DOI 10.1007/s00165-012-0236-5
The Author(s) © 2012. This article is published with open access at Springerlink.com
Formal Aspects of Computing (2012) 24: 727–748

Formal Aspects of Computing

Probabilistic may/must testing: retaining probabilities by restricted schedulers

Sonja Georgievska and Suzana Andova

Department of Mathematics and Computer Science, Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands

Abstract. This paper considers the probabilistic may/must testing theory for processes having external, internal, and probabilistic choices. We observe that the underlying testing equivalence is too strong and distinguishes between processes that are observationally equivalent. The problem arises from the observation that the classical compose-and-schedule approach yields unrealistic overestimation of the probabilities, a phenomenon that has been recently well studied from the point of view of compositionality, in the context of randomized protocols and in probabilistic model checking. To that end, we propose a new testing theory, aiming at preserving the probability information in a parallel context. The resulting testing equivalence is insensitive to the exact moment the internal and the probabilistic choices occur. We also give an alternative characterization of the testing preorder as a probabilistic ready-trace preorder.

Keywords: Probabilistic may/must testing, Restricted schedulers, Ready-trace equivalence

1. Introduction and motivation

Adding probabilistic aspects to concurrency theory has been a field of research for more than two decades. Various formalisms and different semantical models have been proposed, a large number of them focusing on one of the most challenging questions, namely the one of combining nondeterminism and probability. In a pioneering work in this field [Mor96], Carroll Morgan, together with Annabelle McIver, Karen Seidel and Jeff Sanders, studied this issue in the context of Communicating Sequential Processes (CSP) and failures/divergences semantic model (see also [MMS96, SMM97, McM04]). One of the central lines of this work is extending CSP with probabilistic choice, while preserving as many as possible of the standard CSP laws. In addition to this result, not less inspiring, a number of important issues were stated and discussed, to list some of them: the need of linear-time testing semantics, the idea of unobservability of probabilistic choice in testing semantics, and consequently, distributivity of probabilistic choice over other operators, and the observation that idempotency of nondeterministic choice is a rather natural and desired property. The latter was followed up by a reasoning that the demonic nondeterminism *is* not well behaved due to its "unpleasant property of loss of idempotency" [Mor96] and the conclusion that a new type of indifferent nondeterminism is needed. For many researchers these points have been a valuable source of inspiration and motivation up to date. After about 15 years since the paper "Refinement-oriented probability for CSP" [Mor96] was published, the research community has agreed that, what was then called indifferent nondeterminism, and nowadays restricted schedulers, is the right way to combine probabilities, nondeterminism and concurrency. Our contribution to this research area is partially presented in this paper. We were largely inspired and motivated by the work of Carroll Morgan, and moreover, deeply glad to share his viewpoints.

Correspondence and offprint requests to: S. Georgievska, E-mail: sonja.georgievska@gmail.com
A preliminary version of this paper appeared as a conference paper [GeA10b], and in the PhD thesis of the first author [Geo11].

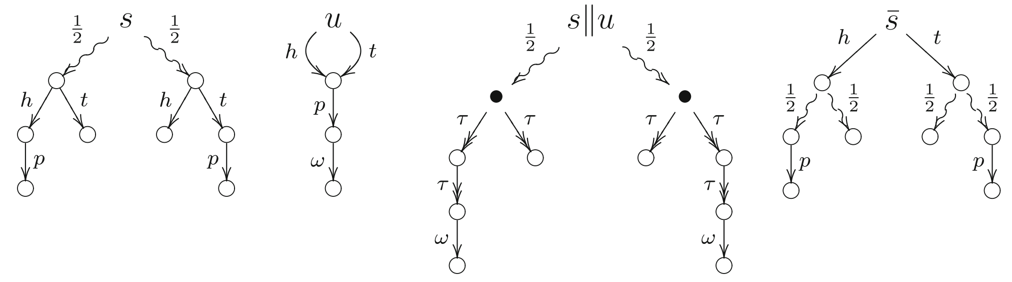


Fig. 1. The coin-flipping machine and the guessing user

A central question in the theory of concurrent processes is to define in which case two processes are to be considered equivalent. Various attempts to answer this question have led to the concepts of observational equivalence, bisimulation, testing equivalence, failure equivalence, etc. [Gla01, Gla93]. In the case of the may/must testing theory for concurrent processes [DeH84, Hen88] and its underlying semantics, two processes are equivalent if and only if they cannot be distinguished when interacting with the environment, which is an arbitrary process itself. Directly related to this positioning is the result that the exact moment an internal nondeterministic choice in a process is resolved is *unobservable*. This is contrary to bisimulation-based process theories where processes that differ only in the moment a choice is resolved are considered not equivalent; and therefore this moment is considered observable, e.g. [Mil80, BBR10]. On the other hand, the same (flexible) aspect of unobservable internal choice appears as well in the failures semantics of the theory of Communicating Sequential Processes (CSP) [BHR84, Hoa85, Ros98]. In fact, it has been shown in [DeN87] that testing equivalence [DeH84, Hen88] and failures equivalence [BHR84] coincide for a broad class of processes. In that sense, the underlying semantics of the may/must testing theory belongs to the linear-time spectrum [Gla01, Gla93].

In order to capture quantitative aspects of the system behaviour, some of the internal choices in a process may be refined by probabilistic choices. In this case, the probability with which a certain event happens depends on the way the remaining unrefined internal nondeterminism is resolved. This nondeterminism is usually resolved by means of a *scheduler* – a function that selects one of the alternatives, every time a process has to make an internal choice. For example, consider the process $s\|u$ in Fig. 1. It has one probabilistic choice, in the initial state, and two internal choices, in the states indicated with black bullets. There are four possible schedulers for this process: one scheduler selects the left τ-transition in the left internal choice and the left τ-transition in the right internal choice; another scheduler selects the left τ-transition in the left internal choice and the right τ-transition in the right internal choice, etc. Application of a scheduler to a process without external action choices (e.g. process $s\|u$), yields a fully probabilistic process, which can be further analyzed using, for instance, standard Markov chain analysis techniques. It is essential to note that different schedulers may yield different fully probabilistic processes. Therefore, the probability of an event occurring, is relative to the particular way the nondeterminism is resolved.

In the probabilistic extension of the may/must testing theory [WaL92], the internal nondeterminism is resolved by means of powerful schedulers, so-called *almighty schedulers*. These schedulers can select any of the possible alternatives, without any constraints. Stated in terms of information that the scheduler is allowed to use when making decisions, this means that an almighty scheduler knows all potentials and the complete structure of the process and the test. However, it was soon observed that this reasoning yields unrealistic probabilities with which a process passes a test [Mor96, Low93, Seg95]. We explain this phenomenon via the following examples.

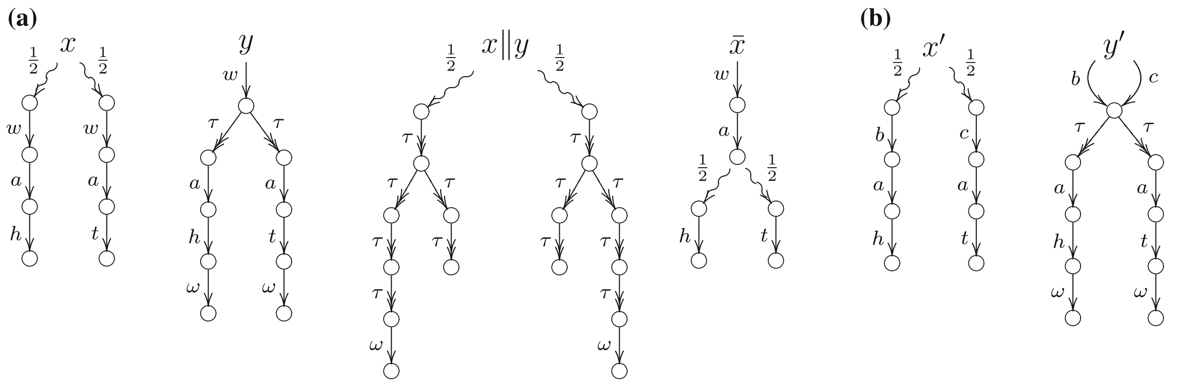


Fig. 2. The coin-flipper and result-guesser game: **a** fair play; **b** unfair play

Example 1.1 **(A gambling machine)** Consider a system consisting of a machine and a user. The machine is equipped with a menu of two buttons: *head* and *tail*, via which the user interacts with the machine. The machine, modeled by process graph s in Fig. 1, first makes a fair choice (i.e. flips a fair coin). This decides the *winning* button: either *head* or *tail*, each being the winning one in a half of the machine runs, in average. The user, modeled by process graph u in Fig. 1, can press any of the two buttons *head* or *tail*. Only if she presses the winning button, she wins a prize, and is happy (denoted by action ω). It is important to note that by no means the user is able to detect the machine's choice beforehand.

The interaction of the user and the machine is obtained by synchronizing the two processes on their common actions, and hiding the synchronizing actions afterwards. In terms of testing theory [DeH84], process s is tested with test u. Considered as a simple game of chance, by means of probability theory it can be calculated that the user wins a prize with probability $\frac{1}{2}$. However, most of the existing approaches for probabilistic testing, in particular probabilistic may/must testing [WaL92, Seg96, JoW02, PDM07, Den08, Den09], do not yield probability $\frac{1}{2}$ for action ω being reported. Namely, consider the result of synchronization of the two processes s and u, represented by process $s\|u$ in Fig. 1. By applying the above mentioned four schedulers to $s\|u$, one obtains the set of probabilities $\{0, \frac{1}{2}, 1\}$ with which action ω can be reported. Therefore, since the power of the schedulers is unrestricted, unrealistic bounds of 0 and 1 for the probability to pass the test are obtained. In fact, when resolving the internal choice in $s\|u$, the above mentioned may/must testing approaches allow schedulers that correspond to strategies the user can define and deploy only if she *knows* the result of the coin-flipping *before* guessing. More concretely, such schedulers ignore the fact that both internal choices in $s\|u$ are resolved in the same way, regardless of the outcome of the probabilistic choice.

Consider now process $\bar{s}$ in Fig. 1. Process $\bar{s}$ can as well represent the behaviour of the coin-flipping machine from the view point of the user: she will win in a half of the cases (which can be easily computed by standard probability theory, too). Hence, according to the user, this machine acts as specified too, regardless of the fact that now the machine flips the coin, not *before* but *after* a button is pressed. However, the probabilistic may/must testing approaches [WaL92, Seg96, JoW02, PDM07, Den08, Den09] do not consider processes s and $\bar{s}$ testing equivalent. Namely, two schedulers can be defined and applied to process $\bar{s}\|u$, both yielding exactly probability of $\frac{1}{2}$ for reporting action ω. Consequently, testing s and $\bar{s}$ with the same test u returns different sets of probabilities, so they are not probabilistically testing equivalent. Note, however, that if the probabilities are ignored, and each probabilistic choice is treated as an internal choice, then processes s and $\bar{s}$ are testing equivalent in the standard non-probabilistic may/must testing theory [DeH84].

Example 1.2 **(Guessing game).** Consider the following game. Player x tosses a fair coin without revealing the outcome and *w*aits. Player y *w*rites down his guess about the outcome of the flipping without showing it to x. Then, both players *a*gree to reveal their outcomes, i.e. x to uncover the coin and y to show what he has written. Players x and y are modeled in Fig. 2a. Process $x\|y$ represents the result of the synchronization of processes x and y, where ω action obviously specifies a correct guess. Obviously, the probability that player y guesses correctly equals $\frac{1}{2}$. However, the schedulers applied to the synchronization of processes x and y give the set of probabilities $\{0, \frac{1}{2}, 1\}$ for reporting ω. This, therefore, suggests that player y has a strategy always to guess the result correctly. On the other hand, if player y plays the same game against process $\bar{x}$ (Fig. 2a), the almighty schedulers approaches yield exactly one probability of $\frac{1}{2}$ for action ω to happen. Hence, in probabilistic may/must

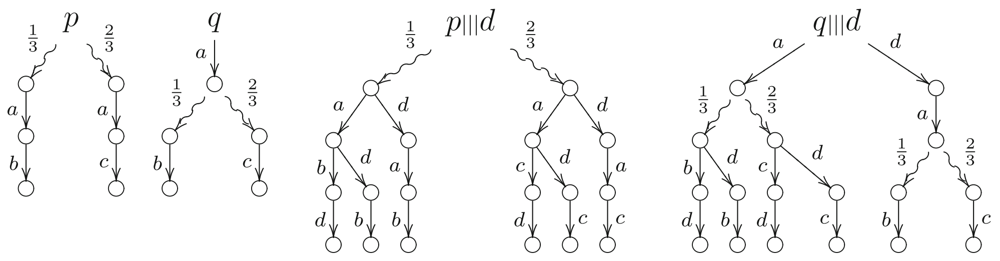


Fig. 3. Processes p and q, interleaved with action d

testing theory, processes x and $\bar{x}$ cannot be equated, although they differ only in the moment the coin is flipped. As a consequence, in this theory prefix does not distribute over probabilistic choice, thus making the moment of resolving an internal probabilistic choice *observable*. Please note that variants of this example were initially discussed in [Low93, Seg95, Mor96].

Note that specifying the *waiting* action of process x with the same action name, regardless of the outcome of the flipping, is essential for player x to keep the outcome unknown to player y. Namely, if player x after flipping the coin offers, depending on the outcome, different actions for synchronization, e.g. b and c, as shown in Fig. 2b, therewith he reveals the outcome of the flipping to the other player. In this case, the guesser can surely guess the result: he makes his guess depending on the action on which both players previously synchronized. In this case it is to be expected to have a scheduler which, for instance, yields probability 1 for successful guess.

It is worth noting that equating processes x and $\bar{x}$ from Fig. 2a, i.e. allowing *distribution of action prefix* over internal probabilistic choice [Hoa85], is closely related to equating processes s and $\bar{s}$ in Fig. 1, i.e. allowing *distribution of external action choice* over internal probabilistic choice [Hoa85], from a point of view of compositionality. Namely, if distribution of external choice over internal probabilistic choice is not allowed under a semantical equivalence, then allowing distribution of action prefix over internal probabilistic choice breaks the congruence property for *interleaving* (or merge) [Hoa85]. This is to be concluded from Fig. 3: relating processes p and q requires relating processes $p|||d$ and $q|||d$, where d denotes a process which can perform only action d and $|||$ is the merge operator.[1]

From the above discussion it follows that, if we are not able to relate processes that differ only in the moment an internal probabilistic choice is resolved, whether it is before or after an action execution, then, for the verification purposes we can only rely on equivalences that inspect the internal structure of processes, as bisimulations and simulations [Gla01]. Indeed, it has been shown in [JoW02, LSV07, Den08] that the testing preorders defined in [WaL92, Seg96] are branching time, simulation-like relations. The most discouraging fact is, however, that in a parallel composition probability information might be lost, as the previous examples show. This questions the reasons for adding probabilities in the model to start with.

In this paper we propose a testing semantics in the style of [DeH84] for processes exhibiting external, internal and probabilistic choices that solves the problem with overestimation of the probabilities with which a process passes a test. The proposed semantics also preserves the concept of unobservability of the internal choice as originally in [DeH84]. The unobservability of the internal choice results from a ready-trace-style characterization of the testing preorder relation, where a ready-trace is an alternating sequence of action menus and performed actions [Pnu85, BBK87, Gla01]. The testing semantics itself is based on a novel method for labeling the internal transitions. The labels on internal transitions indicate different appearance of the same internal choices as, for instance, the internal choices in the two black states in Fig 1. The core idea is that the labels on the internal transitions contain all information based on which the internal choice should be resolved. For example, both internal choices in process $s\|u$ in Fig. 1 are resolved based on the menu $\{h, t\}$ of action-candidates for synchronization, and thus their labels contain exactly this information, together with the synchronized action just executed. In this way, internal choices using the same information are resolved in the same way, regardless of the considered future. As the synchronization of a process and a test may create new internal nondeterminism, our labeling method introduces new labels and guarantees their consistency with labels that may already exist in the process or in the test. Our labeling approach, as we will discuss in more detail in Sect. 4, prevents over- and underestimation of probabilities. Thus, in our semantics processes s and $\bar{s}$ are equated nicely, as well as processes x and $\bar{x}$ (to be discussed in Sect. 4).

[1] We refrain from giving a formal definition of interleaving at the moment, but we emphasize that it is inherited from [Hoa85]. Note that probabilistic transitions, as immediate, have priority over action transitions, which can be delayed (see also [Han91, Seg95]).

We like to stress that the work presented here has paved the way towards more general theory and results, not presented in this paper, but can be found in [GeA10c, GeA10a, GeA12, Geo11]. One of the follow-up results is, the alternative ready-trace characterization of the probabilistic testing semantics defined here has been the basis of an algebraic probabilistic process theory with a rather rich underlying process language, which, among the others, includes a general parallel composition operator [GeA10a, GeA12]. This parallel composition allows processes to synchronize on a set of actions and to interleave on the rest of the actions, as in CSP [Ros98], and also allows action hiding after synchronization, as in CCS [Mil80]. The CSP-style axiomatization of the ready-trace equivalence shows that all the distributivity axioms for internal choice [Hoa85, Ros98] are preserved, and no new axioms regarding the interplay between external and internal choice are added. Thus, with this approach we solve a problem which was open throughout the years, namely to find an extension of CSP with probabilistic choice and proper underlying semantic which is satisfactory w.r.t. the derivation power of the axioms.

Structure of the paper In Sect. 2 we define our semantic model, and in Sect. 3 we define how the internal nondeterminism is resolved, by first performing unfolding and appropriate relabeling. In Sect. 4 we define the synchronization of a process and a test and the testing preorder relation. In Sect. 5 we define the ready-trace preorder relation, and in Sect. 6 we prove that the two preorder relations coincide. We end the paper with a discussion on the related work (Sect. 7) and with concluding remarks (Sect. 8).

2. Process graphs

In this section we define the underlying semantic model of process graphs and operations on them. Three types of choices can be modeled by the process graphs: choice between several different actions—external choice, choice between several internal transitions—internal choice, and probabilistic choice, as a refinement of the internal choice. Note that this kind of modeling is in the spirit of CSP [Hoa85]. However, the internal transitions have been assigned labels, local to the process to which they belong. The labels are meant to contain precisely the information that can be used to resolve the internal choice. In the process of resolving the nondeterminism and obtaining a fully probabilistic process, the labels are used as parameters of unknown probability distributions (Sects. 3 and 4). In this way the most generic notion of randomized schedulers are employed in our setting. It is worth noticing that our semantic model can be also seen as an orthogonal combination of reactive probabilistic processes [LaS91] (or Markov decision processes [Put94]) and parametric discrete-time Markov chains [Doo53], where the transition probabilities are parameters, without a time component.

Given a directed graph r, by $s \xrightarrow{l} t$ we denote that there exists an edge in r originating from a node s and ending in a node t, labeled with l; we may omit s, t, or l from the notation to denote that they are arbitrary. For a finite index set I, by $[\{s \xrightarrow{l_i} s_i\}_{i \in I}]$ we denote that there exist edges $\{s \xrightarrow{l_i} s_i\}_{i \in I}$ and s has no other outgoing edges. We assume a finite set of actions $\mathcal{A}$ and a countable set of *internal labels* $\mathcal{L}$ such that $\mathcal{A} \cap \mathcal{L} = \emptyset$. We let $a, b, c, \ldots$ range over $\mathcal{A}$, τ_i range over $\mathcal{L}$ and π_i range over the interval [0, 1].

Definition 2.1 (**Process graph**). A *process graph* r, or simply *process* r, is a directed, finite-state and finite-edge graph with root r, such that

– there exist three types of edges, or *transitions*: *action* ($\rightarrow$), *internal* ($\twoheadrightarrow$), and *probabilistic* ($\rightsquigarrow$);
– there exist three types of nodes, or *states*: *action*, *nondeterministic*, and *probabilistic*; from an action (resp. nondeterministic, probabilistic) state there can originate only action (resp. internal, probabilistic) transitions;
– the action transitions are labeled with actions from $\mathcal{A}$ such that no two action transitions with the same state of origin are labeled the same;

– the internal transitions are labeled with labels from $\mathcal{L}$ such that
  - no two internal transitions with the same state of origin are labeled the same, and
  - if $s \overset{\tau_1}{\twoheadrightarrow}$ and $t \overset{\tau_1}{\twoheadrightarrow}$, then $[\{s \overset{\tau_i}{\twoheadrightarrow} s_i\}_{i\in I}]$ iff $[\{t \overset{\tau_i}{\twoheadrightarrow} t_i\}_{i\in I}]$, i.e. if two states share a label on their outgoing internal transitions, then they have the same sets of labels on all of their outgoing transitions;

– the probabilistic transitions are labeled with scalars from (0, 1], such that
  - given two states, there is at most one probabilistic transition connecting them, and
  - for each probabilistic state s, if $[\{s \overset{\pi_i}{\rightsquigarrow} s_i\}_{i\in I}]$ then $\sum_{i\in I} \pi_i = 1$, i.e. the sum of the labels on the outgoing transitions equals one;

– all states are reachable from r.

The set of all process graphs is denoted by $\mathcal{G}$. A state without outgoing transitions, i.e. a *deadlock state*, is considered an action state. The deadlock process is denoted by 0. Given an action state s, by s_a we denote the state (if it exists) for which $s \xrightarrow{a} s_a$; by $\mathcal{I}(s)$ we denote the set of actions $\{a_i\}_{i\in I}$ such that $[\{s \xrightarrow{a_i} s_i\}_{i\in I}]$, and if s is a deadlock state, it equals $\emptyset$. $\mathcal{I}(s)$ is called the *menu* of s. Intuitively, $\mathcal{I}(s)$ is the set of actions that process s can perform initially.

Example 2.1 The graphs s, u, and $\bar{s}$ in Fig. 1, and the graphs z, v, and r in Fig. 4a are process graphs.

Each process s defines a set of equations, called *constraints* for s. Similarly as in parametric Markov chains [Doo53], they restrict the values that the internal labels in s can take, such that the values of the labels assigned to an internal choice form a probability distribution, i.e. they are in the interval [0, 1] and sum up to 1. The constraints, formally defined below, play an essential role in our approach.

Definition 2.2 (Constraints). Let s be a process. The set of *constraints* for s, denoted $\mathcal{C}(s)$, is a set of linear equations over labels in $\mathcal{L}$, such that an equation is in $\mathcal{C}(s)$ if and only if it has the form $\sum_{i\in I} \tau_i = 1$ with $[\{t \overset{\tau_i}{\twoheadrightarrow} t_i\}_{i\in I}]$ for some nondeterministic state t in s. $\mathcal{L}$ labels are called *variables* of $\mathcal{C}(s)$. A *resolution* of $\mathcal{C}(s)$ is a function assigning values from [0, 1] to the variables in $\mathcal{C}(s)$, respecting the constraints $\mathcal{C}(s)$.

Process graphs are composed by means of the following parametric operators: $\sqcap$, $\Box$, and $\oplus$. The operators $\sqcap$ and $\Box$ have the same intuitive meaning as the corresponding operators in [Hoa85]: $\sqcap_{i\in I}\tau_i s_i$ is (in our case labeled) internal nondeterministic choice between processes $\{s_i\}_{i\in I}$, and $\Box_{i\in I}\ a_i s_i$ is a choice between action prefixed processes $\{a_i s_i\}_{i\in I}$. $\oplus_{i\in I}\pi_i s_i$ is a probabilistic choice between processes $\{s_i\}_{i\in I}$, specifying that each process s_i is being chosen with probability π_i. Formally,

Definition 2.3 Let I be a finite index set and $\{s_i\}_{i\in I}$ be a set of process graphs. The parametric operators $\oplus$, $\sqcap$, $\Box$ on process graphs are defined as follows:

– For $\{\tau_i\}_{i\in I} \subset \mathcal{L}$ such that $\{\tau_i\}_{i\in I}$ do not appear in the process graphs $\{s_i\}_{i\in I}$, the process graph $\sqcap_{i\in I}\tau_i s_i$ is constructed by creating a new state s as its root, a set $\{s'_i\}_{i\in I}$ of disjoint copies of the process graphs $\{s_i\}_{i\in I}$, and new transitions $\{s \overset{\tau_i}{\twoheadrightarrow} s'_i\}_{i\in I}$.
– For $\{a_i\}_{i\in I} \subseteq \mathcal{A}$, the process graph $\Box_{i\in I}\ a_i s_i$ is constructed by creating a new state s as its root, a set $\{s'_i\}_{i\in I}$ of disjoint copies of the process graphs $\{s_i\}_{i\in I}$, and new transitions $\{s \xrightarrow{a_i} s'_i\}_{i\in I}$.
– For $\{\pi_i\}_{i\in I} \subset (0, 1]$ such that $\sum_{i\in I} \pi_i = 1$, the process graph $\oplus_{i\in I}\pi_i s_i$ is constructed by creating a new state s as its root, a set $\{s'_i\}_{i\in I}$ of disjoint copies of the process graphs $\{s_i\}_{i\in I}$, and new transitions $\{s \overset{\pi_i}{\rightsquigarrow} s'_i\}_{i\in I}$.

We omit the operator sign $\oplus$, $\Box$ or $\sqcap$ if there is only one operand. We also write a rather than $a0$.

Note that our model does not allow choice between an internal transition and an action transition, since this would raise ambiguity in the process of labeling the internal transitions. However, in practice this is not a limitation, since by the testing semantics [DeH84], process $\tau.P + Q$, where + denotes alternative composition, is equivalent to process $\tau.P + \tau.(P + Q)$. Also, processes $a.P + a.Q$ and $a.(\tau.P + \tau.Q)$ are testing equivalent, which is why we only consider choice between different actions. Note that, for clarity, and for being able to define a probabilistic counterpart of CSP later on, we use separate operators for choice between actions and for choice between internal transitions.

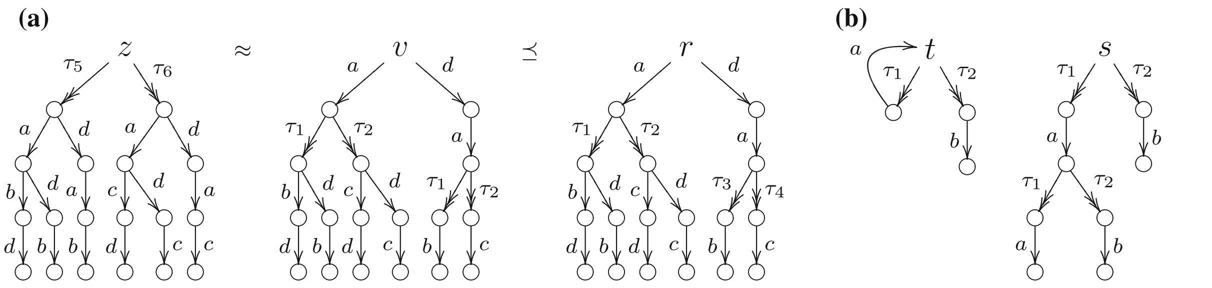


Fig. 4. **a** Processes and relations between them: processes z and v are obtained by interleaving $\tau_5(ab) \sqcap \tau_6(ac)$ and $a(\tau_1 b \sqcap \tau_2 c)$, respectively, with action d. **b** A cyclic process and a finite process

Definition 2.4 A *finite* process graph is a process graph in which only finite paths exist. A *finite process tree* is a finite process graph that has a tree form. A *divergence-free* process graph is a process graph without infinite sequences of probabilistic or internal transitions.

To simplify the technical framework, throughout the paper we assume that processes are divergence-free. Generalization is discussed as future work in Sect. 8.

3. Unfolding and coherent labeling

The idea behind labeling the internal transitions is to be able to identify different appearances of the same internal choice. Please recall from the examples that an internal nondeterministic choice can be cloned when the process it belongs to is synchronized or interleaved. Then, if two internal choices are identified with the same labels, then they are two copies of the same instance, and we ensure that they will be resolved in the same way. For example, as already discussed, the two internal choices in the synchronization $x\|y$ actually represent the same instance of a choice and, thus in our model, they are labeled the same (we defer the discussion for the exact labeling to the next section). A rather more illustrative example is process v in Fig. 4a. It can be seen as the interleaving of process $p \equiv a(\tau_1 b \sqcap \tau_2 c)$ and process $q \equiv d$. Since there is no communication specified between p and q, the internal choice in p cannot and should not be influenced by execution of action d, obviously. Consequently, both internal choices that appear in v must be resolved in the same manner, and this is ensured by the same label sets.

Consider now process s in Fig. 4b as a process which is to be tested. As given, it has two internal choices both having the same labels. This indicates that they both should be resolved in the same way. Nevertheless, it is rather more realistic to reason that the same instance of an internal choice cannot happen again in the future of itself. Consequently, we have to relabel the second internal choice with new labels. Typically, process graphs like s in Fig. 4b result when a cyclic process graph is unfolded: s can be seen as an unfolding of graph t in the same figure. Clearly, in one run, all the internal choices in process t are resolved pairwise independently, due to their nondeterministic nature. Finally, let us recall the example of process y in Fig.2, which has been utilized as a test for process x. It has an internal choice, which has to be resolved as well.

One can, therefore, conclude that computing the set of probabilities by which a process passes a given test requires a proper labeling and, if necessary, relabeling. This is done in three sequential steps. First, (re-)labeling of the internal nondeterminism of a process which is to be tested (e.g. processes in Fig. 4b.) discussed in a moment. Second, labeling of the internal nondeterminism of a test, and, third, labeling of the internal nondeterminism newly created by the synchronization when testing. The last two labeling steps are integrated into one step and are discussed in Sect. 4.1. Once the labeling is completed, the result of the testing in terms of concrete probability values can be computed (Sect. 4.2).

In this section we define how to unfold a process graph up to a finite length and relabel its internal transitions, such that two nondeterministic states have the same label sets only if they represent the same instance of an internal choice. Then, we define, in a rather straightforward way, how the internal nondeterminism in a process is resolved, by assigning probability distributions to the internal choices.

A recursive function that *unfolds* a process up to a certain depth and *relabels* the internal transitions appropriately on-the-fly is defined below. It takes three arguments: a process graph s to be applied to, a set of labels L in $\mathcal{L}$ and a nonnegative integer m. The set of labels L collects the labels used in the unfolding up to the present moment and serves to ease the definition, while the integer m is the *unfolding depth*, i.e. the maximal number of observable actions that appear in a path of the unfolded process graph. The function returns a process graph in the form of a finite tree.

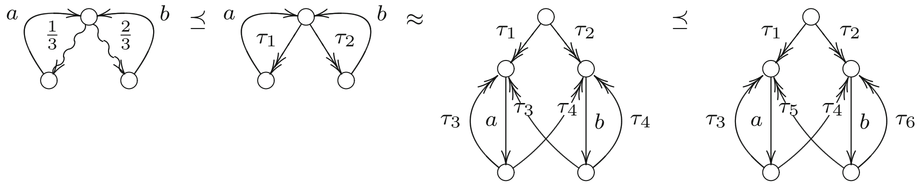


Fig. 5. Cyclic process graphs and relations between them

Definition 3.1 (Unfolding). The partial function $\mathrm{U}\colon \mathcal{G} \times 2^{\mathcal{L}} \times \mathbb{N} \mapsto \mathcal{G}$, called *unfolding*, is defined as

$$\mathrm{U}(s, L, m) \equiv \begin{cases} 0, & \text{if } m = 0 \text{ or } s \equiv 0 \\ \oplus_{i\in I}\pi_i \mathrm{U}(s_i, L, m), & \text{if } m > 0 \text{ and } [\{s \overset{\pi_i}{\rightsquigarrow} s_i\}_{i\in I}] \\ \Box_{i\in I}\ a_i \mathrm{U}(s_i, L, m-1), & \text{if } m > 0 \text{ and } [\{s \xrightarrow{a_i} s_i\}_{i\in I}] \\ \sqcap_{i\in I}\tau_i^{n+1}\mathrm{U}\left(s_i, L \cup \{\tau_i^{n+1}\}_{i\in I}, m\right), & \text{if } m > 0 \text{ and } [\{s \overset{\tau_i}{\twoheadrightarrow} s_i\}_{i\in I}] \text{ and} \\ & n = \max\{k \mid \tau_i^k \in L, i\in I\} \end{cases}$$

where each label τ_i^{n+1} is *fresh*, i.e. it hasn't been used for labeling internal transitions yet,[2] and for a given set of non-negative numbers I, by $\max I$ we denote the maximum of the numbers in I, if $I \neq \emptyset$, or 0, if $I = \emptyset$.

By unfolding a process graph, a finite process tree is obtained. The labels on the internal transitions in the obtained tree are new with respect to the labels in the original process graph. Every time a state, having outgoing internal transitions labeled with the set $\{\tau_i\}_{i\in I}$, is to be unfolded, a fresh set of labels $\{\tau_i^{n+1}\}_{i\in I}$ is used for labeling the corresponding internal transitions in the unfolded tree. This set of labels, in fact, contains implicit information, stored in n, about the number of times a state in the original graph with the label set $\{\tau_i\}_{i\in I}$ has been unfolded up to that moment. This way, internal choices that were labeled the same in the original graph, but are in the same future, are relabeled with different label sets; on the other hand, internal choices that were labeled the same in the original graph, but are "parallel" to each other, remain labeled with equal label sets. This is because, as we already pointed out, in the former case the choices are different instances of the internal choice, while in the later case, the choices are the same instance of the internal choice, happening in different futures.

Example 3.1 The result of function $\mathrm{U}(v, \emptyset, 3)$, for v in Fig. 4a, is process $a(\tau_1^0(bd \,\Box\, db) \sqcap \tau_2^0(cd \,\Box\, dc)) \,\Box\, da(\tau_1^0 b \sqcap \tau_2^0 c)$. Thus, when v is unfolded, both internal choices remain labeled with the same sets of labels.

Example 3.2 The result of function $\mathrm{U}(t, \emptyset, 2)$, for t in Fig. 4b, is process $\tau_1^0(a(\tau_1^1 a \sqcap \tau_2^1 b)) \sqcap \tau_2^0 b$. Thus, when t is unfolded up to depth 2, the two internal choices are relabeled with different sets of labels. The same holds for the unfolding of process s in Fig. 4b.

The following example reveals that assigning labels to cyclic processes requires delicate reasoning.

Example 3.3 Consider the second left-most process in Fig. 5. By unfolding it up to depth 2, we obtain the process $\tau_1^0 a(\tau_1^1 a \sqcap \tau_2^1 b) \sqcap \tau_2^0 b(\tau_1^1 a \sqcap \tau_2^1 b)$. Consider now the right-most process in the same figure. By unfolding it up to depth 2, we obtain the process $\tau_1^0 a(\tau_3^0 a \sqcap \tau_4^0 b) \sqcap \tau_2^0 b(\tau_5^0 a \sqcap \tau_6^0 b)$.

Note that, in order to express full nondeterminism in general cyclic processes, namely to express that all the internal choices are resolved independently from each other, one has to switch to infinite-state graphs. Recall, however, that our present focus is on defining a testing semantics that gives reasonable probabilities for a process to pass a test. Therefore the present focus is on finite behaviour, where this expressivity (of infinite behaviour) does not play a role.

[2] The index n in τ_i^n is not to be confused with the nth power of τ_i; we would denote the latter with $(\tau_i)^n$.

Having labeled the internal choices in an unfolded process tree in a coherent way, the internal nondeterminism can be resolved by assigning probability distributions to the internal choices. Recall from Definition 2.2 that $\mathcal{C}(s)$ denotes the set of constraints for the labels used in process s, which now will impose constraints on distributions that can be assigned to particular internal choice.

Definition 3.2 (**Resolution of a process).** Let s be a process graph, $m \geq 0$, and $\bar{s} = \mathrm{U}(s, \emptyset, m)$. An $m - resolution$ of s is the process graph s^m obtained when, for an arbitrary resolution λ of $\mathcal{C}(\bar{s})$, every transition $t \overset{\tau_i}{\twoheadrightarrow} t'$ in graph $\bar{s}$ is replaced by $t \overset{\lambda(\tau_i)}{\rightsquigarrow} t'$, if $\lambda(\tau_i) \neq 0$, or erased, otherwise.

Thus, an m-resolution of a process s is the process that results when the labels on the internal transitions in the unfolding of s up to depth m have been replaced by probabilities. Note that, speaking in terms of schedulers, we employ randomized schedulers [Seg95, BiA95, Put94], which assign arbitrary probability distributions to the internal transitions, rather than the deterministic schedulers considered in the introduction, that assign only the trivial probability distributions. This is because the former schedulers are more general, and with them, probabilistic choice can be treated as a special type of internal choice.

Remark The unfolding function of Definition 3.1 takes into consideration that a cyclic process may be a parallel composition of simpler processes; with this reasoning, as we saw, infinite-state graphs are necessary in order to model infinite processes where all internal choices are resolved pairwise independently. Another way of defining the unfolding function is to assign different values to all the internal labels of the process. This way, finite state graphs suffice. Clearly, a compromise has to be made in this case, too: we have to assume that the process-argument of the unfolding function is not obtained as a parallel composition. We leave the details of this solution for the future.

4. Testing semantics

In this section we define the testing semantics, and, based on it, a testing preorder relation on processes. To prepare the ground, we define first a systematic labeling of the internal transitions that result from testing a process with a test. Then, simply by treating the internal choices in the synchronization as parametric probabilistic choices, we define a function that calculates the parametric probability with which a process passes a test. Given a value-assignment of the parametric probabilities, we obtain a concrete probability with which a process can pass a test. Based on the parametric probabilities, we define the testing preorder relation on processes.

4.1. Synchronization

A *test* T, as usual, is a finite process tree, such that for a symbol $\omega \notin \mathcal{A}$, there may exist transitions $s \xrightarrow{\omega}$ for some states s in T, denoting success.[3] We assume that all the labels on internal transitions in a test belong to the set $\{\tau_i\}_{i \in I}$ for some index set I. Additionally, we assume that all labels on the internal transitions in a test are distinct (we justify this assumption below).

In the previous section we defined how the internal nondeterminism in a process is labeled in a coherent way and resolved. Recall that submitting a process to a test means synchronizing both of them on all common actions. Therefore, additional nondeterminism arises when a process is being tested. First, there is the nondeterminism with respect to the action on which the process and the test synchronize, if there are multiple candidate-actions for synchronization at one moment, i.e. *synchronization nondeterminism*. Second, there is the internal nondeterminism of the test. In order to determine how the synchronized transitions should be labeled, let us recall Example 1.1 and the synchronization $s \| u$ of the coin-flipping machine s and user u. Clearly, in order to avoid the problem with overestimation of probabilities, we must ensure that both internal choices in $s \| u$ are resolved in the same way. Thus, the resolution of the synchronization nondeterminism should not depend on the probabilistic (or, for that matter, the internal) transitions of the process or the test. Moreover, note that the state in which the process or

[3] It has been shown that infinite tests do not increase the distinguishing power [Seg96, KCS98], because infinite paths cannot report success.

the test, or for that matter their synchronization, resides at the moment, should not play a role in the resolution of synchronization nondeterminism. In fact, in order to keep the probability with which the user guesses the outcome of coin-flipping to $\frac{1}{2}$, the resolution of synchronization nondeterminism should be based at most on the menu that the machine offers, or, in other words, on the actions that are candidates for synchronization. As the process and the test proceed interacting, the entity that resolves synchronization nondeterminism (in our example the user) may actually remember the history of synchronization and take it into account when making the current choice. To conclude, speaking in terms of labels, each label on a synchronized transition should represent exactly the set of actions-candidates for synchronization at that moment, the actual action on which the current synchronization happens, and the history of synchronization.

Regarding the test, recall that we assume that all the labels on internal transitions that it contains are distinct. In other words, for simplicity, we restrict to the subset of tests that are most powerful, as they have "full control" over their internal choices. In fact, if two processes are not distinguished by this subset of tests, then they should not be distinguished by the rest of the (less powerful) tests, too. (In Sect. 6 we actually prove that tests without internal transitions suffice for comparison of processes). Now, to avoid underestimation of the probabilities with which success is reported, we capture the special case when the test itself resolves the synchronization nondeterminism (as in Example 1.1). Therefore, we bear in mind that the test can also take into account the history of resolving the synchronization nondeterminism when resolving its internal nondeterminism. For example, consider the process $\frac{1}{2}(ad \,\Box\, b) \oplus \frac{1}{2}(ac)$ and the test $a(\tau_1 d\omega \sqcap \tau_2 c\omega) \,\Box\, b\omega$. When testing the process, the test can choose transition τ_1 if action a was synchronized out of the candidate-actions a and b, and the transition τ_2 if a was the only candidate for synchronization. In this way, the test can resolve its internal choices such that success is always reported. Thus, when an internal transition originating from the test is labeled in the synchronization of the process and the test, the label includes the history of synchronization.

We now formalize the above discussions, in the form of a synchronization operator for a process and a test. Let $\mathcal{G}_{\not\twoheadrightarrow} \subset \mathcal{G}$ be the set of all process graphs that do not contain internal transitions, i.e. the set of *deterministic processes*. In the following definition we assume that the process is deterministic, and the remark below explains the sufficiency of this for the general case. We presuppose a special label ε that cannot appear in any process or test. The synchronous parallel composition $\|_l$ is parametrized by a label $l \in \mathcal{L}$, equal to the last label created for an internal transition resulting from synchronization nondeterminism, up to the present moment. Thus, in order to pass the history of synchronization to a newly created label l', it is enough to incorporate l in l' as a superscript. Recall that, given an action state s, by s_a we denote the state (if it exists) for which $s \xrightarrow{a} s_a$.

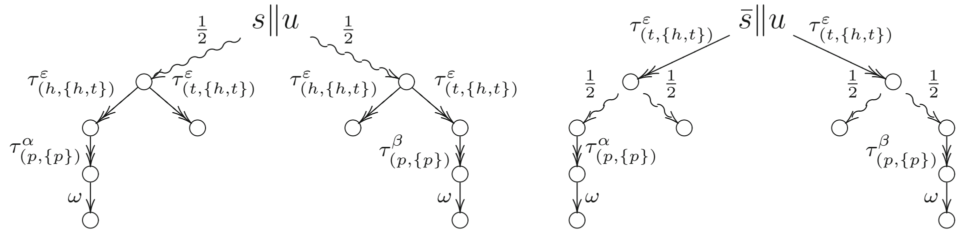
Definition 4.1 (Synchronization). The synchronization $s\|_l T$ of a deterministic process s and a test T is defined as

$$s\|_l T = \begin{cases} \omega, & \text{if } T \xrightarrow{\omega} \\ \bigoplus_{i\in I}\pi_i(s_i\|_l T), & \text{if } [\{s \overset{\pi_i}{\rightsquigarrow} s_i\}_{i\in I}] \text{ and } T \overset{\omega}{\not\rightarrow} \\ \bigoplus_{i\in I}\pi_i\,(s\|_l T_i)\,, & \text{if } [\{T \overset{\pi_i}{\rightsquigarrow} T_i\}_{i\in I}] \text{ and } s \not\rightsquigarrow \\ \sqcap_{i\in I}\tau_i^l\,(s\|_l T_i), & \text{if } [\{T \overset{\tau_i}{\twoheadrightarrow} T_i\}_{i\in I}] \text{ and } s \not\rightsquigarrow \\ \sqcap_{a\in K}\tau^l_{(a,K)}\left(s_a\|_{\tau^l_{(a,K)}} T_a\right), & \text{for } K = \mathcal{I}(s)\cap\mathcal{I}(T) \neq \emptyset, \text{ and } T \overset{\omega}{\not\rightarrow} \\ 0, & \text{otherwise.} \end{cases}$$

If $l = \varepsilon$, then we write $s\|T$ rather than $s\|_\varepsilon\, T$.

Example 4.1 Figure 6 gives the synchronization $s\|u$ of process s and test u, and the synchronization $\bar{s}\|u$ of process $\bar{s}$ and test u from Fig. 1. Note how both internal choices in $s\|u$, that result from the options to synchronize on h or on t, are labeled with the same set of labels. Moreover, the same set of labels is also used for labeling the corresponding internal choice in $\bar{s}\|u$.

Example 4.2 Figure 7 gives the synchronization $x\|y$ of process x and test y and the synchronization $\bar{x}\|y$ of process $\bar{x}$ and test y from Fig. 2a, where test y in our model has been modeled as $w(\tau_1 rh\omega \sqcap \tau_2 rt\omega)$. Note how both internal choices in $x\|y$ are labeled with the same sets of labels, and the same set of labels is used for the internal choice in $\bar{x}\|y$. In Fig. 8 the synchronization $x'\|y'$ of process x' and test y' from Fig. 2b is given, where test y' in our model has been modeled as $a((\tau_1 rh\omega) \sqcap (\tau_2 rt\omega)) \,\Box\, b((\tau_3 rh\omega) \sqcap (\tau_4 rt\omega))$ (thus satisfying our requirements that the test is a process tree with all the labels distinct).



$s\|u$
$\frac{1}{2}$
$\frac{1}{2}$
$\tau^{\varepsilon}_{(h,\{h,t\})}$
$\tau^{\varepsilon}_{(t,\{h,t\})}$
$\tau^{\alpha}_{(p,\{p\})}$
$\tau^{\beta}_{(p,\{p\})}$
ω
$\bar{s}\|u$

Fig. 6. The synchronization of machines s and $\bar{s}$ with user u from Fig. 1. $\alpha = \tau^{\varepsilon}_{(h,\{h,t\})}$, $\beta = \tau^{\varepsilon}_{(t,\{h,t\})}$

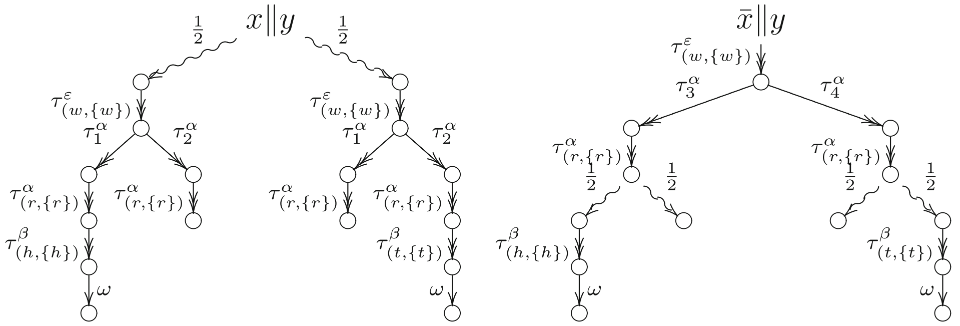
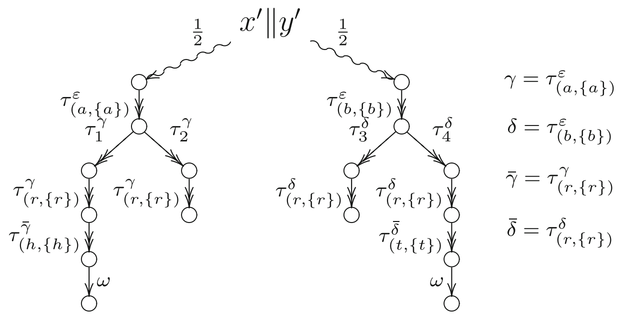


Fig. 7. Synchronized players x and y, and players $\bar{x}$ and y from Fig. 2a. $\alpha = \tau^{\varepsilon}_{(w,\{w\})}$, $\beta = \tau^{\alpha}_{(r,\{r\})}$

Remark Note that the operators defined in Definitions 3.1 and 4.1 can be merged into one synchronization operator that synchronizes any arbitrary process (also if it has internal nondeterminism) with a test. The part regarding the resolution of the process internal nondeterminism would be inherited from Definition 3.1. We separated the definitions for clarity and because Definition 3.1 is also needed in Sect. 5.

4.2. The result of testing

Having labeled all the internal transitions in the synchronization of a deterministic process and a test, we can define the parametric probability with which a deterministic process s passes a test T, i.e. with which action ω is reported in $s\|T$. Namely, by treating every label set as a parametric probability distribution, the probability to report ω in the synchronization is a polynomial over the variables in $\mathcal{L}$. For our convenience, we bypass the creation of the process graph representing the synchronization, and we calculate the resulting polynomial directly.



$x'\|y'$
$\frac{1}{2}$
$\frac{1}{2}$
$\tau^{\varepsilon}_{(a,\{a\})}$
τ^{γ}_{1}
τ^{γ}_{2}
$\tau^{\gamma}_{(r,\{r\})}$
$\tau^{\bar{\gamma}}_{(h,\{h\})}$
$\tau^{\varepsilon}_{(b,\{b\})}$
τ^{δ}_{3}
τ^{δ}_{4}
$\tau^{\delta}_{(r,\{r\})}$
$\tau^{\bar{\delta}}_{(t,\{t\})}$
ω
$\gamma = \tau^{\varepsilon}_{(a,\{a\})}$
$\delta = \tau^{\varepsilon}_{(b,\{b\})}$
$\bar{\gamma} = \tau^{\gamma}_{(r,\{r\})}$
$\bar{\delta} = \tau^{\delta}_{(r,\{r\})}$

Fig. 8. Synchronized players x' and y' from Fig. 2b

Thus, the computation of the parametric probability with which $s\|T$ performs action ω mimics the creation of $s\|T$ itself. Let $\mathcal{T}$ be the set of all tests and $\mathbb{P}$ be the set of all polynomials with variable names in the label set $\mathcal{L}$.

Definition 4.2 **(Result of testing)** The partial function $\mathcal{R}\colon \mathcal{G}_{\not\twoheadrightarrow} \times \mathcal{T} \times \mathcal{L} \mapsto \mathbb{P}$ is defined as

$$\mathcal{R}(s, T, l) = \begin{cases} 1, & \text{if } T \xrightarrow{\omega} \\ \sum_{i\in I} \pi_i \cdot \mathcal{R}(s_i, T, l), & \text{if } [\{s \overset{\pi_i}{\rightsquigarrow} s_i\}_{i\in I}] \text{ and } T \overset{\omega}{\not\rightarrow} \\ \sum_{i\in I} \pi_i \cdot \mathcal{R}(s, T_i, l), & \text{if } [\{T \overset{\pi_i}{\rightsquigarrow} T_i\}_{i\in I}] \text{ and } s \not\rightsquigarrow \\ \sum_{i\in I} \tau_i^l \cdot \mathcal{R}(s, T_i, l), & \text{if } [\{T \overset{\tau_i}{\twoheadrightarrow} T_i\}_{i\in I}] \text{ and } s \not\rightsquigarrow \\ \sum_{a\in K} \tau^l_{(a,K)} \mathcal{R}\left(s_a, T_a, \tau^l_{(a,K)}\right) & \text{for } K = \mathcal{I}(s) \cap \mathcal{I}(T) \neq \emptyset \text{ and } T \overset{\omega}{\not\rightarrow} \\ 0, & \text{otherwise.} \end{cases}$$

If $l = \varepsilon$, then we shall write $\mathcal{R}(s, T)$ rather than $\mathcal{R}(s, T, \varepsilon)$, and we call $\mathcal{R}(s, T)$ the *result* of testing s with T. Given a resolution λ of $\mathcal{C}(s\|T)$ for a deterministic process s and a test T, the value of the polynomial $\mathcal{R}(s, T)$ for the values of its variables given by λ is denoted by $\Pr(s, T, \lambda)$. Intuitively, $\Pr(s, T, \lambda)$ is the probability with which the deterministic process s passes test T, given that the synchronization nondeterminism and the internal nondeterminism of T are resolved by λ.

The following proposition states that if two processes yield the same result when tested by a certain test, then they also yield the same constraints when synchronized with the test. In this case, for every resolution of the constraints, the two processes pass the test with the same probability.

Proposition 4.1 *Let s and t be two deterministic processes and T be a test, such that $\mathcal{R}(s, T) = \mathcal{R}(t, T)$. Then, $\mathcal{C}(s\|T) = \mathcal{C}(t\|T)$, and given an arbitrary resolution λ of $\mathcal{C}(s\|T)$,* $\Pr(s, T, \lambda) = \Pr(t, T, \lambda)$.

Proof. From $\mathcal{R}(s, T) = \mathcal{R}(t, T)$, we have that $s\|T$ and $t\|T$ use the same set of labels L for their internal transitions. Moreover, by Definition 4.2, L can be partitioned into sets $\{L_i\}_{i\in J}$, such that for every internal choice in $s\|T$ there is a set L_j containing the labels assigned to that internal choice, and the same for the internal choices in $t\|T$. Thus, by Definition 2.2, we obtain that $\mathcal{C}(s\|T) = \bigcup_{i\in J} \left\{\sum_{l\in L_i} l = 1\right\} = \mathcal{C}(t\|T)$. The proof of the second part is trivial. □

Example 4.3 Consider processes s and $\bar{s}$, test u in Fig. 1, and the synchronizations $s\|u$ and $\bar{s}\|\bar{u}$ in Fig. 6. It is easy to derive that $\mathcal{R}(s, u) = \mathcal{R}(\bar{s}, u)$, and therefore, by Proposition 4.1, $\mathcal{C}(s\|u) = \mathcal{C}(\bar{s}\|u)$. Moreover, for every resolution λ of $\mathcal{C}(s\|u)$, we have $\Pr(s, u, \lambda) = \Pr(\bar{s}, u, \lambda) = \frac{1}{2}$. That is, no matter how the internal nondeterminism in the synchronization of process s and test u is resolved, the probability with which s passes u is $\frac{1}{2}$. The same holds for process $\bar{s}$ and test u. In other words, the user guesses the outcome of the coin-flipping in either machine with probability $\frac{1}{2}$.

Example 4.4 Consider processes x and $\bar{x}$, test y in Fig. 2a and the synchronizations $x\|y$ and $\bar{x}\|y$ in Fig. 7. We have $\mathcal{R}(x, y) = \mathcal{R}(\bar{x}, y)$ and therefore, by Proposition 4.1, $\mathcal{C}(x\|y) = \mathcal{C}(\bar{x}\|y)$. Moreover, for every resolution λ of $\mathcal{C}(x\|y)$, we have $\Pr(x, y, \lambda) = \Pr(\bar{x}, y, \lambda) = \frac{1}{2}$. That is, no matter how y resolves its internal nondeterminism when synchronized with x, the probability with which success is reported is $\frac{1}{2}$. In other words, no matter how player y makes his guess, it will coincide with the outcome of coin-flipping with probability $\frac{1}{2}$. The same holds for process $\bar{x}$ and test y. On the other hand, consider processes x' and y' in Fig. 2b, and their synchronization $x'\|y'$ in Fig. 8. There exists a resolution λ of $\mathcal{C}(x'\|y')$, assigning value 0 to τ_2^γ and τ_3^δ and value 1 to all the other labels, such that $\Pr(x', y', \lambda) = 1$. In other words, player y' can always rightly guess the outcome of the coin-flipping.

4.3. Testing preorder

In the classical sense of probabilistic testing [WaL92, Seg96, JoW02, PDM07, Den08, Den09], in order for process s to implement process t, it is required that for every test T and every resolution of the nondeterminism in $s\|T$, there exists a resolution of the nondeterminism in $t\|T$ that can mimic the probability to report success in a certain way. In our setting, for the reasons explained in the introduction, we move from the "compose-and-schedule" approach to "schedule-and-compose" (see also [Che06] for this term). Namely, in order for process s to implement process t, we require that for every test T, no matter how s resolves its internal nondeterminism, t can resolve its internal nondeterminism such that T cannot distinguish between s and t.

So far, we can anticipate that a good ground for comparison of two deterministic processes s and t when tested with test T is the function $\mathcal{R}(x, T)$. Namely, assume that $\mathcal{R}(s, T) = \mathcal{R}(t, T)$. Then, by Proposition 4.1, $\mathcal{C}(s \| T) = \mathcal{C}(t \| T)$, i.e. the same internal label sets appear in both $s \| T$ and $t \| T$. This means that the same options for resolving the synchronization nondeterminism and the internal nondeterminism arise when either s or t is tested by T, that is, the test cannot distinguish between s and t based on the actions that s or t offer or have offered for synchronization. By Proposition 4.1, from $\mathcal{R}(s, T) = \mathcal{R}(t, T)$ we also have that for an arbitrary resolution λ of $\mathcal{C}(s \| T)$ (and therefore of $\mathcal{C}(t \| T)$) it holds that $\Pr(s, T, \lambda) = \Pr(t, T, \lambda)$. That is, for every resolution of the synchronization nondeterminism, the probabilities to report success coincide for both s and t. In other words, even if the test repeatedly tests s and t, it cannot make a difference between s and t based on the observed frequency with which success is reported for a particular resolution of the synchronization nondeterminism and the internal nondeterminism in the test. Finally, note that the polynomial $\mathcal{R}(x, T)$ is "insensitive" to the exact moments in time at which x makes its probabilistic choices, i.e. $\mathcal{R}(s, T)$ does not reveal to T the internal structure of x. To summarize, the polynomial $\mathcal{R}(x, T)$ contains the exact information that test T can exploit in order to make a difference between two processes.

The above discussion is formalized in the following definitions of the testing preorder relation and the induced equivalence. Given a test T, by length(T) we denote the maximal number of observable actions T can perform before performing the success action ω. Recall from Definition 3.2, an m-resolution of a process is obtained when it has been unfolded up to length m and the internal nondeterminism has been resolved.

Definition 4.3 (**Testing preorder**). Let s and t be two processes. s *implements* t, denoted $s \preceq_{\mathrm{T}} t$, iff for every natural $m \geq 0$, for every test T with length(T) $= m$ and for every m-resolution s^m of s, there exists an m-resolution t^m of t such that $\mathcal{R}(s^m, T) = \mathcal{R}(t^m, T)$.

Definition 4.4 (**Testing equivalence**). Processes s and t are *testing-equivalent*, denoted $s \approx_{\mathrm{T}} t$, iff $s \preceq_{\mathrm{T}} t$ and $t \preceq_{\mathrm{T}} s$.

Remark Recall that our testing semantics allows for scenarios where the tested process is a machine with menus and the test is a user which resolves the synchronization nondeterminism. In this case the test has the power to see the history of actions-candidates for synchronization, rather than only the history of performed actions (i.e. its local history). In other scenarios a test would resolve its internal nondeterminism the same way the process does, by looking only into its local history. In this case, there is no need to change the labels of the internal transitions in a test while testing as it is done in Definition 4.1. It is important, however, that the way a test resolves its internal nondeterminism has no influence on the testing preorder relation, as we will see in Sect. 6. It rather only influences the obtained maximal/minimal probabilities to report success: the more power a test has, the higher/lower the obtained maximal/minimal probability (over all resolutions of the nondeterminism).

5. Probabilistic ready-trace preorder

In this section we define the probabilistic ready-trace preorder relation on processes. The relation is based on the ability of a process to mimic the probabilistic behaviour of another process under an arbitrary resolution of the internal nondeterminism of the latter. By mimicking the probabilistic behaviour, we mean matching the probability of an arbitrary ready-trace, i.e. a sequence of action-menus and performed actions. As the probability to observe a ready-trace is conditioned upon the actions that are actually performed, we employ conditional probabilities of the ready-traces. For these reasons, we exploit the Bayesian definition of probability [Lid80], in which the probability is naturally conditioned, rather than the measure-theoretic definition.

5.1. Bayesian probability

We consider a sample space, Ω, consisting of points called *elementary events*. Selection of a particular $a \in \Omega$ is referred to as "a has occurred". An *event* $A \subseteq \Omega$ is a set of elementary events. $A, B, C, \ldots$ range over events.

An event A *has occurred* iff a has occurred for some $a \in A$. Let $A_1, A_2, \ldots$ be a sequence of events and C be an event. The members of the sequence are *exclusive given C*, if, whenever C has occurred, no two of them can occur together, that is, if $A_i \cap A_j \cap C = \emptyset$ whenever $i \neq j$. C is called a *conditioning* event. If the conditioning event is Ω, then "given Ω" is omitted.

For certain pairs of events A and B, a real number $P(A \mid B)$ is defined and called the *probability* of A given B. For $P(A \mid \Omega)$ we simply write $P(A)$. These numbers satisfy the following axioms:

Ax1: $0 \leq P(A|B) \leq 1$ and $P(A|A) = 1$.
Ax2: If the events in $\{A_i\}_{i=1}^{\infty}$ are exclusive given B, then $P\left(\bigcup_{i=1}^{\infty} A_i|B\right) = \sum_{i=1}^{\infty} P(A_i|B)$.
Ax3: $P(C|A \cap B) \cdot P(A|B) = P(A \cap C|B)$.

5.2. The preorder relation $\preceq_{\mathsf{RT}}$

We define a ready trace, the conditional probability of a ready trace, and the preorder relation on processes based on the conditional probabilities.

Definition 5.1 (Ready-trace). A *ready-trace of length* n is a sequence $(M_1, a_1, M_2, a_2, \ldots, M_{n-1}, a_{n-1}, M_n)$, where $M_i \subseteq \mathcal{A}$ for all $i \in \{1, 2, \ldots, n\}$ and $a_i \in M_i$ for all $i \in \{1, 2, \ldots, n-1\}$.

We assume that an observer is able to see the actions that the process performs, together with the menus out of which actions are chosen. Intuitively, a ready-trace $(M_1, a_1, M_2, a_2, \ldots, M_{n-1}, a_{n-1}, M_n)$ can be observed if the initial menu is M_1, then action $a_1 \in M_1$ is performed, then the next menu is M_2, then action $a_2 \in M_2$ is performed and so on, until the observation ends at a point when the menu is M_n.

Next, given a deterministic finite process s, we define process $s_{(M,a)}$, that is the process that s becomes, assuming that menu M was offered to s and action a was performed. For example, for process s in Fig. 1, $s_{(\{h,t\},h)} = \frac{1}{2}p \oplus \frac{1}{2}0$. For a state s, we write shortly $s \overset{\pi}{\curvearrowright} s_n$ with s_n being an action state, rather than $s \overset{\pi_1}{\rightsquigarrow} s_1 \overset{\pi_2}{\rightsquigarrow} s_2 \ldots \overset{\pi_n}{\rightsquigarrow} s_n$ for $\pi = \pi_1\pi_2 \cdots \pi_n$.

Definition 5.2 Let s be a finite deterministic process. Let $M \subseteq \mathcal{A}$, $a \in M$. The process graph $s_{(M,a)}$ is obtained from s in the following way:

– if $\mathcal{I}(s) = M$ then $s_{(M,a)} \equiv s_a$;
– if $\{s_i\}_{i \in I} \neq \emptyset$ are all the process graphs such that $\mathcal{I}(s_i) = M$ and $s \overset{\pi_i}{\curvearrowright} s_i$ for $i \in I$, then

$$s_{(M,a)} \equiv \oplus_{i \in I} \frac{\pi_i}{\pi} s_{ia}, \qquad \text{for } \pi = \sum_{i \in I} \pi_i.$$

– in any other case, $s_{(M,a)}$ is undefined.

Next, for a finite deterministic process s and a ready-trace $(M_1, a_1, \ldots, M_{n-1}, a_{n-1}, M_n)$, we define the conditional probability to observe menu M_n in s, given that previously the sequence $M_1, a_1, \ldots, M_{n-1}, a_{n-1}$ was observed. These conditional probabilities are essential for the definition of the ready-trace preorder relation, presented afterwards.

Definition 5.3 Let $(M_1, a_1, \ldots, M_{n-1}, a_{n-1}, M_n)$ be a ready-trace of length n and s be a finite deterministic process. Functions $P_s^1(M)$ and $P_s^n(M_n \mid M_1, a_1, \ldots M_{n-1}, a_{n-1})$, for $n > 1$, are defined in the following way:

$$P_s^1(M) = \begin{cases} \sum_{i \in I} \pi_i P_{s_i}^1(M) & \text{if } [\{s \overset{\pi_i}{\rightsquigarrow} s_i\}_{i \in I}], \\ 1 & \text{if } \mathcal{I}(s) = M, \\ 0 & \text{otherwise.} \end{cases}$$

$$P_s^2(M_2 \mid M_1, a_1) = \begin{cases} P_{s_{(M_1,a_1)}}^1(M_2) & \text{if } P_s^1(M_1) > 0, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

$$P_s^n(M_n \mid M_1, a_1, \ldots, M_{n-1}, a_{n-1}) = \begin{cases} P_{s_{(M_1,a_1)}}^{n-1}(M_n \mid M_2, a_2, \ldots, a_{n-1}) & \text{if } P_s^1(M_1) > 0, \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

When the sample space consists of all the subsets of $\mathcal{A}$, for a given finite deterministic process s, function $P_s^1(M)$ can be interpreted as the probability that menu M is observed when process s starts executing. When the sample space consists of all the ready-traces of length n, function $P^n_s(M_n \mid M_1, a_1, \dots M_{n-1}, a_{n-1})$ can be interpreted as the probability of the event $\{(M_1, a_1, \dots, M_{n-1}, a_{n-1}, M_n)\}$, given the event $\{(M_1, a_1, \dots M_{n-1}, a_{n-1}, X) \mid X \subseteq \mathcal{A}\}$, when observing ready-traces of process s. It can be shown that these probabilities are well defined, i.e. they satisfy axioms Ax1–Ax3 from Sect. 5.1.

Example 5.1 Consider process $p|||d$ in Fig. 3. We have:

$$P^1_{p|||d}(\{a, d\}) = 1, \quad P^2_{p|||d}(\{b, d\} \mid \{a, d\}, a) = \frac{1}{3}, \quad P^3_{p|||d}(\{d\} \mid \{a, d\}, a, \{b, d\}, b) = 1.$$

Definition 5.4 (Ready-trace preorder). Let s and t be two processes. We say s *implements* t *w.r.t. ready-traces*, denoted $s \preceq_{\mathsf{RT}} t$, if and only if for every $m \geq 0$ and every m-resolution $\bar{s}$ of s, there exists an m-resolution $\bar{t}$ of t such that for all $k \leq m$ and for all ready-traces $(M_1, a_1, \dots, M_k)$,

- $P^1_{\bar{s}}(M_1) = P^1_{\bar{t}}(M_1)$ and
- if $k > 1$, then $P^k_{\bar{s}}(M_k \mid M_1, a_1, \dots M_{k-1}, a_{k-1})$ is defined if and only if $P^k_{\bar{t}}(M_k \mid M_1, a_1, \dots, M_{k-1}, a_{k-1})$ is defined, and, in case they are both defined, they are equal.

Informally, a process s implements a process t if and only if for every m-resolution $\bar{s}$ of the nondeterminism in s, there is an m-resolution $\bar{t}$ of the nondeterminism in t, such that for every ready-trace $(M_1, a_1, \dots, M_k)$ of length $k \leq m$, the probability to observe M_k, given that previously the sequence $M_1, a_1, \dots M_{k-1}, a_{k-1}$ was observed, is defined at the same time for both $\bar{s}$ and $\bar{t}$. Moreover, in case both probabilities are defined, they coincide. In general, process s implements process t iff s contains "less" internal nondeterminism than t.

Definition 5.5 (*Ready-trace equivalence*) Let s and t be two processes. s and t are ready-trace equivalent, denoted by $s \approx_{\mathsf{RT}} t$, iff $s \preceq_{\mathsf{RT}} t$ and $t \preceq_{\mathsf{RT}} s$.

Examples Processes s and $\bar{s}$ in Fig. 1 are ready-trace equivalent, and the same holds for the process pairs x and $\bar{x}$ in Fig. 2a, z and v in Fig. 4a, and $s\|u$ and $\bar{s}\|u$ in Fig. 6. Processes z and v in Fig. 4a are ready-trace equivalent and they both implement process r in the same figure. Recall that processes z and v can be actually seen as an interleaving of processes $\tau_5 ab \sqcap \tau_6 ac$, resp. $a(\tau_1 b \sqcap \tau_2 c)$ (none of which can recognize action d) with action d, while process r has "full control" over its nondeterminism. Figure 5 presents relations between several cyclic processes.

6. Equating the two preorders

We show that the testing preorder relation $\preceq_{\mathsf{T}}$ and the ready-trace preorder relation $\preceq_{\mathsf{RT}}$ *coincide*. In addition, we show that deterministic tests, i.e. tests without internal transitions, suffice for comparison of processes.

Theorem 6.1 Let s and t be two processes. If $s \preceq_{\mathsf{RT}} t$, then $s \preceq_{\mathsf{T}} t$.

Proof. Assume that $s \not\preceq_{\mathsf{T}} t$. Then, by Definition 4.3 there exists a test T with length$(T) = m$, for some $m \geq 0$ and there exists an m-resolution $\bar{s}$ of s, such that for every m-resolution $\bar{t}$ of t, it holds that $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$. It is enough to show that for $\bar{s}$ and for every m-resolution $\bar{t}$ of t,

- there exists a menu $M \subseteq \mathcal{A}$, such that $P^1_{\bar{s}}(M) \neq P^1_{\bar{t}}(M)$, or
- for some $k \leq m$ there exists a ready trace $(M_1, a_1, \dots M_k)$, such that $P^k_{\bar{s}}(M_k \mid M_1, a_1, \dots M_{k-1}, a_{k-1})$ and $P^k_{\bar{t}}(M_k \mid M_1, a_1, \dots M_{k-1}, a_{k-1})$ are not defined at the same time, or they are both defined, but different.

We show this by induction on the structure of T. We assume that in the states in which T can perform an ω action, no other actions can be performed. Formally, if for some state u it holds that $u \xrightarrow{\omega}$, then $\mathcal{I}(u) = \{\omega\}$. We can assume this without loss of generality because the other actions, if they exist, do not change the power of the test (see Definition 4.2), i.e. the test immediately reports ω each time it is able to.

Suppose that the test can perform at most one transition before performing ω. Suppose first that there exist transitions $[\{T \overset{\tau_i}{\twoheadrightarrow} T_i\}_{i\in I}]$ such that every T_i for $i \in I$ can perform ω or deadlock. From Definition 4.2 it follows that T is not able to make a difference between s and t. Similarly if $[\{T \overset{\pi_i}{\rightsquigarrow} T_i\}_{i\in I}]$. Therefore, suppose that $[\{T \xrightarrow{a_i} T_i\}_{i\in I}]$ and every T_i for $i \in I$ can perform ω or deadlock. Let $\bar{t}$ be an m-resolution of t. By assumption, $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$. Assume that $P^1_{\bar{s}}(M) = P^1_{\bar{t}}(M)$ for every menu $M \subseteq \mathcal{A}$. Let $\{M_i\}_{i\in I}$ be all the menus such that $P^1_{\bar{s}}(M_i) = P^1_{\bar{t}}(M_i) = \pi_i > 0$ and $\mathcal{I}(T) \cap M_i \neq \emptyset$. Denote $\mathcal{I}(T) \cap M_i$ by M_i^T. From Definition 4.2 we have

$$\mathcal{R}(\bar{s}, T) = \sum_{i\in I} \pi_i \sum_{a\in M_i^T} \tau^{\varepsilon}_{(a,M_i^T)} \mathcal{R}\left(\bar{s}_{(M_i,a)},\ T_a,\ \tau^{\varepsilon}_{(a,M_i^T)}\right), \tag{1}$$

$$\mathcal{R}\left(\bar{t}, T\right) = \sum_{i\in I} \pi_i \sum_{a\in M_i^T} \tau^{\varepsilon}_{(a,M_i^T)} \mathcal{R}\left(\bar{t}_{(M_i,a)},\ T_a,\ \tau^{\varepsilon}_{(a,M_i^T)}\right). \tag{2}$$

Note that

$$\mathcal{R}\left(\bar{t}_{(M_i,a)},\ T_a,\ \tau^{\varepsilon}_{(a,M_i^T)}\right)$$

yields the polynomial 0 or 1 for every $i \in I$ and for every $a \in M_i^T$, depending only on whether T_a deadlocks or performs ω. Therefore, we obtain that $\mathcal{R}(\bar{s}, T) = \mathcal{R}(\bar{t}, T)$, which contradicts the assumption that $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$. Therefore, there must exist a menu M such that $P^1_{\bar{s}}(M) \neq P^1_{\bar{t}}(M)$.

Suppose now that $[\{T \overset{\tau_i}{\twoheadrightarrow} T_i\}_{i\in I}]$, such that T_i for $i \in I$ are arbitrary tests. By assumption, $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$. By Definition 4.2, $\mathcal{R}(\bar{s}, T) = \sum_{i\in I} \tau_i^{\varepsilon}\mathcal{R}(\bar{s}, T_i)$ and $\mathcal{R}(\bar{t}, T) = \sum_{i\in I} \tau_i^{\varepsilon}\mathcal{R}(\bar{t}, T_i)$. Therefore, $\mathcal{R}(\bar{s}, T_i) \neq \mathcal{R}(\bar{t}, T_i)$ for some $i \in I$. The rest follows by the inductive assumption. The case when $[\{T \overset{\pi_i}{\twoheadrightarrow} T_i\}_{i\in I}]$, such that the T_i for $i \in I$ are arbitrary tests, is similar to the previous case.

Suppose now that there exist transitions $[\{T \xrightarrow{a_i} T_i\}_{i\in I}]$, such that each T_i is an arbitrary test. By assumption, $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$. If there exists a menu M such that $P^1_{\bar{s}}(M) \neq P^1_{\bar{t}}(M)$, then we are done. Therefore, assume that $P^1_{\bar{s}}(M) = P^1_{\bar{t}}(M)$ for every menu $M \subseteq \mathcal{A}$. Let $\{M_i\}_{i\in I}$ be all the menus such that $P^1_{\bar{s}}(M_i) = P^1_{\bar{t}}(M_i) = \pi_i > 0$ and $\mathcal{I}(T) \cap M_i \neq \emptyset$. Denote $\mathcal{I}(T) \cap M_i$ by M_i^T. We obtain again that the equations (1) and (2) hold. Then, since $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$, it must be that

$$\mathcal{R}\left(\bar{s}_{(M_i,a)},\ T_a,\ \tau^{\varepsilon}_{(a,M_i^T)}\right) \neq \mathcal{R}\left(\bar{t}_{(M_i,a)},\ T_a,\ \tau^{\varepsilon}_{(a,M_i^T)}\right)$$

for some M_i and $a \in M_i$. From the last, it can be easily concluded that

$$\mathcal{R}\left(\bar{s}_{(M_i,a)},\ T_a\right) \neq \mathcal{R}\left(\bar{t}_{(M_i,a)},\ T_a\right).$$

By the inductive assumption, there exists a menu $M \subseteq \mathcal{A}$, such that

$$P^1_{\bar{s}_{(M_i,a)}}(M) \neq P^1_{\bar{t}_{(M_i,a)}}(M),$$

or there exists a ready trace $(M_1, a_1, \ldots M_k)$ for some $k < m$, such that

$$P^k_{\bar{s}_{(M_i,a)}}(M_k \mid M_1, a_1, \ldots M_{k-1}, a_{k-1}) \text{ and } P^k_{\bar{t}_{(M_i,a)}}(M_k \mid M_1, a_1, \ldots M_{k-1}, a_{k-1})$$

are not defined at the same time, or they are both defined, but different. From $P^1_{\bar{s}}(M_i) = P^1_{\bar{t}}(M_i)$ and from Definition 5.3 we have that in the first case $P^2_{\bar{s}}(M \mid M_i, a) \neq P^2_{\bar{t}}(M \mid M_i, a)$, while in the second case

$$P^{(k+1)}_{\bar{s}}(M_k \mid M_i, a, M_1, a_1, \ldots M_{k-1}, a_{k-1}) \text{ and } P^{(k+1)}_{\bar{t}}(M_k \mid M_i, a, M_1, a_1, \ldots M_{k-1}, a_{k-1})$$

are not defined at the same time, or they are both defined, but different. This completes the proof. $\square$

Theorem 6.2 Let s and t be two processes. If $s \preceq_{\mathsf{T}} t$, then $s \preceq_{\mathsf{RT}} t$.

Proof. Assume that $s \not\preceq_{\mathsf{RT}} t$. Then, by Definition 5.4, for some $m \geq 0$ there exists an m-resolution $\bar{s}$ of s, such that for every m-resolution $\bar{t}$ of t, it holds that

– there exists $M \subseteq \mathcal{A}$ such that $P^1_{\bar{s}}(M) \neq P^1_{\bar{t}}(M)$, or
– for some k, $1 < k \leq m$, there exists a ready trace $(M_1, a_1, \ldots M_k)$, such that $P^k_{\bar{s}}(M_k \mid M_1, a_1, \ldots M_{k-1}, a_{k-1})$ and $P^k_{\bar{t}}(M_k \mid M_1, a_1, \ldots M_{k-1}, a_{k-1})$ are not defined at the same time, or they are both defined, but different.

It is enough to show that there exists a test T with length$(T) = m$ such that, for the given $\bar{s}$, and for an arbitrary m-resolution $\bar{t}$ of t, it holds that $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$. The proof is by induction on the minimal value of m.

In the case $m = 0$ there is nothing to prove. Suppose $m = 1$. Then, $\bar{s}$ is a 1-resolution of s, such that for every 1-resolution $\bar{t}$ of t there exists a menu $M_{\bar{t}}$ with $P^1_{\bar{s}}(M_{\bar{t}}) \neq P^1_{\bar{t}}(M_{\bar{t}})$. Take the test $T = \Box_{a\in\mathcal{A}}\, a\omega$. Let $\bar{t}$ be an arbitrary 1-resolution of t. We have

$$\mathcal{R}(\bar{s}, T) = \sum_{M: P^1_{\bar{s}}(M)>0} P^1_{\bar{s}}(M) \sum_{a\in M} \tau^{\varepsilon}_{(a,M)}, \quad \text{and} \quad \mathcal{R}(\bar{t}, T) = \sum_{M: P^1_{\bar{t}}(M)>0} P^1_{\bar{t}}(M) \sum_{a\in M} \tau^{\varepsilon}_{(a,M)}.$$

Assuming that $\mathcal{R}(\bar{s}, T) = \mathcal{R}(\bar{t}, T)$, we obtain that $P^1_{\bar{s}}(M) = P^1_{\bar{t}}(M)$ for every menu M, which contradicts our assumption that $P^1_{\bar{s}}(M_{\bar{t}}) \neq P^1_{\bar{t}}(M_{\bar{t}})$. Therefore, $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$.

Assume now that $m > 1$. Let $\bar{t}$ be an arbitrary m-resolution of t. If there exists an $M \subseteq \mathcal{A}$ such that $P^1_{\bar{s}}(M) \neq P^1_{\bar{t}}(M)$, then $T = \Box_{a\in\mathcal{A}}\, a\omega$ distinguishes between $\bar{s}$ and $\bar{t}$, similar as in the case $m = 1$. Assume that $P^1_{\bar{s}}(M) = P^1_{\bar{t}}(M)$ for every menu M. Take the test $T = T_m$, defined inductively by

$T_1 = \Box_{a\in\mathcal{A}}\, a\omega$,
$T_n = \Box_{a\in\mathcal{A}}\, aT_{n-1}$ for every $n > 1$.

We have

$$\mathcal{R}(\bar{s}, T) = \sum_{M: P^1_{\bar{s}}(M)>0} P^1_{\bar{s}}(M) \sum_{a\in M} \tau^{\varepsilon}_{(a,M)} \mathcal{R}\left(\bar{s}_{(M,a)},\ T_a,\ \tau^{\varepsilon}_{(a,M)}\right),$$
$$\mathcal{R}\left(\bar{t}, T\right) = \sum_{M: P^1_{\bar{t}}(M)>0} P^1_{\bar{t}}(M) \sum_{a\in M} \tau^{\varepsilon}_{(a,M)} \mathcal{R}\left(\bar{t}_{(M,a)},\ T_a,\ \tau^{\varepsilon}_{(a,M)}\right).$$

Assume that $\mathcal{R}(\bar{s}, T) = \mathcal{R}(\bar{t}, T)$. Then, for every menu M,

$$\mathcal{R}\left(\bar{s}_{(M,a)},\ T_a,\ \tau^{\varepsilon}_{(a,M)}\right) = \mathcal{R}\left(\bar{t}_{(M,a)},\ T_a,\ \tau^{\varepsilon}_{(a,M)}\right). \tag{3}$$

From the assumption that $P^k_{\bar{s}}(M_k \mid M_1, a_1, \ldots M_{k-1}, a_{k-1})$ and $P^k_{\bar{t}}(M_k \mid M_1, a_1, \ldots M_{k-1}, a_{k-1})$ are not defined at the same time, or they are both defined, but different, and from $P^1_{\bar{s}}(M_1) = P^1_{\bar{t}}(M_1)$, we have that

$$P^{k-1}_{\bar{s}_{(M_1,a_1)}}(M_k \mid M_2, a_2, \ldots M_{k-1}, a_{k-1}) \text{ and } P^{k-1}_{\bar{t}_{(M_1,a_1)}}(M_k \mid M_2, a_2, \ldots M_{k-1}, a_{k-1})$$

are not defined at the same time, or they are both defined, but different. Then, by the inductive assumption, we have that

$$\mathcal{R}\left(\bar{s}_{(M_1,a_1)},\ T_{a_1},\ \tau^{\varepsilon}_{(a_1,M_1)}\right) \neq \mathcal{R}\left(\bar{t}_{(M_1,a_1)},\ T_{a_1},\ \tau^{\varepsilon}_{(a_1,M_1)}\right).$$

This contradicts (3). Therefore, $\mathcal{R}(\bar{s}, T) \neq \mathcal{R}(\bar{t}, T)$ and the proof of the theorem is complete. □

Corollary 6.1 Let s and t be two processes. $s \preceq_{\mathsf{T}} t$ if and only if $s \preceq_{\mathsf{RT}} t$.

The following proposition states that tests with internal transitions have no more distinguishing power than deterministic tests, i.e. tests without internal transitions. In other words, if two processes are not related by the testing preorder relation, then this can be concluded using only a certain deterministic test.

Proposition 6.1 *Let s and t be two processes. If $s \not\preceq_{\mathsf{T}} t$, then there exists a deterministic test T with length m, such that for some m-resolution $\bar{s}$ of s and for every m-resolution $\bar{t}$ of t, $\mathcal{R}(s, T) \neq \mathcal{R}(t, T)$.*

Proof. If $s \not\preceq_{\mathsf{T}} t$, then, by Corollary 6.1, $s \not\preceq_{\mathsf{RT}} t$. The rest of the proof follows the lines of the proof of Theorem 6.2. □

7. Related work

As closely related to the results presented here, we consider the research reports that face the challenge of defining a satisfactory linear-time semantics for concurrent systems with probabilistic and nondeterministic choice; by linear-time here we mean allowing distribution of prefix over internal (probabilistic) choice (or, in other words, equating processes x and $\bar{x}$ from Fig. 2a). In the introduction we discussed that this problem is non-trivial. Work addressing it was reported in [Low93, Sei95, Seg95, Mor96, GDR97, KwN98b, AHJ01, Caz03, Che06, LNR06, CSV07, DHR08, ChP07, ABD11]. Our work is also closely related to the research oriented towards restricting the power of the schedulers for concurrent probabilistic-nondeterministic systems [AHJ01, Che06, ChP07, GiD09, And11], to achieve compositionality for trace preorders [AHJ01, Che06], or to obtain realistic probabilities for the behaviour of the system [ChP07, GiD09, And11].

The report [Low93] defines trace-style semantical equivalences for processes with action choice and probabilistic choice; they are extended for processes with internal nondeterminism such that, first, the internal nondeterminism is resolved using the almighty (randomized) schedulers discussed in the introduction, and then processes are compared. Not surprisingly, the equivalences are not congruences, for example they do not equate processes $x \| y$ and $\bar{x} \| y$ from Example 1.2 in the introduction, although processes x and $\bar{x}$ are equated. It is interesting to note that reference [Low93] is one of the earliest that note the problem discussed in this example. Similar problems with compositionality, induced by the underlying almighty schedulers, appear in the trace equivalences defined in [Seg95] and [CSV07]; the latter one characterizes the former one via a testing scenario. We note that the compositionality problem in [Seg95] remains even when restricted to processes without internal nondeterminism, as processes $p|||d$ and $q|||d$ from Fig. 3 are not equated (although p and q are) or, in other words, action choice does not distribute over probabilistic choice.

References [Sei95], [LNR06], and [DHR08] consider processes without internal nondeterminism and define equivalences that equate two processes only if they cannot be distinguished by the environment. However, the environment is not a process itself (as in our case), but rather only a sequence of actions. As a result, although [Sei95], [LNR06], and [DHR08] allow distributivity of external action choice over probabilistic choice, they also make undesirable identifications from the point of view of process theory; for example, they equate processes $\frac{1}{2}ca \oplus \frac{1}{2}cb$ and $\frac{1}{2}c(a \,\Box\, b) \oplus \frac{1}{2}c$.

In [Mor96] a process is a probability distribution over standard CSP processes; two processes are equivalent if the probability distributions are the same. In other words, all the probabilistic choices are resolved before the execution of the process and a resolution is a standard non-probabilistic process. This way, probabilistic choice distributes over most of the operators – for example, processes x and $\bar{x}$ (Fig. 2a) are equated. However, by "lifting" the probabilistic choices to the root, the nondeterministic choices are "pushed" downwards and replicated. As discussed in the introduction, and already in [Mor96], the replication of the nondeterministic choices leads to loss of probability information. With this approach also the idempotence of the internal choice is lost [Mor96], viz. the law $x = x \sqcap x$ does not hold, and action choice does not distribute over probabilistic choice. Interestingly, in [Mor96] it has been discussed that the problem with loss of idempotence and loss of probability information could be overcome by defining an "indifferent" nondeterministic choice operator in addition to the "demonic" one, which would be resolved before the execution of the process, similarly as the probabilistic choice. Translated in terms of schedulers, the authors in [Mor96] propose that, in the future, the power of the schedulers that resolve the nondeterminism should be restricted in order to preserve probability information as well as idempotence of nondeterministic choice.

Similarly as above, in [ABD11] first the probabilistic choices are resolved and the resolutions are compared under the standard may-testing semantics of [DeH84]. Processes in [ABD11] do not contain internal nondeterminism. Again, action choice does not distribute over probabilistic choice, which leads to compositionality problems. The same problems also occur in the testing equivalence of [Caz03], defined with the unrestricted schedulers, similar to the testing semantics of [WaL92] discussed in the introduction.

The loss of idempotence for internal choice was overcome by the button-pushing testing equivalence defined in [KwN98b]; however, here still congruence for parallel composition could not be achieved [KwN98a], viz. action choice does not distribute over probabilistic choice.

We note that ready-trace equivalences for probabilistic systems were previously defined in [GDR97] and [Low93]. Reference [GDR97] defines a ready-trace equivalence for processes given in denotational semantics, with action choice and internal probabilistic choice, that is, without internal nondeterminism. We conjecture that the ready-trace equivalence of [GDR97] coincides with the one defined here, when restricted to processes without internal nondeterminism. However, our definition, unlike [GDR97], yields a black-box testing scenario in the style of [Gla01], with which the equivalence can be determined. Namely, in our case, if the observer can see the action menus at every moment, then she can deduce the conditional probability of a ready-trace via a simple statistical procedure. The definition of ready-trace equivalence given in [Low93] does not yield a testing scenario, since the function that computes the probability of every ready-trace does not yield a probability measure.

The results from the above research reports indicate that mixing probabilistic and nondeterministic choice in concurrent processes creates compositionality problems for linear-time equivalences, no matter whether in order to establish equivalence the internal choices are resolved first [Low93, Seg95, CSV07], or the probabilistic choices are resolved first [Mor96, ABD11]. This observation, together with the problem with overestimation of probabilities under almighty schedulers explained in Example 1.2, noted already in [Low93, Seg95, Mor96], led to a new research direction, aiming to restrict the power of the schedulers used to resolve the nondeterminism. Work in this direction has been reported in [AHJ01, Che06, ChP07, GiD09].

Reference [AHJ01] is the first one that addresses the problem with compositionality for trace semantics [Seg95] in probabilistic systems. The parallel composition in [AHJ01] is synchronous – each time the composed system performs a step, all the components perform a step. The states of the system are valuations over a set of variables, and the information available to each component can be modeled by restricting the variables it is able to read. Reference [Che06] considers asynchronous systems, and restricts the power of the schedulers in a parallel composition, also to obtain compositionality for a trace equivalence for probabilistic systems distinguishing between input and output actions [WSS97]. When components are composed, the local nondeterminism in a component is resolved based only on the history of the component itself; the global nondeterminism, arising from the choice on which component will generate the next output action on which (the rest of) the components synchronize, is resolved by the components themselves, which pass a token one to another. A component that holds the token decides to which component it forwards the token, based on its own local history. Alternatively, as is discussed in [Che06], the global nondeterminism can be resolved by a centralized component-scheduler, which resolves the nondeterminism based on the global history of the composition. In [GiD09] it is observed that such a scheduler, that resolves the global interleaving nondeterminism based on the complete history, may be still too powerful. A restriction is added to the interleaving scheduler in [GiD09], such that it cannot use information from one component in order to decide between two other components. Comparing our approach to [AHJ01] and [Che06], we can conclude that the common feature enabling compositional reasoning under linear-time semantics is resolving the local nondeterminism in a component based on local information only. The three approaches differ in the resolution of the global nondeterminism. While the focus in [AHJ01] and [Che06] is on resolving nondeterminism in special types of concurrent systems, our focus was on deriving a satisfactory extension of a common process language such as CSP, as evidenced in [GeA10a, GeA12, Geo11]. The result is a ready-trace semantics, finer than trace semantics, under which CSP can be extended with a probabilistic choice. The focus in [GiD09, Gir10] is not on devising compositionality, but on improving the probabilistic verification techniques, by obtaining as realistic estimates of probabilities as possible. In the present work, among other things, we were interested in the question “How much freedom can the schedulers retain such that compositionality for linear-time equivalence is preserved?”.

In [Cal10] an algorithm is proposed for computing (time-bounded) optimal reachability probabilities with respect to the distributed schedulers of [Che06] and [GiD09]. The algorithm is based on an interpretation of the model as a parametric Markov chain, such that each state in the latter is a path in the original probabilistic system. The idea is that the scheduler decisions are parameters of the parametric Markov chain, similarly as here. However, there is a conceptual difference between the two approaches. Namely, we are interested in integrating the schedulers in the model itself for a compositional reasoning, and the external choice is left to be resolved by the environment; on the other hand, in [Cal10] the interest is on resolving all nondeterminism in a distributed system in order to compute the bounds of the reachability probabilities, and composing the parametric Markov chains is not an issue. It is interesting that both papers [GeA10b] and [Cal10], that propose integrating the scheduler information into labels, appeared at approximately the same time.

The paper [ChP07] takes a dual approach to the previous approaches, by introducing an *explicit* scheduler that communicates to processes via labels: two sub-processes in the process are indistinguishable to a scheduler if they have the same labels. Thus, by a suitable labeling, the modeler specifies which internal or random choices are visible to the scheduler and which are not. When compared to our and all other approaches, the method in [ChP07] allows for a greater flexibility in equating processes. For example, whether processes x and $\bar{x}$ in Fig. 2a are testing-equivalent depends on the labeling system. Note that this flexibility means that more responsibility is being delegated to the modeler. In our case, on the other hand, it is enough to ensure only that no label appears twice before processes are composed; as the processes start composing, the labels identify which internal choices are multiple instances of the same internal choice, or in terms of [ChP07], which random choices are invisible to the scheduler. Moreover, new labels are automatically assigned to the nondeterminism arising from parallelism, reflecting the information, based on which, this nondeterminism is resolved, and thus serving to identify multiple instances of an internal choice. Also, while in [ChP07] the labels serve to navigate the scheduler, limiting it thus to a deterministic one, in our case the labels represent unknown probabilities, yielding randomized schedulers.

The power of the schedulers for verification of security properties has been also restricted in [And11]; in this paper, tagged probabilistic automata are defined, such that the transitions in a parallel composition are tagged with the identifier of the component that originates the transition, or with a pair of components if synchronization has happened. At each point of the execution, the scheduler of the composition can see the current tags and the history of chosen tags and performed actions. Thus, this scheduler has similarities to our schedulers, showing once again how approaches with different motivations can converge independently to similar solutions. The framework of [And11] has been extended with internal nondeterminism in [Alv10] and corresponding process equivalences (bisimulation and completed trace equivalence), together with a congruence result for bisimulation given in [Alv10].

Finally, note that restricting the power of schedulers that resolve the nondeterminism, for the purpose of realistic modeling, is a research topic in areas other than concurrency theory itself; for example in security (e.g. [Can01]), in operation research (e.g. [Sod71]), and in artificial intelligence (e.g. [KLC98]). We do not make formal comparison to work in those areas, as, reasonably, they target either more specific, or different problems.

8. Concluding remarks

Conclusion In this part, we have proposed a new probabilistic extension of may/must testing theory [DeH84] for concurrent processes, by restricting the schedulers that resolve the nondeterminism. The motivation was that the existing approaches for probabilistic may/must testing [WaL92, Seg96, JoW02, PDM07, Den08, Den09], based on the almighty schedulers for resolving nondeterminism, yielded unrealistic probabilities with which a process passes a test (a problem already observed in [Low93, Mor96, Seg95]). As a result, they equated too few processes when compared to [DeH84], or, in other words, the internal probabilistic choice was observable, unlike in [DeH84]. We have shown that our testing preorder can be characterized with a probabilistic ready-trace preorder relation, by which the exact moment in time in which a probabilistic or an internal choice happens is unobservable. In order to obtain realistic probabilities to pass a test, in our model the internal transitions are augmented with labels. The labels represent the information based on which the internal nondeterminism is resolved and, thus, restrict the schedulers. For finite processes, our model is at least as expressible as the model of probabilistic automata [Seg95], or the alternating model of probabilistic systems [Han91], since each finite process from the latter two can be represented in our model by giving different labels to all internal transitions.

Since [GeA10b], which is the preliminary version of the present paper, we have also defined a generalized parallel composition for our model in [GeA10a, Geo11], such that processes synchronize on a set of actions and interleave on the rest of the actions, as in CSP [Ros98], and also the synchronized actions are hidden, as in CCS [Mil80]. We have shown that our ready-trace preorder is a precongruence, that is, is preserved under parallel composition [GeA10a, Geo11]. Based on the ready-trace equivalence and the new parallel composition, we have given a probabilistic extension of CSP [GeA10a, Geo11, GeA12] which preserves all the nice properties of the internal choice from CSP, such as the distributivity laws and idempotence. Thus, with our approach we have also solved another open problem, namely to give a satisfactory probabilistic extension of CSP, that respects the original laws.

Future work The results presented here open several questions that can be addressed in the future. The synchronization operator defined here, due to its testing nature, applies only to finite processes: tests are finite and unfolded processes are finite. It would be interesting to generalize the operator for synchronization of arbitrary processes, and investigate whether the unfolding could be bypassed in certain cases and whether a finite representation of infinite processes can be obtained. Also, in the present setting we limited ourselves to processes without divergence. There are several ways to treat divergence: one can treat it as an error in the model (e.g. [Hoa85, DeH84]), or can abstract away from it by assuming fairness (e.g. [Mil80, BBR10, ReV07]), or can allow it, but distinguish between processes with and without divergence (see e.g. [GLT09]). In the present setting, divergent processes cannot be obtained by parallel composition; divergence in the non-composed processes could be treated by assuming that the divergence transitions are probabilistic (implying fairness). Then, cycles of probabilistic transitions are easily handled using Markov Chain theory [How71]. However, if a parallel composition that yields cyclic processes is defined, then an internal action may be obtained by synchronization and divergence would require a different treatment.

Another problem that needs to be addressed is the decidability of our ready-trace equivalence. We have defined it using randomized schedulers for the resolution of the nondeterminism, which are in general more powerful than the deterministic schedulers, i.e. those that assign trivial probabilities from the set {0, 1} to the alternatives. Indeed, had we defined our equivalence by deterministic schedulers, then it would have distinguished less. For example, processes $(\tau_1 a \sqcap \tau_2 b)|||(\tau_3 c \sqcap \tau_4 d)$ and $\tau_5(a|||c) \sqcap \tau_6(a|||d) \sqcap \tau_7(b|||c) \sqcap \tau_8(b|||d)$ would be equivalent; by our definition, they are distinguished, as the resolution of the later process such that $\tau_5 = \tau_8 = 0.5$ and $\tau_6 = \tau_7 = 0$ cannot be mimicked by any resolution of the former process. Interestingly, if we ignore the internal labels, those two processes are equated by the failures semantics of CSP [Hoa85]. Thus, we anticipate that if we restrict to deterministic schedulers, no new equalities w.r.t. CSP would arise. Therefore, restricting to deterministic schedulers for decidability reasons should be reasonable.

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

References

[ABD11] Acciai L, Boreale M, De Nicola R (2011) Linear and may-testing semantics in a probabilistic reactive setting. FMOODS-FORTE'11, LNCS 6722. Springer, Berlin, pp 29–43

[Alv10] Alvim MS, Andrés ME, Palamidessi C, van Rossum P (2010) Safe equivalences for security properties. IFIP TCS'10, pp 55–70

[And11] Andrés ME, Palamidessi C, van Rossum P, Sokolova A (2011) Information hiding in probabilistic concurrent systems. Theor Comput Sci 412(28):3072–3089

[BBK87] Baeten JCM, Bergstra JA, Klop JW (1987) Ready-trace semantics for concrete process algebra with the priority operator. Comput J 30(6):498–506

[BBR10] Baeten JCM, Basten T, Reniers MA (2010) Process algebra: equational theories of communicating processes. Cambridge University Press

[BiA95] Bianco A, de Alfaro L (1995) Model checking of probabilistic and nondeterministic systems. FSTTCS '95, LNCS 1026. Springer, Berlin, pp 499–513

[BHR84] Brookes SD, Hoare CAR, Roscoe AW (1984) A theory of communicating sequential processes. J ACM 31(3):560–599

[Cal10] Calin G, Crouzen P, D'Argenio PR, Hahn EM, Zhang L (2010) Time-bounded reachability in distributed input/output interactive probabilistic chains. SPIN'10, LNCS 6349. Springer, Berlin, pp 193–211

[Can01] Canetti R (2001) Universally composable security: a new paradigm for cryptographic protocols. FOCS'01. IEEE, pp 136–145

[Caz03] Cazorla D, Cuartero F, Valero V, Pelayo FL, Pardo JJ (2003) Algebraic theory of probabilistic and nondeterministic processes. J Logic Algebraic Programm 55(1–2):57–103

[ChP07] Chatzikokolakis K, Palamidessi C (2007) Making random choices invisible to the scheduler. CONCUR'07, LNCS 4703. Springer, Berlin, pp 42–58

[Che06] Cheung L, Lynch N, Segala R, Vaandrager F (2006) Switched PIOA: parallel composition via distributed scheduling. Theor Comput Sci 365(1–2):83–108

[CSV07] Cheung L, Stoelinga MIA, Vaandrager FW (2007) A testing scenario for probabilistic processes. J ACM 54(6):29:1–29:45

[AHJ01] de Alfaro L, Henzinger T, Jhala R (2001) Compositional methods for probabilistic systems. In: CONCUR'01, LNCS 2154. Springer, Berlin, pp 351–365

[DeH84] De Nicola R, Hennessy MCB (1984) Testing equivalences for processes. Theor Comput Sci 34:83–133

[Den09] Deng Y, van Glabbeek R, Hennessy M, Morgan C (2009) Testing finitary probabilistic processes (extended abstract). In: CONCUR'09. LNCS 5710. Springer, Berlin, pp 274–288

[Den08] Deng Y, van Glabbeek RJ, Hennessy, M, Morgan C (2008) Characterising testing preorders for finite probabilistic processes. Logical Methods Comput Sci 4(4:4):1–33

[Doo53] Doob JL (1953) Stochastic processes. Wiley, New York

[DHR08] Doyen L, Henzinger TA, Raskin J-F (2008) Equivalence of labeled Markov chains. Int J Found Comput Sci 19(3):549–563

[Geo11] Georgievska S (2011) Probability and hiding in concurrent processes. PhD thesis, Eindhoven University of Technology

[GeA10a] Georgievska S, Andova S (2010) Composing systems while preserving probabilities. EPEW 2010, LNCS 6342. Springer, Berlin, pp 268–283

[GeA10b] Georgievska S, Andova S (2010) Retaining the probabilities in probabilistic testing theory. FOSSACS'10, LNCS 6014. Springer, Berlin, pp 79–93

[GeA10c] Georgievska S, Andova S (2010) Testing reactive probabilistic processes. QAPL'10, EPTCS 28, pp 99–113

[GeA12] Georgievska S, Andova S (2012) Probabilistic CSP: preserving the laws via restricted schedulers. MMB & DFT 2012, LNCS 7201. Springer, Berlin, pp 136–150

[Gir10] Giro S (2010) On the automatic verification of distributed probabilistic automata with partial information. PhD thesis, Universidad Nacional de Córdoba

[GiD09] Giro S, D'Argenio P (2009) On the expressive power of schedulers in distributed probabilistic systems. QAPL'09, ENTCS 253(3). Elsevier, Amsterdam, pp pp 45–71

[Gla93] van Glabbeek RJ (1993) The linear time-branching time spectrum II. CONCUR'93, LNCS 715. Springer, Berlin, pp 66–81

[Gla01] van Glabbeek RJ (2001) The linear time-branching time spectrum I; the semantics of concrete, sequential processes. Handbook of process algebra, chap 1. Elsevier, Amsterdam, pp 3–99

[GLT09] van Glabbeek RJ, Luttik B, Trčka N (2009) Branching bisimilarity with explicit divergence. Fundam Inf 93:371–392

[GDR97] Gomez FC, De Frutos Escrig D., Ruiz VV (1997) A sound and complete proof system for probabilistic processes. ARTS'97, LNCS 1231. Springer, Berlin, pp 340–352

[Han91] Hansson HA (1994) Time and probability in formal design of distributed systems. Elsevier, Amsterdam

[Hen88] Hennessy M (1988) Algebraic theory of processes. MIT Press, New York

[Hoa85] Hoare CAR (1985) Communicating sequential processes. Prentice Hall, Englewood Cliffs

[How71] Howard RA (1971) Semi-Markov and decision processes. Wiley, London

[JoW02] Jonsson B, Wang Y (2002) Testing preorders for probabilistic processes can be characterized by simulations. Theor Comput Sci 282(1):33–51

[KLC98] Kaelbling LP, Littman ML, Cassandra AR (1998) Planning and acting in partially observable stochastic domains. Artif Intell J 101:99–134

[KCS98] Kumar KN, Cleaveland R, Smolka SA (1998) Infinite probabilistic and nonprobabilistic testing. FSTTCS'98, LNCS 1530. Springer, Berlin, pp 209–220

[KwN98b] Kwiatkowska M, Norman G (1998) A testing equivalence for reactive probabilistic processes. EXPRESS'98, ENTCS 16. Elsevier, Amsterdam, pp 1–19

[KwN98a] Kwiatkowska MZ, Norman GJ (1998) A fully abstract metric-space denotational semantics for reactive probabilistic processes. COMPROX '98, ENTCS 13. Elsevier, Amsterdam, pp 1–33

[LaS91] Larsen KG, Skou A (1991) Bisimulation through probabilistic testing. Inf Comput 94:1–28

[Lid80] Lindley DV (1980) Introduction to probability and statistics from a Bayesian viewpoint. Cambridge University Press, Cambridge

[LNR06] López N, Núñez M, Rodríguez I (2006) Specification, testing and implementation relations for symbolic-probabilistic systems. Theor Comput Sci 353(1):228–248

[Low93] Lowe G (1993) Representing nondeterministic and probabilistic behaviour in reactive processes. Technical Report PRG-TR-11-93. Oxford University Computing Labs

[LSV07] Lynch N, Segala R, Vaandrager F (2007) Observing branching structure through probabilistic contexts. SIAM J Comput 37(4):977–1013

[McM04] McIver A, Morgan C (2004) Abstraction, refinement and proof for probabilistic systems (Monographs in Computer Science). Springer, Berlin

[Mil80] Milner R (1980) A calculus of communicating systems. Springer, Berlin

[MMS96] Morgan C, McIver A, Seidel K (1996) Probabilistic predicate transformers. ACM Trans Program Lang Syst 18(3):325–353

[Mor96] Morgan C, McIver A, Seidel K, Sanders JW (1996) Refinement-oriented probability for CSP. Formal Aspects Comput 8(6):617–647

[DeN87] De Nicola R (1987) Extensional equivalences for transition systems. Acta Inf 24(2):211–237

[PDM07] Palmeri MC, De Nicola R, Massink M (2007) Basic observables for probabilistic may testing. QEST '07. IEEE Computer Society, pp 189–200

[Pnu85] Pnueli A (1985) Linear and branching structures in the semantics and logics of reactive systems. ICALP'85, LNCS 194. Springer, Berlin, pp 15–32

[Put94] Puterman ML (1994) Markov decision processes. Wiley, New York

[ReV07] Rensink A, W Vogler W (2007) Fair testing. Inf Comput 205:125–198

[Ros98] Roscoe AW (1998) The theory and practice of concurrency. Prentice Hall, Englewood Cliffs

[Seg95] Segala R (1995) Modeling and verification of randomized distributed real-time systems. PhD thesis, MIT

[Seg96] Segala R (1996) Testing probabilistic automata. CONCUR'96, LNCS 1119. Springer, Berlin, pp 299–314

[Sei95] Seidel K (1995) Probabilistic communicating processes. Theor Comput Sci 152:219–249

[SMM97] Seidel K, Morgan C, McIver A (1997) Probabilistic imperative programming: a rigorous approach. Proceedings of formal methods Pacific '97. Springer Series in Discrete Mathematics and Theoretical Computer Science, Singapore, Springer, Berlin

[Sod71] Sondik EJ (1971) The optimal control of partially observable Markov processes. PhD thesis, Stanford University

[WaL92] Wang Y, Larsen KG (1992) Testing probabilistic and nondeterministic processes. Proceedings of IFIP TC6/WG6.1 twelth international symposium on protocol specification, testing and verification XII, pp 47–61

[WSS97] Wu S-H, Smolka SA, Stark E (1997) Composition and behaviors of probabilistic I/O automata. Theor Comput Sci 176(1–2):1–38

Received 20 December 2011

Accepted in revised form 3 May 2012 by Peter Höfner, Robert van Glabbeek and Ian Hayes

Published online 2 July 2012