Nordic Journal of Computing

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 [2, 3, 15]. 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 [4, 6, 8, 26, 29, 30]. 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., [21, 22].) 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 [8, 26, 30].

In the non-probabilistic setting, there are two principal methods that are used for analyzing labeled transition systems: temporal logic (e.g. [25]),

* Supported by NSF grant CCR-89-15206, and CCR-92-25124, by ARPA contracts N00014-89-J-1988 and N00014-92-J-4033, and by ONR contract N00014-91-J-1046.

Received July 1997.

which is used to establish that a system satisfies certain properties, and equivalence or preorder relations (e.g., [9, 21, 22, 24]), 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 [12, 20, 21]. Thus, it is highly desirable to extend the use of simulations to the probabilistic setting.

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 [8]; as discussed in [8], 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 [8, 30]. The model distinguishes between probabilistic and nondeterministic choices but, unlike the Concurrent Markov Chains of [8, 30], does not distinguish between probabilistic and nondeterministic states. A *probabilistic automaton* is a labeled transition system whose transition relation is a set of pairs $(s, \mathcal{P})$, where $\mathcal{P}$ is a discrete probability distribution over (action, state) pairs and a special symbol δ, representing deadlock. If the distribution $\mathcal{P}$ is only over pairs with the same action, then a transition is called *simple* and can be denoted by $s \xrightarrow{a} \mathcal{P}'$, where $\mathcal{P}'$ 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 [8, 26, 30], choose a next transition to schedule based on the past history of the automaton. In our case, differently from [8, 26, 30], we allow an adversary to choose the next transition 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 [16] in terms of our model, and also define a *strong simulation* relation that generalizes the simulation relation of [13], strengthening it a bit so that some liveness is preserved. We show that strong simulation preserves PCTL formulas without negation and existential quantification. Next, we generalize the strong relations by making them insensitive to probabilistic combination of transitions, 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 transition 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 WPCTL is preserved. For this reason we introduce two other relations, called *branching probabilistic bisimulation* and *branching probabilistic simulation*, which impose new restrictions similar to those of branching bisimulation [7]. Our third main result is that branching probabilistic bisimulation and branching probabilistic simulation are compositional and preserve WPCTL formulas and WPCTL formulas without negation and existential quantification, respectively, up to a condition about divergences.

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, 6 and 7 study the strong and weak relations on our probabilistic model, and show how they preserve PCTL formulas; Section 8 contains some concluding remarks 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 $trans(A) \subseteq states(A) \times acts(A) \times states(A)$, where $acts(A)$ denotes the set $ext(A) \cup int(A)$ of actions. Thus, an automaton is a state machine with labeled 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}) is a transition of 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 executions of A, respectively. A state s of A is *reachable* if there exists a finite execution of A 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 Probability Spaces

Most of our definitions rely on the notion of a probability space, which is used to denote which events can be observed and what are their probabilities.

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 $\mathcal{F}$, $P[\cup_i C_i] = \sum_i P[C_i]$.

The set Ω is called the *sample space* and contains the objects that we want to analyze. For example $\Omega = [0,1]$. The set $\mathcal{F}$ is called the σ*-algebra* and contains the subsets of Ω that we can measure, also called *events*. For example $\mathcal{F}$ can be the set of measurable sets of $[0,1]$ according to Lebesgue. Finally, P is called the *probability measure* and is a function that associates a measure with each element of $\mathcal{F}$. For example, P can associate each element of $\mathcal{F}$ with its Lebesgue measure.

A probability space $(\Omega, \mathcal{F}, P)$ is *discrete* if $\mathcal{F} = 2^\Omega$ and for each $C \subseteq \Omega$, $P[C] = \sum_{x \in C} P[\{x\}]$. It is immediate to verify that for every discrete probability space there are at most countably many points with a non-zero probability measure. Given a set X, we denote by $\mathit{Probs}(X)$ the set of discrete probability spaces $(\Omega, \mathcal{F}, P)$ whose sample space Ω is a subset of X.

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

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

The *product* 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 $(x_1, x_2) \in \Omega_1 \times \Omega_2$. In other words, the product of two discrete probability spaces $\mathcal{P}_1, \mathcal{P}_2$ is a new probability space that describes the operation of picking an element at random from $\mathcal{P}_1$ and $\mathcal{P}_2$ independently.

3.2 Probabilistic Automata

DEFINITION 1. A *probabilistic automaton* M consists of four components: a set $\mathit{states}(M)$ of states, a nonempty set $\mathit{start}(M) \subseteq \mathit{states}(M)$ of start states, an action signature $\mathit{sig}(M) = (\mathit{ext}(M), \mathit{int}(M))$ where $\mathit{ext}(M)$ and $\mathit{int}(M)$ are disjoint sets of external and internal actions, respectively, and a transition relation

$$\mathit{trans}(M) \subseteq \mathit{states}(M) \times \mathit{Probs}((\mathit{acts}(M) \times \mathit{states}(M)) \cup \{\delta\}),$$

where $\mathit{acts}(M)$ denotes the set $\mathit{ext}(M) \cup \mathit{int}(M)$ of actions.

A probabilistic automaton M is *simple* if for each transition $(s, \mathcal{P})$ of $\mathit{trans}(M)$ there is an action a of $\mathit{acts}(M)$ such that $\Omega \subseteq \{a\} \times \mathit{states}(M)$. In such a case a transition can be represented alternatively as $(s, a, \mathcal{P}')$ where $\mathcal{P}' \in \mathit{Probs}(\mathit{states}(M))$, and it is called a *simple transition 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 transition 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 transition is chosen, then the next action is determined and the next state is given by a random distribution.

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

An ordinary automaton is a special case of a probabilistic automaton where each transition leads to a Dirac distribution; the generative model of probabilistic processes of [6] is a special case of a fully probabilistic automaton; simple probabilistic automata are partially captured by the reactive model of [6] in the sense that the reactive model assumes some form of nondeterminism between different actions. However, the reactive model does not allow nondeterministic choices between transitions involving the same action. By restricting simple probabilistic automata to have finitely many states, we obtain objects with a structure similar to that of the Concurrent Labeled Markov Chains of [8]; however, in our model we do not need to distinguish between nondeterministic and probabilistic states. In our model nondeterminism is obtained by means of the structure of the transition relation. This allows us to retain most of the traditional notation that is used for automata.

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

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

Even though we have defined executions for a probabilistic automaton, for the study of the probabilistic behavior of a probabilistic automaton, some more detailed structure is needed. Such a structure, which we call a *probabilistic execution*, is introduced in Section 3.3.

The next definition shows how it is possible to combine several transitions of a probabilistic automaton into a new one. Informally, a combined transition leaving from a state s is obtained by choosing a transition that leaves from s probabilistically, and then behaving according to the transition chosen. Combined transitions play a fundamental role for the definition of probabilistic adversaries and the definition of our probabilistic simulations.

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

- $\Omega = \begin{cases} \cup_i \Omega_i & \text{if } \sum_i p_i = 1 \\ \cup_i \Omega_i \cup \{\delta\} & \text{if } \sum_i p_i < 1 \end{cases}$

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

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

For notational convenience we write $s \xrightarrow{a} \mathcal{P}$ whenever there is a simple transition $(s, a, \mathcal{P})$ in M, and we write $s \xrightarrow{a}_{\mathrm{C}} \mathcal{P}$ whenever there is a simple combined transition $(s, a, \mathcal{P})$ in M. We write $s \xrightarrow{a}$ whenever there exists a probability space $\mathcal{P}$ such that $s \xrightarrow{a} \mathcal{P}$.

We now turn to the parallel composition operator, which is defined in the CSP style [11], i.e., by synchronizing two automata on their common actions. As outlined in [8], it is not clear how to define a parallel composition operator for general probabilistic automata that extends the CSP operator of ordinary automata; thus, we only define it for simple probabilistic automata.

Definition 4. Two simple probabilistic automata M_1 and 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 and 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) $((s_1, s_2), a, \mathcal{P}) \in \mathit{trans}(M)$ iff $\mathcal{P} = \mathcal{P}_1 \otimes \mathcal{P}_2$, such that
    (a) if $a \in \mathit{acts}(M_1)$ then $(s_1, a, \mathcal{P}_1) \in \mathit{trans}(M_1)$, else $\mathcal{P}_1 = \mathcal{D}(s_1)$, and
    (b) if $a \in \mathit{acts}(M_2)$ then $(s_2, a, \mathcal{P}_2) \in \mathit{trans}(M_2)$, else $\mathcal{P}_2 = \mathcal{D}(s_2)$. □

Our analysis in this paper will focus on simple probabilistic automata, and we use general probabilistic automata only for the analysis of probabilistic schedulers. Several systems can be described as simple probabilistic automata. A probabilistic Turing Machine where we assume that each cell of the random tape is instantiated when it is read for the first time is a simple probabilistic automaton with a unique action, say τ, whose states are the instantaneous descriptions of the given machine; an algorithm that at some point can flip a coin or roll a dice can be represented as a simple probabilistic automaton where the flipping and rolling operations are simple transitions. If the outcome of a coin flip or dice roll affects the external behavior of the automaton, then the flip and roll actions can be followed by simple transitions whose actions represent the outcome of the random choice. We emphasize that if we introduce an input/output distinction as in [20], then it is possible to compose general probabilistic automata under the conditions that their input actions appear only in simple transitions. A similar observation appears in [31].

3.3 Schedulers and Adversaries

Several papers in the literature use schedulers, sometimes viewed as adversarial entities, to resolve the nondeterminism in probabilistic systems [5, 8, 17, 30]. An adversary (or scheduler) is an object that schedules the next transition based on the past history of a probabilistic automaton. The next transition can be chosen probabilistically.

DEFINITION 5. An *adversary* for a probabilistic automaton M is a function $\mathcal{A}$ taking a finite execution fragment α of M and returning a combined transition of M that leaves from $lstate(\alpha)$. Formally, $\mathcal{A} : frag^*(M) \rightarrow ctrans(M)$ such that if $\mathcal{A}(\alpha) = (s, \mathcal{P})$, then $s = lstate(\alpha)$. An adversary is *deterministic* if on input α it returns either transitions of M or the pair $(lstate(\alpha), \mathcal{D}(\delta))$, i.e., the next transition is chosen deterministically. Denote the set of adversaries for a probabilistic automaton M by $Advs(M)$. □

Observe that δ can appear in the combined transitions chosen by an adversary. Such an option is useful when the actions enabled from some state are meant to model input from the external environment and the adversary plays the role of an environment that is not providing any input.

DEFINITION 6. An *adversary schema* for a probabilistic automaton M, denoted by $Advs$, is a subset of $Advs(M)$. If $Advs$ is a proper subset of $Advs(M)$ 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 is an example of a restricted adversary schema whenever M is not fully probabilistic. Other examples of restricted adversary schemas are sets of adversaries that base their choices only on partial knowledge of the past history. We refer the reader to [1, 19] for examples of analysis of distributed algorithms based on restricted adversary schemas.

In this paper, in order to guarantee some minimal liveness, we impose a different restriction on our adversaries. Specifically, we denote by $Padvs(M)$ the adversary schema where each adversary can choose δ with a non-zero probability on input α iff there is no transition enabled in M from $lstate(\alpha)$, and we denote by $Dadvs(M)$ the set of deterministic adversaries of $Padvs(M)$. In other words, our adversaries must schedule something whenever something can be scheduled.

We next define what it means for a probabilistic automaton to run under the control of an adversary. Namely, suppose that M has already performed some finite 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 a *probabilistic execution*, where at each point the only transition enabled is the transition due to the choice of $\mathcal{A}$. A similar construction appears in [30]. Unfortunately, the definition of a probabilistic execution is not simple since each state contains the past history of M.

DEFINITION 7. A *probabilistic execution fragment* H of a probabilistic automaton M is a fully probabilistic automaton such that

(1) $states(H) \subseteq frag^*(M)$.

(2) for each transition $(\alpha, \mathcal{P})$ of H there exists a combined transition $(lstate(\alpha), \mathcal{P}')$ of M, called the *corresponding combined transition*, such that

$$\Omega' = \{(a, s) | (a, \alpha a s) \in \Omega\} \cup (\{\delta\} \cap \Omega), \text{ and}$$

$$P'[(a, s)] = P[(a, \alpha a s)]$$

for each $(a, s) \in \Omega'$. If $q = lstate(\alpha)$, then denote $\mathcal{P}$ by $\mathcal{P}_q^H$ and denote $(\alpha, \mathcal{P})$ by tr_q^H.

(3) each state of H is reachable and enables one transition.

A *probabilistic execution* of M is a probabilistic execution fragment of M whose start state is a start state of M. □

Condition (1) says that the states of a probabilistic execution H contain the past history of M; Condition (2) ensures that the transitions of H are derived from transitions of M by including the history in the new states that are reached; Condition (3) is just technical to eliminate useless states and to handle δ uniformly. Observe that a state q may enable a transition $(q, \mathcal{D}(\delta))$.

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

DEFINITION 8. Given a probabilistic automaton M, an adversary $\mathcal{A}$ for M, and a finite execution fragment α of M, the execution $H(M, \mathcal{A}, \alpha)$ of M under adversary $\mathcal{A}$ with starting fragment α is the unique probabilistic execution fragment of M whose start state is α and such that for each state q, if $(q, \mathcal{P}) \in trans(H(M, \mathcal{A}, \alpha))$, then the corresponding combined transition of $(q, \mathcal{P})$ is $\mathcal{A}(q)$. □

3.4 Events

We define a probability space $(\Omega_H, \mathcal{F}_H, P_H)$ for each probabilistic execution fragment H, so that it is possible to analyze the probabilistic behavior of a probabilistic automaton once the nondeterminism is removed. This construction is slightly different from the construction presented in [28].

First of all, we observe that there is a strong correspondence between the extended execution fragments of a probabilistic automaton and the extended executions of one of its probabilistic execution fragments. We express this correspondence by means of an operator $\alpha{\uparrow}q_0^H$ that takes an extended execution fragment of M and gives back the corresponding extended execution of H, and $\alpha{\downarrow}$ that takes an extended execution of H and gives back the corresponding extended execution fragment of M.

Then, the sample space Ω_H can be defined as the set of extended executions of M that correspond to complete extended execution fragments of H, where an extended execution α of H is complete iff it is either infinite or $\alpha = \alpha'\delta$ and $\delta \in \Omega^H_{lstate(\alpha')}$. For each finite extended execution fragment α of M, let C_α, the *cone* with prefix α, be the set $\{\alpha' \in \Omega_H \mid \alpha \leq \alpha'\}$, and let $\mathcal{C}_H$ be the class of cones for H. The probability $\mu_H(C_\alpha)$ of the cone C_α is the product of the probabilities associated with each edge that generates α in H. Formally, let q_0 be the start state of H, and let s_0 be $lstate(q_0)$. If $\alpha = q_0 \frown s_0 a_1 s_1 \cdots s_{n-1} a_n s_n$, then $\mu_H(C_\alpha) \triangleq P^H_{q_0}[(a_1, q_1)] \cdots P^H_{q_{n-1}}[(a_n, q_n)]$, where each q_i is $q_0 \frown s_0 a_1 s_1 \cdots a_i s_i$, and if $\alpha = q_0 \frown s_0 a_1 q_1 \cdots q_{n-1} a_n q_n \delta$, then $\mu_H(C_\alpha) \triangleq P^H_{q_0}[(a_1, q_1)] \cdots P^H_{q_{n-1}}[(a_n, q_n)] P^H_{q_n}[\delta]$, where each q_i is defined as before. In [27] it is shown that there is a unique measure $\bar{\mu}_H$ that extends μ_H to the σ-field $\sigma(\mathcal{C}_H)$ generated by $\mathcal{C}_H$, i.e., the smallest σ-field that contains $\mathcal{C}_H$. $\mathcal{F}_H$ is then obtained from $\sigma(\mathcal{C}_H)$ by extending each event with any set of extended executions taken from 0-probability cones, 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 cones (even uncountable) is measurable. In fact, at most countably many cones have a non-zero measure.

Examples of events are the occurrence of a specific finite execution α, which is C_α, and the occurrence of a specific action a, which can be represented as the union of cones C_α such that action a occurs in α.

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

Definition 9. An *event schema* e for a probabilistic automaton M is a function that associates an event of $\mathcal{F}_H$ with each probabilistic execution fragment H of M. $\square$

An example of an event schema is the function that associates with each probabilistic execution fragment H the event of performing a specific action a.

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 [8], where, unlike in [8], 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:

$$\begin{array}{lll} f & ::= & a \mid \neg f \mid f_1 \wedge f_2 \mid \mathcal{JA} f \\ & & f_1\ EU_{\geq p}\ f_2 \mid f_1\ AU_{\geq p}\ f_2 \mid f_1\ EU_{>p}\ f_2 \mid f_1\ AU_{>p}\ f_2 \end{array}$$

Informally, the atomic formula a means that action a is the only one that can occur during the first transition of a probabilistic automaton and that action a must indeed occur; the formula $\mathcal{JA}f$ 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 example, the property that under any scheduling policy action a occurs eventually with probability at least $1/2$ is expressed by the formula *true* $AU_{\geq 1/2}\ a$, where *true* can be expressed by the formula $\neg(a \wedge \neg a)$. 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 transition $(s', a, \mathcal{D}(s))$, 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 $\vec{M}$ 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 transition $s \xrightarrow{a} \mathcal{P}$ of M where s is a start state, by adding a transition $s' \xrightarrow{\tau} \mathcal{P}$ from the duplicate s' of s, where τ is an internal action that cannot occur in any PCTL formula. In other words $\vec{M}$ makes sure that the first transition of M is invisible.

Let M be a probabilistic automaton, and let α be an extended execution of M. Let $\sqsupseteq$ denote either $\geq$ or $>$. Let an execution α of M be complete iff either it is infinite or it is finite and no transition is enabled in M from $\mathit{lstate}(\alpha)$. Then we define the satisfaction relations $M \models f$ and $\alpha \models_M g$ as follows

$M \models a$	iff each complete execution of M starts 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 is $n > 0$ such that $\alpha = s_0 a_1 s_1 \cdots a_n s_n \frown \alpha'$, for each $i, 1 \leq i < n$, $M[(a_i, s_i)] \models f_1$, and $M[(a_n, s_n)] \models f_2$,
$M \models \mathcal{JA} f$	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 elements α' of Ω_H such that $\alpha' \models_M f_1\ U\ f_2$,
$M \models f_1\ AU_{\sqsupseteq p}\ f_2$	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 elements α' of Ω_H such that $\alpha' \models_M f_1\ U\ f_2$.

Note that for each probabilistic execution H the set $e_{f_1 U f_2}(H)$ can be expressed as a union of cones, 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 [8] 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, then any probabilistic combination of events stays within the bounds. Moreover, deterministic adversaries are sufficient to observe the bounds.

THEOREM 1. *For each probabilistic automaton M and each PCTL formula f, $M \models f$ relative to $Dadvs(M)$ iff $M \models f$ relative to $Padvs(M)$.*

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). The conversion is shown in [27]. □

We now show how to change the syntax and semantics of PCTL to abstract 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 complete extended execution of M has at least one external action, and its first external action is 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 $M[(a_i, s_i)] \models f_1$,
$M \models \mathcal{JA} f$	iff $\overset{\Rightarrow}{M} \models f$,

where $\overset{\Rightarrow}{M}$ hides the first external transitions 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 transition $(s, a, \mathcal{P})$ of M, by adding an internal transition $(s', \tau, \mathcal{P})$ 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 transitions.

THEOREM 2. *For each probabilistic automaton M and each WPCTL formula f, $M \models f$ relative to Dadvs(M) iff $M \models f$ relative to Padvs(M).*

5. Strong Relations

In this section we analyze relations that are sensitive to internal computation. We formalize in our model the bisimulations of [16] (strong bisimulation) following the lines of [8], and the simulations of [13, 16] (strong simulation); then, we show that strong bisimulation preserves PCTL and that strong simulation preserves PCTL formulas that do not contain negation and $EU_{\sqsupseteq p}$. We then introduce two other coarser relations that allow probabilistic combination of transitions and continue to preserve PCTL formulas and PCTL formulas without negation and $EU_{\sqsupseteq p}$, respectively. For convenience, throughout the rest of this paper we assume that no pair of probabilistic automata has any state in common.

DEFINITION 10. 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 every class C of $X/\mathcal{R}$,

$$\sum_{x \in \Omega_1 \cap C} P_1[x] = \sum_{x \in \Omega_2 \cap C} P_2[x].$$

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 $\mathcal{R}$. $\square$

DEFINITION 11. A *strong bisimulation* between two simple probabilistic automata M_1 and 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 transition $s_1 \xrightarrow{a} \mathcal{P}_1$ of either M_1, M_2, there exists a transition $s_2 \xrightarrow{a} \mathcal{P}_2$ of either M_1, M_2 such that $\mathcal{P}_1 \equiv_{\mathcal{R}} \mathcal{P}_2$.

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

Condition 2 of Definition 11 is stated in [16] 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. Strong bisimulation coincides with the strong bisimulation of [22, 24] whenever the involved probabilistic automata represent ordinary automata.

The next definition is used to introduce strong simulations. A similar definition appears in [13]. 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 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 12. 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 $Probs(X)$ and $Probs(Y)$, 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 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 $x \mathcal{R} y$.

The function w is called a *weight function*. □

DEFINITION 13. A *strong simulation* between two simple probabilistic automata M_1 and M_2 is a relation $\mathcal{R} \subseteq states(M_1) \times states(M_2)$ such that

(1) each start state of M_1 is related to at least one start state of M_2;

(2) for each $s_1 \mathcal{R} s_2$ and each transition $s_1 \xrightarrow{a} \mathcal{P}_1$ of M_1, there exists a transition $s_2 \xrightarrow{a} \mathcal{P}_2$ of M_2 such that $\mathcal{P}_1 \sqsubseteq_{\mathcal{R}} \mathcal{P}_2$.

(3) for each $s_1 \mathcal{R} s_2$, if $s_2 \xrightarrow{a}$, then $s_1 \xrightarrow{a}$.

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

If we do not include Condition 3 in the definition of a strong simulation, then we obtain a relation that extends the strong simulation relation of ordinary automata. Here we add Condition 3 to guarantee some minimum liveness requirements, thus extending the 2/3-bisimulation relation of [16]. Condition 3 is fundamental for the preservation of PCTL formulas; however it can be relaxed by requiring s_1 to enable some transition whenever s_2 enables some transition.

PROPOSITION 1. *$\simeq$ and $\sqsubseteq_{\rm SS}$ are compositional. That is, for each M_1 and M_2 such that $acts(M_1) = acts(M_2)$, 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 $M_1 \sqsubseteq_{\rm SS} M_2$, then $M_1 \| M_3 \sqsubseteq_{\rm SS} M_2 \| M_3$.* $\Box$

LEMMA 1. *Let X, Y be two disjoint sets, $\mathcal{R}$ be an equivalence relation on $X \cup Y$, and let $\mathcal{P}_1$ and $\mathcal{P}_2$ be probability spaces of $Probs(X)$ and $Probs(Y)$, respectively, such that $\mathcal{P}_1 \equiv_{\mathcal{R}} \mathcal{P}_2$. Then $\mathcal{P}_1 \sqsubseteq_{\mathcal{R}'} \mathcal{P}_2$, where $\mathcal{R}' {=} \mathcal{R} \cap X \times Y$.*

Lemma 1 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 3. *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 $M_1 \sqsubseteq_{\rm SS} M_2$ and f does not contain any occurrence of $\neg$ and $EU_{\sqsupseteq p}$, then $M_2 \models f$ implies $M_1 \models f$.

Proof sketch. The proofs are by induction on the structure of f, where the nontrivial step is the analysis of $f_1\ AU_{\sqsupseteq p}\ f_2$ and $f_1\ EU_{\sqsupseteq p}\ f_2$. In the first case it is enough to show that for each probabilistic execution H_1 of M_1 obtainable from some adversary there exists a probabilistic execution H_2 of M_2, obtainable from some adversary, such that $P_{H_2}[e_{f_1 U f_2}(H_2)] \leq P_{H_1}[e_{f_1 U f_2}(H_1)]$. In the second case we need to make sure that $P_{H_2}[e_{f_1 U f_2}(H_2)] = P_{H_1}[e_{f_1 U f_2}(H_1)]$.

The probabilistic execution 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 13. Indeed, if δ occurs in a transition enabled from a state q of H_2, then there is some state q' of H_1 that corresponds to q via $\mathcal{R}$ and that contains δ in the transition that it enables. Then, $lstate(q')$ does not enable any transition in M_2, which, using

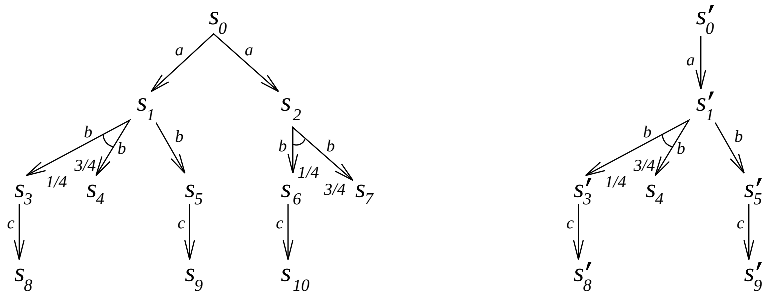

Fig. 1: Strong simulations do not preserve $EU_{\sqsupseteq p}$.

Fig. 2: Probabilistic combination of transitions is useful.

Condition 3, means that *lstate*(q) does not enable any transition in M_2. 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 1. The correspondence between H_1 and H_2 is called an *execution correspondence structure* and it is shown to exist in [27]. Once an execution correspondence structure is built, it is easy to show that $P_{H_2}[e_{f_1 U f_2}(H_2)] \leq P_{H_1}[e_{f_1 U f_2}(H_1)]$ if $\mathcal{R}$ is a strong simulation, and that $P_{H_2}[e_{f_1 U f_2}(H_2)] = P_{H_1}[e_{f_1 U f_2}(H_1)]$ if $\mathcal{R}$ is a strong bisimulation. $\square$

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 of Figure 1. The two probabilistic automata are strong simulation equivalent by matching each s_i with s'_i and by matching s_2, s_6, s_7, s_{10} to s'_1, s'_3, s'_4, s'_8, respectively. However, the right probabilistic automaton satisfies *true* $AU_{\geq 1}$ $(a \wedge (\mathit{true}\ EU_{\geq 1/2}\ c))$, whereas the left probabilistic automaton does not. $\square$

EXAMPLE 2. Consider the two probabilistic automata of Figure 2, where s_0, s'_0 are the start states, s_1, s'_1 enable some transition with action b, and s_2, s'_2 enable some transition with action c. The difference between the left and right probabilistic automata is that the right probabilistic automaton enables an additional transition which is obtained by combining the two transitions of the left probabilistic automaton. Thus, the two probabilistic automata satisfy the same PCTL formulas; however, there is no simulation from the right probabilistic automaton to the left one since the middle transition cannot be reproduced. □

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

DEFINITION 14. A *strong probabilistic bisimulation* between two simple probabilistic automata M_1 and 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 transition $s_1 \xrightarrow{a} \mathcal{P}_1$ of either M_1, M_2, there exists a combined transition $s_2 \xrightarrow{a}_{\mathrm{C}} \mathcal{P}_2$ of either M_1, M_2 such that $\mathcal{P}_1 \equiv_{\mathcal{R}} \mathcal{P}_2$.

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

(1) each start state of M_1 is related to at least one start state of M_2;

(2) for each $s_1 \mathcal{R} s_2$ and each transition $s_1 \xrightarrow{a} (\Omega_1, \mathcal{F}_1, P_1)$ of M_1, there exists a combined transition $s_2 \xrightarrow{a}_{\mathrm{C}} (\Omega_2, \mathcal{F}_2, P_2)$ 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 2. $\simeq_{\mathrm{P}}$ and $\sqsubseteq_{\mathrm{SPS}}$ are compositional. □

THEOREM 4. *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 $M_1 \models f$.* □

Even strong probabilistic bisimulations and strong probabilistic simulations are a generalization of the strong bisimulation and simulation relations of ordinary automata. In fact, if a transition of M_1 leading to a Dirac distribution can be simulated by a combined transition of M_2, then the same transition of M_1 can be simulated by a non-combined transition of M_2, which leads to a Dirac distribution if we are dealing with ordinary automata.

REMARK 1. Strong probabilistic simulations provide us with a simple way to represent the closed interval specification systems of [13]. A *probabilistic specification system* of [13] 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 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 transition for each one of the probability distributions over the next states that are allowed. 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 is 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 given by a combination of the corners. □

6. Weak Transitions

The relations of Section 5 do not abstract from internal computation, while in practice a notion of implementation should ignore the internal transitions of a system as much as possible. In order to do so, we extend our arrow notation in a way similar to the non-probabilistic case [22]. We define the weak arrows $\stackrel{a}{\Longrightarrow}$ and $\stackrel{a}{\Longrightarrow}_{\mathrm{C}}$ to state that a probability distribution over states $\mathcal{P}$ is reached through a sequence of transitions of M, some of which are internal. The main difference from the non-probabilistic case is that in our framework the transitions involved form a tree rather than a linear chain. Formally, $s \stackrel{a}{\Longrightarrow} \mathcal{P}$, where a is either an external action or the empty sequence and $\mathcal{P}$ is a probability distribution over states, iff there is a probabilistic execution fragment H such that

(1) the start state of H is s;

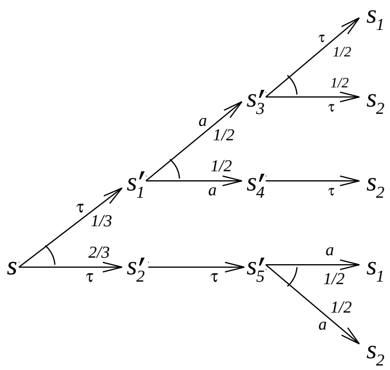

Fig. 3: A representation of a weak transition with action a.

(2) $P_H[\{\alpha\delta \mid \alpha\delta \in \Omega_H\}] = 1$, i.e., the probability of termination in H is 1;

(3) for each $\alpha\delta \in \Omega_H$, $\mathit{trace}(\alpha) = a$;

(4) $\Omega = \{\mathit{lstate}(\alpha) \mid \alpha\delta \in \Omega_H\}$, and for each element s' of Ω, $P[s'] = \sum_{\alpha\delta\in\Omega_H \mid \mathit{lstate}(\alpha)=s'} P_H[C_{\alpha\delta}]$;

(5) for each state q of H, either tr_q^H is the pair $(\mathit{lstate}(q), \mathcal{D}(\delta))$, or the transition that corresponds to tr_q^H is a transition of M.

A weak combined transition, $s \overset{a}{\Longrightarrow}_{\mathrm{C}} \mathcal{P}$, is defined as a weak transition by dropping Condition 5.

Example 3. The diagram of Figure 3 represents graphically a weak transition with action a that leads to state s_1 with probability $5/12$ and to state s_2 with probability $7/12$. We do not represent the states as finite execution fragments since their position in the diagram gives enough information. Similarly, we do not represent δ explicitly. The action τ represents any internal action. From the formal definition of a weak transition, a tree that represents a weak transition may have an infinite branching structure, i.e., it may have transitions that lead to countably many states, and may have some infinite paths; however, each tree representing a weak transition has the property that infinite paths occur with probability 0. This definition of a weak transition is more general than the definition given in [28], where it is required that no infinite path appear in a weak transition.
Figure 4 represents a weak transition of a probabilistic automaton with cycles in its transition relation. Specifically, H represents the weak transition $s_0 \Longrightarrow \mathcal{P}$, where $P[s_0] = 1/8$ and $P[s_1] = 7/8$. If we extend H indefinitely on its right, then we obtain a new probabilistic execution fragment that represents the weak transition $s_0 \Longrightarrow \mathcal{D}(s_1)$. Observe that the new probabilistic execution fragment has an infinite path that occurs with probability 0. Furthermore, observe that there is no other way to reach state s_1 with probability 1. $\square$

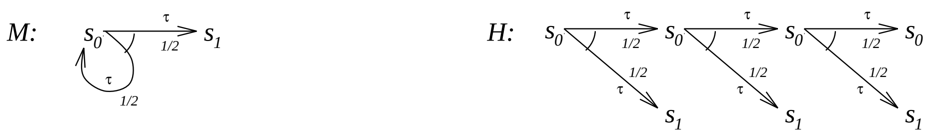

Fig. 4: A weak transition of a probabilistic automaton with cycles.

7. Weak Relations

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 [7].

Weak probabilistic bisimulations and weak probabilistic simulations can be defined in a straightforward manner by changing Condition 2 of Definitions 14 and 15 so that each transition $s_1 \xrightarrow{a} \mathcal{P}_1$ of a probabilistic automaton can be simulated by a weak combined transition $s_2 \overset{a\lceil ext(M_2)}{\Longrightarrow}_{\mathrm{C}} \mathcal{P}_2$ of the other probabilistic automaton, and by using weak transitions in Condition 3. Even in this case, with the opportune arguments about Condition 3, the weak probabilistic relations are an extension of the corresponding relations for ordinary automata. 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.

$$\begin{tikzcd}
s_0 \arrow[r, "\tau"] \arrow[dr, dotted, no head, "\mathcal{R}"'] & s_1 \arrow[r, "a", "f_1"'] & s_2 \arrow[r, "\tau"] & s_3 \arrow[r, "\tau"] & s_4 \arrow[dll, dotted, no head, "\mathcal{R}"] \\
 & s'_0 \arrow[r, "a", "?"'] & s'_1 & &
\end{tikzcd}$$

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 16. A *branching probabilistic bisimulation* between two simple probabilistic automata M_1 and 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 transition $s_1 \xrightarrow{a} \mathcal{P}_1$ of either M_1, M_2, there exists a weak combined transition $s_2 \overset{a\lceil ext(M_2)}{\Longrightarrow_{\mathrm{C}}} \mathcal{P}_2$ of either M_1, M_2 such that $\mathcal{P}_1 \equiv_{\mathcal{R}} \mathcal{P}_2$ and $s_2 \overset{a\lceil ext(M_2)}{\Longrightarrow_{\mathrm{C}}} \mathcal{P}_2$ satisfies the branching condition, i.e., there is a probabilistic execution fragment H that represents $s_2 \overset{a\lceil ext(M_2)}{\Longrightarrow_{\mathrm{C}}} \mathcal{P}_2$ such that for each extended execution $\alpha\delta$ of Ω_H and each occurrence of a state s in α, either

   (a) $s_1 \mathcal{R} s$, $a \in ext(M_2)$ implies that a has not occurred yet, and each state s' preceding s in α satisfies $s_1 \mathcal{R} s'$, or

   (b) $a \in ext(M_2)$ implies that a has occurred, and 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. □

Another way to state the branching condition is the following: there is a probabilistic execution fragment H that represents $s_2 \overset{a\lceil ext(M_2)}{\Longrightarrow_{\mathrm{C}}} \mathcal{P}_2$ such that, viewing H as a tree, all the the states of the tree that occur before action a are related to s_1, and 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_2 to s_2' that occur after action a are related to s_1' as well. In other words, each complete path in the tree satisfies the branching condition of [7].

DEFINITION 17. A *branching probabilistic simulation* between two simple probabilistic automata M_1 and M_2 is a relation $\mathcal{R} \subseteq states(M_1) \times states(M_2)$ such that

(1) each start state of M_1 is related to at least one start state of M_2;

(2) for each $s_1 \mathcal{R} s_2$ and each transition $s_1 \xrightarrow{a} (\Omega_1, \mathcal{F}_1, P_1)$ of M_1, there exists a weak combined transition $s_2 \overset{a\lceil ext(M_2)}{\Longrightarrow_{\mathrm{C}}} (\Omega_2, \mathcal{F}_2, P_2)$ of M_2 such that $(\Omega_1, \mathcal{F}_1, P_1) \sqsubseteq_{\mathcal{R}} (\Omega_2, \mathcal{F}_2, P_2)$, and $s_2 \overset{a\lceil ext(M_2)}{\Longrightarrow_{\mathrm{C}}} (\Omega_2, \mathcal{F}_2, P_2)$ 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 $ext(M_1) = ext(M_2)$ 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 3. *$\simeq_{\mathrm{P}}$ and $\sqsubseteq_{\mathrm{BPS}}$ are compositional.* □

To show that 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 for a probabilistic automaton to exhibit infinite internal computation, but it requires that such a behavior can happen only with probability 0.

DEFINITION 18. A probabilistic automaton M is *probabilistically convergent* if for each probabilistic execution 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 cones. □

THEOREM 5. *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 $EU_{\sqsupseteq p}$, then $M_2 \models f$ implies $M_1 \models f$.

Proof sketch. Similar to the proof of Proposition 3. Here the construction of H_2 is much more complicated than in the proof of Proposition 3 due to the fact that we need to combine several weak transitions. Moreover, we need to show that the branching requirement guarantees the preservation of properties between bisimilar executions. □

8. Concluding Remarks

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 is based on compositionality issues and preservation of PCTL and WPCTL formulas. Throughout the presentation we have shown how our relations are a conservative extension of the corresponding relations defined on ordinary automata. 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 collection of relations whose main idea is that a probabilistic automaton may combine some of its transitions probabilistically in order to simulate another probabilistic automaton.

In [27] this work is pursued further by extending the trace semantics of ordinary automata to the probabilistic framework. The key issue is compositionality, which is not trivial to achieve. We show that all the simulation relations of this paper are sound for the trace semantics, and we introduce other coarser simulation relations that capture the trace semantics better.

A problem that is still open is to derive adequacy results for PCTL and WPCTL. We do not know whether PCTL and WPCTL are adequate for our probabilistic simulation relations. In the non-probabilistic framework adequate logics for strong bisimulation are studied in [10], and adequate logics for branching bisimulation are studied in [23]. An adequate logic for strong bisimulation in the probabilistic framework is studied in [16]; however, no nondeterminism is present in the formalism of [16].

If PCTL and WPCTL are not adequate, then further research should focus either on finding adequate logics (what other operators do we need?) or finding more appropriate simulation relations. An important question is to determine whether WPCTL is sufficiently powerful to express all the properties of practical interest.

Acknowledgments. Thanks to the anonymous referees for their valuable suggestions on a draft version of this paper. In particular, one of the referees pointed out an error in Theorems 19, 23 and 28 of [28], which we have corrected here.

References

[1] Aggarwal, S. 1994. Time Optimal Self-Stabilizing Spanning Tree Algorithms. Tech. Report MIT/LCS/TR-632, MIT Laboratory for Computer Science.

[2] Aspnes, J. and Herlihy, M.P. 1990. Fast randomized consensus using shared memory. *Journal of Algorithms 15*, 1 (September), 441–460.

[3] Ben-Or, M. 1983. Another advantage of free choice: completely asynchronous agreement protocols. In *Proceedings of the* 2^{nd} *Annual ACM Symposium on Principles of Distributed Computing,* Montreal, Quebec, Canada.

[4] Christoff, I. 1990. *Testing Equivalences for Probabilistic Processes.* PhD thesis, Department of Computer Science, Uppsala University.

[5] Fischer, M. and Zuck, L. 1988. Reasoning about Uncertainty in Fault Tolerant Distributed Systems. In *Proceedings of the Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, Volume 331 of *Lecture Notes in Computer Science*. Springer-Verlag, 142–158.

[6] Glabbeek, R.J. van, Smolka, S.A., Steffen, B., and Tofts, C.M.N. 1990. Reactive, generative, and stratified models of probabilistic processes. In *Proceedings* 5^{th} *Annual Symposium on Logic in Computer Science,* Philadelphia, USA. IEEE Computer Society Press, 130–141.

[7] Glabbeek, R.J. van and Weijland, W.P. 1989. Branching Time and Abstraction in Bisimulation Semantics (extended abstract). In *Information Processing 89*. North-Holland, 613–618.

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

[9] Hennessy, M. 1988. *Algebraic Theory of Processes.* MIT Press, Cambridge, Massachusetts.

[10] Hennessy, M. and Milner, R. 1985. Algebraic Laws for Nondeterminism and Concurrency. *Journal of the ACM 32*, 1, 137–161.

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

[12] Jonsson, B. 1991. Simulations between Specifications of Distributed Systems. In *Proceedings of CONCUR 91,* Amsterdam, Volume 527 of *Lecture Notes in*

Computer Science. Springer-Verlag, 346–360.

[13] Jonsson, B. and Larsen, K.G. 1991. Specification and Refinement of Probabilistic Processes. In *Proceedings of the 6th IEEE Symposium on Logic in Computer Science*. Amsterdam, 266–277.

[14] Jonsson, B. and Parrow, J., Editors. 1994. *Proceedings of CONCUR 94,* Uppsala, Sweden, Volume 836 of *Lecture Notes in Computer Science*. Springer-Verlag.

[15] Kushilevitz, E. and Rabin, M. 1992. Randomized Mutual Exclusion Algorithms Revisited. In *Proceedings of the* 11^{th} *Annual ACM Symposium on Principles of Distributed Computing,* Quebec, Canada, 275–284.

[16] Larsen, K.G. and Skou, A. 1991. Bisimulation through Probabilistic Testing. *Information and Computation 94*, 1 (Sept.), 1–28.

[17] Lehmann, D. and Rabin, M. 1981. On the advantage of free choice: a symmetric and fully distributed solution to the dining philosophers problem. In *Proceedings of the* 8^{th} *Annual ACM Symposium on Principles of Programming Languages*, 133–138.

[18] 1990. *Proceedings* 5^{th} *Annual Symposium on Logic in Computer Science,* Philadelphia, USA. IEEE Computer Society Press.

[19] Lynch, N.A., Saias, I., and Segala, R. 1994. Proving Time Bounds for Randomized Distributed Algorithms. In *Proceedings of the* 13^{th} *Annual ACM Symposium on Principles of Distributed Computing,* Los Angeles, CA, 314–323.

[20] Lynch, N.A. and Tuttle., M.R. 1987. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the* 6^{th} *Annual ACM Symposium on Principles of Distributed Computing*. Vancouver, Canada, 137–151.

[21] Lynch, N.A. and Vaandrager, F.W. 1991. Forward and Backward Simulations for Timing-Based Systems. In *Proceedings of the REX Workshop "Real-Time: Theory in Practice"*, Volume 600 of *Lecture Notes in Computer Science*. Springer-Verlag, 397–446.

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

[23] Nicola, R. De and Vaandrager, F. W. 1990. Three Logics for Branching Bisimulation (extended abstract). In *Proceedings* 5^{th} *Annual Symposium on Logic in Computer Science,* Philadelphia, USA. IEEE Computer Society Press, 118–129.

[24] Park, D.M.R. 1981. Concurrency and automata on infinite sequences. In 5^{th} *GI Conference*, Volume 104 of *Lecture Notes in Computer Science*. Springer-Verlag, 167–183.

[25] Pnueli, A. 1982. The temporal semantics of concurrent programs. *Theoretical Computer Science 13*, 45–60.

[26] Pnueli, A. and Zuck, L. 1986. Verification of multiprocess probabilistic protocols. *Distributed Computing 1*, 1, 53–72.

[27] Segala, R. 1995. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT, Dept. of Electrical Engineering and Computer Science.

[28] Segala, R. and Lynch, N.A. 1994. Probabilistic Simulations for Probabilistic Processes. In *Proceedings of CONCUR 94,* Uppsala, Sweden, Volume 836 of *Lecture Notes in Computer Science*. Springer-Verlag, 481–496.

[29] Seidel, K. 1992. Probabilistic Communicating Processes. Tech. Report PRG-102, Ph.D. Thesis, Programming Research Group, Oxford University Computing Laboratory.

[30] Vardi, M.Y. 1985. Automatic Verification of Probabilistic Concurrent Finite-State Programs. In *Proceedings of 26th IEEE Symposium on Foundations of Computer Science*. Portland, OR, 327–338.

[31] Wu, S.H., Smolka, S., and Stark, E.W. 1994. Composition and behaviors of probabilistic I/O automata. In *Proceedings of CONCUR 94,* Uppsala, Sweden, Volume 836 of *Lecture Notes in Computer Science*. Springer-Verlag.