A Probabilistic Applied Pi–Calculus[⋆]

Jean Goubault-Larrecq[1], Catuscia Palamidessi[2], and Angelo Troina[1,2]

[1] LSV - ENS Cachan
61 Avenue du Président Wilson, 94235 Cachan - France
{goubault,troina}@lsv.ens-cachan.fr
[2] LIX - École Polytechnique
Rue de Saclay, 91128 Palaiseau - France
{catuscia,troina}@lix.polytechnique.fr

Abstract. We propose an extension of the Applied Pi–calculus by introducing nondeterministic and probabilistic choice operators. The semantics of the resulting model, in which probability and nondeterminism are combined, is given by Segala's Probabilistic Automata driven by schedulers which resolve the nondeterministic choice among the probability distributions over target states. Notions of static and observational equivalence are given for the enriched calculus. In order to model the possible interaction of a process with its surrounding environment a labeled semantics is given together with a notion of weak bisimulation which is shown to coincide with the observational equivalence. Finally, we prove that results in the probabilistic framework are preserved in a purely nondeterministic setting.

1 Introduction

Security protocols are a critical element of the infrastructures needed for secure communication and processing information. Most security protocols are quite simple if only their length is considered. However, the properties they are supposed to ensure are extremely subtle, hence it is hard to get protocols correct just by informal reasoning. The history of cryptography and security protocols has a lot of examples where weaknesses of supposedly correct algorithms or protocols were discovered even years later. Thus, security protocols are excellent candidates for rigorous formal analysis. They are critical components of distributed security, are very easy to express and very difficult to evaluate by hand.

The use of formal methods for modeling and analyzing cryptographic protocols is now well-established. After the seminal paper by Dolev and Yao [9], which introduced a simple and intuitive description for cryptographic protocols, many alternative definitions have been proposed on the basis of several approaches, ranging from modal logics to process algebras.

Probabilistic models are nowadays widely used in the design and verification of complex systems in order to quantify unreliable or unpredictable behaviour

⋆ This work has been partially supported by the INRIA/ARC project ProNoBiS.

in security, performance and reliability analysis. Probability is taken into account when analyzing quantitative security properties (measuring, in a sense, the security level of the protocol) or when dealing with probabilistic protocols.

In [1], Abadi and Fournet introduce the Applied Pi–calculus, an extension of the Pi–calculus [15] with functions and equations allowing to treat messages not only as atomic names, but also as more complex terms constructed from names and functions. Such an extension gives rise to an important interaction between the *new* construct and value–passing communication. Applications to security are immediate, since the calculus allows to model unforgeable capabilities. Moreover, the Applied Pi–calculus permits a general and systematic development of syntax, operational semantics, equivalences and proof techniques.

It has been remarked that the Applied Pi–calculus, thanks to its explicit substitutions, is similar to Concurrent Constraint calculi like CCP [20], the ρ–calculus [17] and the CC–pi calculus [4].

Bisimulation relations [14] are well–established behavioural equivalences and are now widely used for the verification of properties of computer systems. Actually, a property can be verified by assessing the bisimilarity of the considered system with a specification one knows to enjoy the property. Moreover, bisimulations can sometimes be verified automatically thanks to successful implementations of verification tools like, e.g., the Concurrency Workbench [6] or the Mobility Workbench [23]. It is also extremely important for bisimulations to be congruences in order to account on compositional behavioural equivalences.

Contribution. In this paper we introduce an extension of the Applied Pi–calculus, called Probabilistic Applied Pi–calculus (PAPi for short), where both nondeterministic and probabilistic choices are taken into account. The semantics of the resulting model is given by Segala's Probabilistic Automata [21] driven by schedulers which resolve the nondeterministic choice among the probability distributions over target states (see [22]).

For the enriched calculus, we propose a notion of static equivalence (inherited from the Applied Pi–calculus) and a notion of probabilistic observational congruence. We also give a labeled semantics for modeling the interaction of a process with its surrounding environment. We derive a notion of weak bisimulation and show that it is a congruence relation coinciding with the observational equivalence defined for the unlabeled semantics. Finally, abstracting away from probabilities, we prove that results holding in the probabilistic version of the calculus are preserved within a purely nondeterministic framework.

As an application, we use PAPi to model and analyze the 1-out-of-2 oblivious transfer protocol given in [10]. Such a protocol makes use of cryptographic operations and randomization to achieve fairness in information exchange.

2 Preliminaries

In this section we recall some preliminary notions about terms, equational theories and probability distributions.

Terms. A signature $\Sigma = \{(f_1, a_1), \dots, (f_n, a_n)\}$ consists of a finite set of function symbols f_i each with an arity a_i. A function with arity 0 denotes a constant symbol. Given a signature Σ, and infinite set of names and variables, the set of *terms* is defined by the grammar:

$$M, N \ ::= \ a, b, c, \dots \quad \Big| \quad x, y, z, \dots \quad \Big| \quad f(M_1, \dots, M_l)$$

where M, N are terms, a, b, c are names, x, y, z are variables and $f(M_1, \dots, M_l)$ denotes function application with $(f, l) \in \Sigma$. With $\mathcal{T}$ we denote the set of terms. A term is called *ground* when it does not contain free variables and we use $\mathcal{T}_G$ to denote the set of ground terms. Metavariables u, v range over both names and variables. Tuples $u_1, \dots, u_l$ and $M_1, \dots, M_l$ are abbreviated to $\tilde{u}$ and $\widetilde{M}$, respectively.

As in [1], we rely on a sort system for terms. It may include a set of base types, such as $\mathsf{Integer}$, Key, etc., or simply a universal base type Data. In addition, if $\mathcal{S}$ is a sort, then $\mathsf{Channel}(\mathcal{S})$ is the sort of those channels that convey messages of sort $\mathcal{S}$. Variables and names can have any sort. We would use a, and c as channel names, s and k as names of some base type, and m and n as names of any sort. For simplicity, function symbols take arguments and produce results of base types only. In the following of the paper we always assume that terms are well-sorted and that substitutions preserve sorts.

Equational Theories. Given a signature Σ, we equip it with an *equational theory* E. An equational theory is a congruence over terms closed under substitutions of terms for variables (see [16, 8, 11]). We require this equational theory to be also closed under one-to-one substitutions on names. We use the standard notation $\Sigma \vdash M =_E N$ when the equation $M = N$ is in the theory E of Σ, and $\Sigma \nvdash M =_E N$ for the negation of $\Sigma \vdash M =_E N$.

In [1] one may found several examples of equational theories for the modeling of different kinds of cryptographic applications such as pairing, symmetric and asymmetric encryption, hashing, probabilistic encryption (modeled in a nondeterministic sense), signatures and XOR. We recall just some of them.

Algebraic data types such as pairs and lists could be defined by equipping a signature Σ with the binary function symbol pair and the unary function symbols fst and snd, with equations $\mathsf{fst}(\mathsf{pair}(x, y)) = x$ and $\mathsf{snd}(\mathsf{pair}(x, y)) = y$.

Now, the equational theory for algebraic data types consists of these equations and all the ones obtained by reflexivity, symmetry and transitivity and by substituting terms for variables. The sort system should enforce that fst and snd are applied only to pairs (alternatively a boolean function recognizing pairs may be added). Equations can be added to describe particular behaviours. For example, a constant symbol wrong can be considered such that $\mathsf{fst}(M) = \mathsf{snd}(M) = \mathsf{wrong}$ for appropriate ground terms M which are not pairs. In the following we use the abbreviations (M, N) for $\mathsf{pair}(M, N)$ and (L, M, N) for $\mathsf{pair}(\mathsf{pair}(L, M), N)$.

A one-way hash function can be represented as a unary function symbol h with no equations. The one-wayness of h is modeled by the absence of an inverse while the fact that h is collision-free results from $\mathsf{h}(M) = \mathsf{h}(N)$ only for $M = N$.

Symmetric cryptography (shared-key cryptography), is modeled via binary function symbols enc and dec for encryption and decryption with equation $\mathsf{dec}(\mathsf{enc}(x, y), y) = x$, where x represents the plaintext and y the key.

Asymmetric encryption can be modeled introducing two unary function symbols pk and sk for generating the public and the secret keys form a seed with the equation $\mathsf{dec}(\mathsf{enc}(x, \mathsf{pk}(y)), \mathsf{sk}(y)) = x$.

Sometimes, it may be useful to assume that encrypted messages come with sufficient redundancy such that decryption with a wrong key is evident. We may incorporate this property by adding equations $\mathsf{dec}(M, N) = \mathsf{wrong}$ for all ground terms M and N such that $M \neq \mathsf{enc}(L, N)$ for all L.

Probability Measures. A *discrete probability measure* over a set X is a function $\mu : 2^X \to [0, 1]$ such that $\mu(X) = 1$ and for each countable family $\{X_i\}$ of pairwise disjoint elements of 2^X, $\mu(\cup_i X_i) = \sum_i \mu(X_i)$. We adopt the convenient abuse of notation $\mu(x)$ for $\mu(\{x\})$. Let us denote by $D(X)$ the set of discrete probability measures over X. Given an element $x \in X$, we denote by δ_x the *Dirac measure* on x, namely, the probability measure μ such that $\mu(x) = 1$.

Given two probability measures μ_1, μ_2 and a real number $p \in [0, 1]$, we define the *convex combination* $\mu_1 +_p \mu_2$ to be the probability measure μ such that for each set $Y \in 2^X$, $\mu(Y) = p \cdot \mu_1(Y) + (1 - p) \cdot \mu_2(Y)$.

Recall that any discrete probability measure is the countable linear combination $\sum_{x.\mu(x)\neq 0} \mu(x) \cdot \delta_x$.

3 The Probabilistic Applied Pi Calculus

In this section we introduce the Probabilistic Applied Pi–calculus (PAPi).

3.1 Syntax

The grammar of PAPi processes is obtained by extending the one for the Applied Pi–calculus with a nondeterministic ($+$) and a probabilistic ($\oplus_p$) choice operator:

$$\begin{aligned} P, Q \ ::= \ & \mathbf{0} \ \mid \ \overline{u}\langle M \rangle.P \ \mid \ u(x).P \ \mid \ P + Q \ \mid \ P \oplus_p Q \ \mid \\ & P \mid Q \ \mid \ !P \ \mid \ \nu n.P \ \mid \ \text{if } M = N \text{ then } P \text{ else } Q \end{aligned}$$

The null process $\mathbf{0}$ does nothing; $\overline{u}\langle M \rangle.P$ outputs the term M on channel u and then behaves like P; $u(x).P$ is ready to perform an input on channel u, then to behave like P with the actual received message replacing the formal parameter x; $P + Q$ denotes a process which may behave either like P or Q; $P \oplus_p Q$ behaves like P with probability p, like Q with probability $1 - p$; $P \mid Q$ is the parallel composition of P and Q; the replication $!P$ behaves as an infinite number of copies of P running in parallel; $\nu n.P$ generates a fresh private name

n and then behaves like P; if $M = N$ then P else Q is the usual conditional process, it behaves like P if $M = N$ and like Q otherwise. Note that $M = N$ represents equality (i.e. with respect to some equational theory) rather than syntactic identity. We may omit a process when it is equal to $\mathbf{0}$.

As for the Applied Pi–calculus, we extend plain processes with *active substitutions*:

$$A, B ::= P \quad | \quad \nu n.A \quad | \quad \nu x.A \quad | \quad A \,|\, B \quad | \quad \{M/x\}$$

where P is a plain process. We denote with $\mathcal{A}$ the set of extended processes. We write $\{M/x\}$ for the active substitution that replaces the variable x with the term M. The substitution $\{M/x\}$ is like *let* $x = M$ *in...*, with the ability to *float* and to apply to any process that comes in contact with it. By applying a restriction $\nu x.(\{M/x\} \,|\, P)$ we obtain exactly *let* $x = M$ *in* P. Intuitively, a substitution $\{M/x\}$ denotes either a static public information known to every participant of the protocol, or it may appear when the term M has been sent to the environment, and the environment may not contain the atomic names appearing in M; in this situation, the variable x is just a way to refer to M. We write $\{M_1/x_1, \ldots, M_l/x_l\}$ for the parallel substitutions $\{M_1/x_1\} \,|\, \ldots \,|\, \{M_l/x_l\}$. We denote substitutions by σ, the image of a variable x according to σ as $x\sigma$ and the result of applying σ to the free variables of a term T as $T\sigma$. In the following we identify the empty frame and the null process $\mathbf{0}$.

Extending the sort system for terms, we rely on a sort system for extended processes. This should enforce that M and N are of the same sort in the conditional expression, that u has sort $\mathsf{Channel}(\mathcal{S})$ for some $\mathcal{S}$ in the input and output expressions, and that x and M have the corresponding sort $\mathcal{S}$ in those expressions. As done before, we omit the details of the sort system, and we just assume that extended processes are well-sorted.

Names and variables have scopes which are delimited by restrictions and by inputs. As usual, we denote with $fv(A)$ and $fn(A)$ the *free* variables and names of A which do not occur within the scope of any binder νx and $u(x)$. With $bv(A)$ and $bn(A)$ we denote the *bound* variables and names of A, respectively.

An extended process is *closed* when every variable is either bound or defined by an active substitution. With $\mathcal{A}_C$ we denote the set of closed extended processes. We may use the abbreviation $\nu\tilde{u}$ for the (possibly empty) series of pairwise-distinct binders $\nu u_1.\nu u_2 \ldots \nu u_l$.

Intuitively, we may see extended processes as plain processes extended with a context for the interpretation of their variables. As usual, an *evaluation context* is an expression (an extended process) with a hole. Formally, an evaluation context $C[\_]$ is defined by the following grammar:

$$C[\_] ::= \Box \quad | \quad \nu n.C[\_] \quad | \quad \nu x.C[\_] \quad | \quad A \,|\, C[\_] \quad | \quad C[\_] \,|\, A$$

where $A \in \mathcal{A}$ is an extended process. A context $C[\_]$ *closes* A when $C[A]$ is closed.

A *frame* is an extended process built up from $\mathbf{0}$ and active substitutions by parallel composition and restriction. The domain $dom(\varphi)$ of a frame φ is the set of variables that φ exports (those variables x for which φ has an active substitution $\{M/x\}$ not under a restriction on x). We assume all substitutions in a frame to be cycle-free, and that there is at most one substitution for each variable (and exactly one when the variable is restricted).

A frame can be viewed as an approximation of an extended process A that accounts for the static knowledge exposed by A to its environment, but not for A's dynamic behaviour. Given a probabilistic extended process A, with $\varphi(A)$ we denote the frame obtained from A by replacing every plain process embedded in A with $\mathbf{0}$. For example, given the process $A = (P \oplus_p Q) \mid \{M/x\} \mid \{N/x\}$, we have that $\varphi(A) = \mathbf{0} \mid \{M/x\} \mid \{N/x\}$. The domain $dom(A)$ of A is the domain of its frame $\varphi(A)$; namely, $dom(A) = dom(\varphi(A))$.

3.2 Semantics

Structural congruence ($\equiv$) is the smallest equivalence relation on extended processes that is closed (i) by α-conversion on both names and variables, (ii) by application of evaluation contexts, and such that:

$$\begin{array}{c}
(\textsc{Par-}\mathbf{0}) \quad A \equiv A \mid \mathbf{0} \qquad (\textsc{Par-C}) \quad A \mid B \equiv B \mid A \\
(\textsc{Par-A}) \quad A \mid (B \mid C) \equiv (A \mid B) \mid C \qquad (\textsc{Repl}) \quad !P \equiv P \mid !P \\
(\textsc{New-}\mathbf{0}) \quad \nu n.\mathbf{0} \equiv \mathbf{0} \qquad (\textsc{New-C}) \quad \nu u.\nu v.A \equiv \nu v.\nu u.A \\
(\textsc{New-Par}) \quad A \mid \nu u.B \equiv \nu u.(A \mid B) \text{ if } u \notin fv(A) \cup fn(A) \\
(\textsc{Alias}) \quad \nu x.\{M/x\} \equiv \mathbf{0} \qquad (\textsc{Subst}) \quad \{M/x\} \mid A \equiv \{M/x\} \mid A\{M/x\} \\
(\textsc{Rewrite}) \quad \{M/x\} \equiv \{N/x\} \text{ if } \Sigma \vdash M =_E N
\end{array}$$

Rules for parallel composition and restriction are standard. ALIAS enables the introduction of an arbitrary active substitution, SUBST describes the application of an active substitution to a process in contact with it, and REWRITE deals with equational term rewriting. As pointed out in [1], ALIAS and SUBST yield $A\{M/x\} \equiv \nu x.(\{M/x\} \mid A)$ for $x \notin fv(M)$.

We let μ range over distributions over the classes of extended processes defined by the structural congruence relation. Namely, $\mu : 2^{\mathcal{A}/\equiv} \to [0,1]$. In the following we abbreviate $\mu([B])$ with $\mu(B)$, where $[B]$ is the equivalence class of B up to structural congruence $\equiv$.

The *internal probabilistic reduction* $A \to \mu$, which describes a transition that leaves from A and leads to a probability distribution μ, is the smallest relation satisfying the following axioms:

$$\begin{array}{c}
(\textsc{Id}) \quad P \to \delta_P \qquad (\textsc{Comm}) \quad \overline{a}\langle x\rangle.P \mid a(x).Q \to \delta_{P \mid Q} \qquad (\textsc{NdBran}) \quad \dfrac{P \to \mu}{P+Q \to \mu} \\
(\textsc{PrBran}) \quad \dfrac{P \to \mu_1 \quad Q \to \mu_2}{P \oplus_p Q \to \mu_1 +_p \mu_2} \qquad (\textsc{Then}) \quad \text{if } M = M \text{ then } P \text{ else } Q \to \delta_P \\
(\textsc{Else}) \quad \text{if } M = N \text{ then } P \text{ else } Q \to \delta_Q \quad \text{ for } M, N \in \mathcal{T}_G \text{ s.t.} \Sigma \not\vdash M =_E N \\
(\textsc{EvCon}) \quad \dfrac{A \to \mu}{C[A] \to \mu_C}
\end{array}$$

A stuttering reduction (Id) is needed to deal with $+$ and $\oplus_p$ (see Example 1). Communication (Comm) is kept simple considering as a variable the message sent. There is no loss of generality since Alias and Subst can introduce a variable to stand for a term (see [1]). Nondeterministic branching (NdBran) is as usual, the symmetric reduction is omitted. Probabilistic branching (PrBran) results from the convex combination of probability measures. Comparisons (Then and Else) rely on the underlying equational theory E; using Else may sometimes require to apply active substitutions in the context in order to get ground terms M and N. Note that the only rule that gives rise to a probabilistic choice is PrBran, the other ones just return a Dirac measure.

Since reduction rules should be closed under application of evaluation contexts, we need to define extensions of the distributions μ such that given $A \to \mu$ we could define μ_C such that $C[A] \to \mu_C$. Formally, given an evaluation context $C[\_]$ and a distribution μ, we define the unique distribution μ_C such that $\mu_C(C[A]) = \mu(A)$. For example, with $\mu_{\square \,|\, B}$ we denote the distribution μ' such that $\mu'(A \,|\, B) = \mu(A)$, with $\mu_{\nu_u.\square}$ we denote the distribution μ' such that $\mu'(\nu u.A) = \mu(A)$.

Example 1. Consider the process $A = (\overline{a}\langle M\rangle + \overline{b}\langle M\rangle) \oplus_p \overline{c}\langle M\rangle$. We have $A \to \mu$ and $A \to \mu'$, where $\mu = \delta_{\overline{a}\langle M\rangle} +_p \delta_{\overline{c}\langle M\rangle}$ and $\mu' = \delta_{\overline{b}\langle M\rangle} +_p \delta_{\overline{c}\langle M\rangle}$. Moreover, we have $A \,|\, B \to \mu_{\square \,|\, B}$ and $A \,|\, B \to \mu'_{\square \,|\, B}$ for any process B.

There is a step from a process A to a process B through the distribution μ (denoted $A \to_\mu B$) if $A \to \mu$ and $\mu([B]) > 0$.

An *execution* of A is a finite (or infinite) sequence of steps $e = A \to_{\mu_1} A_1 \to_{\mu_2} \ldots \to_{\mu_k} A_k$, where $A_0, \ldots, A_k \in \mathcal{A}$ and $\mu_i \in D(\mathcal{A}/_\equiv)$. With $Exec_A$ we denote the set of executions starting from A. For the finite execution $e = A \to_{\mu_1} A_1 \to_{\mu_2} \ldots \to_{\mu_k} A_k$ we define $last(e) = A_k$ and $|e| = k$. For any $j \leq |e|$, with e^j we define the sequence of steps $A \to_{\mu_1} A_1 \to_{\mu_2} \ldots \to_{\mu_j} A_j$.

Finally, with $e\!\uparrow$ we denote the set of executions e' such that $e \leq_{prefix} e'$, where $\leq_{prefix}$ is the usual prefix relation over sequences.

Example 2. Consider again process A of Example 1, and process $B = a(x)$. We have $A \,|\, B \to_{\mu_{\square \,|\, B}} \overline{a}\langle M\rangle \,|\, a(x) \to_{\delta_{\mathbf{0}}} \mathbf{0}$, with $\mu = \delta_{\overline{a}\langle M\rangle} +_p \delta_{\overline{c}\langle M\rangle}$ and $\overline{a}\langle M\rangle \,|\, a(x) \equiv \nu x.(\overline{a}\langle x\rangle \,|\, a(x) \,|\, \{M/x\})$. Note that we also have $A \,|\, B \to_{\mu_{\square \,|\, B}} \overline{c}\langle M\rangle \,|\, a(x)$.

Since we allow nondeterministic choices, an extended process may behave in several different ways. Intuitively, the nondeterministic choice is among the possible probability distributions that a process may follow. Given a process A, we denote with $behave(A)$ the set of the possible behaviours of A, i.e., $behave(A) = \{\mu \,|\, A \to \mu\}$. Hence, each possible probabilistic transition $A \to_\mu$ can be seen as arising from a *scheduler* resolving the nondeterminism in A (see [22]). A *scheduler* is a total function F assigning to a finite execution e a distribution $\mu \in behave(last(e))$. Given a scheduler F and a process A, we define $Exec_A^F$ as the set of executions starting from A and driven by F, namely the set of executions $\{e = A \to_{\mu_1} A_1 \to_{\mu_2} A_2 \to_{\mu_3} \ldots \,|\, \forall i, \mu_i(A_i) > 0 \text{ where } \mu_i = F(e^{i-1})\}$.

Given the finite execution $e = A \rightarrow_{\mu_1} A_1 \rightarrow_{\mu_2} \ldots \rightarrow_{\mu_k} A_k \in Exec_A^F$, we define $P_A^F(e) = \mu_1(A_1) \cdot \ldots \cdot \mu_k(A_k)$.

We define the probability space on the executions starting from a given process $A \in \mathcal{A}$, as follows. Given a scheduler F, $\sigma Field_A^F$ is the smallest sigma field on $Exec_A^F$ that contains the basic cylinders $e\uparrow$, where $e \in Exec_A^F$. The probability measure $Prob_A^F$ is the unique measure on $\sigma Field_A^F$ such that $Prob_A^F(e\uparrow) = P_A^F(e)$.

Example 3. Consider again the process A of Example 1, and the scheduler F such that $F(A) = \mu = \delta_{\overline{a}\langle M\rangle} +_p \delta_{\overline{c}\langle M\rangle}$. We have that the executions $e = A \rightarrow_\mu \overline{a}\langle M\rangle$ and $e' = A \rightarrow_\mu \overline{c}\langle M\rangle$ are in $Exec_A^F$ with $P_A^F(e) = p$ and $P_A^F(e') = 1 - p$. Note that with the chosen F, action $\overline{b}\langle M\rangle$ is never performed.

Given a scheduler F, a process A and a measurable set of processes $H \subseteq \mathcal{A}$, with $Exec_A^F(H)$ we denote the set of executions starting from A that cross a process in the set H. Namely, $Exec_A^F(H) = \{e \in Exec_A^F \mid last(e^i) \in H, \text{ for some } i\}$.

We define the probability of reaching a process in H starting from A according to the policy given by F as $Prob_A^F(H) = Prob_A^F(Exec_A^F(H))$.

4 Equivalences

In this section we recall the definition of *static equivalence* for frames introduced in [1]. We also introduce a notion of *observational congruence* allowing to argue when PAPi extended processes cannot be distinguished by any context. Contexts can be used to represent active attackers and observational congruence may capture security properties. For example, secrecy and authentication properties have been defined in this way in [2] for the Spi–calculus.

4.1 Static Equivalence

Two frames should be considered equivalent when they behave equivalently when applied to terms obeying a certain equational theory E. We denote this equivalence (also called *static equivalence*) with $\approx_E$. As pointed out in [1], defining a static equivalence in presence of the ν construct becomes somehow delicate. Consider, for instance, the three frames:

$$\varphi_0 = \nu k.\{k/x\} \mid \nu s.\{s/y\} \quad \varphi_1 = \nu k.\,\{\mathsf{f}(k)/x, \mathsf{g}(k)/y\} \quad \varphi_2 = \nu k.\,\{k/x, \mathsf{f}(k)/y\}$$

where f and g are unary functions with no equations (two independent one-way hash functions). In φ_0, since k and s are new, variables x and y are mapped to unrelated values different from any value a context may build. This also holds for φ_1 (even if $\mathsf{f}(k)$ and $\mathsf{g}(k)$ are based on the same fresh value, they look unrelated). Thus, a context obtaining values for x and y cannot distinguish between φ_0 and φ_1. However, a context may discriminate φ_2 by checking the predicate $\mathsf{f}(x) = y$. Hence, static equivalence is defined so that $\varphi_0 \approx_E \varphi_1 \not\approx_E \varphi_2$.

Definition 1. *Given an equational theory E, two terms M and N are equal in the frame $\varphi \equiv \nu\tilde{n}.\sigma$ (written $(M =_E N)\varphi$), if and only if $M\sigma =_E N\sigma$ and $\{\tilde{n}\} \cap (fn(M) \cup fn(N)) = \emptyset$.*

Hence, for the previous example, we have $(\mathsf{f}(x) = y)\varphi_2$ but not $(\mathsf{f}(x) = y)\varphi_0$.

Definition 2. *Given an equational theory E, two closed frames φ and ψ are statically equivalent (written $\varphi \approx_E \psi$) when $dom(\varphi) = dom(\psi)$ and for all terms M and N, $(M =_E N)\varphi$ iff $(M =_E N)\psi$.*

We say that two closed extended processes A and B are statically equivalent (written $A \approx_E B$) iff $\varphi(A) \approx_E \varphi(B)$.

Note that deciding static equivalence can be quite hard to check (it depends on E and Σ) [7]. The next lemma, proved in [1], states a basic property of $\approx_E$.

Lemma 1. *Static equivalence is closed by structural equivalence, by reduction, and by application of closing evaluation contexts.*

4.2 Observational Congruence

We write $A \Downarrow_p^F a$ (a *probabilistic barb*) when A can send a message on a with probability p according to the scheduler F, namely, when $Prob_A^F(H) = p$ where $A' \in H$ if and only if $A' = C[\overline{a}\langle x\rangle.P]$ for some evaluation context $C[\_]$ that does not bind a. Notice that the set of executions starting from A and crossing a process in H is measurable since it can be seen as the countable union of measurable sets $\bigcup_{C,P,x,e.e\in Exec_A^F \wedge last(e)=C[\overline{a}\langle x\rangle.P]} e\!\uparrow$.

Definition 3. Observational congruence *($\approx$) is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A\mathcal{R}B$ implies:*

1. *for all schedulers F such that $A \Downarrow_p^F a$, there exists a scheduler F' such that $B \Downarrow_p^{F'} a$;*
2. *for all schedulers F and classes $\mathcal{C} \in \mathcal{A}_C/_\mathcal{R}$, there exists a scheduler F' such that $Prob_A^F(\mathcal{C}) = Prob_B^{F'}(\mathcal{C})$;*
3. *$C[A]\mathcal{R}C[B]$ for all closing evaluation contexts $C[\_]$.*

The quantification on the schedulers means, intuitively, that given $A \approx B$, for any possible behaviour (scheduler) of A there exists an analogous behaviour of B and viceversa.

As pointed out in [1], if $A \approx B$, then, for any test C of the form if $M = N$ then $\overline{a}\langle s\rangle$ else $\mathbf{0}$, where a does not occur in A or B, $A \mid C$ and $B \mid C$ should have the same barbs, thus implying static equivalence for A and B. As a consequence, the following lemma holds, stating that observational congruence is finer than static equivalence.

Lemma 2. *Given $A, B \in \mathcal{A}$, $A \approx B$ implies $A \approx_E B$.*

4.3 Labeled Semantics and Weak Bisimulation

In process calculi theory, a labeled semantics usually allows describing the potential interactions of a process with other ones that could occur in its environment. Such interactions are modeled by allowing the process to perform as many transitions as its active actions are, each transition having the corresponding action as label and leading to a new process which corresponds to the result of the execution of that action. Moreover, a labeled semantics may include silent (or internal) transitions, usually labeled with τ, which describe the internal activity of the process, namely the interactions occurring between internal components of the system. Furthermore, the actions performed may include parameters. As an example, since the action of sending or receiving a message on a channel may require the transmitted message as parameter, one should explicitly show the parameter within the transition label.

Thus, to model the interaction of PAPi processes with the environment, a labeled operational semantics can be provided which defines a relation $A \xrightarrow{\alpha} \mu$, where α is a label of one of the following forms:

- the symbol τ (corresponding to an internal reduction);
- a label $a(M)$, where M may contain names and variables (corresponding to an input of M on a);
- a label $\overline{a}\langle u\rangle$ or $\nu u.\overline{a}\langle u\rangle$, where u is either a channel name or a variable of base type (corresponding to an output of u on a).

In addition to the structural congruence rules and the internal reduction semantics of Section 3.2 (where each reduction rule should be equipped with the label τ), we adopt the following rules:

$$\text{(In)}\quad a(x).P \xrightarrow{a(M)} \delta_{P\{M/x\}} \qquad \text{(Out-Atom)}\quad \overline{a}\langle u\rangle.P \xrightarrow{\overline{a}\langle u\rangle} \delta_P$$

$$\text{(Open-Atom)}\quad \frac{A \xrightarrow{\overline{a}\langle u\rangle} \mu \quad u \neq a}{\nu u.A \xrightarrow{\nu u.\overline{a}\langle u\rangle} \mu} \qquad \text{(Scope)}\quad \frac{A \xrightarrow{\alpha} \mu \quad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.\mu}$$

$$\text{(Par)}\quad \frac{A \xrightarrow{\alpha} \mu \quad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset}{A \,|\, B \xrightarrow{\alpha} \mu \,|\, B}$$

$$\text{(Struct)}\quad \frac{A \equiv B \quad B \xrightarrow{\alpha} \mu}{A \xrightarrow{\alpha} \mu}$$

There is a step from a process A to a process B through the distribution μ with label α (denoted $A \xrightarrow{\alpha}_{\mu} B$) if $A \xrightarrow{\alpha} \mu$ and $\mu(B) > 0$. Given a process A, different reaction rules $A \xrightarrow{\alpha} \mu$ may be applied according to α and μ. As a consequence, we redefine the set of possible behaviours of A as $\mathit{behave}_l(A) = \{(\alpha, \mu) \mid A \xrightarrow{\alpha} \mu\}$.

A *labeled execution* of A is a finite (or infinite) sequence of steps $e = A \xrightarrow{\alpha_1}_{\mu_1} A_1 \xrightarrow{\alpha_2}_{\mu_2} \dots \xrightarrow{\alpha_k}_{\mu_k} A_k$, where $A_0, \dots, A_k \in \mathcal{A}$ and $\mu_i \in D(\mathcal{A}/_{\equiv})$. With abuse of notation, we define Exec_A, $\mathit{last}(e) = A_k$, $|e|$, e^j and $e\uparrow$ as for unlabeled executions.

Executions arise by resolving the nondeterminism on both α and μ. As a consequence, a scheduler for the labeled semantics is a function F assigning to a finite labeled execution e a pair $(\alpha, \mu) \in \mathit{behave}_l(\mathit{last}(e))$.

Given a scheduler F and a process A, we define $Exec_A^F$ as the set of executions starting from A and driven by F, namely the set of executions $\{e = A \xrightarrow{\alpha_1}_{\mu_1} A_1 \xrightarrow{\alpha_2}_{\mu_2} A_2 \xrightarrow{\alpha_3}_{\mu_3} \ldots \mid \forall i, \mu_i(A_i) > 0$ where $(\alpha_i, \mu_i) = F(e^{i-1})\}$. Given the finite execution $e = A \xrightarrow{\alpha_1}_{\mu_1} A_1 \xrightarrow{\alpha_2}_{\mu_2} \ldots \xrightarrow{\alpha_k}_{\mu_k} A_k \in Exec_A^F$, we define $P_A^F(e) = \mu_1(A_1) \cdot \ldots \cdot \mu_k(A_k)$.

Example 4. Consider the process A of Example 1 and the scheduler F such that $F(A) = (\tau, \mu)$, with μ defined as in Example 1, and, trivially, $F(A \xrightarrow{\tau}_{\mu} \overline{a}\langle M\rangle) = (\overline{a}\langle M\rangle, \delta_{\mathbf{0}})$ and $F(A \xrightarrow{\tau}_{\mu} \overline{c}\langle M\rangle) = (\overline{c}\langle M\rangle, \delta_{\mathbf{0}})$. We have $e = A \xrightarrow{\tau}_{\mu} \overline{a}\langle M\rangle \xrightarrow{\overline{a}\langle M\rangle}_{\delta_{\mathbf{0}}} \mathbf{0}$ and $e' = A \xrightarrow{\tau}_{\mu} \overline{c}\langle M\rangle \xrightarrow{\overline{c}\langle M\rangle}_{\delta_{\mathbf{0}}} \mathbf{0}$ with $P_A^F(e) = p$ and $P_A^F(e') = 1 - p$. Note, again, that with such a scheduler the label $\overline{b}\langle M\rangle$ does never appear. Also note that the process $\nu c.A$ may reach with probability $(1-p)$ the process $\nu c.\overline{c}\langle M\rangle$ from which it cannot perform any other step.

Again, given a scheduler F, a finite execution e and a measurable set H, $Prob_A^F(e\uparrow)$, $Exec_A^F(H)$ and $Prob_A^F(H)$ are defined analogously as for the unlabeled case. Let $Exec_A^F(\tau^*\alpha\tau^*, H)$ be the set of executions that, starting from A, lead to a process in H via an execution performing an α action preceded and followed by an arbitrary number of τ steps. We define the probability $Prob_A^F(\tau^*\alpha\tau^*, H) = Prob_A^F(Exec_A^F(\tau^*\alpha\tau^*, H))$.

Definition 4. Weak bisimulation *($\approx_l$) is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A\mathcal{R}B$ implies:*

1. *$A \approx_E B$;*
2. *for all schedulers F and classes $\mathcal{C} \in \mathcal{A}_C/_{\mathcal{R}}$, there exists a scheduler F' such that $Prob_A^F(\mathcal{C}) = Prob_B^{F'}(\mathcal{C})$;*
3. *for all schedulers F, $\alpha \neq \tau$ and classes $\mathcal{C} \in \mathcal{A}_C/_{\mathcal{R}}$, there exists a scheduler F' such that $Prob_A^F(\alpha, \mathcal{C}) = Prob_B^{F'}(\tau^*\alpha\tau^*, \mathcal{C})$, with $fv(\alpha) \subseteq dom(A)$ and $bn(\alpha) \cap fn(B) = \emptyset$.*

The following lemma states that given $A \approx_l B$ and a closing evaluation context $C[\_]$, $C[A] \approx_l C[B]$ holds.

Lemma 3. *$\approx_l$ is closed under application of closing evaluation contexts.*

Proof. (Sketch) Given $A \approx_l B$ we should prove that $C[A] \approx_l C[B]$ for any closing evaluation context $C[\_]$. The proof is done by induction on the structure of $C[\_]$.

– For $C[\_] = \Box$ we have $C[A] = A$ and $C[B] = B$, thus the result trivially holds.
– For $C[\_] = \bar{C} \mid \Box$ or $C[\_] = \Box \mid \bar{C}$ with $\bar{C} \in \mathcal{A}_C$, the only non trivial case is for $\bar{C} =$ if $M = N$ then P else Q. Now, since by definition of $\approx_l$ we have that $A \approx_E B$, we have that A and B should pass the same tests (in this case defined by all the possible contexts $\bar{C}$), thus implying that $\bar{C} \mid A \approx_l \bar{C} \mid B$.

– For $C[\_] = \nu u.\Box$, we have that for any label α such that $u \notin \alpha$, scheduler F and equivalence class $\mathcal{C}$, $Prob^F_A(\alpha, \mathcal{C}) = Prob^F_B(\alpha, \mathcal{C}) = Prob^F_{C[A]}(\alpha, \mathcal{C}) = Prob^F_{C[B]}(\alpha, \mathcal{C})$. For any α such that $\alpha = \overline{a}\langle u \rangle$ and $a \neq u$, $Prob^F_A(\alpha, \mathcal{C}) = Prob^F_B(\alpha, \mathcal{C}) = Prob^F_{C[A]}(\nu u.\alpha, \mathcal{C}) = Prob^F_{C[B]}(\nu u.\alpha, \mathcal{C})$. For any α such that $\alpha = \overline{a}\langle u \rangle$ and $a = u$, $Prob^F_{C[A]}(\alpha, \mathcal{C}) = Prob^F_{C[B]}(\alpha, \mathcal{C}) = Prob^F_{C[A]}(\nu u.\alpha, \mathcal{C}) = Prob^F_{C[B]}(\nu u.\alpha, \mathcal{C}) = 0$. □

We can also show that $\approx_l$ and $\approx$ coincide. Even if the notion of weak bisimulation does not include an explicit condition about contexts, it is still closed under application of evaluation contexts. As a consequence, $\approx_l$ is simpler than the notion of observational congruence given in Definition 3.

Theorem 1. *$\approx_l$ is a congruence.*

Proof. Immediate, from Lemma 3. □

Theorem 2. *$A \approx_l B$ if and only if $A \approx B$.*

Proof. (Sketch) $\Rightarrow$) Condition 1 of Definition 3, is guaranteed by condition 3 of Definition 4. Condition 2 of Definition 3 is identical to condition 2 in Definition 4. Condition 3 in Definition 3 is guaranteed by Lemma 3. $\Leftarrow$) Condition 1 of Definition 4 is guaranteed by Lemma 2. Condition 2 is equal in both definitions. Condition 3 of Definition 4 can be proved by induction on the structure of the label α using condition 1 of Definition 3 at the induction step for $\alpha = \overline{a}\langle M \rangle$. □

5 An Application

We give an implementation of the *1-out-of-2-oblivious transfer* protocol (OT^1_2) in PAPi. The notion of oblivious transfer (OT) was first introduced by Rabin [18] in a number theoretic context and then generalized by Even, Goldreich and Lampel [10] with the OT^1_2 notion. Intuitively, OT^1_2 allows one party (S) to transfer exactly one secret, out of two different recognizable secrets (M_0, M_1), to his counterpart (R). Each secret is received with probability one half and the sender is completely ignorant of which secret has been received. Intuitively, $\mathrm{OT}^1_2(S, R, M_0, M_1)$ is a protocol that should satisfy the following axioms: (A) R can read exactly one message: either M_0 or M_1, the probability of each to be read is one half; (B) if R does not read M_i he gains no useful information about M_i by the execution of OT^1_2; (C) for S, the a posteriori probability that R got M_0 (M_1) remains one half. Oblivious transfer is widely used in protocols for secure multiparty computation and has been shown to be rather efficient.

In order to describe OT^1_2 in PAPi, and recalling the notation in [10], we should extend the equational theory for asymmetric encryption with two binary functions $\boxplus$ and $\boxminus$ such that $(x \boxplus y) \boxminus y = x$ and the mappings $x \mapsto x \boxplus y$ and $y \mapsto x \boxplus y$ are permutations on the set of terms. Intuitively, when using RSA [19],

$x \boxplus y$ is implemented as reduction modulo N (the RSA modulus) of $x+y$, while $x \boxminus y$ is the reduction modulo N of $x-y$. The full list of equations is:

$$\begin{array}{llll}
(1) & \mathsf{fst}(\mathsf{pair}(x,y)) = x & (2) & \mathsf{snd}(\mathsf{pair}(x,y)) = y \\
(3) & \mathsf{dec}(\mathsf{enc}(x,\mathsf{pk}(y)),\mathsf{sk}(y)) = x & (4) & \mathsf{enc}(\mathsf{dec}(x,\mathsf{sk}(y)),\mathsf{pk}(y)) = x \\
(5) & (x \boxplus y) \boxminus y = x & (6) & (x \boxplus y) \boxminus x = y \\
(7) & x \boxplus (y \boxminus x) = y & (8) & x \boxplus y = y \boxplus x
\end{array}$$

We are now ready to implement OT_2^1 in PAPi in the following way:

$\mathrm{OT}_2^1(S,R,M_0,M_1) ::= S(M_0,M_1) \mid R \qquad$ where:

$S(M_0,M_1) ::= \nu e.\nu m_0.\nu m_1.\ \Big(\overline{c}\langle \mathsf{pk}(e), m_0, m_1\rangle.c(y).(\overline{c}\langle T_{00}, T_{11}, 0\rangle \oplus_{\frac{1}{2}} \overline{c}\langle T_{01}, T_{10}, 1\rangle)\Big)$

with $\quad T_{ij} = M_i \boxplus \mathsf{dec}(y \boxminus m_j, \mathsf{sk}(e)) \quad$ and:

$R ::= \nu l.\ \Big(c(z,x_0,x_1).(\overline{c}\langle \mathsf{enc}(l,z) \boxplus x_0\rangle.P_0 \oplus_{\frac{1}{2}} \overline{c}\langle \mathsf{enc}(l,z) \boxplus x_1\rangle.P_1)\Big)$

with, for $i \in \{0,1\} \quad P_i ::= c(y_0,y_1,y_2).\,(\text{if } y_2 =_E 0 \text{ then } \overline{a}\langle y_i \boxminus l\rangle \text{ else } \overline{a}\langle y_{1-i} \boxminus l\rangle)\,.$

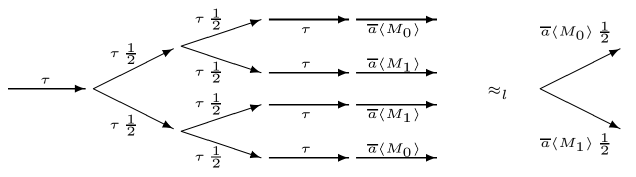
For simplicity we write input actions with multiple variables (this can be easily encoded with pair, fst and snd). S picks two fresh messages m_0 and m_1 and transmits them to R, together with the public key of the fresh secret e. The receiver R receives this triple and randomly (with probability $\frac{1}{2}$) sends back to S the term $T = \mathsf{enc}(l,\mathsf{pk}(e)) \boxplus m_i$, for $i \in \{0,1\}$. Since S does not know the secret value l, it cannot tell whether T has been obtained from m_0 or m_1. S generates the messages T_{ij} obtained by combining M_i and m_j and with probability $\frac{1}{2}$ sends to R the M_i combined with the right m_j used by R. The flag 0 (1, resp.) is used to indicate that S used m_0 (m_1, resp.) for the first part of the message. The receiver can now compute the secret (M_0 or M_1) from the right T_{ij} and l. At the final step, R sends the value of the received secret on channel a.

Note that we do not consider equations of the form $\mathsf{dec}(M,\mathsf{sk}(e)) = \mathsf{wrong}$ when M is not encrypted with $\mathsf{sk}(e)$. Otherwise, S may be able to know which m_j was used by R through the test $\mathsf{dec}(\mathsf{enc}(l,\mathsf{pk}(e)) \boxplus m_i \boxminus m_j, \mathsf{sk}(e)) = \mathsf{wrong}$. Such a test is true only if $i \neq j$. In the case of $i = j$, S is able to compute the secret l as $\mathsf{dec}(\mathsf{enc}(l,\mathsf{pk}(e)) \boxplus m_i \boxminus m_j, \mathsf{sk}(e))$. This problem is avoided by using an asymmetric cipher (e.g., RSA), obtained with equations (4) and (5). In this way, the test never returns the value wrong and S cannot tell whether the result of $\mathsf{dec}(\mathsf{enc}(l,\mathsf{pk}(e)) \boxplus m_i \boxminus m_j, \mathsf{sk}(e))$ is l or just a random decryption.

By means of our notion of weak bisimulation we can show that the OT_2^1 implementation in PAPi satisfies the axioms. In particular, we can show that the receiver R receives M_0 or M_1 with probability $\frac{1}{2}$ by checking the weak bisimulation of the protocol implementation with the process that simply outputs M_0 or M_1 on a channel a with probability $\frac{1}{2}$. Such a system, which captures axioms (A), (B) and (C) required by OT_2^1, may be seen as the secure behaviour of the protocol. Namely, imposing a restriction on channel c, thus forcing synchronization among S and R, it holds that:

$$\nu c.\mathrm{OT}_2^1(S,R,M_0,M_1) \approx_l \overline{a}\langle M_0\rangle \oplus_{\frac{1}{2}} \overline{a}\langle M_1\rangle.$$

This can be proved easily, since $\nu c.\mathrm{OT}_2^1(S,R,M_0,M_1)$ performs only internal reductions labeled with τ before performing the output of M_0 or M_1 (with probability $\frac{1}{2}$, resp.) on channel a. The two bisimilar labeled probabilistic automata modeling the behaviour of $\nu c.\mathrm{OT}^1_2(S, R, M_0, M_1)$ and $\overline{a}\langle M_0\rangle \oplus_{\frac{1}{2}} \overline{a}\langle M_1\rangle$ are shown in Figure 1 (probabilities equal to 1 are omitted). Notice that at each step there is just a probability distribution that a scheduler can chose (the only nondeterministic choices are among blocking schedulers).



τ
$\tau\ \frac{1}{2}$
$\tau\ \frac{1}{2}$
$\tau\ \frac{1}{2}$
$\tau\ \frac{1}{2}$
$\tau\ \frac{1}{2}$
$\tau\ \frac{1}{2}$
τ
τ
τ
τ
$\overline{a}\langle M_0\rangle$
$\overline{a}\langle M_1\rangle$
$\overline{a}\langle M_1\rangle$
$\overline{a}\langle M_0\rangle$
$\approx_l$
$\overline{a}\langle M_0\rangle\ \frac{1}{2}$
$\overline{a}\langle M_1\rangle\ \frac{1}{2}$

Fig. 1. $\nu c.\mathrm{OT}^1_2(S, R, M_0, M_1) \approx_l \overline{a}\langle M_0\rangle \oplus_{\frac{1}{2}} \overline{a}\langle M_1\rangle$.

6 A Conservative Extension

Many process algebraic approaches are non–probabilistic and, in general, probabilistic choice can be approximated by suitable nondeterministic mechanisms. Using probabilistic features, however, provides stronger safety and security guarantees. We give formal substance to this claim (Proposition 1 below), by showing that $\approx$ is a conservative extension of an appropriate notion of observational congruence for the purely Nondeterministic Applied Pi–calculus (NAPi), obtained by removing the probabilistic choice operators from the syntax of plain processes.

With $\mathcal{A}_{NP}$ we denote the set of extended processes in NAPi. The *internal reduction* $A \rightarrow A'$, becomes now the smallest relation on $\mathcal{A}_{NP}$ closed by structural equivalence and application of evaluation contexts such that:

$$\overline{a}\langle x\rangle.P \,|\, a(x).Q \rightarrow P \,|\, Q \qquad \frac{P \rightarrow P'}{P{+}Q \rightarrow P'} \qquad \text{if } M = M \text{ then } P \text{ else } Q \rightarrow P$$

$$\text{if } M = N \text{ then } P \text{ else } Q \rightarrow Q \quad \text{for } M, N \in \mathcal{T}_G \text{ s.t.} \Sigma \not\vdash M =_E N$$

Given a process $A \in \mathcal{A}$ we define the plain process $A_{NP} \in \mathcal{A}_{NP}$ obtained by replacing each probabilistic choice operator appearing in A with a purely nondeterministic choice operator. As an example, given $A = (P \oplus_p Q) \,|\, \{M/x\}$, we get $A_{NP} = (P{+}Q) \,|\, \{M/x\}$.

The notion of observational congruence introduced in the probabilistic framework (see Definition 3) can be rewritten for the purely nondeterministic case.

For $A \in \mathcal{A}_{NP}$, we write $A \Downarrow a$ when A can send a message on a, namely when $A \rightarrow^* C[\overline{a}\langle x\rangle.P]$ for some evaluation context $C[\_]$ that does not bind a.

Definition 5. Nondeterministic observational congruence *($\approx_{NP}$) is the largest symmetric relation $\mathcal{R}$ between closed extended processes in $\mathcal{A}_{NP}$ with the same domain such that $A\mathcal{R}B$ implies:*

1. *if $A \Downarrow a$, then $B \Downarrow a$;*

2. *if $A \to^* A'$, then $B \to^* B'$ and $A' \mathcal{R} B'$ for some B';*
3. *$C[A] \mathcal{R} C[B]$ for all closing evaluation contexts $C[\_]$.*

The following proposition states that removing probabilities form two observationally equivalent probabilistic extended processes the equivalence is preserved in the purely nondeterministic setting.

Proposition 1. *Given $A, B \in \mathcal{A}$ such that $A \approx B$, then $A_{NP} \approx_{NP} B_{NP}$.*

Proof. (Sketch) For each execution in the probabilistic setting there exist a corresponding execution in the nondeterministic one. Hence, for each probabilistic barb of A and B (with the same probability), there exist nondeterministic ones for A_{NP} and B_{NP}. The converse implication does not hold, since, in general, two barbs in the nondeterministic setting may have two different probabilities in the richer framework. $\square$

Hence, if a system satisfies an observational equivalence property in the probabilistic setting, its nondeterministic counterpart does still satisfy the property in the nondeterministic setting. The converse implication does, in general, not hold, since systems satisfying a property in the nondeterministic setting may turn out to lose the property in the more expressive probabilistic framework.

Example 5. Consider the process $A = \nu c.\mathrm{OT}^1_2(S, R, M_0, M_1)$ introduced in Section 5 and the family of processes $B = \overline{a}\langle M_0 \rangle \oplus_p \overline{a}\langle M_1 \rangle$. It is easy to see that $A_{NP} \approx_{NP} B_{NP}$ (both processes have just a barb on channel a). However, it is not true that $A \approx B$ for all p. Actually, the equivalence holds just for $p = \frac{1}{2}$.

7 Conclusions

In this paper we have introduced the Probabilistic Applied Pi–calculus (PAPi), an extension of the Applied Pi–calculus ([1]) for dealing with probability, nondeterminism and equations (which are shown to be rich enough for modeling the most common cryptographic operations). We have given a labeled operational semantics and a labeled weak bisimulation, which we have then shown to be a congruence. As one expects, the results given in the probabilistic framework are preserved with respect to the results given in the non-probabilistic one.

As an application, we have shown how PAPi applies to the OT^1_2 protocol where probability and cryptographic operations play an important role. As another possible future application, we mention, just as an example, sensor networks, for which: (a) environmental distributed sensing can be modeled with a nondeterministic choice among input channels waiting for external stimuli; (b) randomization is crucial (see the probabilistic routing policies introduced in [3], or the randomized sleeping architecture proposed in [5]); (c) cryptography is fundamental when dealing with secure wireless communication. Notice, moreover, that thanks to the generality of equational theories, PAPi can also be applied to domains different from security.

References

1. M. Abadi and C. Fournet. *Mobile Values, New Names, and Secure Communication.* In Proc. of POPL'01, ACM Press, 104–115, 2001.
2. M. Abadi and A. D. Gordon. *A Calculus for Cryptographic Protocols: The Spi Calculus.* Information and Computation 148(1):1–70, 1999.
3. C. L. Barrett, S. J. Eidenbenz, L. Kroc, M. Marathe and J. P. Smith. *Parametric Probabilistic Sensor Network Routing.* In Proc. of WSNA'03, ACM Press, 122–131, 2003.
4. M. G. Buscemi and U. Montanari. *CC–pi: A Constraint–based Language for Specifying Service Level Agreements.* In Proc. of ESOP'07, Springer LNCS 4421, 19–32, 2007.
5. Q. Cao, T. Abdelzaher, T. He and J. Stankovic. *Towards Optimal Sleep Scheduling in Sensor Networks for Rare-event Detection.* In Proc. of IPSN'05, IEEE CS Press, 20–27, 2005.
6. R. Cleaveland, J. Parrow and B. Steffen. *The concurrency workbench: a semantics-based tool for the verification of concurrent systems.* ACM Trans. Program. Lang. Syst. 15(1):36–72, 1993.
7. V. Cortier and M. Abadi. *Deciding Knowledge in Security Protocols under Equational Theories.* Theoretical Computer Science 367(1–2):2–32, 2006.
8. N. Dershowitz and J.-P. Jouannaud. *Rewrite Systems.* Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)", 243–320, 1990.
9. D. Dolev and A. C. Yao. *On the Security of Public Key Protocols.* IEEE Transactions on Information Theory 29(12):198–208, 1983.
10. S. Even, O. Goldreich and A. Lempel. *A Randomized protocol fo Signing Contracts.* Communications of the ACM 28(6):637–647, 1985.
11. J. A. Goguen, J. W. Thatcher, E .G. Wagner and J. B. Wright. *Initial Algebra Semantics and Continuous Algebras.* Journal of the ACM 24(1):68–95, 1977.
12. A. Jung and R. Tix. *The Troublesome Probabilistic Powerdomain.* In Proc. of Workshop on Computation and Approximation, Elsevier ENTCS 13, 1998.
13. M. W. Mislove, J. Ouakine and J. Worrell. *Axioms for Probability and Nondeterminism.* In Proc. of EXPRESS'03, Elsevier ENTCS 96, 7–28, 2004.
14. R. Milner. *Communication and Concurrency.* Prentice–Hall, 1989.
15. R. Milner. *Communicating and Mobile Systems: the π–Calculus.* Cambridge University Press, 1999.
16. J. C. Mitchell. *Foundations for Programming Languages.* MIT Press, 1996.
17. J. Niehren and M. Mueller. *Constraints for Free in Concurrent Computation.* In Proc. of ACSC'95, Springer LNCS 1023, 171–186, 1995.
18. M. O. Rabin. *how to Exchange Secrets by Oblivious Transfer.* Unpublished manuscript, 1981.
19. R. Rivest, A. Shamir and L. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* Communications of the ACM 21(2):120–126, 1978. Previously released as an MIT "Technical Memo" in April 1977.
20. V. A. Saraswat, M. C. Rinard and P. Panangaden. *Semantic Foundations of Concurrent Constraint Programming.* POPL'91, ACM press, 333–352, 1991.
21. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems.* PhD thesis, MIT, Laboratory for Computer Science, 1995.
22. R. Segala and N. Lynch. *Probabilistic Simulations for Probabilistic Processes.* In Proc. of CONCUR'94, Springer LNCS 839, 481–496, 1994.
23. B. Victor and F. Moller, *The Mobility Workbench - A Tool for the pi-Calculus.* In Proc. of CAV '94, Springer LNCS 818, 428–440, 1994.