Expressiveness of probabilistic π-calculi*

Sylvain Pradalier† Catuscia Palamidessi‡

Abstract

In this work we propose a probabilistic extension of the π-calculus. The main novelty is a probabilistic *mixed choice* operator, that is, a choice construct with a probability distribution on the branches, and where input and output actions can both occur as guards. We develop the operational semantics of this calculus, and then we investigate its expressiveness. In particular, we compare it with the sublanguage with the two *separate choices*, where input and output guards are not allowed together in the same choice construct. Our main result is that the separate choices can encode the mixed one. Further, we show that *input-guarded* choice can encode *output-guarded* choice and viceversa. In contrast, we conjecture that neither of them can encode the pair of the two separate choices.

1 Introduction

In the field of concurrent languages, expressiveness is an important and intriguing problem. Differently from the case of sequential languages, the purpose of a program is not just to compute a function, but also to control the communication and the interaction of the various parallel components of a system. There are therefore more parameters and perspectives which must be taken into account when assessing the expressive power of a new formalism.

Most of the main process calculi proposed in literature have been widely investigated from the point of view of the expressive power, both in absolute terms, i.e. their capability to solve problems, and in relative terms, i.e. their comparison. In particular, there has been a lot of work aiming at establishing the relation between different calculi, thus providing some structure for the huge plethora of formalisms that have been proposed in the field of Concurrency. One of the goals of such investigation is, of course, to individuate languages that have the same expressive power but can be

*This work has been partially supported by the INRIA ARC project ProNoBiS and by the INRIA DREI Équipe Associée PRINTEMPS.

†Lix, Ecole Polytechnique, 91128 Palaiseau Cedex, France

‡INRIA Futurs and Lix, Ecole Polytechnique, 91128 Palaiseau Cedex, France

implemented in a more efficient way. The encoding itself can be valuable, as the source language, even if less efficient, may still be useful as a specification language. Another goal is to find out the constraints to implementation. For instance, if a language can solve a problem that is known to be not solvable in a distributed asynchronous model (like for instance the symmetric leader election), then we know that language cannot be implemented in a totally distributed manner. The interested reader can find in [11] an extended discussion on these issues.

Surprisingly however, for an important class of calculi, the probabilistic ones, the question of relative expressiveness has not been investigated much (as far as we know), despite the fact that there have been many proposals already and that the area is rather mature. Among the several approaches, we mention the one in [17] which is similar to ours in spirit. We suggest the interested reader to consult [1] for a recent overview and classification of the main probabilistic calculi and models that have been proposed.

In this paper, we make a first step towards the study of relative expressiveness in the probabilistic setting. We focus on one of the key mechanisms in Concurrency: the choice operator. This construct represents a choice between alternative computations, and may be controlled by means of *guards*. Its importance relies on the fact that it is very useful in distributed systems for allowing processes to interact and coordinate.

One can define various kinds of choice operator depending on the guards that are allowed to appear in it. In process calculi, guards are usually communication actions (input and output), and it is then natural to consider the following classification:

- *input-guarded choice*: the guards can only be input actions,
- *output-guarded choice*: the guards can only be output actions,
- *separate choice*: a choice can contain input or output guards, but not both,
- *mixed choice*: a choice can contain both input and output guards.

In the non-probabilistic world it has been proved that the asynchronous π-calculus (no choice, and only asynchronous outputs) can encode input-guarded choice [10] and also the separate choices [9]. On the contrary, it cannot encode the mixed choice [11]. The mixed choice had been already proved to be strictly more powerful than the other kinds of choice also in CSP [3]. In both these cases, the proof of the separation result relies on the capability/incapability of expressing the solution to certain consensus problems.

We are interested in exploring whether the probabilistic extensions of the above choice constructs presents a similar gap. In particular, we consider this question in the context of the π-calculus. We know that probabilities

add expressive power: in fact, in [12] a probabilistic extension of the π-calculus with input-guarded choice has been proved able to encode the π-calculus with the mixed choice. On the other hand this result *per-se* is not a proof that everything collapses in the same expressiveness class. In fact, the combination of the mixed choice and probability may generate new capabilities. This is one motivation for exploring the probabilistic extension of the mixed choice.

It is however not obvious how to define such extension. There are in fact various subtle issues related to the compositionality of the parallel operator, as explained in [16]. A stochastic version of the π-calculus with the mixed choice has been defined in [15], but in that case the definition of the parallel operator rests on the *synchronous*[1] assumption, according to which all the parallel components move at the same time. This is the case for all the proposals of the probabilistic mixed choice that we know of from literature. We, on the contrary, want to investigate the expressiveness of choice in an *asynchronous* setting, since we are interested in distributed systems, where the assumption of a global time would be unrealistic.

1.1 Contribution

We propose a probabilistic version of the π-calculus with the mixed choice coherent with the proposal in [6, 12] for the case of input-guarded choice. We then investigate its expressive power relatively to that of its sublanguages, that are obtained by restricting the choice construct.

We will see that, in contrast to the findings in [11] for the non-probabilistic case, the mixed choice can be encoded in the separate choices. Our result presents some analogy with the one in [12], where the mixed choice was encoded using probabilistic input-guarded choice, The difference with the latter is that in our case we have the separate choices available, and we present a much simpler encoding, based on a default possibility of backtracking. The encoding in [12] is based on a sophisticated extension of the dining cryptographers protocol. We think that such idea cannot be extended to our setting (this is part of our conjecture about the gap between the separate choices and one single choice, see below). On the other hand, we are not sure either that our simpler encoding can be adapted somehow to the setting in [12], because ours requires a choice construct with both output and τ prefixes, which is not present in the probabilistic asynchronous π-calculus considered in [12].

The importance of our result relies on the fact that the distributed implementation of the mixed choice is much more difficult than the one of

[1]The term synchronous (resp. asynchronous) is used here in the sense of Distributed Computing, i.e. it means that the underlying model of computation is based on a global clock (resp. local clocks). This is different from the use in Concurrency, where synchronous and asynchronous usually refer to communication.

separate choices. Under certain conditions, it is even impossible. Again, see [11] for a discussion on this topic.

On the lower level of the hierarchy, we will show that each form of choice (input-guarded or output-guarded) can encode the other. We do not know, at the moment, whether the hierarchy actually collapses into just one class of expressiveness, but our expectation is that this is not the case: we believe that there is a gap in between the language with the two separate choices and the two languages with one kind of choice.

We consider two semantics for our language. The first one follows the approach of [6] in which the coefficients on the branches of a choice add up to 1 (and for this reason we regard them as probabilities). The second approach is based in considering these coefficients as *activities*. Then, after the execution we re-normalize the quantities associated to the runs to get probabilities. Part 2.3 explains in detail the philosophy of both semantics.

1.2 Overview:

We begin by describing and explaining the syntax and the two operational semantics (Section 2). Then we present an encoding of the probabilistic mixed choice by the probabilistic separate choices (Section 3). Further we conjecture that the two separate choices (input and output) cannot be encoded by using only one of them (Section 4). On the other hand, we show that input and output guarded choices can encode each other (Section 5). We then discuss some controversial design decision for our calculus (Section 6), and we conclude with the plans for future research related to this work (Section 7).

For reasons of space, we do not include the proofs here. They can be found in [14].

2 An operational semantic of the quantitative and probabilistic π-calculus with mixed choice

The introduction of probabilities in a calculus does not necessarily rule out the nondeterminism. Indeed the variations of behaviors due to the environment, the scheduler or the adversary, may be more naturally considered as nondeterministic, since we may not have any information on it a priori.

This is why, as in [6], we consider a syntax with a probabilistic choice and a classical parallel operator. The first will represent the behavioral variations due to the process in a given scheduling scenario. The latter will generate nondeterministically different scheduling scenarios. Correspondingly, the operational semantics will generate groups of transitions corresponding to the probabilistic choices in different scenarios. We call such a group a *step*.

2.1 Syntax

We use a syntax very close to the classical π-calculus. The only modification on the standard constructors is the addition of a (positive) coefficient on each branch of a choice. This is the method used in [6, 15] for instance. Intuitively, the greater the coefficient of a branch is, the more probable is its execution. We explain in Section 2.3 how probabilities are computed from these coefficients.

The syntax is defined by the following grammar

$$\text{Prefixes} \quad \alpha ::= \quad \overline{x}y \quad | \quad x(y) \quad | \quad \tau$$

$$\text{Processes} \quad P ::= \quad \Sigma_{i\in I}(\alpha_i, p_i).P_i \quad | \quad \nu x P \quad | \quad P|P \quad | \quad X \quad | \quad rec_X.P$$

The prefixes can be of the form $\overline{x}y$ (output), $x(y)$ (input), and τ (silent action). A process is either of the form $\Sigma_{i\in I}(\alpha_i, p_i).P_i$ (a weighted choice between different guarded process), or $\nu x P$ (a restriction), or $P|Q$ (a parallel composition), or X (a recursion variable), or $rec_X.P$ (a recursive process).

We sometimes write $(\alpha_1, p_1).P_1 + ... + (\alpha_n, p_n).P_n$ instead of $\Sigma_{i\leq n}(\alpha_i, p_i).P_i$. The empty sum represents a terminated process and is denoted by 0.

Both our operational semantics use this syntax, but for the probabilistic one we require that, for each choice $\Sigma_{i\leq n}(\alpha_i, p_i).P_i$, the sum $\Sigma_{i\in I} p_i$ be 1. See Section 2.3.

Let us briefly recall the terminology relevant for our investigation: a (sub-)language has *mixed choice*, *separate choice*, *input-guarded choice*, *output-guarded choice*, or *blind choice*, if its choice constructs of cardinality greater than 1 can have as guards, respectively: both outputs and inputs, either outputs or inputs, only inputs, only outputs, or no guards at all (i.e. only τ prefixes).

Following the terminology used in Concurrency Theory, we call *asynchronous* the sub-language in which outputs can only be followed by the terminated process (*asynchronous outputs*), and occur only in trivial choices, i.e. in choices of cardinality 1. In other words, outputs can only appear in constructs of the form $\overline{x}y.0$.

2.2 Structural congruence

We use the usual structural congruence $\equiv$ of the π-calculus, that is, α-conversion, commutativity and associativity of the parallel operator, neu-

trality of the 0, commutativity of the choice, scope extrusion, and commutativity of the restriction. See [14] for the precise definition.

2.3 Operational Semantics

Weighted choices associate a coefficient to each action. Intuitively this coefficient represents the chances for the action to be executed: the higher the coefficient, the more probable the action.

To obtain probabilities from these coefficients, they only need to sum up to 1. It is therefore sufficient to re-normalize each coefficient. Here we have two alternatives: we can re-normalize for each choice, directly in the term and at each step of execution, or we can re-normalize after the derivation of the transitive closure of each possible step. Both possibilities seem reasonable and they correspond to our two semantics: the *probabilistic* semantics and the and the *quantitative* one, respectively. The probabilistic case requires the sum of the probabilities associated to each step to be equal to 1, and that each rule keeps the sum equal to 1.

The difference between these two semantics is that the coefficient in the quantitative case may be used to represent also other kinds of information. For instance, it could be associated to the expected speed of some reaction which enables the guard (i.e. the inverse of the expected time that takes for the guard to become enabled). As an example, let $P = (a_1, 10).P_1 + (a_2, 10).P_2$ and $Q = (b_1, 5).Q_1 + (b_2, 5).Q_2$, and consider their parallel composition $P \,|\, Q$. In the probabilistic case, all coefficients are re-normalized to $1/2$ and have equal chance to occur. But in the quantitative case, a_1 and a_2 are more likely to occur first. Furthermore b_1 and b_2 could be in competition with a_1 and a_2, for instance if they all are input actions on a channel where there is only an output available. So, the probability to occur first may translate into the probability to occur at all.

In this paper, we have also another reason to consider the quantitative approach: our main result, the encoding of the probabilistic mixed choice into the probabilistic separate choices, presents some problems with respect to the restriction operator in the probabilistic semantics, while this problem disappears in the quantitative semantics.

2.3.1 Rules common to both semantics

In the classical π-calculus, one writes $P \xrightarrow{\alpha} Q$ for the transition from P to Q. Here we write $P\{\xrightarrow{\alpha_i}_{p_i} P_i\}_{i\in I}$ for the step from P to the P_i's with coefficients p_i. The reason for this grouping are explained at the beginning of Section 2 We omit the notation $i \in I$ when there is no ambiguity.

Rules CONG, SUM and REC are the probabilistic extensions of the corresponding rules in the classical π-calculus.

$$\textsf{SUM}: \quad \frac{}{\Sigma_i(p_i,\mu_i).P_i\{\xrightarrow{\mu_i}_{p_i} P_i\}}$$

$$\textsf{CONG}: \quad \frac{P\equiv P' \quad P\{\xrightarrow{\mu_i}_{p_i} P_i\} \quad \forall i. P_i \equiv P_i'}{P'\{\xrightarrow{\mu_i}_{p_i} P_i'\}}$$

$$\textsf{REC:} \quad \frac{P[rec_X P/X]\{\xrightarrow{\mu_i}_{p_i} P_i\}}{rec_X P\{\xrightarrow{\mu_i}_{p_i} P_i\}}$$

The COM rule corresponds to the fusion of the three classical rules of the π-calculus for interleaving, communication and communication with scope extrusion (called PAR, COM and CLOSE classically). It is more complicated than other probabilistic calculi in literature because we are dealing with an asynchronous model (asynchronous in the sense of no global clock): Each process can proceed at his own speed and decide whether to synchronize or not, on each of the branches, hence several different cases can occur when combining the steps of two parallel processes.

Given two steps $P\{\xrightarrow{\mu_i}_{p_i} P_i\}_{i\in I}$ and $Q\{\xrightarrow{\eta_j}_{q_j} Q_j\}_{j\in J}$, we want to build a step from $P|Q$. To this end, for each pair of transitions of $(P \xrightarrow{\mu_i}_{p_i} P_i, Q \xrightarrow{\eta_j}_{q_j} Q_j)$ we build a transition of $P|Q$ using one of the three classical rules for the parallel composition. For instance, if $P\{\xrightarrow{\overline{x}y}_{1/2} P', \ldots\}$ and $Q\{\xrightarrow{x(z)}_{1/3} Q', \ldots\}$, then from $P|Q$ we will have steps of the form $P|Q\{\xrightarrow{\tau}_{1/6} P'|Q'[y/z]\}$ (communication), and of the form $P|Q\{\xrightarrow{\overline{x}y}_{1/6} P'|Q, \ldots\}$ (left interleaving), and of the form $P|Q\{\xrightarrow{x(z)}_{1/6} P|Q', \ldots\}$ (right interleaving).

One may wonder why we do not put these steps together in one single step from $P|Q$. This is because the alternative between these three cases should be nondeterministic rather than probabilistic, as in the classical π-calculus. The condition: $\forall i,j.\quad bn(\mu_i)\bigcap fn(Q_j)=\emptyset \quad\wedge\quad bn(\eta_j)\bigcap fn(P_i)=\emptyset$ comes from the condition $bn(\mu)\bigcap fn(Q)=\emptyset$ of the classical rule PAR: $\frac{P\xrightarrow{\mu}P' \quad bn(\mu)\bigcap fn(Q)=\emptyset}{P|Q\xrightarrow{\mu}P'|Q}$.

Note that the nondeterminism of the calculus derives from this case.

$$\textsf{COM}: \quad \frac{P\{\xrightarrow{\mu_i}_{p_i} P_i\} \quad Q\{\xrightarrow{\eta_j}_{q_j} Q_j\} \quad \forall i,j. \quad bn(\mu_i)\bigcap fn(Q_j)=\emptyset \quad\wedge\quad bn(\eta_j)\bigcap fn(P_i)=\emptyset}{(P|Q)\{\xrightarrow{\alpha_{i,j}}_{p_i q_j} R_{i,j}\}}$$

where $R_{i,j}$ and $\alpha_{i,j}$ are defined by:

- if $\mu_i = \overline{y}x,\ \eta_j = y(z)$,
  - either $R_{i,j} = P_i|Q_j[x/z] \quad \wedge \quad \alpha_{i,j} = \tau$: communication.
  - or $R_{i,j} = P_i|Q \quad \wedge \quad \alpha_{i,j} = \mu_i$: left interleaving.
  - or $R_{i,j} = P|Q_j \quad \wedge \quad \alpha_{i,j} = \eta_j$: right interleaving.
- symmetric case: $\mu_i = y(z),\ \eta_j = \overline{y}x$
- if $\mu_i = \overline{y}(x),\ \eta_j = y(z)$,
  - either $R_{i,j} = (\nu x)(P_i|Q_j[x/z]) \quad \wedge \quad \alpha_{i,j} = \tau$: communication and scope extrusion
  - or $R_{i,j} = P_i|Q \quad \wedge \quad \alpha_{i,j} = \mu_i$: left interleaving.
  - or $R_{i,j} = P|Q_j \quad \wedge \quad \alpha_{i,j} = \eta_j$: right interleaving.
- symmetric case: $\mu_i = y(z),\ \eta_j = \overline{y}(x)$
- otherwise,
  - either $R_{i,j} = P_i|Q \quad \wedge \quad \alpha_{i,j} = \mu_i$: left interleaving.
  - or $R_{i,j} = P|Q_j \quad \wedge \quad \alpha_{i,j} = \eta_j$: right interleaving.

Let us illustrate the rule COM with an example. For simplicity we omit the parameters in the communication. Furthermore, if in a step we have two transition with the same label and the same continuation, then we write the transition only once, of course with probability equal to the sum of the probabilities.

Example 1 *Consider the processes* $P = (1/2, \overline{y}).P_1 + (1/2, x).P_2$ *and* $Q = (1/3, y).Q_1 + (2/3, z).Q_2$. *The possible steps of* $P \,|\, Q$ *are* 24, *in fact* 3 *possible outcomes derive from the combination between the first branch of* P *and the first of* Q, 2 *from the first of* P *and the second of* Q, 2 *from the second of* P *and the first of* Q, *and* 2 *from the second of* P *and the second of* Q. *All the possible steps are:*

$$
\begin{array}{l}
P \,|\, Q\{\xrightarrow{\overline{y}}_{1/2} P_1 \,|\, Q, \quad \xrightarrow{x}_{1/2} P_2 \,|\, Q\} \\
P \,|\, Q\{\xrightarrow{\overline{y}}_{1/2} P_1 \,|\, Q, \quad \xrightarrow{x}_{1/6} P_2 \,|\, Q, \quad \xrightarrow{z}_{1/3} P \,|\, Q_2\} \\
P \,|\, Q\{\xrightarrow{\overline{y}}_{1/2} P_1 \,|\, Q, \quad \xrightarrow{x}_{1/3} P_2 \,|\, Q, \quad \xrightarrow{y}_{1/6} P \,|\, Q_1\} \\
P \,|\, Q\{\xrightarrow{\overline{y}}_{1/2} P_1 \,|\, Q, \quad \xrightarrow{y}_{1/6} P \,|\, Q_1, \quad \xrightarrow{z}_{1/3} P \,|\, Q_2\}
\end{array}
$$

$$
\begin{array}{l}
P \,|\, Q\{\xrightarrow{\overline{y}}_{1/6} P_1 \,|\, Q, \quad \xrightarrow{z}_{1/3} P \,|\, Q_2, \quad \xrightarrow{x}_{1/2} P_2 \,|\, Q\} \\
P \,|\, Q\{\xrightarrow{\overline{y}}_{1/6} P_1 \,|\, Q, \quad \xrightarrow{z}_{1/3} P \,|\, Q_2, \quad \xrightarrow{x}_{1/6} P_2 \,|\, Q, \quad \xrightarrow{z}_{1/3} P \,|\, Q_2\} \\
P \,|\, Q\{\xrightarrow{\overline{y}}_{1/6} P_1 \,|\, Q, \quad \xrightarrow{z}_{1/3} P \,|\, Q_2, \quad \xrightarrow{x}_{1/3} P_2 \,|\, Q, \quad \xrightarrow{y}_{1/6} P \,|\, Q_1\} \\
P \,|\, Q\{\xrightarrow{\overline{y}}_{1/6} P_1 \,|\, Q, \quad \xrightarrow{z}_{1/3} P \,|\, Q_2, \quad \xrightarrow{y}_{1/6} P \,|\, Q_1, \quad \xrightarrow{z}_{1/3} P \,|\, Q_2\}
\end{array}
$$

$$
\begin{array}{llll}
P\,|\,Q\{\xrightarrow{\overline{y}}_{1/3} P_1\,|\,Q, & \xrightarrow{y}_{1/6} P\,|\,Q_2, & \xrightarrow{x}_{1/2} P_2\,|\,Q\} & \\
P\,|\,Q\{\xrightarrow{\overline{y}}_{1/3} P_1\,|\,Q, & \xrightarrow{y}_{1/6} P\,|\,Q_2, & \xrightarrow{x}_{1/6} P_2\,|\,Q, & \xrightarrow{z}_{1/3} P\,|\,Q_2\} \\
P\,|\,Q\{\xrightarrow{\overline{y}}_{1/3} P_1\,|\,Q, & \xrightarrow{y}_{1/6} P\,|\,Q_2, & \xrightarrow{x}_{1/3} P_2\,|\,Q, & \xrightarrow{y}_{1/6} P\,|\,Q_1\} \\
P\,|\,Q\{\xrightarrow{\overline{y}}_{1/3} P_1\,|\,Q, & \xrightarrow{y}_{1/6} P\,|\,Q_2, & \xrightarrow{y}_{1/6} P\,|\,Q_1, & \xrightarrow{z}_{1/3} P\,|\,Q_2\}
\end{array}
$$

$$
\begin{array}{llll}
P\,|\,Q\{\xrightarrow{y}_{1/6} P\,|\,Q_1, & \xrightarrow{z}_{1/3} P\,|\,Q_2, & \xrightarrow{x}_{1/2} P_2\,|\,Q\} & \\
P\,|\,Q\{\xrightarrow{y}_{1/6} P\,|\,Q_1, & \xrightarrow{z}_{1/3} P\,|\,Q_2, & \xrightarrow{x}_{1/6} P_2\,|\,Q, & \xrightarrow{z}_{1/3} P\,|\,Q_2\} \\
P\,|\,Q\{\xrightarrow{y}_{1/6} P\,|\,Q_1, & \xrightarrow{z}_{1/3} P\,|\,Q_2, & \xrightarrow{x}_{1/3} P_2\,|\,Q, & \xrightarrow{y}_{1/6} P\,|\,Q_1\} \\
P\,|\,Q\{\xrightarrow{y}_{1/6} P\,|\,Q_1, & \xrightarrow{z}_{1/3} P\,|\,Q_2, & \xrightarrow{y}_{1/6} P\,|\,Q_1, & \xrightarrow{z}_{1/3} P\,|\,Q_2\}
\end{array}
$$

$$
\begin{array}{llll}
P\,|\,Q\{\xrightarrow{\tau}_{1/6} P_1\,|\,Q_1, & \xrightarrow{\overline{y}}_{1/3} P_1\,|\,Q, & \xrightarrow{x}_{1/2} P_2\,|\,Q\} & \\
P\,|\,Q\{\xrightarrow{\tau}_{1/6} P_1\,|\,Q_1, & \xrightarrow{\overline{y}}_{1/3} P_1\,|\,Q, & \xrightarrow{x}_{1/6} P_2\,|\,Q, & \xrightarrow{z}_{1/3} P\,|\,Q_2\} \\
P\,|\,Q\{\xrightarrow{\tau}_{1/6} P_1\,|\,Q_1, & \xrightarrow{\overline{y}}_{1/3} P_1\,|\,Q, & \xrightarrow{x}_{1/3} P_2\,|\,Q, & \xrightarrow{y}_{1/6} P\,|\,Q_1\} \\
P\,|\,Q\{\xrightarrow{\tau}_{1/6} P_1\,|\,Q_1, & \xrightarrow{\overline{y}}_{1/3} P_1\,|\,Q, & \xrightarrow{y}_{1/6} P\,|\,Q_1, & \xrightarrow{z}_{1/3} P\,|\,Q_2\}
\end{array}
$$

$$
\begin{array}{llll}
P\,|\,Q\{\xrightarrow{\tau}_{1/6} P_1\,|\,Q_1, & \xrightarrow{z}_{1/3} P\,|\,Q_2, & \xrightarrow{x}_{1/2} P_2\,|\,Q\} & \\
P\,|\,Q\{\xrightarrow{\tau}_{1/6} P_1\,|\,Q_1, & \xrightarrow{z}_{1/3} P\,|\,Q_2, & \xrightarrow{x}_{1/6} P_2\,|\,Q, & \xrightarrow{z}_{1/3} P\,|\,Q_2\} \\
P\,|\,Q\{\xrightarrow{\tau}_{1/6} P_1\,|\,Q_1, & \xrightarrow{z}_{1/3} P\,|\,Q_2, & \xrightarrow{x}_{1/3} P_2\,|\,Q, & \xrightarrow{y}_{1/6} P\,|\,Q_1\} \\
P\,|\,Q\{\xrightarrow{\tau}_{1/6} P_1\,|\,Q_1, & \xrightarrow{z}_{1/3} P\,|\,Q_2, & \xrightarrow{y}_{1/6} P\,|\,Q_1, & \xrightarrow{z}_{1/3} P\,|\,Q_2\}
\end{array}
$$

2.3.2 The two ν rules

The four previous rules are the same for both semantics. But as restrictions can erase some actions (if $x \in fn(\mu) \wedge \neg(\mu = \overline{z}x \wedge z \neq x)$, then νx prevents P from doing μ) one has, in the probabilistic case, to re-normalize coefficients so that the sum stays equal to 1. In the quantitative case it is sufficient to erase transitions which do not satisfy the condition without modifying coefficients.

As for the rule COM, each of these two rules correspond to several rules of the classical π-calculus. In each rule, the first set is for the actions that output the name bound by the ν. The second set is for the action that do not involve any name bound by the ν. These corresponds to the classic rules OPEN and NU.

Quantitative case

$$\nu_q : \quad \frac{P\{\xrightarrow{\mu_i}_{p_i} P_i\} \quad \exists i.(x \notin fn(\mu_i) \vee (\mu_i = \overline{z}x \wedge z \neq x))}{\nu x.P\{\xrightarrow{\overline{z_i}(x)}_{p_i} P_i, \quad \mu_i = \overline{z_i}x, z_i \neq x\} \bigcup \{\xrightarrow{\mu_i}_{p_i} \nu x P_i, \quad x \notin fn(\mu_i)\}}$$

Probabilistic case The rule for the probabilistic case is obtained from previous one by renormalizing the coefficients.

$$\nu_p : \quad \frac{P\{\xrightarrow{\mu_i}_{p_i} P_i\} \quad \exists i.(x \notin fn(\mu_i) \vee (\mu_i = \overline{z}x \wedge z \neq x))}{\nu x.P\{\xrightarrow{\overline{z_i}(x)}_{q_i} P_i, \quad \mu_i = \overline{z_i}x, z_i \neq x\} \bigcup \{\xrightarrow{\mu_i}_{q_i} \nu x P_i, \quad x \notin fn(\mu_i)\}}$$

with $\forall i.q_i = p_i/(\Sigma_{j:x \notin fn(\mu_j) \vee (\mu_j = \overline{z}x \wedge z \neq x)} p_j)$

Note that each rule, except ν_q which is for the quantitative case, preserves the sum of the coefficients. Thus in the probabilistic case we derive only steps where the sum of the coefficients is equal to 1.

2.4 Weak steps

The weak steps are defined by:

$$\mathsf{wea1} : \quad \frac{P\{\xrightarrow{\mu_i}_{p_i} P_i\}}{P\{\stackrel{\mu_i}{\Longrightarrow}_{p_i} P_i\}}$$

$$\mathsf{wea2} : \quad \frac{P\{\stackrel{\mu_i}{\Longrightarrow}_{p_i} P_i\} \biguplus \{\stackrel{\tau}{\Longrightarrow}_q Q\} \quad Q\{\stackrel{\eta_j}{\Longrightarrow}_{r_j} R_j\}}{P\{\stackrel{\mu_i}{\Longrightarrow}_{p_i} P_i\} \biguplus \{\stackrel{\eta_j}{\Longrightarrow}_{q.r_j} R_j\}}$$

$$\mathsf{wea3} : \quad \frac{\forall n.P\{\stackrel{\mu_i}{\Longrightarrow}_{p_{in}} P_i\} \quad lim_{n \to \infty} p_{in} = \overline{p_i}}{P\{\stackrel{\mu_i}{\Longrightarrow}_{\overline{p_i}} P_i\}}$$

The first two rules are inspired from [5]. The last one is new and represents the limit case of the second one. Note that the limit only concerns probabilities.This last rule is very important because our encoding is based on a loop that allows to backtrack whenever we take the wrong decision. Eventually, the right decision will be taken with probability 1 but this may happen only in the limit.

Note that also here the sum of the coefficients is preserved.

3 Encoding of the mixed choice by the separates choices

We will show that in the probabilistic π-calculus without the nu operator and in the quantitative π-calculus, the mixed choice is encodable by the separate ones. In both cases we prove the correctness of our encoding by showing that it preserves a sort of weak bisimulation. The reason why we wrote "without the ν operator" is because at present we don't know how to encode the ν in a correct way in the probabilistic case. To translate it homomorphically does not work, because of the re-normalizations induced by the rule ν_p that interferes with weak steps. We will explain the problem in more details in Section 3.2.1.

3.1 The encoding

Our encoding is homomorphic w.r.t. all constructors except the choice. Thus we only explain the encoding of the choice. Let P be a mixed choice $\Sigma_{i\in I}(p_i, \overline{x_i}y_i).P_i + \Sigma_{j\in J}(q_j, \tau).Q_j + \Sigma_{k\in K}(r_k, x_k(z_k)).R_k$. If I or K are empty, then the choice is not mixed and $[\![P]\!] = P$. Otherwise we begin by making a blind choice between three branches corresponding to outputs, inputs or τ's. Then in each branch we make a (separate) choice between the outputs, inputs or τ. As one can go in the outputs branch (for instance) even if the context enforces communication on an input, we have to include a mechanism to backtrack. To this end, the separate choices contain also a τ-prefixed branch going back recursively to the beginning, with probability ϵ, which needs to be smaller than 1. This means that the process can, in principle, loop forever. However, such event will have probability the product of ϵ with itself infinitely many times, which is 0.

3.1.1 Encoding of the weighted mixed choice

$$\begin{aligned} [\![\nu x.P]\!] &= \nu x.[\![P]\!] \\ [\![P|Q]\!] &= [\![P]\!]|[\![Q]\!] \\ [\![rex_X.P]\!] &= rec_X.[\![P]\!] \\ [\![X]\!] &= X \end{aligned}$$

Let P be a mixed choice of the form $\Sigma_{i\in I}(p_i, \overline{x_i}y_i).P_i + \Sigma_{j\in J}(q_j, \tau).Q_j + \Sigma_{k\in K}(r_k, x_k(z_k)).R_k$. We define the encoding of P as follows:

$$\begin{aligned} [\![P]\!] = \quad rec_X.(\quad & (\Sigma_{i\in I}(p_i), \tau).P_{\text{send}} \\ & +(\Sigma_{j\in J}(q_j), \tau).P_\tau \\ & +(\Sigma_{k\in K}(r_k), \tau).P_{\text{receive}} \\ ) & \end{aligned}$$

where:

$$\begin{aligned} P_{\text{send}} = \quad & \Sigma_{i\in I}(\frac{p_i(1-\epsilon)}{\Sigma_{i\in I}(p_i)}, \overline{x_i}y_i).[\![P_i]\!] + (\epsilon, \tau).X \\ P_\tau = \quad & \Sigma_{j\in J}(\frac{q_j(1-\epsilon)}{\Sigma_{j\in J}(q_j)}, \tau).[\![Q_j]\!] + (\epsilon, \tau).X \\ P_{\text{receive}} = \quad & \Sigma_{k\in K}(\frac{r_k(1-\epsilon)}{\Sigma_{k\in K}(r_k)}, x_k(z_k)).[\![R_k]\!] + (\epsilon, \tau).X \end{aligned}$$

3.2 Correctness of the encoding

3.2.1 The structure of the proof

We establish the correctness of the encoding by Theorems 1 and 2 in the probabilistic case, and 3 and 4 in the quantitative case. In both cases the theorems correspond to a sort of weak bisimulation.

Theorems 1 and 3 correspond exactly to one direction of weak bisimulation. They state that if P can perform a step, then $[\![P]\!]$ can perform the corresponding weak step. To prove these results, we use Lemma 1. Essentially, the properties stated by points *i*-*v* of this lemma show that the weak variants of Rules SUM, REC, CONG, COM and ν_q, respectively, are sound with respect to Rules wea1, wea2 and wea3, i.e, the definition of $\Longrightarrow$. By "weak variant" of rule X here we mean the rule obtained by replacing $\rightarrow$ with $\Longrightarrow$ in X.

Unfortunately the same result does not hold for the rule ν_p. The following is a counterexample.

Example 2 *Let* $P = (1/2, c).0 + (1/2, \tau).X$ *and* $X = (1/2, b).0 + (1/2, a).0$. *We have* $P\{\xrightarrow{c}_{1/2} 0, \stackrel{b}{\Longrightarrow}_{1/4} 0, \stackrel{a}{\Longrightarrow}_{1/4} 0\}$. *If the equivalent of Lemma 1.v for* ν_p *were to hold, then we should have also* $\nu a.P\{\stackrel{c}{\Longrightarrow}_{2/3} 0, \stackrel{b}{\Longrightarrow}_{1/3} 0\}$ *(the coefficient are different from the* ν_q *case because we need to apply renormalization). However, we have only the strong steps* $\nu a.P\{\xrightarrow{c}_{1/2} 0, \xrightarrow{\tau}_{1/2} \nu a.X\}$ *and* $\nu a.X\{\xrightarrow{b}_1 0\}$, *so by Rule* wea2 *we can only obtain* $\nu a.P\{\stackrel{c}{\Longrightarrow}_{1/2} 0, \stackrel{b}{\Longrightarrow}_{1/2} 0\}$.

The discrepancy illustrated by the above counterexample is due to the fact that the renormalization in the weak variant of Rules SUM-COM would take place at a different time than in Rules weak1-weak3.

Lemma 1

i. *If* $Q\{\xrightarrow{\eta_j}_{q_j} Q_j\}$ *can be derived by using only* SUM, *then* $Q\{\stackrel{\eta_j}{\Longrightarrow}_{q_j} Q_j\}$,

ii. *If* $P[rec_X P/X]\{\stackrel{\mu_i}{\Longrightarrow}_{p_i} P_i\}$, *then* $rec_X P\{\stackrel{\mu_i}{\Longrightarrow}_{p_i} P_i\}$,

iii. *If* $P \equiv P'$, $P\{\stackrel{\mu_i}{\Longrightarrow}_{p_i} P_i\}$ *and* $\forall i. P_i \equiv P_i'$, *then* $P'\{\stackrel{\mu_i}{\Longrightarrow}_{p_i} P_i'\}$,

iv. *If* $P\{\stackrel{\mu_i}{\Longrightarrow}_{p_i} P_i\}$, $Q\{\stackrel{\eta_j}{\Longrightarrow}_{q_j} Q_j\}$, $\forall i,j. bn(\mu_i) \bigcap fn(Q_j) = \emptyset$ *and* $bn(\eta_j) \bigcap fn(P_i) = \emptyset$, *then* $(P|Q)\{\stackrel{\alpha_{i,j}}{\Longrightarrow}_{p_i q_j} R_{i,j}\}$, *where the* $\alpha_{i,j}$*'s and the* $R_{i,j}$*'s are defined as in the* COM *rule,*

v. *If* $P\{\stackrel{\mu_i}{\Longrightarrow}_{p_i} P_i\}$ *and* $\exists i.(x \notin fn(\mu_i) \vee (\mu_i = \overline{z}x \wedge z \neq x))$, *then* $\nu x.P\{\stackrel{\overline{z_i}(x)}{\Longrightarrow}_{p_i} P_i,\ \mu_i = \overline{z_i}x, z_i \neq x\} \bigcup \{\stackrel{\mu_i}{\Longrightarrow}_{p_i} \nu x P_i,\ x \notin fn(\mu_i)\}$.

In the other direction, to get a standard weak bisimulation, we should have that if $[\![P]\!]$ can perform a step, then P can perform the corresponding weak step. However we cannot get this result: since the translation divides a mixed choice into various separate choices, we get more possibilities in the translated term than in the original.

Indeed, as a translated term can only make a blind choice, to get a standard bisimulation we would need that the P_{send}, P_{receive} et P_τ resulting from the encoding are associated to P by the bisimulation. This does not work, one can easily see that these terms have in general steps different from those of the initial term.

However, all the weak steps that $[\![P]\!]$ can perform can be continued so to get one that is included in the ones of P. For instance, after $[\![P]\!]$ has performed the blind choice, it can perform the output choices, which will result in a weak step of the form $[\![P]\!]\{\stackrel{\overline{x_i}y_i}{\Longrightarrow}_{p_i} [\![P_i]\!]\} \biguplus \{\xrightarrow{\tau}_{\Sigma_j q_j} P_\tau\} \biguplus \{\xrightarrow{\tau}_{\Sigma_k r_k} P_{\text{receive}}\}$. By repeating this for the other two kinds of branches, we will get a weak step of the form $trP\{\stackrel{\overline{x_i}y_i}{\Longrightarrow}_{p_i} P_i\} \biguplus \{\stackrel{\tau}{\Longrightarrow}_{q_j} Q_j\} \biguplus \{\stackrel{x_k(z_k)}{\Longrightarrow}_{r_k} R_k\}$, that corresponds exactly to the step of P.

This situation is well known in the classical (non-probabilistic) setting: it is often the case that an encoding does not preserve the operational semantics at each step. In other words, it may happen that some intermediate states in the computation of an encoded process do not correspond to the encoding of any derivative of the original process. However, it is often the case that the encoding satisfies a property of the following form: if $[\![P]\!] \stackrel{\mu}{\Longrightarrow} Q$, then there exists P' such that $[\![P]\!] \stackrel{\mu}{\Longrightarrow} Q \stackrel{\tau}{\Longrightarrow} [\![P']\!]$ and $P \stackrel{\mu}{\Longrightarrow} P'$.

To formalize the above idea in the probabilistic setting, we introduce the notion of *completion* of a step:

Definition 1 *Let P be a process and let $P\{\stackrel{\mu_i}{\longrightarrow}_{p_i} P_i\}$ be one of its steps. A completion of this step is a weak step resulting of a derivation that begins with a* wea1 *applied to $P\{\stackrel{\mu_i}{\longrightarrow}_{p_i} P_i\}$, and then continues with successive applications of* wea2 *and* wea3*.*

Intuitively, completing a step means to explore the possible continuations of this step, going deeper each time we get a τ. The wea3 rule indeed only modifies coefficients and the rule wea2 permits to extend only silent transitions. Thus the completion of a step does not go further, in each branch, than the first non-silent action.

We remark that the notion of completion is quite robust, in the sense that it does not reduce the interactions possibilities, as shown by the following proposition.

Proposition 1 *Let P be a process. Let $P\{\stackrel{\mu_i}{\longrightarrow}_{p_i} P_i\}$ be one of its steps, and let $P\{\stackrel{\eta_j}{\Longrightarrow}_{q_j} Q_j\}$ be one of its completions. Consider now a process R, and let $R\{\stackrel{\alpha_k}{\longrightarrow}_{r_k} R_k\}$ be one of its steps. Let X be a step of the form $P \mid R\{\ldots\}$ obtained by applying the* COM *rule to $P\{\stackrel{\mu_i}{\longrightarrow}_{p_i} P_i\}$ and $R\{\stackrel{\alpha_k}{\longrightarrow}_{r_k} R_k\}$. By using Lemma 4 with premises $P\{\stackrel{\eta_j}{\Longrightarrow}_{q_j} Q_j\}$ and $R\{\stackrel{\alpha_k}{\longrightarrow}_{r_k} R_k\}$, we obtain a step Y of the form $P \mid R\{\ldots\}$ which is a completion of X.*

We are now ready to complete the formal assessment of the correctness of the encodings: Theorems 2 and 4 represent the other direction of bisimulation, but only "modulo completion". More precisely, they state that each step of $[\![P]\!]$ can be completed into a step corresponding to one of P.

3.2.2 Correctness results for the probabilistic case

Theorem 1 *In the probabilistic semantics, if $P\{\stackrel{\mu_i}{\longrightarrow}_{p_i} P_i\}$ can be derived without using the rule ν_p, then $[\![P]\!]\{\stackrel{\mu_i}{\Longrightarrow} p_i[\![P]\!]_i\}$.*

Theorem 2 *In the probabilistic semantics, if $[\![P]\!]\{\stackrel{\mu_i}{\longrightarrow}_{p_i} P_i\}$, then there exist Q_j's such that $[\![P]\!]\{\stackrel{\eta_j}{\Longrightarrow}_{q_j} [\![Q_j]\!]\}$ is a derivable completion of the above step and $P\{\stackrel{\eta_j}{\Longrightarrow}_{q_j} Q_j\}$.*

3.2.3 Correctness results for the quantitative case

Theorem 3 *In the quantitative semantics, if* $P\{\xrightarrow{\mu_i}_{p_i} P_i\}$*, then* $[\![P]\!]\{\stackrel{\mu_i}{\Longrightarrow} p_i[\![P]\!]_i\}$*.*

Theorem 4 *In the quantitative semantics, if* $[\![P]\!]\{\xrightarrow{\mu_i}_{p_i} P_i\}$*, then there exist* Q_j*'s such that* $[\![P]\!]\{\stackrel{\eta_j}{\Longrightarrow}_{q_j} [\![Q_j]\!]\}$ *is a derivable completion of the above step and* $P\{\stackrel{\eta_j}{\Longrightarrow}_{q_j} Q_j\}$*.*

3.3 Reduction of the size of a separate choice to two

This encoding reduces the language to a very simple form of separate choices: blind choices, and choices of size 2 in which one of the branches is prefixed by a τ.

The encoding works exactly in the same way as the previous one. We separate each branch of the separate choice, as we separated inputs from outputs and from τ. The theorem of correctness are identical.

$$
\begin{array}{rcl}
[\![\Sigma_{i\in I}(p_i,\mu_i).P_i]\!] & = & \\
 & rec_X.(& \\
 & & \Sigma_{i\in I} \quad (p_i,\tau).Q_i \\
 &) & \\
\text{where: } Q_i & = & \quad (1-\epsilon,\mu_i).[\![P_i]\!] + (\epsilon,\tau).X
\end{array}
$$

---

4 Expressiveness of the separate choices

We just showed how the mixed choice can be reduced to separate choices of size two. The question is now to compare the pair of the two separate choices to only one (typically the probabilistic asynchronous π-calculus proposed in [6]).

We conjecture that there is no encoding of the two separates choices by one of them.

Note that the encodings of Section 5 prove that input guarded choice and output guarded choice are equivalent. So we can restrict to compare the pair of separate choices to input guarded choice.

4.1 A failed attempt to encode the separate choices

We present here our best attempt to encode separate choices by input guarded choices, we discuss the reason why it does not work, and we conjecture that it is not possible to define such an encoding.

In a non probabilistic setting, various kinds of choice have been encoded by using the parallel operator. The basic idea is to put in parallel the branches of the choice, making sure that only one of them would be executed. See for instance [10, 9]. This idea however cannot work here, since the transitions would not be in the same step anymore. In other words, we cannot encode choice by parallelism since in the choice the decision between two branches is ruled by probabilities, while in the parallel product it is ruled by non-determinism. So a choice can only be translated in another choice similar for actions and probabilities.

So, our only hope is to translate the separate choices into input guarded choices. Let us use, for instance, the idea of [7] for translating output prefixes into inputs and asynchronous outputs:

$$\begin{array}{rcl} [\![x(y).P]\!] & = & \nu z(\overline{x}z \quad | \quad z(y).[\![P]\!]) \\ [\![\overline{x}y.Q]\!] & = & x(z).(\overline{z}y \quad | \quad [\![Q]\!]) \end{array}$$

This encoding can be lifted to choices in the following way:

$$\begin{array}{rcl} [\![(p, x(y)).P_x + (1-p, \tau).P_\tau]\!] & = & \\ & & \nu z(\\ & & \overline{x}z \quad | \quad (p, z(y)).[\![P_x]\!] + (1-p, \tau).[\![P_\tau]\!] \\ & &) \end{array}$$

$$\begin{array}{rcl} [\![(p_x, \overline{x}y).P_x + (1-p_x, \tau).P_\tau]\!] & = & \\ & & (p_x, x(z)).(\quad \overline{z}y|[\![P_x]\!] \quad) \\ & & + \quad (1-p_x, \tau).[\![P_\tau]\!] \end{array}$$

However this does not work since the translation of the output guarded choice can synchronize with $\overline{x}z$ while simultaneously the translation of the input guarded choice can execute its τ. The input choice performed its τ but the output choice began its branch of synchronization, and there is no way to go backwards. As we do not have output choices, the backward mechanism of the previous encoding can not work for the outputs. The translation of the output guarded choices is now in a deadlock which was not possible in the original term.

A solution could be to replace the τ of the translation of the input choice by an input on channel x so that the $\overline{x}z$ can synchronize either with this branch or with the input branch of the translation of the outputs guarded choice. But then another input guarded choice on x can interfere. It also

contains a $\overline{x}z'$ that can be received by the first input guarded choice while the $\overline{x}z$ would be received by the output guarded choice. Here again we get an unexpected deadlock.

5 Encodings between input and output guarded choices

Finally, we present encodings between input and output guarded choices. We use the idea of [7] already illustrated in previous section.

These encodings highlights the symmetry between the two choices and establishes that they are equally expressive. One has to note that the encoding only use asynchronous outputs (although they can be used within choices). So all the four languages with either input or output choice, and with asynchronous output or not, are equivalent.

Encoding the input guarded choice into the output one

$$\begin{array}{rcl} [\![\Sigma_{i\in I}(p_i, x_i(y_i)).P_i]\!] & = & (\nu_{z_i})_{i\in I}(\quad \Sigma_{i\in I}(p_i, \overline{x_i}z_i) \quad | \quad \Pi_{i\in I} z_i(y_i).[\![P_i]\!] \quad) \\ [\![\overline{x}y.P]\!] & = & x(z).(\overline{z}y \quad | \quad [\![P]\!]) \end{array}$$

Encoding the output guarded choice into the input one

$$\begin{array}{rcl} [\![x(y).P]\!] & = & \nu z(\overline{x}z | z(y).[\![P]\!]) \\ [\![\Sigma_{i\in I}(p_i, \overline{x_i}y_i).P_i]\!] & = & \Sigma_{i\in I}(p_i, x_i(z)).(\overline{z}y_i | [\![P_i]\!]) \end{array}$$

---

6 Discussion on the design of our calculus

In the design of our language we took some design decisions that are strictly related to the nature of choice in calculi with synchronous communication (in the sense of Concurrency Theory), such as CCS [8], TCSP [4], and ACP [2], besides the π-calculus. In these calculi, the commitment to a certain branch in a choice takes place at the same time as the transition, and in agreement with the partner of the communication action. In other words, there is no possibility for a partner to commit on a communication action

before the other partner decides what to do. As a consequence, the choice in these languages is memoryless.

In order to illustrate this point, consider the following example, suggested by Roberto Segala during a discussion:

- $P = (1/2, \overline{a}).0 + (1/2, \overline{b}).0$
- $Q = rec_X((1/2, a).X + (1/2, c).X)$

A possible step for $P \,|\, Q$ is:

$$P \,|\, Q\{\xrightarrow{\tau}_{1/4} Q, \xrightarrow{a}_{1/4} P \,|\, Q, \xrightarrow{c}_{1/2} P \,|\, Q\}$$

If the communication does not occur then the system $P \,|\, Q$ stays in the same state. Thus by scheduling always the above step, we have a computation where the communication occurs eventually with probability 1.

The key point is the above example is that it is not possible to commit unilaterally on a certain communication action, and to keep memory of such commitment in the subsequent transitions. Indeed if the synchronization does not occur, it could be the case that P had tried to make a $\overline{b}$. But at the next step, we can reschedule the same step (with probability 1/4 of synchronizing) without taking into account the fact that P had selected $\overline{b}$ in previous transition.

Roberto Segala did not find this acceptable: since $\overline{a}$ has probability 1/2 in P, he argued that there should be no context where the synchronization on the channel a occurs with a probability greater than 1/2.

In our opinion, on the contrary, it is perfectly plausible that the context may increase the probability of a communication event, and it is in line with the philosophy of synchronous communication in concurrency theory. So in the above example we find correct that the communication occurs eventually with probability 1.

7 Future work

We are interested in proving (or disproving) our conjecture (see part 4). We are currently looking at some impossibility results for probabilistic systems in the Distributed Computing literature.

We also aim at studying whether our results extend to the stochastic calculi proposed in [15, 13]. The author of [15] claims that the mixed choice is necessary to model certain biological phenomena, so it would be interesting to see in which way the stochastic aspects may interfere with our result. If, on the contrary, our result extends to the stochastic case, then it may be useful to know it in order to get a more efficient implementation.

Acknowledgments: We thank Roberto Segala for insightful discussion and for helping to fix some problems in the operational semantics.

References

[1] Falk Bartels, Ana Sokolova, and Erik P. de Vink. A hierarchy of probabilistic system types. *Theoretical Computer Science*, 327(1-2):3–22, 2004.

[2] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1,3):109–137, 1984.

[3] Luc Bougé. On the existence of symmetric algorithms to find leaders in networks of communicating sequential processes. *Acta Informatica*, 25(2):179–201, February 1988.

[4] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, 1984.

[5] Yuxin Deng and Catuscia Palamidessi. Axiomatizations for probabilistic finite-state behaviors. In *Proceedings of FOSSACS'05*, volume 3441 of *Lecture Notes in Computer Science*, pages 110–124. Springer, 2005. `http://www.lix.polytechnique.fr/~catuscia/papers/Prob_Axiom/fossacs05.pdf`.

[6] Oltea Mihaela Herescu and Catuscia Palamidessi. Probabilistic asynchronous π-calculus. In Jerzy Tiuryn, editor, *Proceedings of FOSSACS 2000 (Part of ETAPS 2000)*, volume 1784 of *Lecture Notes in Computer Science*, pages 146–160. Springer, 2000. `http://www.lix.polytechnique.fr/~catuscia/papers/Prob_asy_pi/fossacs.ps`.

[7] Kohei Honda and Mario Tokoro. An object calculus for asynchronous communication. In Pierre America, editor, *Proceedings of the European Conference on Object-Oriented Programming (ECOOP)*, volume 512 of *Lecture Notes in Computer Science*, pages 133–147. Springer, 1991.

[8] R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.

[9] Uwe Nestmann. What is a 'good' encoding of guarded choice? *Journal of Information and Computation*, 156:287–319, 2000. An extended abstract appeared in the *Proceedings of EXPRESS'97*, volume 7 of *ENTCS*.

[10] Uwe Nestmann and Benjamin C. Pierce. Decoding choice encodings. *Journal of Information and Computation*, 163:1–59, 2000. An extended abstract appeared in the *Proceedings of CONCUR'96*, volume 1119 of *LNCS*.

[11] Catuscia Palamidessi. Comparing the expressive power of the synchronous and the asynchronous pi-calculus. *Mathematical Structures in Computer Science*, 13(5):685–719, 2003. A short version of this paper appeared in POPL'97. `http://www.lix.polytechnique.fr/~catuscia/papers/pi_calc/mscs.pdf`.

[12] Catuscia Palamidessi and Oltea M. Herescu. A randomized encoding of the π-calculus with mixed choice. *Theoretical Computer Science*, 335(2-3):73–404, 2005. `http://www.lix.polytechnique.fr/~catuscia/papers/prob_enc/report.pdf`.

[13] Andrews Phillips and Luca Cardelli. A correct abstract machine for the stochastic pi-calculus. Technical report, 2005.

[14] Sylvain Pradalier, 2005. Rapport de stage. Master Parisien de Recherche en Informatique. `http://mpri.master.univ-paris7.fr/attached-documents/stages-2005-rapports/rapport-2005-pradalier.pdf`.

[15] Corrado Priami. Stochastic pi-calculus. *The Computer Journal*, 38(7):578–589, 1995.

[16] Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1995. Available as Technical Report MIT/LCS/TR-676.

[17] Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995. An extended abstract appeared in *Proceedings of CONCUR '94*, LNCS 836: 481-496.