M_2. Such a combination can be visualized as a tree of combined steps. The branching condition says that all the states of the tree that occur before action a are related to s_1, and that whenever a state s'_2 of Ω_2 is related to some state s'_1 of Ω_1, then all the states in the path from s_1 to s'_2 that occur after action a are related to s'_1 as well. In other words, each maximal path in the tree satisfies the branching condition of [6].

Definition 25. A *branching probabilistic simulation* between two simple probabilistic automata M_1, M_2 is a relation $\mathcal{R} \subseteq \mathit{states}(M_1) \times \mathit{states}(M_2)$ such that

1. each start state of M_1 is related to at least one start state of M_2;
2. for each $s_1 \mathcal{R} s_2$ and each step $s_1 \xrightarrow{a} (\Omega_1, \mathcal{F}_1, P_1)$ of M_1, there exists a weak combined step $s_2 \overset{a\lceil \mathit{ext}(M_2)}{\Longrightarrow}_{\mathrm{P}} (\Omega_2, \mathcal{F}_2, P_2)$ of M_2 such that $(\Omega_1, \mathcal{F}_1, P_1) \sqsubseteq_{\mathcal{R}} (\Omega_2, \mathcal{F}_2, P_2)$, and $s_2 \overset{a\lceil \mathit{ext}(M_2)}{\Longrightarrow}_{\mathrm{P}} (\Omega_2, \mathcal{F}_2, P_2)$ satisfies the branching condition.
3. for each $s_1 \mathcal{R} s_2$, if $s_2 \overset{a}{\Longrightarrow}$, then $s_1 \overset{a}{\Longrightarrow}$.

We write $M_1 \sqsubseteq_{\mathrm{BPS}} M_2$ whenever $\mathit{ext}(M_1) = \mathit{ext}(M_2)$ and there is a branching probabilistic simulation between M_1 and M_2. The kernel of branching probabilistic simulation is denoted by $\equiv_{\mathrm{BPS}}$. □

Proposition 26. $\simeq_{\mathrm{P}}$ *and* $\sqsubseteq_{\mathrm{BPS}}$ *are compositional.* □

To show that a WPCTL formulas are preserved by the different simulation relations, we need to guarantee that a probabilistic automaton is free from divergences with probability 1. The definition below allows a probabilistic automaton to exhibit infinite internal computation, but it requires that such a behavior can happen only with probability 0.

Definition 27. A probabilistic automaton M is *probabilistically convergent* if for each execution automaton H of M and each state q of H, the probability of diverging (performing infinitely many internal actions and no external actions) from q is 0, i.e., $P_H[\Theta_q] = 0$, where Θ_q is the set of infinite executions of H that pass through state q and that do not contain any external action after passing through state q. Note that Θ_q is measurable since it is the complement of a union of rectangles. □

Theorem 28. *Let M_1 and M_2 be two probabilistically convergent, simple probabilistic automata, and f be a WPCTL formula.*

1. *If $M_1 \simeq_{\mathrm{P}} M_2$, then $M_1 \models f$ iff $M_2 \models f$.*
2. *If $M_1 \sqsubseteq_{\mathrm{BPS}} M_2$ and f does not contain any occurrence of $\neg$ and $EU_{\exists p}$, then $M_2 \models f$ implies $M_1 \models f$.*
3. *If $M_1 \equiv_{\mathrm{BPS}} M_2$ and f does not contain any occurrence of $EU_{\exists p}$, then $M_1 \models f$ iff $M_2 \models f$.*

Proof sketch. Similar to the proof of Proposition 19. Here the construction of H_2 is much more complicated than in the proof of Proposition 19 due to the fact that we need to combine several weak steps. Moreover, we need to show that the branching requirement guarantees the preservation of properties between bisimilar executions. □

7 Concluding Remarks

7.1 Summary

We have extended some of the classical simulation relations to a new probabilistic model that distinguishes naturally between probabilistic and nondeterministic choice and that allows us to represent naturally randomized and/or restricted forms of scheduling policies. Our method of analysis was based on compositionality issues and preservation of PCTL and WPCTL formulas. We have observed that the distinguishing power of PCTL does not change if we allow randomization in the schedulers. Based on that, we have introduced a new set of relations whose main idea is that an automaton may combine probabilistically some of its steps in order to simulate another automaton.

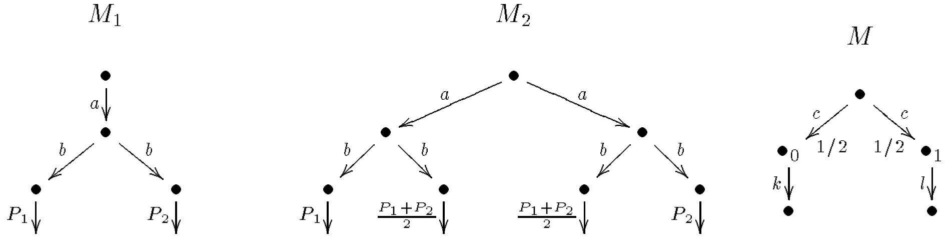
7.2 Refinement-Based Preorders

In [11] there is a notion of refinement between probabilistic specification systems stating (up to a notion of image-finiteness that is not important here) that S_1 refines S_2 if each probabilistic process system that satisfies S_1, also satisfies S_2. The notion of refinement of [11] suggests a new set of preorders based on the relations of Sections 5 and 6. Namely, given a preorder $\sqsubseteq_{\mathrm{X}}$, where X ranges over SS, SPS, WS, WPS, BS, BPS, we can define a new preorder $\preceq_{\mathrm{X}}$ as

$$M_1 \preceq_{\mathrm{X}} M_2 \text{ iff for each fully probabilistic } M_3, \text{ if } M_3 \sqsubseteq_{\mathrm{X}} M_1 \text{ then } M_3 \sqsubseteq_{\mathrm{X}} M_2.$$

Note, for example, that by restricting ourselves to the non-probabilistic case, $\preceq_{\mathrm{SS}}$ reduces to complete trace inclusion (including internal actions), while $\preceq_{\mathrm{WS}}$ and $\preceq_{\mathrm{BS}}$ reduce to complete external trace inclusion. Moreover, by removing Condition 3 in the refinements of Sections 5 and 6, $\preceq_{\mathrm{SS}}$ reduces to trace inclusion, while $\preceq_{\mathrm{WS}}$ and $\preceq_{\mathrm{BS}}$ reduce to external trace inclusion. Thus, we do not expect these preorders to be very strong. Unfortunately, none of the preorders above is compositional.

Example 3. Consider the three probabilistic automata



where the P_1-branch performs an action d and reaches a state enabling a new action z with probability 1/2, the P_2-branch performs an action d and reaches a state enabling a new action u with probability 1/2, and the $(P_1+P_2)/2$ branch is a combination of P_1 and P_2. It is easy to see that $M_1 \preceq_{\mathrm{X}} M_2$ and $M_2 \preceq_{\mathrm{X}} M_1$ for any X since any combination of steps of M_1 can be obtained from M_2 and vice versa; however, it is not the case that $M_1||M \preceq_{\mathrm{X}} M_2||M$. Consider the following execution automaton H of $M_1||M$: perform action a followed by action c; if state $\bullet_0$ is reached then perform the left b in M_1, perform d, k and possibly z, else, perform the right b in M_1, perform d, l and possibly u. Clearly H satisfies $M_1||M$; however, H does not satisfy $M_2||M$ since in $M_2||M$ there is no state that corresponds to the state reached in H after the occurrence of action a. In other words H correlates the occurrence of action k with action z, and the occurrence of action l with action u, whereas such a correlation is not possible in $M_2||M$. □

Based on Example 3 we may conclude that the preorders $\preceq_{\mathrm{X}}$ are not suitable for the compositional analysis of probabilistic systems. In reality it is still possible to use similar preorders if we make some additional assumptions. If we view a scheduler as an adversary, then we can say that an adversary chooses the next step based on the past history of the system. In Example 3 we have allowed the adversary to solve an internal choice of M_1 based on an internal condition of M. However, our counterexample would not work if the internal choices of each probabilistic automaton cannot be resolved by looking at the internal structure of other automata. A similar assumption is common for cryptographic systems.

7.3 Further Work

We are currently working on the definitions of adversary schemas that view other automata as black boxes, so that refinement-based preorders are compositional. Other further work includes finding some good notion of external behavior for probabilistic automata, studying applications of our results to the task of verifying probabilistic distributed systems, and extending our model and our results to handle real-time systems.

References

1. J. Aspnes and M.P. Herlihy. Fast randomized consensus using shared memory. *Journal of Algorithms*, 15(1):441–460, September 1990.
2. M. Ben-Or. Another advantage of free choice: completely asynchronous agreement protocols. In *Proceedings of the* 2^{nd} *PODC*, August 1983.
3. I. Christoff. Testing equivalences for probabilistic processes. Technical Report DoCS 90/22, Ph.D. Thesis, Department of Computer Science, Uppsala University, Sweden, 1990.
4. M. Fischer and L. Zuck. Reasoning about uncertainty in fault tolerant distributed systems. In *Proceedings of the Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 331, pages 142–158, 1988.
5. R.J. van Glabbeek, S.A. Smolka, B. Steffen, and C.M.N. Tofts. Reactive, generative, and stratified models of probabilistic processes. In *Proceedings* 5^{th} *LICS*, pages 130–141. IEEE Computer Society Press, 1990.
6. R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). In *Information Processing 89*, pages 613–618. North-Holland, 1989. Full version available as Report CS-R9120, CWI, Amsterdam, 1991.
7. H. Hansson. *Time and Probability in Formal Design of Distributed Systems.* PhD thesis, Department of Computer Science, Uppsala University, 1991.
8. M. Hennessy. *Algebraic Theory of Processes.* MIT Press, 1988.
9. C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, 1985.
10. B. Jonsson. Simulations between specifications of distributed systems. In *Proceedings CONCUR 91*, LNCS 527, pages 346–360, 1991.
11. B. Jonsson and K. G. Larsen. Specification and refinement of probabilistic processes. In *Proceedings of the 6th LICS*, July 1991.
12. E. Kushilevitz and M. Rabin. Randomized mutual exclusion algorithms revisited. In *Proceedings of the* 11^{th} *PODC*, pages 275–284, 1992.
13. K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, September 1992.
14. D. Lehmann and M. Rabin. On the advantage of free choice: a symmetric and fully distributed solution to the dining philosophers problem. In *Proceedings of the* 8^{th} *POPL* pages 133–138, January 1981.
15. N.A. Lynch and M.R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the* 6^{th} *PODC*, pages 137–151, Vancouver, Canada, August 1987. A full version is available as MIT Technical Report MIT/LCS/TR-387.
16. N.A. Lynch and F.W. Vaandrager. Forward and backward simulations for timing-based systems. In *Proceedings of the REX Workshop "Real-Time: Theory in Practice"*, LNCS 600, pages 397–446, 1991.
17. R. Milner. *Communication and Concurrency.* Prentice-Hall International, 1989.
18. A. Pnueli. The temporal semantics of concurrent programs. *TCS*, 13:45–60, 1982.
19. A. Pnueli and L. Zuck. Verification of multiprocess probabilistic protocols. *Distributed Computing*, 1(1):53–72, 1986.
20. K. Seidel. Probabilistic communicating processes. Technical Report PRG-102, Ph.D. Thesis, Programming Research Group, Oxford University Computing Laboratory, 1992.
21. M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings of 26th FOCS*, pages 327–338, Portland, OR, 1985.