An Overview of Probabilistic Process Algebras and their Equivalences

Natalia López and Manuel Núñez

Dpt. Sistemas Informáticos y Programación
Universidad Complutense de Madrid
{natalia,mn}@sip.ucm.es

Abstract. In order to describe probabilistic processes by means of a formal model, some considerations have to be taken into account. In this paper we present some of the ideas appeared in the literature that could help to define appropriate formal frameworks for the specification of probabilistic processes. First, we will explain the different interpretations of the probabilistic information included in this kind of models. After that, the different choice operators used in the most common probabilistic languages are enumerated. Once we have an appropriate language, we have to give its semantics. Thus, we will review some of the theories based on bisimulation and testing semantics. We will conclude by studying the extensions of the chosen languages with other operators such as parallel composition and hiding.

1 Introduction

In order to specify the functional behavior of concurrent and distributed systems, process algebras [Hoa85,Hen88,Mil89,BW90,BPS01] have been shown to be a powerful mechanism. In addition to several languages, there has been intensive research in the study of semantics that could appropriately capture equivalent behavior of syntactically different processes. Nevertheless, the original formulations were not able to accurately represent systems where quantitative information, as time or probabilities, play a fundamental role. For example, if a lossy channel is specified without using a probabilistic estimation of the faulty rate, all that can be known is that a message may arrive or not. On the contrary, if one specifies such a probability and the sending of the message is iterated, it can be proved that, with probability 1, the message will arrive.

In order to define a process algebra there exists two main decisions that have to be taken:

- The mechanism to model the *choice*[1] among a set of available actions. Usually, process algebraic languages consider either a (unique) CCS-like choice operator or a pair of choice operators as in CSP.

[1] Let us remark that in the majority of the semantic frameworks we have that other operators, such as parallel and hiding, can be *derived* from the choice operator (some notable exceptions are the π-calculus and true concurrency semantics). Thus, the *choice* of the choice operator is usually more relevant than other design decisions.

– The *semantics* to assign meaning to processes. In this case, we can consider testing semantics, bisimulation semantics, trace semantics, etc.

In the case of process algebras with probabilistic information these two decisions are far from obvious. Specifically, in order to model the choice there are several possibilities. First, we may have probabilistic and non-probabilistic versions of the CCS and CSP choice operators as well as their possible combinations. In this paper, the symbol + will represent the usual CCS choice while $+_p$ will denote a probabilistic version of this operator. Besides, $\Box$ represents the usual external choice operator appearing in CSP-like languages while $\oplus$ denotes the internal choice operator. Their probabilistic versions are denoted by $\Box_p$ and $\oplus_p$, respectively. In addition to the problem of the *choice of probabilistic choices*, we may have different probabilistic models depending on the way probabilities are treated.[2] For example, we may have a unique probability distribution to (probabilistically) resolve choices among actions, that is, either a generative or a stratified model. Another option is that probabilities are used only to resolve the non-determinism generated by different occurrences of the same action, that is, a reactive model. Besides, we will consider several approaches that are adaptations of one or several of these standard models. Finally, as it can be expected, the incorporation of probabilistic information makes harder the formal definition and study of semantic frameworks for the new languages.

The research on probabilistic models of computation is not new. Actually, the first work on probabilistic automata [Rab63] originated back in the 1960's. During the 80's, there were several studies extending previous concepts, mainly logics, with probabilistic information (e.g. [FH82,Koz83,HS84,Var85,CY88,JP89]). However, it is in the last fifteen years that an explosion of models for probabilistic processes has happened, with different languages, different interpretations of the probabilistic information, etc. The first models related to labelled transition systems and probabilistic information appeared in the end of the 1980's. In [LS89], Larsen and Skou introduce a probabilistic extension of the classical notion of strong bisimulation. They characterize this relation by using a testing semantics where the discriminatory power of the tests is increased with respect to the tests in the classical framework [dNH84,Hen88]. Specifically, they allow the use of multiple copies of the tested process so that the tester may experiment with one copy at a time. They show that if two processes are not probabilistically (strongly) bisimilar then there exists a test to distinguish them with a certain probability $1 - \epsilon$, where ϵ is arbitrarily small. This first notion of probabilistic strong bisimulation has become the standard definition and it will be shown with more detail in Section 4. Following this work, Bloom and Meyer [BM89] introduce a notion of probabilistic bisimulation in terms of *probabilizations* of non-probabilistic processes. They show that two labelled transition systems are bisimilar (without probabilistic information) iff weights can be assigned to obtain new systems that cannot be distinguished by a general notion of probabilistic

[2] Following the nomenclature introduced in [GSST90], we consider the reactive, generative, and stratified models of probabilities. We will elaborate on the differences between these models in the next section.

testing. The first *probabilistic process algebra* appears one year later. In [GJS90] a probabilistic version of SCCS [Mil83], called PCCS, is presented. They replace the (binary) non-deterministic choice operator of SCCS by a (n-ary) probabilistic choice operator, so that they use a syntax as $\sum_{i\in I}[p_i]P_i$. In this case, the probabilities p_i are assigned to the actions offered by the corresponding process P_i. They present a system of equational rules for PCCS based on the strong bisimulation defined in [LS89].

The aim of this paper is to be useful as an *index* covering the main contributions related to models of probabilistic processes and their equivalences. In the first case, we will concentrate both on probabilistic extensions of process algebras and on those probabilizations of labelled transition systems such that there exists a straight translation into a process algebraic notation. In the latter case we will only consider probabilistic extensions of bisimulations and testing semantics. Let us clearly state that this paper does not try to be self-contained in the sense that all the models are completely presented. On the contrary, we have omitted most of the technicalities so that the reader can get an overview of probabilistic models without getting lost in technical details. That is, technical concepts will be given only in the case that they facilitate the understanding of the notions. Thus, it can be said that in the trade-off between *descriptiveness* and *formality* we have preferred to choose the first one. Nevertheless, the reader who desires to acquire a more thorough knowledge of some of the models is pointed to the original papers. Moreover, there are topics that would definitevely fit in our study but that, in order to keep a reasonable length of the paper, we could not include. Most notably, we may remark the study of the probabilistic counterparts of simulation relations (e.g. [SL94,SL95,SV99]). Besides, we mainly focus on the different possibilities for choice operators while other operators are set aside (in Section 6, we review some of the proposals for parallel and hiding operators).

In order to get a first contact with probabilistic process algebras, next we present a simple language and its operational semantics. Despite its simplicity, most of the models presented in this paper are based on variations of this language. In particular, so-called fully probabilistic models can be described by languages similar to this one. As usual, Act denotes the set of visible actions that processes may perform. We also consider a special action $\tau \notin Act$ that represents internal behavior of processes. The set $Act \cup \{\tau\}$ is denoted by Act_τ. The set of probabilistic processes, denoted by $Proc$, is given by the following EBNF expression:

$$P ::= X \mid \mathrm{stop} \mid \alpha\,;P \mid P +_p P \mid \mathrm{rec}X.P$$

where X is a process variable, $\alpha \in Act_\tau$, and $p \in (0,1)$. The term stop represents a process that may perform no action, that is, a deadlocked process. The term $\alpha\,;P$ represents the process that performs the action α and then it behaves as P. $P +_p Q$ denotes the probabilistic choice between P and Q. Intuitively, p (resp. $1-p$) denotes the *weight* assigned to the actions that P (resp. Q) may perform. The idea is that the probabilities with which one of the components, for example P, performs actions are multiplied by a probability taken from the choice operator (in this case p). Let us note that we do not consider *extreme*

$$\frac{}{\alpha;P \xrightarrow{\alpha}_1 P} \qquad \frac{P \xrightarrow{\alpha}_q P'}{P +_p Q \xrightarrow{\alpha}_{p\cdot q} P'} \qquad \frac{Q \xrightarrow{\alpha}_q Q'}{P +_p Q \xrightarrow{\alpha}_{(1-p)\cdot q} Q'} \qquad \frac{P \xrightarrow{\alpha}_p P'}{\mathrm{rec}X.P \xrightarrow{\alpha}_p P'[\mathrm{rec}X.P/X]}$$

Fig. 1. Operational semantics of a basic process algebra.

values of probabilities, that is, the parameter associated with a probabilistic choice fulfills $0 < p < 1$. If we would allow, for example, $p = 1$ we would obtain a notion of priority. Such models are complex (even more if probabilities and priorities are mixed) and they are out of the scope of this paper (see [CLN01] for an overview of the inclusion of priority in process algebras). Finally, the term $\mathrm{rec}X.P$ is used to define (possibly) recursive processes. During the rest of the paper we will omit trailing occurrences of stop. For example, we will write b instead of $b\,;\mathrm{stop}$.

In Figure 1 we present the operational semantics of our language. The intuitive meaning of a transition as $P \xrightarrow{\alpha}_p P'$ is that the process P may perform the action α with probability p. After this action is performed, the process behaves as P'. Let us note that, despite the probabilities labelling transitions, the operational rules are similar to those for the corresponding subset of CCS. The first rule defines the behavior of the prefix operator: The term $\alpha\,;P$ performs α with probability 1 and after that it behaves as P. The last rule in Figure 1 represents the usual definition for the behavior of recursive processes. The remaining two rules describe the behavior of the choice operator. If the left hand side of a choice $P +_p Q$ (that is, the process P) can perform an action with a certain probability q, the choice is resolved and the action is performed with probability $p\cdot q$. A similar situation appears for the right hand side of the choice. In this case the corresponding probabilities associated with actions of Q are multiplied by $1 - p$. This simple definition of the choice operator considers that the choice is resolved only by using its probabilistic parameter. In other words, the operator $+_p$ behaves as a pure (probabilistic) internal choice. For instance, we have that the processes $a\,;P +_p \mathrm{stop}$ and $a\,;P$ are not equivalent. In order to achieve such equivalence we need to include a *normalization* factor. Let us consider the function *live* defined as:

$$\begin{array}{ll} \mathit{live}(\mathrm{stop}) = \mathit{live}(X) = 0 & \mathit{live}(a\,;P) = 1 \\ \mathit{live}(P +_p Q) = \max(\mathit{live}(P), \mathit{live}(Q)) & \mathit{live}(\mathrm{rec}X.P) = \mathit{live}(P) \end{array}$$

It is easy to check that $\mathit{live}(P) = 1$ iff P is able to perform, at least, a transition. Then, the rules for the choice operators have to be modified accordingly:

$$\frac{P \xrightarrow{\alpha}_q P'}{P +_p Q \xrightarrow{\alpha}_{q\cdot \frac{p}{p+(1-p)\cdot \mathit{live}(Q)}} P'} \qquad \frac{Q \xrightarrow{\alpha}_q Q'}{P +_p Q \xrightarrow{\alpha}_{q\cdot \frac{1-p}{p\cdot \mathit{live}(P)+(1-p)}} Q'}$$

One of the classical problems when defining an operational semantics for probabilistic process algebras consists in taking into account different occurrences of the same transition. For example, consider the process $P = a +_{\frac{1}{2}} a$. If we were not careful, we would have the transition $P \xrightarrow{a} {}_{\frac{1}{2}}\mathsf{stop}$ only once, while we should have this transition twice. Some approaches to solve this problem are to index transitions (e.g. [GSS95]), to increase the number of rules (e.g. [LS91]), to define a transition probability function (e.g. [SS00]), to add the probabilities associated with the same transition (e.g. [YL92]), or to consider that if a transition can be derived in several ways then each derivation generates a different instance (e.g. [NdFL95]). In the last case, multisets of transitions are considered instead of sets of transitions.

The rest of the paper is structured as follows. Section 2 is devoted to show the different interpretations of probabilistic information in process algebras. Section 3 describes the different possibilities to add probabilities to the choice operator(s). In Section 4, the most significant examples of *strong* and *weak bisimulation* definitions will be given. Meanwhile, Section 5 is devoted to review the main contributions on probabilistic *testing semantics*. In Section 6 we sketch some of the proposals for including parallel and hiding operators in probabilistic process algebras. Finally, in Section 7 we present our conclusions.

2 Interpretation of Probabilities

As we have already commented, one of the design decisions when defining a probabilistic process algebra is the way probabilistic information is related to usual actions. In [GSST90,GSS95] van Glabbeek et al. present three models of probabilistic processes based on the language PCCS. They are called *reactive*, *generative*, and *stratified*. While the first two models have been widely used when defining probabilistic processes, the stratified model has received less attention (maybe due to its inherent complexity). Actually, all the proposals analyzed in this paper take (somehow) either the reactive or the generative interpretation. We will introduce these models in terms of the actions that can be offered by the environment.

The *reactive model* was already introduced by Larsen and Skou in [LS89] for labelled transition systems. In [GSST90], however, this interpretation of probabilities is considered in the context of probabilistic process algebras. A process *reacts* to the stimuli given by its environment, that is, only one action is offered by the environment. So, the process chooses among the actions of that type taking into account their associated probabilities. Thus, if we consider a process as

$$P = \left[\frac{1}{5}\right] a\,;P_1 + \left[\frac{4}{5}\right] a\,;P_2 + \left[\frac{1}{3}\right] b\,;Q_1 + \left[\frac{2}{3}\right] b\,;Q_2$$

and the environment offers the action a, then P will perform a and, after that, it will behave as P_1 with probability $\frac{1}{5}$ and as P_2 with probability $\frac{4}{5}$. Besides, if the environment offers b then P will perform it, and after that it will behave either as Q_1 or as Q_2, with probabilities $\frac{1}{3}$ and $\frac{2}{3}$, respectively. As this simple

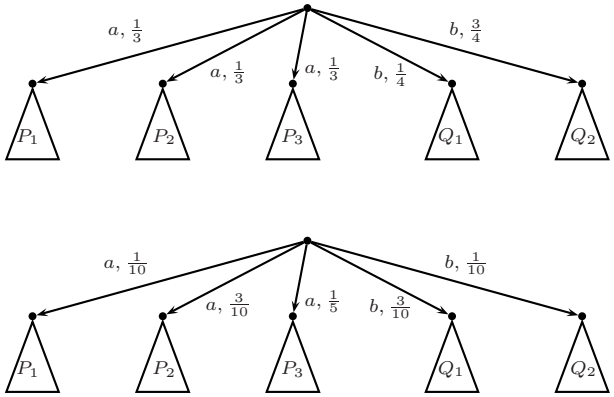


Fig. 2. Examples of a reactive process (top) and a generative process (bottom).

example shows, there is no probabilistic relation among different actions in the reactive model.

Intuitively, we say that a process as $P = \sum_{i \in I} [p_i] a_i \,;\, P_i$ is a *reactive process* if the following condition holds:

$$\forall \alpha \in Act \text{ we have } \sum \{\!\!\{\, p_i \mid a_i = \alpha \ \wedge \ i \in I \,\}\!\!\} \in \{0, 1\}$$

That is, the (sum of the) probability associated with each action of the alphabet is equal either to 0 or to 1. In Figure 2 a graphical example of a reactive process is given.[3]

In the *generative model* the environment may simultaneously offer several actions and the process makes a choice among them, by considering the probability distribution assigned to these actions. In this model, there is a probabilistic relation among all the actions. For instance, an example of generative process is given by

$$P = \left[\frac{1}{6}\right] a \,;\, P_1 + \left[\frac{1}{3}\right] a \,;\, P_2 + \left[\frac{1}{2}\right] b \,;\, P_3$$

Let us remark that in this case the sum of the probabilities associated with all the actions is equal to 1. However, there is a *redistribution* of probabilities among those actions that are offered by the environment. For instance, let us suppose

[3] Let us remark that the reactive model, as explained before, allows only a unique probabilistic distribution for each type of action, that is, a unique *bundle* for each type of action. However, there exist other reactive-based formalisms where this constraint is relaxed (e.g. [SL94] allows several bundles for a given type of action, being the choice between these bundles non-deterministically resolved).

that the environment offers only the action a.[4] Then, after performing a (with probability 1) the process P behaves as P_1 with probability $\frac{\frac{1}{6}}{\frac{1}{6}+\frac{1}{3}} = \frac{1}{3}$ and as P_2 with probability $\frac{\frac{1}{3}}{\frac{1}{6}+\frac{1}{3}} = \frac{2}{3}$. Let us suppose now that the environment offers both a and b. Then P performs a with a total probability equal to $\frac{1}{2}$ and b with the same probability. Afterwards, it behaves as P_1 with probability $\frac{1}{6}$, as P_2 with probability $\frac{1}{3}$, and as P_3 with probability $\frac{1}{2}$.

The syntax for *generative processes* can be given as $\sum_{i\in I}[p_i]a_i\ ;\ P_i$ where $\sum_{i\in I} p_i = 1$. An example of generative process is given in Figure 2 (bottom). The main difference with respect to the reactive model is that in the generative model there is a unique probability distribution among all the different action types.

In the *stratified* model the interpretation of probabilities is similar to the generative one but the probabilistic branching is kept, that is, the redistribution of probabilities is made *locally*. We will illustrate this difference by means of a simple example.

Example 1. Let us consider the process $P = [\frac{1}{2}]a + [\frac{1}{2}]([\frac{1}{2}]b + [\frac{1}{2}]c)$, and let us suppose that the occurrences of c are restricted in P, that is, let $P' = P\backslash\{c\}$. In the generative model, we have that P' will perform a with probability $\frac{\frac{1}{2}}{\frac{1}{4}+\frac{1}{2}} = \frac{2}{3}$. That is, the probability associated with c is proportionally distributed between the offered actions, a and b. In contrast, by using a stratified interpretation, we have that P' performs a with probability $\frac{1}{2}$ while b is also performed with probability $\frac{1}{2}$. This is so because the probability associated with c is *given* only to b. In Figure 3 we show a graphical representation of P in both the generative and stratified models. □

Even though most of the models for probabilistic processes are based on one of the previous interpretations of probabilities, there are a few proposals combining more than one of these interpretations. This is the case of the probabilistic extension of I/O automata [LT87] presented in [WSS94,WSS97] and recently formalized in a process algebraic style in [SCS03]. The idea is that input actions can be controlled by the environment, that is, the environment may decide which action is performed. Thus, it is more appropriate to give a probability distribution for each of the input actions. In other words, the model is reactive for input actions. On the contrary, output and internal actions cannot be controlled by the environment. So, a unique probability distribution is appropriate to resolve the choice among these actions. That is, the model is generative for output and internal actions. In addition, there is no probabilistic relation either among different input actions or between them and output/internal actions. The process will non-deterministically decide whether it performs either the input provided by the environment or an output or internal action. It is worth to point out that an exponentially timed delay is associated with each state of the model in order

[4] Actually, this is equivalent to consider that the action b has been *restricted*, that is, a restriction operator with parameter $\{b\}$ has been applied to P.

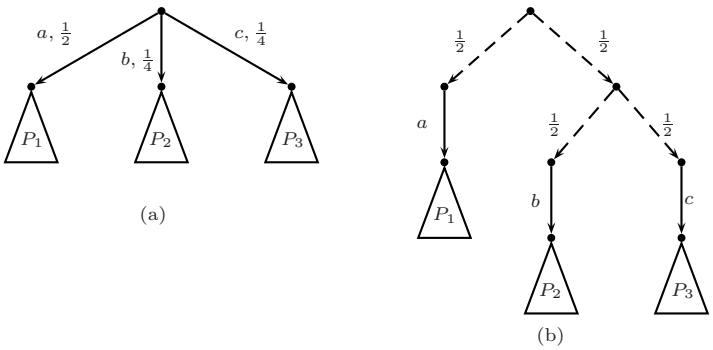


Fig. 3. Interpretation of $P = [\frac{1}{2}]a + [\frac{1}{2}]([\frac{1}{2}]b + [\frac{1}{2}]c)$ in the generative model (a) and in the stratified model (b).

to avoid the inclusion of non-determinism within input/output actions in the context of the parallel operator. A similar combination of probabilistic models appears in [BA01,BA03] to define an algebraic model where processes with different advancing speeds are modelled. This advancing speed is achieved by using a probabilistic parallel operator where, in contrast with the previous model, time delays are not needed. In general, generative-reactive models present some very interesting properties. For example, if we consider a complete system where all the inputs are resolved, then a fully probabilistic system from the specification of the complete system is obtained. That is, non-determinism is used just to express a clear form of internal system control via input/output synchronization.

3 The Choice of Probabilistic Choices

In this section we briefly discuss how the choice among actions should be modeled. Let us remark that this question has not a standard answer even in the non-probabilistic setting. Most of the models have decided to choose either the CCS [Mil89] or the CSP [Hoa85] approaches. In CSP there is an external choice, denoted in this paper by $\Box$, and there is also an internal choice, denoted by $\oplus$. The difference between these two operators is that while an external choice must be resolved by a decision taken from the environment (in other words, a visible action) internal choices are non-deterministically resolved. Thus, internal transitions do not resolve external choices. For instance, a process as $a \,\Box\, (b \oplus c)$ can internally evolve either into $a \Box b$ or into $a \Box c$. The unique choice operator of CCS, denoted by $+$, is a mixture between external and internal choices. In this case, internal actions do resolve choices. For example, the process $(a\,;c) + (\tau\,;b)$ evolves after performing an internal transition into the process b.

It is worth to point out that, in the non-probabilistic setting, the choice operators of CSP can be (more or less) simulated with the ones from CCS, and

vice versa. For example, the CSP process $a \oplus b$ can be expressed in CCS as the process $\tau\,;a+\tau\,;b$. Thus, the process $(a \oplus b) \,\Box\, c$ can be translated into CCS as $\tau;(a+c)+\tau;(b+c)$. Processes in CCS as $a+b$ and $\tau;a+\tau;b$ can be simulated by processes in CSP as $a \,\Box\, b$ and $a \oplus b$, respectively. *Mixed choices*, that is choices between visible and internal actions, can also be described in CSP, but in this case we need to use the hiding operator. Let us consider the CCS process $a+\tau;b$. This process can be translated into CSP as $(a \,\Box\, c\,;b)\backslash\{c\}$, where $P\backslash\{c\}$ means that all the occurrences of c in P are hidden, that is, they are transformed into internal actions.

Unfortunately, in a probabilistic setting the choice of choice operators is highly relevant because the previous simulations are not that easy. Most of the probabilistic extensions appeared in the literature are based on the ideas underlying CCS. That is, the most used notation is similar to the process algebra that we were describing in the introduction of the paper. However, there are other CCS-like models combining probabilistic choice operators and non-probabilistic ones. This is the case, for example, of the process algebra described in [YL92]. This model has two choice operators: The usual (non-probabilistic) CCS choice operator and a *purely* probabilistic choice operator. This last operator is the probabilization of the CSP internal choice operator, that is, the choice is resolved in a purely probabilistic manner.

Extensions based in CSP are more scarce. In [Low95,NdFL95] both CSP choice operators are extended with probabilities. A probabilistic external choice operator behaves as the probabilistic CCS choice operator if internal actions are not involved in the choice. A probabilistic internal choice as $P \oplus_p Q$ is equivalent to the probabilistic CCS process $\tau\,;a +_p \tau\,;b$. In the case of CSP there are also proposals including both probabilistic and non-deterministic choices. In [Sei95] the external choice operator is non-probabilistic but it is parameterized by a set of traces. This parameter must be considered as a *scheduler* to resolve non-deterministic choices. In [CdFV97] the authors introduce a language combining a (non-probabilistic) external choice and a probabilistic internal choice. Finally, in [CCVP01,CCV$^+$03] they considered both (non-probabilistic) CSP choice operators as well as a probabilistic internal choice.

Even more important than the choice operators that a language is using, it is to consider whether all the choices are (probabilistically) quantified. Actually, this is the main distinction that we will use during the rest of the paper. That is, probabilistic models can be roughly classified in the following two groups:

- *Fully Probabilistic Models.* In the case of work following these models we have that choices are always quantified. Intuitively, the underlying (probabilistic) transition systems have transitions labelled by both an action and a probability, while the interpretation of probabilities is generative. Examples of fully probabilistic models have appeared in [CSZ92,NdFL95,BH97,CDSY99]. Finally, let us remark that the process algebra defined in the introduction of this paper is a basic language to describe this type of processes. We will use this fact during the rest of the paper when introducing semantic frameworks for fully probabilistic models.

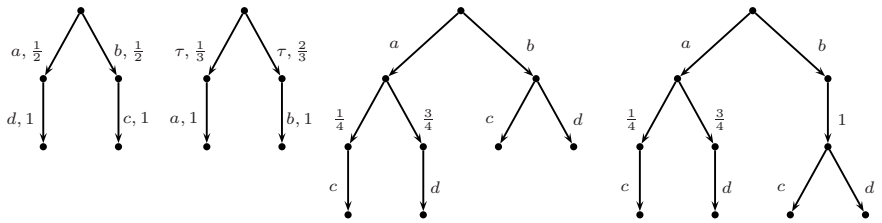


Fig. 4. Fully probabilistic and non-determinism probabilistic processes.

– *Non-deterministic Probabilistic Models.* In these models some choices are quantified (usually by means of a purely probabilistic internal choice) while there are also non-deterministic choice operators. Thus, the reactive interpretation of probabilities as well as its variations fall into this category. Intuitively, the underlying (probabilistic) transition systems have transitions labelled either by an action or by a probability. In this case, we have a further distinction between *alternating models* (where there is a strict alternation of probabilistic and action transitions) and *non-alternating models*. Examples of studies where a non-deterministic probabilistic model has been used are [HJ89,YL92,SL95,PLS00,BS01].

In Figure 4 some examples of fully probabilistic processes (the first two ones) and non-deterministic probabilistic processes (the other two processes) are given. From now on we consider that $\xrightarrow{\alpha}_p$, with $\alpha \in Act_\tau$ and $p \in (0, 1]$, denotes a transition of a fully probabilistic process. While fully probabilistic models can be, more or less, represented by using the process algebra presented in the introduction of the paper, we need to introduce some notation to deal with non-deterministic probabilistic models. In the rest of this section we will go through the main differences between the alternating and the non-alternating models. We will also present definitions of basic process algebras for both models. In the following we will consider that in the context of non-determinism probabilistic processes the transition $\xrightarrow{\alpha}$ denotes a non-determinism transition performing action α while the transition $\longrightarrow_p$ denotes a probabilistic transition performed with probability p.

3.1 Alternating model

Two types of processes can be distinguished in an alternating model: *Probabilistic* processes and *non-deterministic* processes. Besides, there usually exists two types of transitions: *Probabilistic* transitions, outgoing from probabilistic states, labelled with probabilities, reaching non-deterministic processes, and *non-deterministic* transitions, outgoing from non-deterministic states, labelled with actions, reaching probabilistic processes.

$$\frac{}{\bigoplus_{i\in I}[p_i]N_i \longrightarrow_{p_j} N_j} \qquad \frac{}{\sum_{i\in I}\alpha_i;P_i \xrightarrow{\alpha_j} P_j}$$

$$\frac{P \longrightarrow_p N}{\mathrm{rec}X.P \longrightarrow_p N[\mathrm{rec}X.P/X]} \qquad \frac{N \xrightarrow{\alpha} P}{\mathrm{rec}X.N \xrightarrow{\alpha} P[\mathrm{rec}X.N/X]}$$

Fig. 5. Operational semantics for an alternating basic language.

Definition 1. Let $\mathcal{P}_N$ be the set of *non-deterministic* process expressions, ranging over $N, N', \ldots$, and let $\mathcal{P}_P$ be the set of *probabilistic* process expressions, ranging over $P, Q, \ldots$. We denote by *Proc* the set of processes in $\mathcal{P}_N \cup \mathcal{P}_P$, ranging over $G, G', \ldots$ The syntax for *probabilistic processes* is given by the EBNF expression:

$$P ::= \mathrm{stop} \mid X \mid \bigoplus_{i\in I} [p_i]N_i \mid \mathrm{rec}X.P$$

The syntax for *non-deterministic processes* is given by the EBNF expression:

$$N ::= \mathrm{stop} \mid X \mid \sum_{i\in I} \alpha_i \,;\, P_i \mid \mathrm{rec}X.N$$

where $p_i \in [0,1]$, $\sum_{i\in I} p_i = 1$, X is a process variable, and I is a set of indexes. The probabilistic operator $\bigoplus$ represents a pure probabilistic (internal) choice, while $\sum$ represents a non-deterministic choice. Finally, rec$X.G$ is used to define recursive processes. □

The operational behavior of this process algebra is given in Figure 5. The left hand side rules describe probabilistic behaviors, while the right hand side rules describe non-deterministic behaviors. The first rule shows how a probabilistic choice is resolved by performing a probabilistic transition. In the second rule, non-deterministic choices are resolved by performing an action. Finally, the last two rules express the behavior of the recursive processes. Examples of studies using alternating models are given in [HJ89,BS01].

3.2 Non-Alternating Model

As in the previous case, we have to consider the same two types of processes and transitions: *Probabilistic* and *non-deterministic*. The main difference among non-alternating models is the kind of states reached after a transition. For example, in the particular basic process algebra described below we consider that probabilistic transitions lead to non-deterministic states, while non-deterministic transitions may reach any kind of states. Thus, the syntax of our process algebra is defined as follows:

Definition 2. Let $\mathcal{P}_N$ be the set of non-deterministic process expressions ranging over $N, N', \ldots$, and let $\mathcal{P}_P$ be the set of probabilistic process expressions,

$$\frac{}{\oplus_{i\in I}[p_i]N_i \longrightarrow_{p_j} N_j} \qquad \frac{}{\sum_{i\in I}\alpha_i;G_i \xrightarrow{\alpha_j} G_j}$$

$$\frac{P \longrightarrow_p N}{\mathrm{rec}X.P \longrightarrow_p N[\mathrm{rec}X.P/X]} \qquad \frac{N \xrightarrow{\alpha} G}{\mathrm{rec}X.N \xrightarrow{\alpha} G[\mathrm{rec}X.N/X]}$$

Fig. 6. Operational semantics for a non-alternating basic language.

ranging over $P, Q, \ldots$. We denote by *Proc* the set of processes in $\mathcal{P}_N \cup \mathcal{P}_P$, ranging over $G, G', \ldots$ The syntax for *probabilistic processes* is given by the EBNF expression:

$$P ::= \mathrm{stop} \mid X \mid \bigoplus_{i\in I} [p_i]N_i \mid \mathrm{rec}X.P$$

The syntax for *non-deterministic processes* is given by the EBNF expression:

$$N ::= \mathrm{stop} \mid X \mid \sum_{i\in I} \alpha_i\,;G_i \mid \mathrm{rec}X.N$$

where $p_i \in [0,1]$, $\sum_{i\in I} p_i = 1$, X is a process variable, and I is a set of indexes. The probabilistic operator $\bigoplus$ represents a pure probabilistic (internal) choice, while $\sum$ represents a non-deterministic choice. As usually, rec$X.G$ defines a recursive process. □

The operational semantics of this process algebra appears in Figure 6. Examples of languages following a non-alternating model are [YL92,SL94,PLS00].

4 Bisimulation Semantics

In this section we will review some of the proposals for probabilistic extensions of the classical notions of strong and weak bisimulation. The definition of these two kinds of equivalence for fully probabilistic processes and for non-determinism probabilistic processes differ in several points. Thus, we will present them separately.

4.1 Probabilistic Strong Bisimulation

The definition of strong bisimulation for non-probabilistic processes is very intuitive because it represents the idea of a game of *imitations*. That is, if one process performs an action then the other one has to imitate it by performing the same action. Afterwards, the new processes have to be able to imitate each other, and so on.

Definition 3. Let *Proc* be a set of (non-probabilistic) processes. We say that an equivalence relation $\mathcal{R}$ is a *strong bisimulation* on *Proc* iff for any pair of processes $P, Q \in Proc$ we have that $P\mathcal{R}Q$ implies

$$\forall \alpha \in Act_\tau \text{ we have } P \xrightarrow{\alpha} P' \text{ implies } \exists Q' : Q \xrightarrow{\alpha} Q' \ \wedge \ P'\mathcal{R}Q'$$

We say that two processes $P, Q \in Proc$ are *strongly bisimilar*, denoted by $P \sim Q$, if there exists a strong bisimulation that contains the pair (P, Q). □

In the probabilistic setting a new problem appears: The probabilities of reaching an equivalence class have to be computed. In the non-probabilistic setting we simply require $P \xrightarrow{a} P'$ implies $Q \xrightarrow{a} Q'$. That is, it is enough that the second *player* has a possibility to imitate the step of the first player. If we work with probabilistic processes, in addition to require that the second process is able to imitate the first one, we also need that it does it with the same probability. In other words, we have to consider all the possible ways to imitate the execution of the action and we have to add the probabilities associated with these possibilities. Moreover, we have to consider all the probabilities associated with evolutions into equivalent processes.

As we have already mentioned the first definition of (strong) probabilistic bisimulation was given in [LS89]. In that work, Larsen and Skou introduce a notion of strong bisimulation for probabilistic processes considering that two processes are bisimilar if they perform the same actions with the same *cumulative* probability. In [HJ90] this notion is adapted to the alternating model while [SL94] also considers this equivalence for the case of the non-alternating model.

Strong Bisimulation for Fully Probabilistic Processes The definition of probabilistic strong bisimulation has not varied since it was firstly introduced in [LS89,LS91]. It is worth to note that this notion of bisimulation is based on the corresponding notion of *lumpability* initially introduced for Markov chains (see e.g. [KS76]).

Definition 4. Let *Proc* be the set of fully probabilistic processes defined by the EBNF given in the introduction of this paper. For any equivalence relation $\mathcal{R}$ on *Proc* we denote by $Proc/\mathcal{R}$ the set of equivalence classes induced by $\mathcal{R}$. We say that an equivalence relation $\mathcal{R}$ is a *strong bisimulation* on *Proc* iff for any pair of processes $P, Q \in Proc$ we have that $P\mathcal{R}Q$ implies

$$\forall \alpha \in Act_\tau,\ C \in Proc/\mathcal{R} \text{ we have } Prob(P, \alpha, C) = Prob(Q, \alpha, C)$$

where $Prob(P, \alpha, C) = \sum \{\!|\ p \mid P \xrightarrow{\alpha}_p P' \ \wedge \ P' \in C\ |\!\}$. We say that two probabilistic processes $P, Q \in Proc_p$ are *strongly bisimilar*, denoted by $P \sim Q$, if there exists a strong bisimulation that contains the pair (P, Q). □

Intuitively, $Prob(P, \alpha, C)$ computes the probability associated with those transitions leaving from P, labelled by α, and reaching a process belonging to the set C. For example, in the left hand side of Figure 7 we can see that if P_2

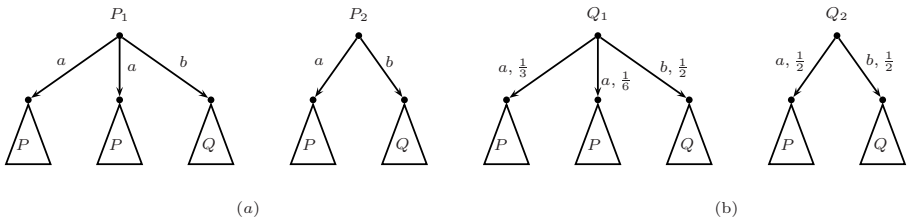


Fig. 7. Strongly bisimilar processes, in the non-probabilistic (a) and probabilistic (b) settings.

performs a then P_1 can perform any of its a-transitions to imitate it. However, in the probabilistic case (right hand side of Figure 7) the situation is slightly different. If Q_2 performs a with probability $\frac{1}{2}$ then the (sum of the) probabilities of both a-transitions in Q_1 have to be considered in order to imitate the movement performed by Q_2.

Strong Bisimulation for Non-deterministic Probabilistic Processes As a representative case we consider the notion of strong bisimulation introduced in [HJ90] for the alternating model. Their language is defined in two steps. First, probabilistic information is included. Then, this language is extended with temporal information. As we are only interested in probabilistic languages, we will limit our comments to the probabilistic features. Actually, the syntax and operational semantics of the corresponding probabilistic sub-language are already given in the previous section while describing a basic alternating process algebra (see Definition 1).

In order to define a notion of probabilistic bisimulation, the considerations about adding probabilities that were commented for the fully probabilistic case must also be taken into account.

Definition 5. Let $P \in \mathit{Proc}$ be a process. For any set of non-deterministic processes $C \subseteq \mathcal{P}_N$ the *cumulative distribution function* of P to reach C, denoted by $\mu(P, C)$, is defined as

$$\mu(P, C) = \sum \{\!\{\, p \mid P \longrightarrow_p N \ \wedge\ N \in C \,\}\!\}$$

□

Intuitively, $\mu(P, C)$ computes the probability of reaching a process in C from P after performing a probabilistic transition.

Definition 6. Let $\mathcal{R}$ be an equivalence relation on $\mathcal{P}_N$. We say that $\mathcal{R}$ is an *alternating bisimulation* iff for any pair of processes $N_1, N_2 \in \mathcal{P}_N$ we have that $N_1 \mathcal{R} N_2$ implies

$$\forall \alpha \in \mathit{Act}_\tau \text{ we have } N_1 \xrightarrow{\alpha} P_1 \text{ implies } \exists P_2 : N_2 \xrightarrow{\alpha} P_2 \wedge \forall\, C \in \mathcal{P}_N/\mathcal{R} :\ \mu(P_1, C) = \mu(P_2, C)$$

We say that two processes $N_1, N_2 \in \mathcal{P}_N$ are *alternating bisimulation equivalent*, denoted by $N_1 \sim N_2$, if there exists an alternating bisimulation that contains the pair (N_1, N_2). □

Intuitively, for non-deterministic processes their strong bisimulation behaves as the classical notion while for probabilistic processes the probabilities are computed following [LS89].

4.2 Probabilistic Weak Bisimulation

The imitation metaphor that we were using for strong bisimulation can be also applied to weak bisimulation. The main difference consists in how internal movements are imitated. In the case of strong bisimulation, an internal movement of the first player has to be imitated by exactly one internal movement of the second player. In the case of weak bisimulation, an internal movement can be imitated by any number (including zero) of internal movements. Moreover, in order to imitate the performance of a visible action, the second process may use as many internal transitions as needed.

Definition 7. Let *Proc* be a set of (non-probabilistic) processes. We say that an equivalence relation $\mathcal{R}$ is a *weak bisimulation* on *Proc* iff for any pair of processes $P, Q \in Proc$ we have that $P\mathcal{R}Q$ implies

$$\forall \alpha \in Act_\tau \text{ we have } P \xrightarrow{\alpha} P' \text{ implies } \exists Q' : Q \stackrel{\alpha}{\Longrightarrow} Q' \ \wedge \ P'\mathcal{R}Q'$$

We say that two processes $P, Q \in Proc$ are *weakly bisimilar*, denoted by $P \approx Q$, if there exists a weak bisimulation that contains the pair (P, Q). □

In the previous definition, if $\alpha \in Act$ then the transition $\stackrel{\alpha}{\Longrightarrow}$ represents the sequence of transitions $\xrightarrow{\tau}{}^* \xrightarrow{\alpha} \xrightarrow{\tau}{}^*$, while we have that $\stackrel{\tau}{\Longrightarrow}$ denotes the sequence $\xrightarrow{\tau}{}^*$; in turn $\xrightarrow{\tau}{}^*$ represents the reflexive and transitive closure of $\xrightarrow{\tau}$, that is, a (possibly empty) sequence of internal transitions.

In contrast with the probabilistic extension of strong bisimulation, there is no common agreement about what a *good* definition of probabilistic weak bisimulation is. Actually, it is worth to point out that a similar situation appears in the non-probabilistic case where several alternative *weak* notions of bisimulation have appeared (see e.g. [MS92,GW96,dFLN99]). In the rest of this section we introduce some of the definitions that have appeared in the probabilistic setting. As we did for strong bisimulation, we split the presentation in two parts: Fully probabilistic processes and non-deterministic probabilistic processes.

Weak Bisimulation for Fully Probabilistic Processes The first proposal for fully probabilistic processes of a probabilistic weak bisimulation does not appear until 1997 [BH97]. If we consider that probabilistic strong bisimulation was already introduced in 1989, we may see that it was not a trivial task to define an appropriate extension of weak bisimulation for probabilistic processes.

In [BH97] processes are described as *fully probabilistic transition systems*, that is, labelled transition systems where transitions are labelled by an action and a probability. Thus, these transition systems are equivalent to the ones induced by the simple process algebra presented in the introduction of this paper when describing a fully probabilistic process algebra. First, we need to consider an auxiliary function to compute the probability of performing a given action from a given process reaching another process.

Definition 8. Let *Proc* be the set of fully probabilistic processes, $P, Q \in Proc$, and $\alpha \in Act_\tau$. We define the probability of P to reach Q by performing α, denoted by $\mathrm{prob}(P, \alpha, Q)$, as

$$\mathrm{prob}(P, \alpha, Q) = \sum \{\!|\, p \mid P \xrightarrow{\alpha}_p Q \,|\!\}$$

□

Given that internal movements have to be (somehow) abstracted, it is also necessary to define the probability of reaching a set of processes after a sequence of actions is performed.

Definition 9. Let $P \in Proc$ be a process and $C \subseteq Proc$ be a set of processes. The *probability* to *reach* C from P by performing the action $\alpha \in Act_\tau$, denoted by $Prob(P, \alpha, C)$, is defined as $Prob(P, \alpha, C) = \sum_{Q \in C} \mathrm{prob}(P, \alpha, Q)$.

Let $P \in Proc$ be a process, $\gamma \subseteq Act_\tau^*$ be a set of sequences of actions, and $C \subseteq Proc$ be a set of processes. The *probability* of P to *reach* C *after* the sequences belonging to γ are performed, denoted by $Prob(P, \gamma, C)$, is defined as:

$$Prob(P, \gamma, C) = \begin{cases} 1 & \text{if } P \in C \ \wedge\ \epsilon \in \gamma \\ 0 & \text{if } P \notin C \ \wedge\ \epsilon \in \gamma \\ \displaystyle\sum_{(\alpha, Q) \in Act_\tau \times Proc} \mathrm{prob}(P, \alpha, Q) \cdot Prob(Q, \gamma', C) & \text{if } \gamma' = \gamma/\alpha \end{cases}$$

where ϵ denotes the empty trace and $\gamma/\alpha = \{\lambda \mid \alpha\lambda \in \gamma\}$. □

In order to define this notion of probabilistic weak bisimulation, it is necessary to compute the probability with which different sets of processes (actually, different equivalence classes) are reached after performing sequences as $\tau^* a \tau^*$. That is, if a process performs an external action a, possibly preceded and succeeded by sequences of internal actions, then the other one has to simulate the action a in the same way. Let us remind that in order to simulate an internal movement a process may perform no action at all.

Definition 10. An equivalence relation $\mathcal{R}$ is a *probabilistic weak bisimulation* iff for any pair of processes $P, Q \in Proc$ we have that $P \mathcal{R} Q$ implies:

$$\forall \alpha \in Act \cup \{\epsilon\},\ C \in Proc/\mathcal{R} \text{ we have } Prob(P, \tau^* \alpha \tau^*, C) = Prob(Q, \tau^* \alpha \tau^*, C)$$

where we consider $\tau^* \epsilon \tau^* = \tau^*$. We say that two processes $P, Q \in Proc$ are *probabilistically weakly bisimilar*, denoted by $P \approx Q$, if there exists a weak bisimulation that contains the pair (P, Q). □

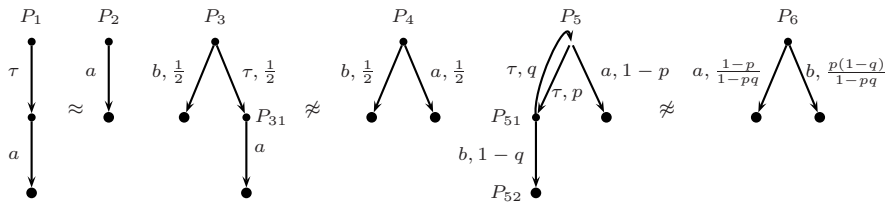


Fig. 8. Examples of weakly bisimilar and not bisimilar processes.

In Figure 8 we present some examples illustrating this notion of weak bisimulation. The first pair of processes P_1 and P_2 shows that the processes $\tau\,;a$ and a are weakly equivalent. However, the processes P_3 and P_4 are not equivalent. The idea is that abstracting internal transitions is not the same as deleting them. If P_3 performs its τ transition then the process P_{31} is reached. At that point, P_4 cannot simulate the movement performed by P_3. Actually, P_4 may try to *simulate* this movement by performing no transitions. However, P_4 is not equivalent to P_{31} because the former may perform, with a probability greater than zero, the action b while the latter cannot. Let us note that even if we delete probabilities, these two processes are not (non-probabilistic) weakly bisimilar.

Let us consider now the processes P_5 and P_6. They are not equivalent for the notion introduced in [BH97]. In order to show this non-equivalence, let us note that these two processes would be equivalent only if they were also equivalent to the process P_{51}. But this is not the case since, for example, the probabilities of reaching the (class of the) process P_{52} from P_5 and P_{51} are not equal. In the first case we obtain that this probability t_b is defined as $t_b = p \cdot (q \cdot t_b + 1 - q)$, that is, $t_b = \frac{p \cdot (1-q)}{1-p \cdot q}$. In the case of P_{51} we obtain $t'_b = (1-q) + q \cdot p \cdot t'_b$, that is, $t'_b = \frac{1-q}{1-p \cdot q}$. So, these states are not equivalent. A similar situation appears if we consider the action a. In this case we obtain for P_5 and P_{51} the values $t_a = \frac{1-p}{1-p \cdot q}$ and $t'_a = \frac{q \cdot (1-p)}{1-p \cdot q}$, respectively.

It has been sometimes argued that this notion of bisimulation is too *fine*, that is, it distinguishes processes that should be equivalent. As an example, it may be questionable that P_5 and P_6 are not equivalent. Based on this result, [AB01] proposes an alternative notion of weak bisimulation without considering intermediate processes in sequences of τ transitions. In particular, these two processes are equivalent. Intuitively, they consider that it is not possible to decide at a given moment whether we are in the process P_5 or in the process P_{51} because each other can be reached by performing internal actions. So, there are some internal processes that are not considered in order to decide whether two processes are equivalent. In this case, by taking into account the previously computed probabilities, we have that the probability of performing both a and b, possibly preceded by a sequence of internal actions, is the same for P_5 and P_6, while the corresponding probabilities for P_{51} are not computed. Thus, the idea

underlying [AB01] is that, in some cases, abstracting internal transitions is the same as (somehow) deleting them.

Weak Bisimulation for Non-deterministic Probabilistic Processes The model described in [BH97] has also been used, as a starting point, for models considering both probabilistic and non-deterministic choices. In [PLS00] a probabilistic weak bisimulation for a class of *labelled concurrent Markov chains* is introduced. Let us remark that this formalism can be easily translated into the language for the non-alternating model that we have introduced in Section 3.2. Thus, during the rest of this section we will consider that the set of processes *Proc* is the one presented in Definition 2.

In order to compute the probability with which a process performs a particular sequence of actions, it is necessary to resolve the non-determinism appearing in the process. For this reason they introduce the notion of *scheduler*. The idea is that for any partial computation (ending in a non-deterministic process) the scheduler chooses the next transition to be performed.

Before introducing their notion of weak bisimulation we will explain the role of two auxiliary functions. Given a process $G \in Proc$, $\gamma \in Act_\tau^*$, $C \subseteq Proc$, and a scheduler σ, the function $Prob(G, \gamma, C, \sigma)$ computes the cumulative probability to perform the sequence γ from G to reach any process in C with respect to σ. The main difference between this function and the one in [BH97] (Definition 9 in this survey) consists in the addition of a scheduler to resolve non-determinism. The formal definition of this function can be found in the original paper [PLS00]. Besides, the function $\mu_{\mathcal{R}}(G, C)$ computes the cumulative probability of reaching any state of C by performing only one probabilistic transition, but considering only the probabilities of leaving the equivalence class of G.

Definition 11. Let $\mathcal{R}$ be an equivalence relation, $G \in Proc$, and $C \subseteq Proc$. The function $\mu_{\mathcal{R}}(G, C)$ is defined as:

$$\mu_{\mathcal{R}}(G, C) = \begin{cases} \frac{\mu(G,C)}{1-\mu(G,[G]_{\mathcal{R}})} & \text{if } \mu(G, [G]_{\mathcal{R}}) \neq 1 \\ \\ \mu(G, C) & \text{otherwise} \end{cases}$$

where for any $G \in Proc$ and $C \subseteq Proc$ we have that $\mu(G, C) = \sum_{G' \in C} \mathrm{pr}(G, G')$. Additionally, the value $\mathrm{pr}(G, G')$ is defined as:

$$\mathrm{pr}(G, G') = \begin{cases} p & \text{if } G \longrightarrow_p G' \\ 1 & \text{if } G = G' \ \wedge \ G \in \mathcal{P}_N \\ 0 & \text{otherwise} \end{cases}$$

□

Definition 12. Let $Proc = \mathcal{P}_N \cup \mathcal{P}_P$ be the set of processes, where $\mathcal{P}_N$ denotes the set of non-deterministic processes and $\mathcal{P}_P$ denotes the set of probabilistic processes. An equivalence relation $\mathcal{R} \in Proc \times Proc$ is a *weak bisimulation* if for any pair of processes $G, G' \in Proc$ whenever $G \mathcal{R} G'$ we have

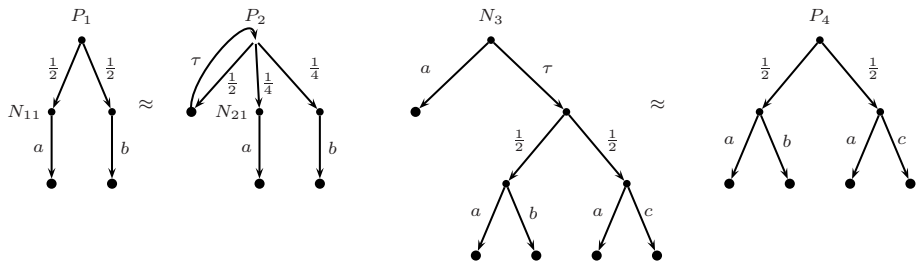


Fig. 9. Examples of weakly bisimilar processes

- $\forall \alpha \in Act_\tau$, if $G, G' \in \mathcal{P}_N$ and $G \xrightarrow{\alpha} G'$ then there exists a scheduler σ such that $Prob(G', \tau^* \overline{\alpha} \tau^*, [G']_{\mathcal{R}}, \sigma) = 1$, where $\overline{\alpha}$ denotes α if $\alpha \in Act$ and the empty string ϵ if $\alpha = \tau$.
- There exists a scheduler σ such that

$$\forall\, C \in (Proc/\mathcal{R}) - [G]_{\mathcal{R}} :\ \mu_{\mathcal{R}}(G, C) = Prob(G', \tau^*, C, \sigma)$$

We say that the processes G and G' are *weakly bisimilar*, denoted by $G \approx G'$, if there exists a weak bisimulation that contains the pair (G, G'). □

Intuitively, an equivalence relation is a weak bisimulation if two conditions hold. First, for any pair of related non-deterministic processes, if one of them can perform a visible action then the other one imitates it by performing several internal actions before and after this action; if the performed action is internal then the other process can imitate it by performing any number of internal actions, in particular none. Second, for any probabilistic process the cumulative probability to reach a set C through probabilistic transitions is equal to the probability needed by the other process to reach the same set by performing τ actions and probabilistic transitions.

Example 2. Consider the processes depicted in Figure 9. We have that P_1 and P_2 are weakly bisimilar since the probability with which both processes reach the non-deterministic processes that are able to perform a, that is, N_{11} and N_{21} respectively, is the same. In particular, we have that $\mu_{\mathcal{R}}(P_1, [N_{11}]_{\mathcal{R}}) = \frac{1}{2}$ while $Prob(P_2, \tau^*, [N_{11}]_{\mathcal{R}}, \sigma) = \sum_{i=0}^{\infty} \frac{1}{4} \cdot \frac{1}{2^i} = \frac{1}{2}$, for the appropriate scheduler σ.

The processes N_3 and P_4 are also weakly bisimilar since N_3 can perform a non-deterministic transition labelled by a, and for a particular scheduler σ we have $Prob(P_4, \tau^* a \tau^*, C, \sigma) = \frac{1}{2} + \frac{1}{2} = 1$, where C contains the processes associated with the leaves of the tree. □

This model as well as the notion of weak bisimulation is also considered in [BS01]. They deal with both an alternating and a non-alternating model. Actually, the syntax of the languages for both models is the same. The difference

appears in the definition of the operational semantics. In the alternating model a process as $\alpha;P$ performs α and it reaches the probabilistic distribution of P after an internal step. On the contrary, in the non-alternating model the same process performs the action α and automatically reaches the probabilistic distribution of P. In the non-alternating model, their notions of strong and weak bisimulation turn to be equal to those introduced in [SL94]. Meanwhile, in the alternating model strong bisimulation is equivalent to the definition given in [HJ89] while weak bisimulation coincides with [PLS00]. Another study whose notion of weak bisimulation is also equivalent to the weak bisimulation defined in [PLS00], but in the alternating case, is presented in [DGJP02].

5 Testing Semantics

In the previous section we showed that bisimulation semantics can be interpreted in terms of a game of simulation. In the case of the classical theory of testing [dNH84,Hen88] we can give an intuitive interpretation as well. The idea consists in an experimenter testing a process with a test. Given two processes, if there does not exist a test which returns different results for these two processes, then we say that they are equivalent. Depending on the way single tests are carried out, different testing semantics can be defined. Next we briefly sketch the different possibilities to define such a testing semantics. In order to test a process we consider the different interactions between the tested process and a set of tests. This interaction is usually modelled by the parallel composition of the process and the test. A test is (essentially) defined as a process but it may contain a special action, denoted by ω, indicating the successful termination of the testing procedure. If the interaction between a process and a test leads the test to a state where ω can be performed, we say that this computation is *successful*. Thus, if we consider all the possible interactions (i.e. computations) between a process P and a test T we may have three results:

- All of the computations are successful, and we say P *must* pass T.
- Some of the computations are successful, and we say P *may* pass T.
- There are no successful computations, and we say P does not pass T.

The previous possibilities induce, according to [dNH84,Hen88] and depending on how two processes react to a set of tests, the following three testing equivalences.

- We say that two processes P and Q are *may* equivalent if for any test T we have P may pass T iff Q may pass T.
- We say that two processes P and Q are *must* equivalent if for any test T we have P must pass T iff Q must pass T.
- We say that two processes P and Q are *may-must* equivalent if for any test T we have both P may pass T iff Q may pass T and P must pass T iff Q must pass T.

It turns out that *may* equivalence identifies the same processes as trace equivalence does. Besides, if we consider non-divergent processes (that is, processes that cannot perform an infinite sequence of internal actions) we obtain that *must* equivalence is *equivalent* to the usual semantic model for CSP: Failure semantics. Finally, *may-must* equivalence is a little bit finer than *must* equivalence because it may partially *see* the behavior of a process in the case of a divergent behavior. For non-divergent processes, *must* and *may-must* equivalences identify the same processes. In addition to these equivalences, testing preorders can be defined in the usual way. For example, we may say that Q is *better* than P in the *may* sense if for any test T we have P may pass T implies Q may pass T. Finally, let us remark that several alternative notions of testing for non-probabilistic processes have appeared in the literature (e.g. [Hen87,Phi87,NC95,BRV95,dFLN97]).

In order to define probabilistic testing semantics, the main question that has to be answered is: *With which probability does a process pass a test?* Depending on the underlying probabilistic model we have two possibilities. In the case of fully probabilistic models, there is a well established mechanism: The probability with which a process passes a test is given by the sum of the probabilities associated with successful computations. Thus, two processes are equivalent if they pass with the same probability any test. If we consider non-deterministic probabilistic models, we do not obtain a unique probability for passing a test, because of the presence of non-deterministic choices that are not quantified. In this case we have several possibilities. For example, we may consider that two processes are equivalent if the minimum/maximum probabilities with which these processes pass any test are the same. Even though probabilistic testing semantics have not received so much attention as the probabilizations of bisimulation, there are also numerous proposals dealing with the topic (e.g. [Chr90,YL92,CSZ92,NdF95,NCS98,KN98]). In this section we will review the main concepts underlying the definition of these semantic frameworks.

5.1 Testing Semantics for Fully Probabilistic Processes

As we have explained before, testing semantics for fully probabilistic processes share a common pattern: We compute the (unique) probability with which a process passes a test. Nevertheless, depending on the characteristics that we are interested to analyze there exist more than a unique possibility to define an equivalence.

In [Chr90] three equivalences, as well as their corresponding denotational characterizations, are introduced. In that study, both processes and tests are defined in terms of labelled transitions systems. As usual, processes are identified with the initial state of the corresponding labelled transition system. However, processes include probabilistic information (by using a generative interpretation of probabilities) while tests are non-probabilistic and deterministic. In this approach, *testing systems* are defined as the parallel composition of a process and a test. In order to relate his syntax to the ones previously presented in this survey we have that his processes are similar to the probabilistic labelled transition

systems induced by the basic process algebra given in the introduction. Moreover, tests can be generated in the same way but replacing $+_p$ by a CCS choice operator where non-determinism is not allowed. Thus, the relevant choices are quantified because tests cannot introduce non-determinism.

Definition 13. Let *Proc* be the set of fully probabilistic processes, $\mathcal{T}$ the set of tests, P be a process, T be a test, and $\gamma \in Act^*$ be a sequence of visible actions. The *total probability of performing* γ starting from (P,T) is given by

$$\text{prob}((P,T),\gamma) = \sum_{(Q,T')\in Proc\times\mathcal{T}} \text{prob}'((P,T),\gamma,(Q,T'))$$

where $\text{prob}'((P,T),\gamma,(Q,T'))$ computes the probability of performing γ from (P,T) reaching (Q,T') and is defined as $Prob((P,T),\overline{\gamma},\{(Q,T')\})$ (see Definition 9). In addition, $\overline{\gamma}$ is defined as follows: $\overline{\epsilon} = \epsilon$, $\overline{a} = \tau^* a$, and $\overline{a\sigma} = \tau^* a\overline{\sigma}$. □

According to the different properties that the tester can be interested in, three partial orders are introduced.

- We write $P \leq_{tr} Q$ if for any sequential test T (that is, a trace of visible actions finishing with an acceptance action), and any trace of visible actions γ, we have that the probability of deadlock in the testing system conformed by Q and T after performing γ is less than or equal to the one for the testing system associated with P and T. Formally,

$$\forall T \in \mathcal{T}_{seq}, \gamma \in Act^* \text{ we have } \text{prob}((P,T),\gamma) \leq \text{prob}((Q,T),\gamma)$$

- We write $P \leq_{wte} Q$ if for any test T we have that after performing any sequence of visible actions the probability of deadlock, with respect to the visible actions that can be (weakly) performed in the next step, in the testing system conformed by Q and T is less than or equal to the one for the testing system associated with P and T. Formally,

$$\forall T \in \mathcal{T}, \gamma \in Act^* \text{ we have } \sum_{a\in Act} \frac{\text{prob}((P,T),\gamma a)}{\text{prob}((P,T),\gamma)} \leq \sum_{a\in Act} \frac{\text{prob}((Q,T),\gamma a)}{\text{prob}((Q,T),\gamma)}$$

- We write $P \leq_{ste} Q$ if for any test T we have that after performing any sequence of visible actions the probability of deadlock in the testing system conformed by Q and T is less than or equal to the one for the testing system associated with P and T. Formally,

$$\forall T \in \mathcal{T}, \gamma \in Act^* \text{ we have } \text{prob}((P,T),\gamma) \leq \text{prob}((Q,T),\gamma)$$

It is easy to show that the relation between these partial orders is as follows:

$$(P \leq_{tr} P') \Longleftarrow (P \leq_{wte} P') \Longleftarrow (P \leq_{ste} P')$$

The first *fully probabilistic* testing semantics, in the sense that both processes and tests are probabilistic, appears in [CSZ92]. This work was extended

in [YCDS94,CDSY99] with alternative characterizations of the corresponding equivalence. They consider probabilistic labelled transition systems to describe both processes and tests. However, in order to introduce this work, we will use the probabilistic process algebra defined in the introduction. The only difference between processes and tests is that the latter can perform a distinguished action ω that represents that the computation is successful. If the test can perfom an acceptance action then the testing procedure will (successfully) finish. Let us remark that in addition to visible actions, that is those belonging to Act, internal actions may appear both in processes and tests. The interaction between processes and tests is given by composing them in parallel and assuming a total synchronization.

Definition 14. Let *Proc* be the set of fully probabilistic processes and $\mathcal{T}$ be the set of tests. Let us consider a process $P \in Proc$ and a test $T \in \mathcal{T}$. The *interaction system* for P and T, denoted by $P \parallel T$, is defined as a test where the new probability distribution function, denoted by prob_I, is defined as follows:

$$\text{prob}_I(P \parallel T, \alpha, Q \parallel T') = \begin{cases} 0 & \text{if } \nu(P,T) = 0 \\ \frac{\text{prob}(T,\tau,T') \cdot (1 - \text{prob}_P(P,\tau))}{\nu(P,T)} & \text{if } \nu(P,T) \neq 0 \wedge \alpha = \tau \wedge P = P' \\ \frac{\text{prob}(P,\tau,Q) \cdot (1 - \text{prob}_T(T,\tau))}{\nu(P,T)} & \text{if } \nu(P,T) \neq 0 \wedge \alpha = \tau \wedge T = T' \\ \frac{\text{prob}(P,\alpha,Q) \cdot \text{prob}(T,\alpha,T')}{\nu(P,T)} & \text{otherwise} \end{cases}$$

where $\alpha \in Act_\tau$, the cumulative probability transition functions prob, for P and T, follow Definition 8, and $\text{prob}_R(r,\alpha) = \sum_{r' \in R} \text{prob}_R(r,\alpha,r')$. We write $P \parallel T \xrightarrow{\alpha}_p Q \parallel T'$ if $\text{prob}_I(P \parallel T, \alpha, Q \parallel T') = p$.

In the definition of the function prob_I we have used a *normalization factor* $\nu : Proc \times \mathcal{T} \longrightarrow [0,1]$. The formal definition of this factor is given by

$$\begin{aligned} \nu(P,T) = & \textstyle\sum_{a \in Act} \text{prob}_P(P,a) \cdot \text{prob}_T(T,a) \\ & + \text{prob}_T(T,\tau) + \text{prob}_P(P,\tau) - \text{prob}_T(T,\tau) \cdot \text{prob}_P(P,\tau) \end{aligned}$$

□

Intuitively, the normalization factor computes the total probability with which $P \parallel T$ may perform actions. So, by dividing by this factor we automatically get that the sum of the probabilities associated with outgoing transitions of the interaction system equals 1. Let us justify why the normalization factor ν is defined in this way. First, an interaction system may perform an external action only if both process and test are able to. Thus, the probability of performing any visible action is given by

$$\sum_{a \in Act} \text{prob}_P(P,a) \cdot \text{prob}_T(T,a) \tag{1}$$

Regarding internal actions, the interaction system $P \parallel T$ can internally evolve if either the process or the test may autonomously perform an internal action. Nevertheless, if both of them decide to do it then these two transitions

are merged into one (this is contemplated in the last case of the definition of prob_I). The probability with which P may perform τ while T does not is given by $\text{prob}_P(P,\tau) \cdot (1 - \text{prob}_T(T,\tau))$. Symmetrically, $P \parallel T$ may internally evolve because T performs a τ action and P does not with probability $\text{prob}_T(T,\tau)\cdot(1-\text{prob}_P(P,\tau))$. Finally, the probability with which the process and the test simultaneously perform a τ action is given by $\text{prob}_P(P,\tau) \cdot \text{prob}_T(T,\tau)$. If we put together these three values and we unfold the products we obtain

$$\text{prob}_T(T,\tau) + \text{prob}_P(P,\tau) - \text{prob}_T(T,\tau) \cdot \text{prob}_P(P,\tau) \tag{2}$$

If we add the values given in expressions (1) and (2) we finally get the definition of the normalization factor ν.

The probability with which a process P passes a test T is given by the sum of all the successful computations from $P \parallel T$. In order to compute this probability we need to define some auxiliary concepts.

Definition 15. Let P be a process, T be a test, and $P \parallel T$ be the associated interaction system. A *computation* is a sequence of transitions

$$C = P \parallel T \xrightarrow{\alpha_1}_{p_1} P_1 \parallel T_1 \xrightarrow{\alpha_2}_{p_2} \cdots P_{n-1} \parallel T_{n-1} \xrightarrow{\alpha_n}_{p_n} P_n \parallel T_n$$

such that for any $i < n$ we have $T_i \not\xrightarrow{\omega}$ and there do not exist $p > 0$, α, Q, and T' such that $P_n \parallel T_n \xrightarrow{\alpha}_p Q \parallel T'$. If $T_n \xrightarrow{\omega}$ then we say that the computation is *successful*. We denote by $S_{P\parallel T}$ the set of *successful computations* from $P \parallel T$. The probability of a successful computation C, denoted by $Prob(C)$, is inductively defined as

$$\begin{aligned} Prob(P \parallel T) &= 1 \\ Prob(P \parallel T \xrightarrow{\alpha}_p C') &= p \cdot Prob(C') \end{aligned}$$

For any $\mathcal{C} \subseteq S_{P\parallel T}$ we define $Prob(\mathcal{C}) = \sum_{C\in\mathcal{C}} Prob(C)$. Finally, we say that p is the *probability* with which the process P *passes* the test T, denoted by $P\, pass_p\, T$, if $Prob(S_{P\parallel T}) = p$.

Let $\mathcal{T}_0 \subseteq \mathcal{T}$ be a set of probabilistic tests and P, Q two processes. We say that Q is *better* than P with respect to $\mathcal{T}_0$, denoted by $P \sqsubseteq_{\mathcal{T}_0} Q$, if for any test $T \in \mathcal{T}_0$, we have $P\, pass_p\, T$ and $Q\, pass_q\, T$ implies $p \leq q$. Besides, we say that P and Q are *testing equivalent* with respect to $\mathcal{T}_0$, denoted by $P \approx_{\mathcal{T}_0} Q$, if for any test $T \in \mathcal{T}_0$, if $P\, pass_p\, T$ and $Q\, pass_q\, T$ then we have $p = q$. □

Let us note that successful computations have finite length since, by definition, they reach in a finite amount of steps a deadlocked process (since no more actions can be performed). The testing equivalence previously defined is parameterized by a set of tests. In [YCDS94] they consider two of these sets: The whole family of tests and the set of tests that do not contain the internal action τ. The following example shows that the former set of tests has (strictly) more distinguishing power than the latter. We consider that $\sqsubseteq_0$ denotes the previous preorder with respect to τ-free tests while $\sqsubseteq$ denotes the preorder with respect to the whole family of tests.

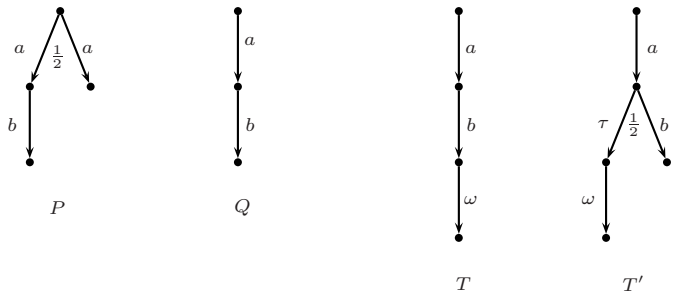


Fig. 10. Examples of probabilistic processes and probabilistic tests.

Example 3. Let us consider the processes P and Q depicted in Figure 10. These processes are related if we consider only τ-free tests, that is, $P \sqsubseteq_0 Q$. However, this is not the case if tests are not restricted. For instance, let us consider the tests T and T' (see Figure 10). We obtain $Prob(S_{P\|T}) = \frac{1}{2}$ and $Prob(S_{Q\|T}) = 1$. Thus, we have $Q \not\sqsubseteq P$. Besides, $Prob(S_{P\|T'}) = \frac{3}{4}$ and $Prob(S_{Q\|T'}) = \frac{1}{2}$. Thus, we have $P \not\sqsubseteq Q$. □

Actually, the whole set of test is too discriminative since the preorder coincides in fact with the induced equivalence relation.

In [NdF95] testing semantics for a probabilistic extension of a subset of the specification language LOTOS [LOT88] are studied. The underlying model, that is the induced probabilistic labelled transition systems, is equivalent to the one in [CSZ92] since probability is introduced by replacing the LOTOS choice operator by a probabilistic one. In this work the reactive and generative interpretations of probabilities are characterized in terms of tests. So, two processes are *reactive* testing equivalent if they pass with the same probability all the tests consisting of sequences of actions. The intuition is that only one action may be offered simultaneously, being that the reason why tests are traces. In the *generative* testing equivalence all the possible tests are allowed, that is, more than one action may be offered at the same time. In a third testing semantics, called *limited generative*, tests may offer more than an action simultaneously but all of them must be offered with the same probability.

Another approach for probabilistic testing, based also on the notion introduced in [CSZ92], is presented in [NR99]. In this work, fair testing [NC95,BRV95] is characterized in terms of a probabilistic testing semantics. Intuitively, it is shown that a (non-probabilistic) process P *fairly* passes a test T iff for any probabilizations of P and T, that is P_p and T_p, we have that P_p passes T_p with probability equal to 1. Let us remark that in the probabilistic setting the following two properties are not equal: all the computations of the system $P \parallel T$ are successful and $P\, pass_1\, T$.

Example 4. Let $P = \mathrm{rec}X.\ (a\,;\mathrm{stop} + \tau\,;X)$ be a (non-probabilistic) process and $T = a\,;\omega$ be a the test. We have that P must pass T does not hold because of the infinite sequence of τ actions. However, this computation can be considered *unfair* because the left hand side of the choice is never taken. On the contrary, if we consider the probabilistic process $P_p = \mathrm{rec}X.\ (a\,;\mathrm{stop} +_p \tau\,;X)$ then for any $0 < p < 1$ we have $P_p\ pass_1\ T$. □

The models described above have in common that they consider probabilistic versions of basic CCS (or, equivalently, probabilistic labelled transition systems). Even though these languages are usually simpler, the corresponding testing semantics present a drawback: They do not (sufficiently) abstract internal actions.

Example 5. Let us consider the processes $P_1 = a$ and $P_2 = \tau\,;a$. If we take the notion of composition defined in [CSZ92] or in [NdF95] then these two processes are not testing equivalent. For example, for the test $T = a\,;\omega +_{\frac{1}{2}} \tau\,;\mathrm{stop}$ we have that, on the one hand, P_1 passes T with probability $\frac{1}{2}$ while, on the other hand, P_2 passes T with probability $\frac{1}{4}$. □

Thus, it is worth to define formalisms where the previous shortcomings are corrected. Actually, this problem appears because of the possibility of *mixed* choices, that is, choices where both internal and external actions are offered simultaneously. We may either restrict the language (as they do in [YCDS94], where a testing semantics excluding τ actions in tests is studied) or to consider a new language. In [NdFL95], a CSP like language is considered where both choice operators are extended with probabilities. Tests are defined as processes but possibly containing the ω action. In addition to the definition of a testing equivalence, in [NdFL95] an alternative characterization and a fully abstract denotational semantics are also provided. Both of them are based on the corresponding ones for (non-probabilistic) must testing. In [Núñ03] the framework is extended with a sound and complete axiomatization. This language is also used in [GN99] in order to define a probabilistic extension of refusal testing [Phi87].

5.2 Testing Semantics for Non-deterministic Probabilistic Processes

In [YL92] a language featuring both a (non-probabilistic) CCS choice operator and a probabilistic internal choice operator is presented. In this model, tests are also considered as processes that can perform an acceptance action to express the success of a computation. They introduced a notion of probabilistic testing by considering the (set of) probabilities with which a process may pass a test. Actually, given the fact that not all the choices are quantified, a unique probability cannot be (in general) determined.

Example 6. Let us consider the process $Q = \tau\,;a + \tau\,;(a \oplus_{\frac{1}{2}} b) + \tau\,;b$ and the test $T = b\,;\omega$. The process Q has three possible choices for resolving its non-determinism: a, $a \oplus_{\frac{1}{2}} b$, and b. Considering the first choice, Q fails the test T, that is, Q passes the test T with probability 0; with the second choice, Q passes the test T with probability $\frac{1}{2}$, meanwhile for the last choice, the test is passed

with probability 1 by Q. Thus, Q passes the test T with the set of probabilities $\{0, \frac{1}{2}, 1\}$. □

In [YL92] the authors define probabilistic interpretations of the classical may and must semantics as follows:

- A process P *must* pass a test T if the minimum of the set of probabilities with which P passes T is equal to 1.
- A process P *may* pass a test T if the maximum of the set of probabilities with which P passes T is greater than 0.

Another proposal for testing semantics in the case of non-deterministic probabilistic processes is presented in [CCVP01,CCV+03]. They consider a language featuring the two non-probabilistic choice operators from CSP as well as a probabilistic internal choice. Even though their notions of testing are similar to those of [YL92], in order to introduce alternative characterizations they follow [NdFL95,CdFV97] by providing adequate probabilizations of acceptance sets and acceptance trees. In addition, they also present sound and complete axiomatizations for their testing equivalences.

6 Other Operators

In the previous sections we have concentrated on the possible considerations for defining probabilistic choice operators as well as on the semantic models for probabilistic processes. Given the fact that the considered semantic frameworks, either bisimulations or testing semantics, are defined from the operational behavior of processes, the language that it is used to compose processes can be set (partially) apart. For instance, even though in testing semantics the application of tests to processes is usually defined by means of a parallel operator, this operator can be *hidden* in the presentation of the models. Actually, the only relevant feature that we have considered so far was whether the language had only quantified choices or whether non-deterministic choices were also allowed. In this section we will study how other operators are included in probabilistic process algebras. Specifically, we will consider the parallel composition and hiding operators.

6.1 Parallel Composition Operator

The inclusion of parallel operators in probabilistic process algebras presents some additional problems, with respect to the non-probabilistic setting, depending on how probabilities associated with the different components are combined (see [DHK99] for a discussion and classification of different possibilities).

The first probabilistic process algebraic notations included a *synchronous parallel composition* because the definition was simpler in the context of generative models. Intuitively, we suppose that there is an operation $*$ on the set of

Fig. 11. Synchronous Parallel Composition

actions Act such that $(Act, *)$ is a monoid. Thus, if we have the parallel composition of P and Q, P may perform a, and Q may perform b, then the parallel composition will perform the action belonging to Act given by $a*b$. For example, this is the case of the probabilistic version of SCCS defined in [GJS90]. In that language, the parallel composition of two processes P and Q, denoted by $P \times Q$, behaves as the following inference rule describes

$$\frac{P \xrightarrow{\alpha}_p P',\ Q \xrightarrow{\beta}_q Q'}{P \times Q \xrightarrow{\alpha * \beta}_{p \cdot q} P' \times Q'}$$

The probabilities of the transitions of $P \times Q$ are simply determined by the product of the corresponding probabilities. The action performed by the composition, that is $\alpha * \beta$, is considered as the simultaneous (unordered) occurrence of both actions.

Example 7. Let us consider the processes $P = a +_{\frac{1}{2}} b$ and $Q = \overline{a} +_{\frac{1}{3}} c$ depicted in Figure 11. The parallel composition of both processes, $P \times Q$, will have four possible *atomic* actions to be performed. The first action will be 1 (i.e. the result of $a * \overline{a}$) with probability $\frac{1}{6}$, another action will be $a * c$ with probability $\frac{1}{3}$, the third one will be $b * \overline{a}$ with probability $\frac{1}{6}$, and, finally, $b * c$ with probability $\frac{1}{3}$ (see Figure 11). □

The definition of an asynchronous parallel composition is not so simple.[5] First, we consider the case of a CCS-like operator. Let us remind that the parallel composition of two non-probabilistic processes P and Q, denoted by $P \mid Q$, is defined by the following operational rules:

$$\frac{P \xrightarrow{a} P'}{P \mid Q \xrightarrow{a} P' \mid Q} \qquad \frac{Q \xrightarrow{a} Q'}{P \mid Q \xrightarrow{a} P \mid Q'} \qquad \frac{P \xrightarrow{a} P',\ Q \xrightarrow{\overline{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

[5] Let us remark that a first approach has been already explained in Definition 14 (see also [CDSY99]). There, the composition of process and test is based on the general idea of considering all possible pairs of actions arising from an independent choice in the two processes (product of probabilities) and then restricting to the acceptable pairs. However, in this composition interleaving actions (actually, the main source of problems when introducing parallel operators in a probabilistic setting) are not considered.

The first two rules express the fact that processes are allowed to asynchronously perform actions. The third rule defines the behavior in the presence of synchronization. If one of the processes may perform an action a and the other one can perform the complementary one, that is $\overline{a}$, then a synchronization may take place. However, it is worth to point out that a simple adaptation of the previous rules to the case of fully probabilistic processes may add non-determinism to the model. The following example illustrates this problem.

Example 8. Let us consider again the processes $P = a +_{\frac{1}{2}} b$ and $Q = \overline{a} +_{\frac{1}{3}} c$. If we try to make an analogy with the synchronous case then we have that there exist four different possibilities: P and Q synchronize by performing a and $\overline{a}$ respectively, P performs a and Q performs c, P performs b and Q performs $\overline{a}$, or P performs b and Q performs c. The problem appears because the performance of any two actions is not *atomic*. Thus, there are two ways of performing b and c by P and Q, respectively: First b and after that c, or vice versa. However, we only know the total probability of performing b and c, that is, $\frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$. A similar situation appears for the other cases. □

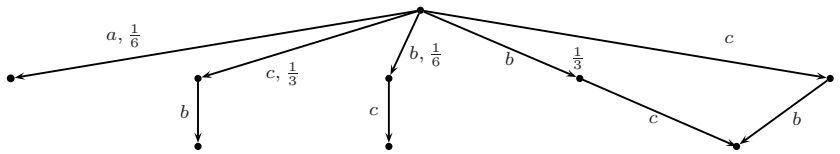
In [BBS95] a way to solve the problems described before in a generative framework is presented. They propose an operator $\|^{p,q}$ where $p, q \in (0, 1)$. The term $P\|^{p,q}Q$ represents a process that asynchronously performs actions from P with probability p, provided that P and Q are not going to synchronize; the probability $1 - p$ plays a similar role for Q. The value q denotes the probability of either P or Q to perform an action asynchronously when a synchronization is possible. Thus, synchronization is performed with probability $1 - q$.

In the non-probabilistic setting, a CSP-like asynchronous parallel composition operator has a parameter representing the set of synchronization actions. All the actions included in it have to be performed synchronously by the components of the parallel composition, while the rest of the actions are performed autonomously. The operational rules for the parallel composition of processes P and Q synchronizing in A, denoted by $P \|_A Q$, is defined as follows:

$$\frac{P \xrightarrow{a} P',\ a \notin A}{P \|_A Q \xrightarrow{a} P' \|_A Q} \qquad \frac{Q \xrightarrow{a} Q',\ a \notin A}{P \|_A Q \xrightarrow{a} P \|_A Q'} \qquad \frac{P \xrightarrow{a} P',\ Q \xrightarrow{a} Q',\ a \in A}{P \|_A Q \xrightarrow{a} P' \|_A Q'}$$

In contrast with the CCS operator previously presented, synchronization actions cannot be performed autonomously, as it was the case for a and $\overline{a}$ in Example 8. However, non-determinism still arises as the following example shows.

Example 9. Let us consider the processes $P = a \,\square_{\frac{1}{2}}\, b$ and $Q = a \,\square_{\frac{1}{3}}\, c$, and their parallel composition $R = P \|_{\{a\}} Q$. Intuitively, R may perform a with probability $\frac{1}{6}$. In addition, R can also asynchronously perform either b or c. However, there are several ways to perform them. If P performs b then Q has two options: It can try to perform either a or c. Thus, if Q decides to synchronize with P by performing a, then P should perform b with probability $\frac{1}{6}$, that is, $\frac{1}{2} \cdot \frac{1}{3}$. For the same reason, c can be performed with probability $\frac{1}{3} = \frac{1}{2} \cdot \frac{2}{3}$ when P desires to perform a. Non-determinism appears in the other possible option: P decides to



$$(a \,\square_{\frac{1}{2}}\, b) \parallel_{\{a\}} (a \,\square_{\frac{1}{3}}\, c)$$

Fig. 12. Asynchronous Parallel Composition for CSP

perform b while Q chooses to perform c. As in the CCS case, there are again two possibilities: Either b or c is the first action to be performed. In this case, all we know is the combined probability of these two options, that is, $\frac{1}{3}$. This example is graphically presented in Figure 12. □

In the case of this parallel operator we have an additional problem: We need a *normalization factor*. Let us explain this point by using the previous example.

Example 10. Let us consider again the processes defined in Example 9. If the process $R = P \parallel_{\{a\}} Q$ performs c with probability $\frac{1}{3}$ then we obtain the process $P \parallel_{\{a\}} \text{stop}$. At that moment, P can perform a with probability $\frac{1}{2}$ but there exists no possibility of synchronization. So, all the probability should be assigned to b, that is, it seems logical that $P \parallel_{\{a\}} \text{stop}$ performs b with probability 1. □

The problem may be solved by introducing a parallel operator with two parameters, denoted by $\|^p_A$, where $p \in (0, 1)$ and A is the synchronization set (see e.g. [NdF95]). The process $P \|^p_A Q$ can evolve either by performing non-synchronizing actions from P or Q (multiplying its associated probabilities by p and $1 - p$ respectively) or by synchronizing in an action belonging to A.

6.2 Hiding Operators

There exist some proposals to include either a hiding operator or a restriction operator into probabilistic process algebras. For example in [CCV+03], where a process algebra featuring both probabilistic and non-deterministic behaviors is presented, the operational definition of hiding is given by the following rules:

$$\frac{P \longrightarrow_p P'}{P\backslash A \longrightarrow_p P'\backslash A} \qquad \frac{P \longrightarrow P'}{P\backslash A \longrightarrow P'\backslash A} \qquad \frac{P \xrightarrow{a} P' \ \wedge \ a \notin A}{P\backslash A \xrightarrow{a} P'\backslash A} \qquad \frac{P \xrightarrow{a} P' \ \wedge \ a \in A}{P\backslash A \longrightarrow P'\backslash A}$$

where $\longrightarrow_p$ represents the resolution of a probabilistic choice, $\longrightarrow$ the internal evolution of a process, and $\xrightarrow{a}$ represents the evolution of the process by performing the visible action a. That is, the definition is essentially the same as in the non-probabilistic case. A similar operator is given in [AB01].

7 Conclusions

In this paper we have presented the most relevant adaptations to the probabilistic setting of the classical notions of bisimulation and testing semantics. We have distinguished between fully probabilistic models, that is where all the choices are quantified, and non-deterministic probabilistic processes, that is where some choices are quantified and others are non-deterministically resolved. We expect that this paper can serve both to those researchers who are only interested in getting a brief overview of the field as well as to those who desire to get a more thorough knowledge by following the references that we provide.

Acknowledgments

The authors would like to thank the anonymous referees of this paper for the careful reading. Their very valuable comments have undoubtedly improved the quality of this survey.

References

[AB01] S. Andova and J.C.M. Baeten. Abstraction in probabilistic process algebra. In *TACAS 2001, LNCS 2031*, pages 204–219. Springer, 2001.

[BA01] M. Bravetti and A. Aldini. Expressing processes with different action durations through probabilities. In *PAPM-PROBMIV 2001, LNCS 2165*, pages 168–183. Springer, 2001.

[BA03] M. Bravetti and A. Aldini. Discrete time generative-reactive probabilistic processes with different advancing speeds. *Theoretical Computer Science*, 290(1):355–406, 2003.

[BBS95] J.C.M. Baeten, J.A. Bergstra, and S.A. Smolka. Axiomatizing probabilistic processes: ACP with generative probabilities. *Information and Computation*, 121(2):234–255, 1995.

[BH97] C. Baier and H. Hermanns. Weak bisimulation for fully probabilistic processes. In *Computer Aided Verification'97, LNCS 1254*, pages 119–130. Springer, 1997.

[BM89] B. Bloom and A. R. Meyer. A remark on bisimulation between probabilistic processes. In *Logic at Botik'89, LNCS 363*, pages 26–40. Springer, 1989.

[BPS01] J.A. Bergstra, A. Ponse, and S.A. Smolka, editors. *Handbook of Process Algebra*. North Holland, 2001.

[BRV95] E. Brinksma, A. Rensink, and W. Vogler. Fair testing. In *CONCUR'95, LNCS 962*, pages 313–327. Springer, 1995.

[BS01] E. Bandini and R. Segala. Axiomatizations for probabilistic bisimulation. In *ICALP 2001, LNCS 2076*, pages 370–381. Springer, 2001.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge Tracts in Computer Science 18. Cambridge University Press, 1990.

[CCV+03] D. Cazorla, F. Cuartero, V. Valero, F.L. Pelayo, and J.J. Pardo. Algebraic theory of probabilistic and non-deterministic processes. *Journal of Logic and Algebraic Programming*, 55(1–2):57–103, 2003.

[CCVP01] D. Cazorla, F. Cuartero, V. Valero, and F.L. Pelayo. A process algebra for probabilistic and nondeterministic processes. *Information Processing Letters*, 80:15–23, 2001.

[CdFV97] F. Cuartero, D. de Frutos, and V. Valero. A sound and complete proof system for probabilistic processes. In *4th International AMAST Workshop on Real-Time Systems, Concurrent and Distributed Software, LNCS 1231*, pages 340–352. Springer, 1997.

[CDSY99] R. Cleaveland, Z. Dayar, S.A. Smolka, and S. Yuen. Testing preorders for probabilistic processes. *Information and Computation*, 154(2):93–148, 1999.

[Chr90] I. Christoff. Testing equivalences and fully abstract models for probabilistic processes. In *CONCUR'90, LNCS 458*, pages 126–140. Springer, 1990.

[CLN01] R. Cleaveland, G. Lüttgen, and V. Natarajan. Priority in process algebra. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of process algebra*, chapter 12. North Holland, 2001.

[CSZ92] R. Cleaveland, S.A. Smolka, and A.E. Zwarico. Testing preorders for probabilistic processes. In *19th ICALP, LNCS 623*, pages 708–719. Springer, 1992.

[CY88] C. Courcoubetis and M. Yannakakis. Verifying temporal properties of finite-state probabilistic programs. In *29th IEEE Symposium on Foundations of Computer Science*, pages 338–345. IEEE Computer Society Press, 1988.

[dFLN97] D. de Frutos-Escrig, L.F. Llana-Díaz, and M. Núñez. Friendly testing as a conformance relation. In *Formal Description Techniques for Distributed Systems and Communication Protocols (X), and Protocol Specification, Testing, and Verification (XVII)*, pages 283–298. Chapman & Hall, 1997.

[dFLN99] D. de Frutos-Escrig, N. López, and M. Núñez. Global timed bisimulation: An introduction. In *Formal Description Techniques for Distributed Systems and Communication Protocols (XII), and Protocol Specification, Testing, and Verification (XIX)*, pages 401–416. Kluwer Academic Publishers, 1999.

[DGJP02] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Weak bisimulation is sound and complete for PCTL*. In *CONCUR 2002, LNCS 2421*, pages 355–370. Springer, 2002.

[DHK99] P.R. D'Argenio, H. Hermanns, and J.-P. Katoen. On generative parallel composition. In *PROBMIV'98, Electronics Notes in Theoretical Computer Science 22*. Elsevier, 1999.

[dNH84] R. de Nicola and M.C.B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[FH82] Y.A. Feldman and D. Harel. A probabilistic dinamic logic. In *14th ACM Symposium on Theory of Computing*, pages 181–195. ACM Press, 1982.

[GJS90] A. Giacalone, C.-C. Jou, and S.A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proceedings of Working Conference on Programming Concepts and Methods, IFIP TC 2*. North Holland, 1990.

[GN99] C. Gregorio and M. Núñez. Denotational semantics for probabilistic refusal testing. In *PROBMIV'98, Electronic Notes in Theoretical Computer Science 22*. Elsevier, 1999.

[GSS95] R. van Glabbeek, S.A. Smolka, and B. Steffen. Reactive, generative and stratified models of probabilistic processes. *Information and Computation*, 121(1):59–80, 1995.

[GSST90] R. van Glabbeek, S.A. Smolka, B. Steffen, and C.M.N. Tofts. Reactive, generative, and stratified models of probabilistic processes. In *5th IEEE Symposium on Logic In Computer Science*, pages 130–141. IEEE Computer Society Press, 1990.

[GW96] R. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996.

[Hen87] M. Hennessy. An algebraic theory of fair asynchronous communicating processes. *Theoretical Computer Science*, 49:121–143, 1987.

[Hen88] M. Hennessy. *Algebraic Theory of Processes*. MIT Press, 1988.

[HJ89] H. Hansson and B. Jonsson. A framework for reasoning about time and realibility. In *10th IEEE Real-Time Systems Symposium*. IEEE Computer Society Press, 1989.

[HJ90] H. Hansson and B. Jonsson. A calculus for communicating systems with time and probabilities. In *11th IEEE Real-Time Systems Symposium*, pages 278–287. IEEE Computer Society Press, 1990.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[HS84] S. Hart and M. Sharir. Probabilistic temporal logics for finite and bounded models. In *16th ACM Symposium on Theory of Computing*, pages 1–13. ACM Press, 1984.

[JP89] C. Jones and G.D. Plotkin. A probabilistic powerdomain of evaluations. In *4th IEEE Symposium on Logic In Computer Science*, pages 186–195. IEEE Computer Society Press, 1989.

[KN98] M. Kwiatkowska and G.J. Norman. A testing equivalence for reactive probabilistic processes. In *EXPRESS'98, Electronic Notes in Theoretical Computer Science 16*. Elsevier, 1998.

[Koz83] D. Kozen. A probabilistic PDL. In *15th ACM Symposium on Theory of Computing*, pages 291–297. ACM Press, 1983.

[KS76] J.G. Kemeny and J.L. Snell. *Finite Markov Chains*. Springer, 1976.

[LOT88] LOTOS. A formal description technique based on the temporal ordering of observational behaviour. IS 8807, TC97/SC21, 1988.

[Low95] G. Lowe. Probabilistic and prioritized models of timed CSP. *Theoretical Computer Science*, 138:315–352, 1995.

[LS89] K. Larsen and A. Skou. Bisimulation through probabilistic testing. In *16th ACM Simposium on Principles of Programming Languages*, pages 344–352. ACM Press, 1989.

[LS91] K. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

[LT87] N.A. Lynch and M.R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *6th ACM Symp. on Principles of Distributed Computing*, pages 137–151. ACM Press, 1987.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 253:267–310, 1983.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[MS92] R. Milner and D. Sangiorgi. Barbed bisimulation. In *19th ICALP, LNCS 623*, pages 685–695. Springer, 1992.

[NC95] V. Natarajan and R. Cleaveland. Divergence and fair testing. In *22nd ICALP, LNCS 944*, pages 648–659. Springer, 1995.

[NCS98] K. Narayan Kumar, R. Cleaveland, and S.A. Smolka. Infinite probabilistic and nonprobabilistic testing. In *18th Conference on Foundations of Software Technology and Theoretical Computer Science, LNCS 1530*, pages 209–220. Springer, 1998.

[NdF95] M. Núñez and D. de Frutos. Testing semantics for probabilistic LOTOS. In *Formal Description Techniques VIII*, pages 365–380. Chapman & Hall, 1995.

[NdFL95] M. Núñez, D. de Frutos, and L. Llana. Acceptance trees for probabilistic processes. In *CONCUR'95, LNCS 962*, pages 249–263. Springer, 1995.

[NR99] M. Núñez and D. Rupérez. Fair testing through probabilistic testing. In *Formal Description Techniques for Distributed Systems and Communication Protocols (XII), and Protocol Specification, Testing, and Verification (XIX)*, pages 135–150. Kluwer Academic Publishers, 1999.

[Núñ03] M. Núñez. Algebraic theory of probabilistic processes. *Journal of Logic and Algebraic Programming*, 56(1–2):117–177, 2003.

[Phi87] I. Phillips. Refusal testing. *Theoretical Computer Science*, 50(3):241–284, 1987.

[PLS00] A. Philippou, I. Lee, and O. Sokolsky. Weak bisimulation for probabilistic systems. In *CONCUR 2000, LNCS 1877*, pages 334–349. Springer, 2000.

[Rab63] M.O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.

[SCS03] E.W. Stark, R. Cleaveland, and S.A. Smolka. A process-algebraic language for probabilistic I/O automata. In *CONCUR'03, LNCS*. Springer, 2003. to appear.

[Sei95] K. Seidel. Probabilistic communicating processes. *Theoretical Computer Science*, 152:219–249, 1995.

[SL94] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. In *CONCUR'94, LNCS 836*, pages 481–496. Springer, 1994.

[SL95] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

[SS00] E.W. Stark and S.A. Smolka. A complete axiom system for finite-state probabilistic processes. In *Proof, Language and Interaction: Essays in Honour of Robin Milner*. MIT Press, 2000.

[SV99] M. Stoelinga and F.W. Vaandrager. Root contention in IEEE 1394. In *5th AMAST Workshop on Real-Time and Probabilistic Systems, LNCS 1601*, pages 53–74. Springer, 1999.

[Var85] M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *26th IEEE Symposium on Foundations of Computer Science*, pages 327–338. IEEE Computer Society Press, 1985.

[WSS94] S.-H. Wu, S.A. Smolka, and E.W. Stark. Composition and behaviors of probabilistic I/O automata. In *CONCUR'94, LNCS 836*, pages 513–528. Springer, 1994.

[WSS97] S.-H. Wu, S.A. Smolka, and E.W. Stark. Composition and behaviors of probabilistic I/O automata. *Theoretical Computer Science*, 176(1-2):1–37, 1997.

[YCDS94] S. Yuen, R. Cleaveland, Z. Dayar, and S.A. Smolka. Fully abstract characterizations of testing preorders for probabilistic processes. In *CONCUR'94, LNCS 836*, pages 497–512. Springer, 1994.

[YL92] W. Yi and K.G. Larsen. Testing probabilistic and nondeterministic processes. In *Protocol Specification, Testing and Verification XII*, pages 47–61. North Holland, 1992.