Two Formal Approaches for Approximating Noninterference Properties

Alessandro Aldini[1], Mario Bravetti[2], Alessandra Di Pierro[3], Roberto Gorrieri[2], Chris Hankin[4], and Herbert Wiklicky[4]

[1] Istituto STI, Università di Urbino *Carlo Bo*, Italy
[2] Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy
[3] Dipartimento di Informatica, Università di Pisa, Italy
[4] Department of Computing, Imperial College, London, UK

Abstract. The formalisation of security properties for computer systems raises the problem of overcoming also in a formal setting the classical view according to which confidentiality is an absolute property stating the complete absence of any unauthorised disclosure of information. In this paper, we present two formal models in which the notion of noninterference, which is at the basis of a large variety of security properties defined in the recent literature, is approximated. To this aim, the definition of indistinguishability of process behaviour is replaced by a similarity notion, which introduces a quantitative measure ε of the behavioural difference among processes. The first model relies on a programming paradigm called Probabilistic Concurrent Constraint Programming, while the second one is presented in the setting of a probabilistic process algebra. In both models, appropriate notions of distance provide information (the ε) on the security level of the system at hand, in terms of the capability of an external observer of identifying illegal interferences.

1 Introduction

The exact estimation of properties of computer systems is a problem that was widely and successfully attacked via several different formal approaches (see, e.g., [CT02,BHK01,HHHMR94,HS95,Hil96,BDG98,Ber99,BB00,Bra02]). However, a number of factors make the use of approximation techniques necessary to enhance the reliability of "exact" solutions obtained through the formal analysis of the mathematical model of a real, complex system. On the one hand, the confidence we can have in the answers computed by a software tool, which are delivered with certainty, strictly depends on the likelihood of obtaining precise information needed to formally specify the system at hand. On the other hand, even when such information is exact, the results of the mathematical analysis definitely assert that the considered property is or is not satisfied by the system model, while in practice it often happens that a system that approximately behaves like a perfect one is not only acceptable but also the only possible implementation. In practice, in a realistic scenario, a qualitative binary answer to the classical question "does the system satisfy my property?" is too restrictive and, in many cases, not significant.

In this work, we concentrate on formal techniques that employ probabilistic information to give a *quantitative* answer to the kind of question above in the restricted framework of security properties. Indeed, the motivations surveyed above apply also to the problem of verifying the security requirements of real systems. It is well-accepted that the unauthorised disclosure of confidential information cannot be completely avoided in real, complex systems, where typically the interplay between the portion of the system handling secrets and the other components that instead manage public information is more tight than that we expect [RMMG01]. In practice, part of the information flowing through the system cannot be controlled, and a portion of such an unavoidable information flow is sometimes illegal, in the sense that it reveals confidential data to unauthorised users. In such a case, the goal of the designer consists of minimising the illegal information leakage, and, as a consequence, the aim of the analyst must be the provision of an approximated estimation of such an information leakage. As a simple, real example, consider a password-based authentication system, like, e.g., an automatic teller machine. It is trivial to verify that absolute secrecy cannot be guaranteed. In fact, a brute-force based attack has the possibility, even if negligible, of guessing the password, thus violating the secrecy requirements. The analysis of such a kind of system is beyond the scope of possibilistic information flow techniques, which reject programs that do not guarantee absolute secrecy. A more interesting analysis should state that a potential information leakage is not troubling. From a quantitative viewpoint, this corresponds to verify whether or not the probability of detecting a potential illegal information flow is beyond a threshold for which the observer considers the system to be secure "enough". In case of the automatic teller machine, the probability of cracking the system depends on the length of the password and on the number of attempts at disposal of the attacker. By playing on these parameters, the designer can limit to a negligible (as desired) value the probability of accessing the system without knowing the appropriate password.

The approach to information flow analysis we consider is based on the idea of noninterference, originally proposed in [GM82], which states that "one group of users, using a certain set of commands, is noninterfering with another group of users if what the first group does with those commands has no effect on what the second group of users can see". In a security context, the first group is represented by the high-level users, which execute confidential, secret activities, while the second group is given by the low-level users, which instead see public data only. The intuition is that the low-level view of the system to be analysed is not to be altered by the behaviour of the high-level users. If this is the case, we say that any covert channel cannot be set up from the high level to the low level. The verification of the condition above is based on the idea of indistinguishability of behaviours: in order to establish that there is no information flow between a high-level component H and a low-level object L, it is sufficient to check if for any pair of behaviours of the system that differ only in H's behaviour, L's observations cannot distinguish these two behaviours. Depending on the nature of the information flow, an external observer can characterise different kinds of interference, due, e.g., to the deterministic, nondeterministic, timed, or probabilistic behaviour of the system. In particular, possibilistic noninterference for nondeterministic programs is weaker than probabilistic noninterference, which helps to reveal those covert channels that arise from the analysis of the frequency of the possible observations in several consecutive executions of the system [Gra90,McL90]. Consider, e.g., a program P that handles pin numbers needed to access the automatic teller machine mentioned above. At a certain point of the execution, the following statement is executed:

$$\mathit{low\_variable} := \mathrm{PIN}_i +^p \mathtt{rand}(9999)$$

where $+^p$ is a probabilistic choice operator that selects the left-hand command (which assigns a secret pin to a public, low variable) with probability p and the right-hand command (which assigns a random value from the range $[0 \ldots 9999]$ to the low variable) with probability $1-p$. According to a purely nondeterministic behaviour, the program above is secure, since the set of possible outcomes does not change depending on which command will be executed. However, statistical inferences derived from the relative frequency of outcomes of repeated executions of the program allow an external observer to disclose the secret pin with a confidence that depends on the number of executed experiments.

Probabilistic noninterference also offers the means for approximating noninterference properties, by quantifying the real effectiveness of each possibilistic covert channel. More precisely, the key idea of an approach based on probabilistic noninterference is to replace the notion of indistinguishability by an appropriate notion of similarity. For instance, consider again program P and assume that parameter p is a value very close to 0. Obviously, the behaviour of P is not the same as that of the following secure program P':

$$\mathit{low\_variable} := \mathtt{rand}(9999)$$

since if we execute "infinitely often" both programs, then the limit of the frequencies of the possible outcomes allow the observer to distinguish P from P'. However, in practice we have that P and P' are similar and the probability of distinguishing the two programs is still negligible even after a large number n of experiments. In other words, P is considered to be an acceptable approximation of a secure program. As a result of an approach that replaces the restrictive idea of indistinguishability by a relaxed, more realistic notion of similarity, we can accept as secure systems a number of programs that somehow suffer from an information leakage but in practice offer sufficient security guarantees.

In this work, we survey two semantics-based security models (i.e., models that analyse the program behaviour to verify security properties) in which the notion of noninterference is approximated in the sense that they allow for some exactly quantified information leakage. The first one formalises such an approach in the context of a particular probabilistic declarative language, while the second one is based on a probabilistic process algebraic framework.

Language-based formalisms provide a suitable framework for analysing the confidentiality properties of real, complex computing systems. Particularly promising is the use of program semantics and analysis for the specification of information-flow policies and information-flow controls which guarantee data confidentiality (see, e.g., [SM03] for a survey).

On the other hand, process algebras provide all the main ingredients needed to specify and analyse noninterference properties of computer systems (see, e.g., the several process algebraic approaches described in [FG01]). They are designed with the aim of describing concurrent systems that may interact through the exchange of messages, so that they can be used to naturally express each information flow occurring within the system to be modeled. They deal with both nondeterminism and, as we will focus in this work, probability, so that several kinds of information leakage can be revealed. They also deal in an elegant way with abstraction thanks to the hiding operator, which can be used to specify the observational power of each external observer, depending on the security level of such an observer. Last but not least, there exists a strong, natural similarity between the notion of indistinguishability for processes and semantic equivalences over process algebraic terms.

In the following (Sect. 2), we first introduce the language-based approach by presenting a formalisation of a noninterference property called *confinement* together with its probabilistic and approximated versions in the setting of the probabilistic programming language PCCP (Probabilistic Concurrent Constraint Programming) [DW98a,DW98b]. In this language nondeterminism is completely replaced by probabilistic choice, which makes it possible to develop a statistical interpretation of the approximation of the security property. Moreover, the different role played by variables in imperative and constraint programming hinders a direct translation of previous formalisation of noninterference based on the imperative paradigm into the PCCP setting, where a more appropriate notion must consider process identity rather than variables values.

Then (Sect. 3), we introduce a process algebraic framework for approximating probabilistic noninterference [ABG03]. The basic calculus integrates the characteristics of the classical CCS [Mil89] and CSP [Hoa85] and employs the probabilistic model introduced in [BA03], which is a mixture of the reactive and generative models of probability [GSS95]. Such an approach permits the modeler to specify both nondeterministic behaviour and probabilistic information in the same system model. The behavioural equivalence of process expressions is defined in terms of weak probabilistic bisimulation [BH97], a probabilistic extension of the classical weak bisimulation by Milner [Mil89]. Moreover, the behavioural similarity among processes is defined in terms of a relation called weak probabilistic bisimulation with ε-precision, an approximated version of the weak probabilistic bisimulation, where ε provides information on "how much" two behaviours differ from each other.

Finally (Sect. 4), some conclusions and comments about related work terminate the paper.

2 Language-based Approach to Noninterference

2.1 Probabilistic Concurrent Constraint Programming

Probabilistic Concurrent Constraint Programming (PCCP) [DW98a,DW98b] is a probabilistic version of the Concurrent Constraint Programming (CCP) paradigm [SRP91,SR90]. This can be seen as a kind of process algebra enhanced with a notion of computational state. More precisely, CCP as well as PCCP are based on the notion of a generic *constraint system* $\mathcal{C}$, defined as a cylindric algebraic complete partial order (see [SRP91,dDP95] for more details), which encodes the information ordering. This is referred to as the *entailment* relation $\vdash$ and is sometimes denoted by $\sqsupseteq$. A cylindric constraint system includes constraints of the form $\exists_x c$ (cylindric elements) to model *hiding* of local variables, and constraints of the form d_{xy} (diagonal elements) to model *parameter passing*. The axioms of the constraint system include laws from the theory of cylindric algebras [HMT71] which model the cylindrification operators $\exists_x$ as a kind of first-order existential quantifiers, and the diagonal elements d_{xy} as the equality between x and y.

$A ::= \mathbf{tell}(c)$		adding a constraint
$\mid$	$\Box_{i=1}^{n}\ \mathbf{ask}(c_i) \rightarrow p_i : A_i$	probabilistic choice
$\mid$	$\|_{i=1}^{n}\ q_i : A_i$	prioritised parallelism
$\mid$	$\exists_x A$	hiding, local variables
$\mid$	$p(x)$	procedure call, recursion

Table 1. The Syntax of PCCP Agents

In PCCP probability is introduced via a probabilistic choice and a form of probabilistic parallelism. The former replaces the nondeterministic choice of CCP, while the latter replaces the pure nondeterminism in the interleaving semantics of CCP by introducing a probabilistic scheduling. This allows us to implement mechanisms for differentiating the relative advancing speed of a set of agents running in parallel.

The concrete syntax of a PCCP agent A is given in Table 1, where c and c_i are *finite* constraints in $\mathcal{C}$, and p_i and q_i are real numbers representing probabilities. Note that at the syntactic level no restrictions are needed on the values of the numbers p_i and q_i; as explained in the next section, they will be turned into probability distributions by a normalisation process occurring during the computation. The meaning of $p(x)$ is given by a procedure declaration of the form $p(y) : -A$, where y is the formal parameter. We will assume that for each procedure name there is at most one definition in a fixed set of declarations (or program) P.

2.2 Operational Semantics

The operational model of PCCP can be intuitively described as follows. All processes share a common store consisting of the least upper bound, denoted by $\sqcup$, (with respect to the inverse $\sqsubseteq$ of the entailment relation) of all the constraints established up to that moment by means of **tell** actions. These actions allow for communication. Synchronisation is achieved via an **ask** guard which tests whether the store entails a given constraint. The probabilistic choice construct allows for a random selection of one of the different possible synchronisations making the program similar to a *random walk*-like stochastic process. Parts of the store can be made local by means of a *hiding operator* corresponding to a logical existential quantifier.

The operational semantics of PCCP is formally defined in terms of a probabilistic transition system, $(\mathrm{Conf}, \longrightarrow_p)$, where Conf is the set of configurations $\langle A, d\rangle$ representing the state of the system at a certain moment and the transition relation $\longrightarrow_p$ is defined in Table 2. The state of the system is described by the agent A which has still to be executed, and the common store d. The index p in the transition relation indicates the probability of the transition to take place. In order to describe all possible stages of the evolution of agents, in Table 2 we use an extended syntax by introducing an agent **stop** which represents successful termination, and an agent $\exists_x^d A$ which represents the evolution of an agent of the form $\exists_x B$ where d is the local information on x produced during this evolution. The agent $\exists_x B$ can then be seen as the particular case where the local store is empty, that is $d = \mathit{true}$. In the following we will identify all agents of the form $\|_{i=1}^n\, q_i : \mathbf{stop}$ and $\exists_x^d\mathbf{stop}$ with the agent **stop** as they all indicate a successful termination.

The rules of Table 2 are closely related to the ones for nondeterministic CCP, and we refer to [dDP95] for a detailed description. The rules for probabilistic choice and prioritised parallelism involve a normalisation process needed to re-distribute the probabilities among those agents A_i which can actually be chosen for execution. Such agents must be enabled (i.e. the corresponding guards $\mathbf{ask}(c_i)$ succeed) or active (i.e. able to make a transition). This means that we have to re-define the probability distribution so that only enabled/active agents have non-zero probabilities and the sum of these probabilities is one. The probability after normalisation is denoted by $\tilde{p}_j$. For example, in rule **R2** the normalised transition probability can be defined for all enabled agents by

$$\tilde{p}_i = \frac{p_i}{\sum_{\vdash c_j} p_j},$$

where the sum $\sum_{\vdash c_j} p_j$ is over all enabled agents. When there are no enabled agents normalisation is not necessary. We treat a zero probability in the same way as a non-entailed guard, i.e. agents with zero probability are not enabled; this guarantees that normalisation never involves a division by a zero value. Analogous considerations apply to the normalisation of active agents in **R3**. It might be interesting to note that there are alternative ways to deal with the situation where $\sum_{\vdash c_j} p_j = 0$ (all enabled agents have probability zero). In

[DW00] normalisation is defined in this case as the assignment of a uniform distribution on the enabled agents; such a normalisation procedure allows, for example, to introduce a quasi-sequential composition.

The meaning of rule **R4** is intuitively explained by saying that the agent $\exists_x^d A$ behaves "almost" like A, with the difference that the variable x which is possibly present in A must be considered local, and that the information present in d has to be taken into account. Thus, if the store which is visible at the external level is c, then the store which is visible internally by A is $d \sqcup (\exists_x c)$. Now, if A is able to make a step, thus reducing itself to A' and transforming the local store into d', what we see from the external point of view is that the agent is transformed into $\exists_x^{d'} A'$, and that the information $\exists_x d$ present in the global store is transformed into $\exists_x d'$.

The semantics of a procedure call $p(x)$, modelled by Rule **R5**, consists in the execution of the agent A defining $p(x)$ with a parameter passing mechanism similar to call-by-reference: the formal parameter x is linked to the actual parameter y in such a way that y inherits the constraints established on x and vice-versa. This is realised in a way to avoid clashes between the formal parameter and occurrences of y in the agent via the operator Δ_y^x defined by:

$$\Delta_y^x A = \begin{cases} \exists_y^{d_{xy}} A & \text{if } x \neq y \\ A & \text{if } x = y. \end{cases}$$

R1	$\langle \mathbf{tell}(c), d\rangle \longrightarrow_1 \langle \mathbf{stop}, c \sqcup d\rangle$	
R2	$\langle \square_{i=1}^n \ \mathbf{ask}(c_i) \to p_i : A_i, d\rangle \longrightarrow_{\tilde{p}_j} \langle A_j, d\rangle$	$j \in [1,n]$ and $d \vdash c_j$
R3	$\dfrac{\langle A_j, c\rangle \longrightarrow_p \langle A_j', c'\rangle}{\langle \Vert_{i=1}^n \ p_i : A_i, c\rangle \longrightarrow_{p \cdot \tilde{p}_j} \langle \Vert_{j \neq i=1}^n p_i : A_i \parallel p_j : A_j', c'\rangle}$	$j \in [1,n]$
R4	$\dfrac{\langle A, d \sqcup \exists_x c\rangle \longrightarrow_p \langle A', d'\rangle}{\left\langle \exists_x^d A, c\right\rangle \longrightarrow_p \left\langle \exists_x^{d'} A', c \sqcup \exists_x d'\right\rangle}$	
R5	$\langle p(y), c\rangle \longrightarrow_1 \langle \Delta_y^x A, c\rangle$	$p(x) : -A \in P$

Table 2. The Transition System for PCCP

Observables We will consider a notion of observables which captures the probabilistic input/output behaviour of a PCCP agent. We will define the observables $\mathcal{O}(A, d)$ of an agent A in store d as a probability distribution on constraints. Formally, this is defined as an element in the real vector space:

$$\mathcal{V}(\mathcal{C}) = \left\{ \sum x_c c \;\middle|\; x_c \in \mathbb{R},\ c \in \mathcal{C} \right\},$$

that is the free vector space obtained as the set of all formal linear combinations of elements in $\mathcal{C}$. The coefficients x_c represent the probability associated to constraints c.

Operationally, a distribution $\mathcal{O}(A,d)$ corresponds to the set of all pairs $\langle c,p\rangle$, where c is the result of a computation of A starting in store d and p is the probability of computing that result. For the purpose of this paper we will restrict to agents which only exhibit computations whose length is bounded. Note that since our programs are finitely branching this implies by König's lemma that the vector space of constraints is finite-dimensional.

We formally define the set of results for an agent A as follows.

Definition 1. *Let A be a PCCP agent. A computational path π for A in store d is defined by*

$$\pi \equiv \langle A_0, c_0\rangle \longrightarrow_{p_1} \langle A_1, c_1\rangle \longrightarrow_{p_2} \ldots \longrightarrow_{p_n} \langle A_n, c_n\rangle\,,$$

where $A_0 = A$, $c_0 = d$, $A_n = \mathbf{stop}$ and $n < \infty$.

We denote by $\mathrm{Comp}(A,d)$ the set of all computational paths for A in store d.

Definition 2. *Let $\pi \in \mathrm{Comp}(A,d)$ be a computational path for A in store d,*

$$\pi \equiv \langle A,d\rangle = \langle A_0, c_0\rangle \longrightarrow_{p_1} \langle A_1, c_1\rangle \longrightarrow_{p_2} \ldots \longrightarrow_{p_n} \langle A_n, c_n\rangle\,.$$

We define the result of π as $res(\pi) = c_n$ and its probability as $prob(\pi) = \prod_{i=1}^{n} p_i$.

Because of the probabilistic choice, there might be different computational paths for a given PCCP program which lead to the same result. The probability associated to a given result c is then the sum of all probabilities $prob(\pi)$ associated to all paths π such that $res(\pi) = c$. This suggests that we introduce the following equivalence relation on $\mathrm{Comp}(A)$.

Definition 3. *Let $\pi, \pi' \in \mathrm{Comp}(A)$ be two computational paths for A in store d. We define $\pi \approx \pi'$ iff $res(\pi) = res(\pi')$. The equivalence class of π is denoted by $[\pi]$.*

The definitions of $res(\pi)$ and $prob(\pi)$ are extended to $\mathrm{Comp}(A)_{/\approx}$ in the obvious way by $res([\pi]) = res(\pi)$ and $prob([\pi]) = \sum_{\pi' \in [\pi]} prob(\pi')$.

We can now define the probabilistic input/output observables of a given agent A in store d as the set

$$\mathcal{O}(A,d) = \left\{ \langle res([\pi]), prob([\pi])\rangle \mid [\pi] \in \mathrm{Comp}(A)_{/\approx} \right\}.$$

In the following we will adopt the convention that whenever the initial store is omitted then it is intended to be *true*.

Example 1. [CHM02] Consider an ATM (Automatic Teller Machine) accepting only a single PIN number n out of m possible PINs, e.g. $m = 10000$:

$$\begin{aligned}\mathrm{ATM}n \equiv & \textstyle\prod_{i=1, i\neq n}^{m} \mathbf{ask}(\mathrm{PIN}i) \rightarrow 1 : \mathbf{tell}(alarm)\\ & \textstyle\prod\ \mathbf{ask}(\mathrm{PIN}n) \rightarrow 1 : \mathbf{tell}(cash)\end{aligned}$$

This agent simulates an ATM which recognises PINn: if PINn has been told the machine dispenses *cash*, otherwise — for any incorrect PINi — it sounds an *alarm*.

Consider now the following agent representing the client whose PIN is i:

$$C_i \equiv \mathbf{ask}(true) \to 1 : \mathbf{tell}(\text{PIN}i).$$

The computational paths for the parallel composition $M_i \equiv p : \text{ATM}n \parallel q : C_i$ are given in Figure 1 respectively for the case in which $i = n$ and $i \neq n$. When run in the initial store *true*, agent ATMn is not active (no constraints PINj, $1 \leq j \leq m$ is entailed by the store); thus C_i is scheduled with probability 1 (obtained by q after normalisation). The resulting configuration is $\langle p : \text{ATM}n \parallel q : \mathbf{stop}, \text{PIN}i\rangle$. Now the only active agent is ATMn which is then executed with (normalised) probability 1 leading to the final configuration $\langle \mathbf{stop}, cash\rangle$ in the case where the PIN number is correct (right hand side derivation in Figure 1) and $\langle \mathbf{stop}, alarm\rangle$ in the case where the PIN number is wrong (left hand side derivation in Figure 1).

$$\begin{array}{ccc}
\langle p : ATM_n \parallel q : C_i, true\rangle & \qquad & \langle p : ATM_n \parallel q : C_n, true\rangle \\
\downarrow 1 & & \downarrow 1 \\
\langle p : ATM_n \parallel q : \mathbf{stop}, PIN_i\rangle & & \langle p : ATM_n \parallel q : \mathbf{stop}, PIN_n\rangle \\
\downarrow 1 & & \downarrow 1 \\
\langle p : \mathbf{stop} \parallel q : \mathbf{stop}, alarm \sqcup PIN_i\rangle & & \langle p : \mathbf{stop} \parallel q : \mathbf{stop}, cash \sqcup PIN_i\rangle
\end{array}$$

Fig. 1. Execution of a program simulating the interaction with an ATM.

The observables are then $\mathcal{O}(M_i) = \{\langle \text{PIN}_i \sqcup alarm, 1\rangle\}$ for $i \neq n$, $\mathcal{O}(M_i) = \{\langle \text{PIN}_n \sqcup cash, 1\rangle\}$ for $i = n$.

2.3 Probabilistic noninterference and identity confinement

The original idea of noninterference as stated in [GM82] can be expressed in the PCCP formalism via the notion of *identity confinement*. Roughly, this notion establishes whether it is possible to identify which process is running in a given program. Therefore, given a set of agents and a set of potential intruders, the latter cannot see what the former set is doing, or more precisely, no spy is able to find out which of the agents in the first group is actually being executed.

This formulation is the natural translation in the context of PCCP of the notion of confinement typically expressed in imperative languages via the values of variables [SS00].

The following example illustrates the notion of identity confinement as compared to the imperative formulation. It also shows the difference between nondeterministic and probabilistic (identity) confinement.

Example 2. In an imperative language, confinement — as formulated for example in [SS99,SS00] — usually refers to a standard two-level security model consisting of high and low level variables. One then considers the (value of the) high variable h as confined if the value of the low level variable l is not "influenced" by the value of the high variable, i.e. if the observed values of l are independent of h.

The following statement illustrates the difference between nondeterministic and probabilistic confinement:

$$h := h \bmod 2; \;\; (l := h \;{}_{\frac{1}{2}}\!\Box_{\frac{1}{2}}\; (l := 0 \;{}_{\frac{1}{2}}\!\Box_{\frac{1}{2}}\; l := 1))$$

The value of l clearly depends "somehow" on h. However, if we resolve the choice nondeterministically it is impossible to say anything about the value of h by observing the possible values of l. Concretely, we get the following dependencies between h and possible values of l:

- For $h \bmod 2 = 0$: $\{l = 0, l = 1\}$
- For $h \bmod 2 = 1$: $\{l = 1, l = 0\}$,

i.e. the possible values of l are the same independently from the fact that h is even or odd. In other words, h is nondeterministically confined.

In a probabilistic setting the observed values for l and their probabilities allow us to distinguish cases where h is even from those where h is odd. We have the following situation:

- For $h \bmod 2 = 0$: $\left\{\left\langle l = 0, \frac{3}{4}\right\rangle, \left\langle l = 1, \frac{1}{4}\right\rangle\right\}$
- For $h \bmod 2 = 1$: $\left\{\left\langle l = 0, \frac{1}{4}\right\rangle, \left\langle l = 1, \frac{3}{4}\right\rangle\right\}$

Therefore, the probabilities to get $l = 0$ and $l = 1$ reveal if h is even or odd, i.e. h is probabilistically *not* confined.

Example 3. We can re-formulate the situation above in our declarative setting by considering the following agents:

$$\begin{aligned}
\texttt{hOn} &\equiv \mathbf{ask}(\mathit{true}) \to \frac{1}{2} : \mathbf{tell}(\texttt{on}) \;\Box\; \mathbf{ask}(\mathit{true}) \to \frac{1}{2} : \texttt{Rand} \\
\texttt{hOff} &\equiv \mathbf{ask}(\mathit{true}) \to \frac{1}{2} : \mathbf{tell}(\texttt{off}) \;\Box\; \mathbf{ask}(\mathit{true}) \to \frac{1}{2} : \texttt{Rand} \\
\texttt{Rand} &\equiv \mathbf{ask}(\mathit{true}) \to \frac{1}{2} : \mathbf{tell}(\texttt{on}) \;\Box\; \mathbf{ask}(\mathit{true}) \to \frac{1}{2} : \mathbf{tell}(\texttt{off})
\end{aligned}$$

The constraint system consists of four elements:

$$\mathcal{C} = \{\mathit{true}, \texttt{on}, \texttt{off}, \mathit{false} = \texttt{on} \sqcup \texttt{off}\},$$

where $\mathit{true} \sqsubseteq \texttt{on} \sqsubseteq \mathit{false}$ and $\mathit{true} \sqsubseteq \texttt{off} \sqsubseteq \mathit{false}$.

The constraints `on` and `off` represent the situations in which the low variable $l = 1$ or $l = 0$ respectively. The agent `hOn` corresponds then to the behaviour of the imperative program fragment in case that $h \mod 2 = 1$, while `hOff` corresponds to the case where $h \mod 2 = 0$. The auxiliary agent `Rand` corresponds to the second choice in the above imperative fragment. The imperative notion of confinement now translate in our framework into a problem of identity confinement: getting information about h in the previous setting is equivalent to discriminating between `hOn` and `hOff`, i.e. revealing their identity. The two agents will be identity confined if they are observationally equivalent in any context.

As explained in Section 2.2, the observables of a PCCP agent correspond to a distribution on the constraint system, that is a vector in the space $\mathcal{V}(\mathcal{C})$. Thus, the difference between two observables corresponds to the vector difference between the given observables and can be measured by means of a *norm*. We adopt here the supremum norm $\|\cdot\|_\infty$ formally defined as

$$\|(x_i)_{i\in I}\|_\infty = \sup_{i\in I} |x_i|,$$

where $(x_i)_{i\in I}$ represents a probability distribution. However, as long as we are interested in defining the identity of two vectors, any p-norm: $\|(x_i)_{i\in I}\|_p = \sqrt[p]{\sum_{i\in I} |x_i|^p}$ would be appropriate.

Probabilistic identity confinement is then defined as follows [DHW01]:

Definition 4. *Two agents A and B are probabilistically identity confined iff their observables are identical in any context, that is for all agent S,*

$$\mathcal{O}(p : A \parallel q : S) = \mathcal{O}(p : B \parallel q : S)$$

or equivalently,

$$\Big\|\mathcal{O}(p : A \parallel q : S) - \mathcal{O}(p : B \parallel q : S)\Big\| = 0,$$

for all scheduling probabilities p and $q = 1 - p$.

Example 4. It is easy to check that any context can distinguish between the agents `hOn` and `hOff` of Example 3. In fact, even if executed on their own their observables are different (cf. Figure 2):

$$\Big\|\mathcal{O}(\texttt{hOn}, \mathit{true}) - \mathcal{O}(\texttt{hOff}.\mathit{true})\Big\| =$$
$$\left\| \left\{ \left\langle \texttt{on}, \frac{3}{4} \right\rangle, \left\langle \texttt{off}, \frac{1}{4} \right\rangle \right\} - \left\{ \left\langle \texttt{on}, \frac{1}{4} \right\rangle, \left\langle \texttt{off}, \frac{3}{4} \right\rangle \right\} \right\| = \frac{1}{2}.$$

Therefore `hOn` and `hOff` are *not* probabilistically identity confined.

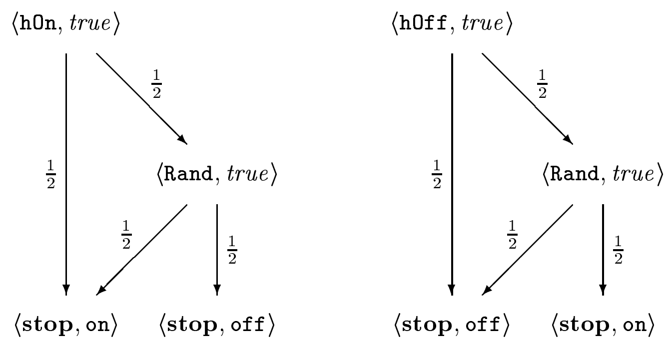


Fig. 2. Transitions for `hOn` and `hOff`

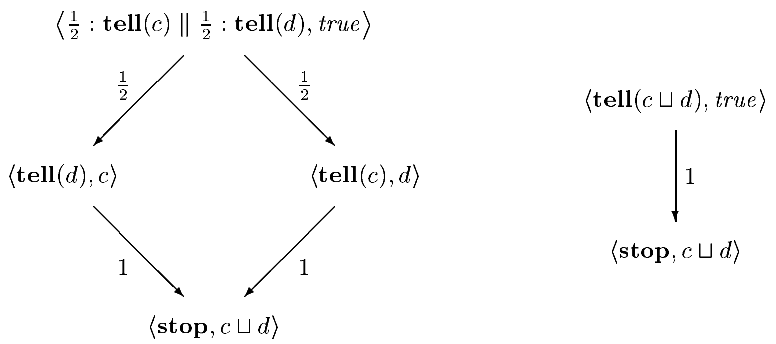


Fig. 3. Independent Executions of A and B

Example 5. Consider the following two PCCP agents [DHW01]:

$$A \equiv \frac{1}{2} : \mathbf{tell}(c) \parallel \frac{1}{2} : \mathbf{tell}(d)$$
$$B \equiv \mathbf{tell}(c \sqcup d).$$

It is easy to see that in their nondeterministic versions A and B executed in any context give the same observables. A and B are thus nondeterministically identity confined.

Treating the choice probabilistically still gives us the same observables for A and B if they are executed on their own (cf. Figure 3), but they are not probabilistically confined. A context which reveals the identity of A and B is for example the agent:

$$C \equiv \mathbf{ask}(c) \rightarrow \frac{2}{3} : \mathbf{tell}(e) \,[]\, \mathbf{ask}(d) \rightarrow \frac{1}{3} : \mathbf{tell}(f),$$

$$\begin{tikzcd}[column sep=small]
 & & \left\langle \frac{1}{2} : A \parallel \frac{1}{2} : C, \mathit{true} \right\rangle \arrow[dl, "\frac{1}{2}"'] \arrow[dr, "\frac{1}{2}"] & & \\
 & \left\langle \frac{1}{2} : \mathbf{tell}(d) \parallel \frac{1}{2} : C, c \right\rangle \arrow[dl, "\frac{1}{2}"'] \arrow[dr, "\frac{1}{2}"] & & \left\langle \frac{1}{2} : \mathbf{tell}(c) \parallel \frac{1}{2} : C, d \right\rangle \arrow[dl, "\frac{1}{2}"'] \arrow[dr, "\frac{1}{2}"] & \\
\langle \mathbf{tell}(d), c \sqcup e \rangle \arrow[dr, "1"'] & & \langle C, c \sqcup d \rangle \arrow[dl, "\frac{2}{3}"] \arrow[dr, "\frac{1}{3}"'] & & \langle \mathbf{tell}(c), d \sqcup f \rangle \arrow[dl, "1"] \\
 & \langle \mathbf{stop}, c \sqcup d \sqcup e \rangle & & \langle \mathbf{stop}, c \sqcup d \sqcup f \rangle &
\end{tikzcd}$$

Fig. 4. Executions of A in Context C

$$\begin{tikzcd}[column sep=small]
 & \left\langle \frac{1}{2} : B \parallel \frac{1}{2} : C, \mathit{true} \right\rangle \arrow[d, "1"] & \\
 & \langle C, c \sqcup d \rangle \arrow[dl, "\frac{2}{3}"'] \arrow[dr, "\frac{1}{3}"] & \\
\langle \mathbf{stop}, c \sqcup d \sqcup e \rangle & & \langle \mathbf{stop}, c \sqcup d \sqcup f \rangle
\end{tikzcd}$$

Fig. 5. Executions of B in Context C

as the executions of A and B in this context give different observables (cf. Figure 4 and Figure 5):

$$\mathcal{O}\left(\frac{1}{2} : A \parallel \frac{1}{2} : C\right) = \left\{\left\langle c \sqcup d \sqcup e, \frac{7}{12}\right\rangle, \left\langle c \sqcup d \sqcup f, \frac{5}{12}\right\rangle\right\}$$

$$\mathcal{O}\left(\frac{1}{2} : B \parallel \frac{1}{2} : C\right) = \left\{\left\langle c \sqcup d \sqcup e, \frac{2}{3}\right\rangle, \left\langle c \sqcup d \sqcup f, \frac{1}{3}\right\rangle\right\}.$$

We observe that if we restrict to a particular class of contexts, namely those of the form:

$$D \equiv \mathbf{ask}(g) \to 1 : \mathbf{tell}(h),$$

then A and B are probabilistically identity confined with respect to these agents: for any choice of the scheduling probabilities p and $q = 1-p$, we obtain the same observables for the parallel compositions of D with A and B respectively.

If neither c nor d entails g then D will never be executed, and the executions of $p : A \parallel q : D$ and $p : B \parallel q : D$ are essentially the same as for A and B alone (cf. Figure 3).

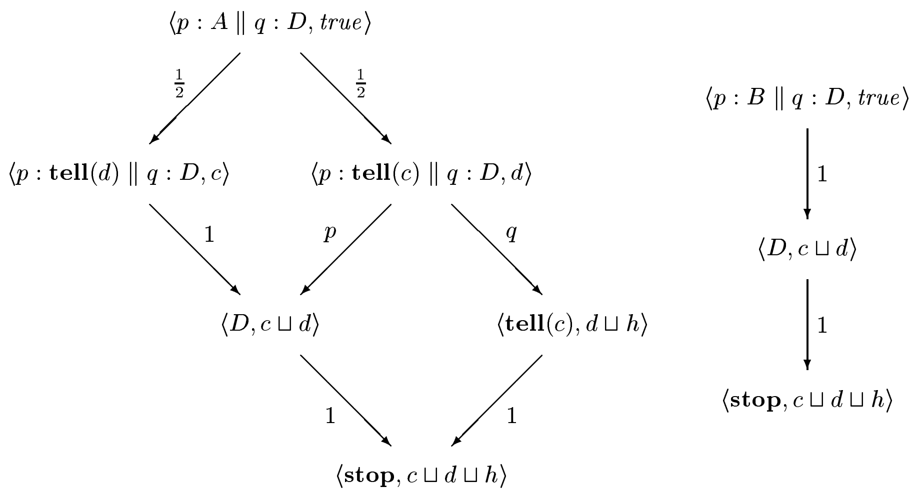


Fig. 6. Executions in Context D when d entails g

If only d entails g we obtain the derivations in Figure 6. The case where g is entailed by c alone is analogous. In all cases we end up with a single result $c \sqcup d \sqcup h$ with probability one.

The derivations of $p : A \parallel q : D$ and $p : B \parallel q : D$ in the case that both c and d entail g are depicted in Figure 7: again we obtain the same result $c \sqcup d \sqcup h$ with probability one.

$$
\begin{tikzcd}
 & \langle p : A \parallel q : D, \mathit{true} \rangle \arrow[dl, "\frac{1}{2}"'] \arrow[dr, "\frac{1}{2}"] & \\
\langle p : \mathbf{tell}(d) \parallel q : D, c \rangle \arrow[d, "q"'] \arrow[dr, "p"] & & \langle p : \mathbf{tell}(c) \parallel q : D, d \rangle \arrow[dl, "p"'] \arrow[d, "q"] \\
\langle \mathbf{tell}(d), c \sqcup h \rangle \arrow[dr, "1"'] & \langle D, c \sqcup d \rangle \arrow[d, "1"] & \langle \mathbf{tell}(c), d \sqcup h \rangle \arrow[dl, "1"] \\
 & \langle \mathbf{stop}, c \sqcup d \sqcup h \rangle &
\end{tikzcd}
\qquad
\begin{tikzcd}
\langle p : B \parallel q : D, \mathit{true} \rangle \arrow[d, "1"] \\
\langle D, c \sqcup d \rangle \arrow[d, "1"] \\
\langle \mathbf{stop}, c \sqcup d \sqcup h \rangle
\end{tikzcd}
$$

Fig. 7. Executions in Context D when both c and d entail g

In general, identical behaviour in all contexts is hardly ever achievable. It therefore makes sense to ask for identical observables if A and B are executed in parallel with agents with only limited capabilities. Moreover, the power of a context can be evaluated in terms of its ability to distinguish the behaviours of two agents. It is also reasonable to think that its effectiveness will depend on the probabilities of the scheduling in the interleaving with the given agents. This leads to the definition of a weaker (and yet more practical) notion of probabilistic identity confinement which is parametric in the type of context S and the scheduling probability p. We will introduce such a notion, which we call *approximate identity confinement*, in the next section.

2.4 Approximate Identity Confinement

In Section 1 we argued that it is practically more useful to base noninterference properties on some *similarity* notions instead of equivalence once.

The confinement notion discussed above is *exact* in the sense that it refers to the equivalence of the agents' behaviour. In this section, we introduce a technique which allows us to relax confinement to an approximate and yet more effective notion.

The intuitive idea behind such a notion is that we look at *how much* the behaviours of two agents differ, instead of qualitatively asserting whether they are identical or not. In particular, in the probabilistic case we can measure the distance ε between the distributions representing the agents' observables instead of checking whether this difference is 0. We can then say that the agents are ε-confined for some $\varepsilon \geq 0$.

We illustrate this idea by means of the ATM example introduced in Section 2.2.

Example 6. Consider the program in Example 1 which simulates an ATM (Automatic Teller Machine) accepting only a single PIN number n out of m possible PINs, e.g. $m = 10000$:

$$\begin{aligned} \mathrm{ATM}n \equiv & \textstyle\Box_{i=1, i\neq n}^{m} \mathbf{ask}(\mathrm{PIN}i) \rightarrow 1 : \mathbf{tell}(alarm) \\ & \Box\ \mathbf{ask}(\mathrm{PIN}n) \rightarrow 1 : \mathbf{tell}(cash) \end{aligned}$$

The following agent simulates a spy which tries a random PIN number i:

$$S \equiv \textstyle\Box_{i=1}^{m} \mathbf{ask}(true) \rightarrow 1 : \mathbf{tell}(\mathrm{PIN}i)$$

If we consider two such machines $\mathrm{ATM}n_1$ and $\mathrm{ATM}n_2$ for $n_1 \neq n_2$ and execute them in context S we obtain two slightly different observables, namely:

$$\begin{aligned} \mathcal{O}\left(p : \mathrm{ATM}n_1 \parallel q : S\right) = & \left\{\left\langle \mathrm{PIN}n_1 \sqcup cash, \frac{1}{m}\right\rangle\right\} \\ & \cup \bigcup_{i=1, i\neq n_1}^{m} \left\{\left\langle \mathrm{PIN}i \sqcup alarm, \frac{1}{m}\right\rangle\right\} \\ \text{and} & \\ \mathcal{O}\left(p : \mathrm{ATM}n_2 \parallel q : S\right) = & \left\{\left\langle \mathrm{PIN}n_2 \sqcup cash, \frac{1}{m}\right\rangle\right\} \\ & \cup \bigcup_{i=1, i\neq n_2}^{m} \left\{\left\langle \mathrm{PIN}i \sqcup alarm, \frac{1}{m}\right\rangle\right\}. \end{aligned}$$

Clearly, $\mathcal{O}(p : \mathrm{ATM}n_1 \parallel q : S)$ and $\mathcal{O}(p : \mathrm{ATM}n_2 \parallel q : S)$ are different.

For most PINs both machines will sound an alarm in most cases, but if we are lucky, the spy will use the correct PINs in which case we are able to distinguish the two machines (besides earning some *cash*). The chances for this happening are small but are captured essentially if we look at the difference between the observables:

$$\left\| \mathcal{O}(p : \mathrm{ATM}n_1 \parallel q : S) - \mathcal{O}(p : \mathrm{ATM}n_2 \parallel q : S) \right\| = \frac{1}{m}.$$

The set $\{\mathrm{ATM}n\}_n$ is ε-confined with respect to S with $\varepsilon = \frac{1}{m}$ but not strictly confined. In the practical applications, m is usually very large, that is ε is very small, which makes it reasonable to assume the ATM's agents as secure although not exactly confined.

The notion of approximate identity confinement we will define in the following is based on the idea of measuring how much the behaviour of two agents differs if we put them in a certain context. We will refer to such a context as *spy* or *attacker*. This restriction makes sense as no system is secure against an omnipotent attacker [LMMS98] and its security depends on the quality of the possible attacker. We will discuss in the following different kinds of such attackers.

As an example, consider the class of attackers expressed in PCCP by:

$$\mathcal{S}_n = \left\{\Box_{i=1}^{n} \mathbf{ask}(c_i) \to p_i : \mathbf{tell}(f_i)\right\},$$

where $f_i \in \mathcal{C}$ are *fresh* constraints, that is constraints which never appear in the execution of the host agents, and $c_i \in \mathcal{C}$. These agents are passive and memoryless attackers. They do not change the behaviour of the hosts, and are only allowed to interact with the store in one step. Nevertheless, they are sufficient for formalising quite powerful attacks such as the timing attacks in [Koc95].

A generalisation of this class is to consider active spies (e.g. Example 7 and Example 1) and/or spies with memory such as $\mathbf{ask}(c) \to p : \mathbf{ask}(d) \to q : \mathbf{tell}(f)$.

Example 7. Consider the two agents:

$$A \equiv \mathbf{ask}(c) \to 1 : \mathbf{tell}(d)$$
$$B \equiv \mathbf{stop}.$$

If executed in store *true*, A and B are obviously confined with respect to any *passive* spy. They both do nothing, and it is therefore impossible to distinguish them by just observing. However, for an *active* spy like $S \equiv \mathbf{tell}(c)$ it is easy to determine if it is being executed in parallel with A or B. Note that if executed in any store d such that $d \vdash c$, the two agents A and B are always distinguishable because their observables are different.

The notion of approximate confinement which we introduce in the following is a generalisation of the identity confinement introduced in [DHW01] and defined in Section 2.3. The definition we give is parametric with respect to a set of admissible spies $\mathcal{S}$ and scheduling probabilities p and $q = 1-p$. We say that two agents A and B are approximately confined with respect to a set of spies $\mathcal{S}$ iff there exists an $\varepsilon \geq 0$ such that for all $S \in \mathcal{S}$ the *distance* between the observables of $p : A \parallel q : S$ and $p : B \parallel q : S$ is smaller than ε. We consider as a measure for this distance the supremum norm $\|\cdot\|_\infty$ as in Definition 4. In this case, the choice of this norm is particularly appropriate because it allows us to identify a single constraint c for which the associated probabilities are maximally different. In the following we will usually omit the index ∞.

Definition 5. *Given a set of admissible spies $\mathcal{S}$, we call two agents A and B ε-confined for some $\varepsilon \geq 0$ iff:*

$$\sup_{S \in \mathcal{S}} \left\| \mathcal{O}(p : A \parallel q : S) - \mathcal{O}(p : B \parallel q : S) \right\| = \varepsilon.$$

This definition can be generalised to a set of more than two agents.

The number ε associated to a given class of spies $\mathcal{S}$ can be seen as a measure of the "power" of $\mathcal{S}$. In fact, it is strongly related to the number of tests a spy needs to perform in order to reveal the identity of the host agents. We will make this argument more precise in the next section. Note that this number depends on the scheduling probability. This is because the effectiveness of a spy can only be evaluated depending on the internal design of the host system which is in general not known to the spy. For example, in [DHW03b] we have presented an analysis which shows that the "best" spy of the class $\mathcal{S}_2$ defined above is one with a choice distribution where p_1 is very close to 0 and p_2 is very close to 1, or vice versa.

Obviously, if two agents A and B are ε-confined with $\varepsilon(p) = 0$ for all scheduling probability p then they are probabilistically identity confined.

2.5 Statistical Interpretation

The notion of approximate confinement is strongly related to statistical concepts, in particular to so-called *hypothesis testing* (see e.g. [Sha99]).

Identification by Testing Let us consider the following situation. We have two agents A and B which are attacked by a spy S. Furthermore, we assume that A and B are ε-confined with respect to S. This means that the observables $\mathcal{O}(p : A \parallel q : S)$ and $\mathcal{O}(p : B \parallel q : S)$ are ε-similar. In particular, as the observables do not include infinite results, we can identify some constraint $c \in \mathcal{C}$ such that $|p_A(c) - p_B(c)| = \varepsilon$, where $p_A(c)$ is the probability of the result c in an execution of $p : A \parallel q : S$ and $p_B(c)$ is the probability that c is a result of $p : B \parallel q : S$.

Following the standard interpretation of probabilities as "long-run" relative frequencies, we can thus expect that the number of times we get c as result of an execution of $p : A \parallel q : S$ and $p : B \parallel q : S$ will differ "on the long run" by exactly a factor ε. That means if we execute $p : A \parallel q : S$ or $p : B \parallel q : S$ "infinitely" often we can determine $p_A(c)$ and $p_B(c)$ as the limit of the frequencies with which we obtain c as result.

In fact, for any unknown agent X we can attempt do determine $p_X(c)$ experimentally by executing $p : X \parallel q : S$ over and over again. Assuming that X is actually the same as either A or B we know that the $p_X(c)$ we obtain must be either $p_A(c)$ or $p_B(c)$. We thus can easily determine this way if $X = A$ or $X = B$, i.e. reveal the identity of X (if $\varepsilon \neq 0$), simply by testing.

Unfortunately — as J.M. Keynes pointed out — we are all dead on the long run. The above described experimental setup is therefore only of theoretical value. For practical purposes we need a way to distinguish A and B by finite executions of $p : A \parallel q : S$ and $p : B \parallel q : S$. If we execute $p : A \parallel q : S$ and $p : B \parallel q : S$ only a finite number of — say n — times, we can observe a certain experimental frequency $p_A^n(c)$ and $p_B^n(c)$ for getting c as a result. Each time we repeat this finite sequence of n executions we may get different values for $p_A^n(c)$ and $p_B^n(c)$ (only the infinite experiments will eventually converge to the same constant values $p_A(c)$ and $p_B(c)$).

Analogously, we can determine the frequency $p_X^n(c)$ for an unknown agent X by testing, i.e. by looking at n executions of $p : X \parallel q : S$. We can then try to compare $p_X^n(c)$ with $p_A^n(c)$ and $p_B^n(c)$ or with $p_A(c)$ and $p_B(c)$ in order to find out if $X = A$ or $X = B$. Unfortunately, there is neither a single value for either $p_X^n(c)$, $p_A^n(c)$ or $p_B^n(c)$ (each experiment may give us different values) nor can we test if $p_X^n(c) = p_A^n(c)$ or $p_X^n(c) = p_B^n(c)$ nor if $p_X^n(c) = p_A(c)$ or $p_X^n(c) = p_B(c)$.

For example, it is possible that c is (coincidentally) not the result of the first execution of $p : X \parallel q : S$, although the (long-run) probabilities of obtaining c by executing $p : A \parallel q : S$ or $p : B \parallel q : S$ are, let's say, $p_A = 0.1$ and $p_B = 0.5$. If we stop our experiment after $n = 1$ executions we get $p_X^1(c) = 0$. We know that $X = A$ or $X = B$ but the observed $p_X^1(c)$ is different from both p_A and p_B.

Nevertheless, we could argue that it is more likely that $X = A$ as the observed $p_X^1(c) = 0$ is closer to $p_A = 0.1$ than to $p_B = 0.5$. The problem is now to determine, on the basis of such experiments, how much the identification of X with A is "more correct" than identifying X with B on the basis of such experiments.

For finite experiments we can only make a guess about the true identity of X, but never definitely reveal its identity. The *confidence* we can have in our guess or *hypothesis* about the identity of an unknown agent X — i.e. the probability that we make a correct guess — depends obviously on two factors: the number of tests n and the difference ε between the observables of $p : A \parallel q : S$ and $p : B \parallel q : S$.

Hypothesis Testing The problem is to determine experimentally if the unknown agent X is one of two known agents A and B. The only way we can obtain information about X is by executing it in parallel with a spy S. In this way we can get an experimental estimate for the observables of $p : X \parallel q : S$. We then can compare this estimate with the observables of $p : A \parallel q : S$ and $p : B \parallel q : S$.

That means: based on the outcome of some finite experiments (involving an unknown agent X) we formulate a hypothesis H about the identity of X, namely either that "X is A" or that "X is B". Our hypothesis about the identity of X will be formulated according to a simple rule: depending if the experimental estimate for the observables of $p : X \parallel q : S$ are closer to $\mathcal{O}(p : A \parallel q : S)$ or to $\mathcal{O}(p : B \parallel q : S)$ we will identify X with A or B respectively.

More precisely, the method to formulate the hypothesis H about the identity of the unknown process X consists of the two following steps:

1. We execute $p : X \parallel q : S$ exactly n times in order to obtain an experimental approximation, i.e. average, for its observables

$$\overline{\mathcal{O}}_n(p : X \parallel q : S) = \left\{ \left\langle c, \frac{\#\text{ of times } c \text{ is the result}}{n} \right\rangle \right\}_{c \in \mathcal{C}},$$

2. Depending if $\overline{\mathcal{O}}_n(p : X \parallel q : S)$ is closer to the observables $\mathcal{O}_n(p : A \parallel q : S)$ or $\mathcal{O}_n(p : B \parallel q : S)$ we formulate the hypothesis

$$H : \begin{cases} X = A & \text{if } \left\|\mathcal{O}_n(p : X \parallel q : S) - \mathcal{O}(p : A \parallel q : S)\right\| \\ & \quad \le \left\|\mathcal{O}_n(p : X \parallel q : S) - \mathcal{O}(p : B \parallel q : S)\right\| \\ X = B & \text{otherwise.} \end{cases}$$

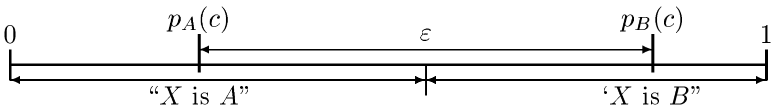
The question is now whether the guess expressed by the hypothesis H about the true identity of the black box X, which we formulate according to the above procedure, is correct; or more precisely: what is the probability that the hypothesis H holds? To do this we have to distinguish two cases or scenarios:

X **is actually** A**:** What is the probability (in this case) that we formulate the *correct* hypothesis $H : X$ is A and what is the probability that we formulate the *incorrect* hypothesis $H : X$ is B?

X **is actually** B**:** What is the probability (in this case) that we formulate the *correct* hypothesis $H : X$ is B and what is the probability that we formulate the *incorrect* hypothesis $H : X$ is A?

Clearly, in each case the probability to formulate a correct hypothesis and the probability to formulate an incorrect hypothesis add up to one. Furthermore, it is obvious that both scenarios "X is actually A" and "X is actually B" are symmetric. We will therefore investigate only one particular problem. Suppose that X is actually agent A, what is the probability that — according to the above procedure — we formulate the — in this case — correct hypothesis $H : X$ is A.

In the following we use the notation $p_X(c)$ and $p^n_X(c)$ to denote the probability assigned to $c \in \mathcal{C}$ in the distribution representing the observables $\mathcal{O}(p : X \parallel q : S)$ and in the experimental average $\mathcal{O}_n(p : X \parallel q : S)$ respectively. Furthermore, we look at a simplified situation where we are considering only a single constraint c where the difference between $p_A(c)$ and $p_B(c)$ is maximal. Let us assume without loss of generality that $p_A(c) < p_B(c)$ as in the diagram below:



If the experimental value $p^n_X(c) = p^n_A(c)$ we obtained in our test is anywhere to the left of $p_A(c) + \varepsilon/2$ then the hypothesis H we formulate (based on $p^n_A(c)$) will be the correct one: "X is A"; if the experimental value is to the right of $p_A(c) + \varepsilon/2$ we will formulate the incorrect hypothesis: "X is B".

Under the assumption that "X is actually A" the probability $\mathbf{P}(H)$ that we will formulate the correct hypothesis "X is A" is therefore:

$$\mathbf{P}\left(p^n_A(c) < p_A(c) + \frac{\varepsilon}{2}\right) = 1 - \mathbf{P}\left(p_A(c) + \frac{\varepsilon}{2} < p^n_A(c)\right).$$

To estimate $\mathbf{P}(H)$ we have just to estimate the probability $\mathbf{P}(p^n_A(c) < p_A(c) + \varepsilon/2)$, i.e. that the experimental value $p^n_A(c)$ will be left of $p_A(c) + \varepsilon/2$.

Confidence Estimation The confidence we can have in the hypothesis H we formulate is true can be determined by various statistical methods. These methods allow us to estimate the probability that an experimental average X_n — in our case $p^n_A(c)$ — is within a certain distance from the corresponding expectation value $\mathbf{E}(X)$ — here $p_A(c)$ — i.e. the probability

$$\mathbf{P}\left(|X_n - \mathbf{E}(X)| \leq \varepsilon\right)$$

for some $\varepsilon \geq 0$. These statistical methods are essentially all based on the *central limit theorem*, e.g. [Bil86,GS97,Sha99].

The type of tests we consider here to formulate a hypothesis about the identity of the unknown agent X are described in statistical terms by so called *Bernoulli Trials* which are parametric with respect to two probabilities p and $q = 1 - p$ (which have nothing to do with the scheduling probabilities above). The central limit theorem for this type of tests [GS97, Thm 9.2] gives us an estimate for the probability that the experimental value $S_n = n \cdot X_n$ after n repetitions of the test will be in a certain interval $[a, b]$:

$$\lim_{n\to\infty} \mathbf{P}(a \leq S_n \leq b) = \frac{1}{\sqrt{2\pi}} \int_{a^*}^{b^*} \exp\left(\frac{-x^2}{2}\right)$$

where

$$a^* = \frac{a - np}{\sqrt{npq}} \quad \text{and} \quad b^* = \frac{b - np}{\sqrt{npq}}.$$

Unfortunately, the integral of the so called *standard normal density* on the right hand side of the above expression is not easy to obtain. In practical situations one has to resort to numerical methods or statistical tables, but it allows us — at least in principle — to say something about $\mathbf{P}(H)$.

Identifying S_n with $n \cdot p^n_A$ we can utilise the above expression to estimate the probability $\mathbf{P}(p_A(c) + \varepsilon/2 \leq p^n_A)$ which determines $\mathbf{P}(H)$. In order to do this we have to take:

$$\begin{aligned} a &= p_A(c) + \frac{\varepsilon}{2} \\ b &= \infty \\ p &= p_A(c) \\ q &= 1 - p_A(c). \end{aligned}$$

This allows us — in principle — to compute the probability:

$$\lim_{n\to\infty} \mathbf{P}\left(p_A(c) + \frac{\varepsilon}{2} \leq p^n_A(c) \leq \infty\right).$$

Approximating — as it is common in statistics — $\mathbf{P}(p_A(c) + \varepsilon/2 \leq p^n_A)$ by $\lim \mathbf{P}(p_A(c) + \varepsilon/2 \leq p^n_A)$ we get:

$$\mathbf{P}(H) = 1 - \mathbf{P}\left(p_A(c) + \frac{\varepsilon}{2} \leq p^n_A(c)\right)$$

$$
\begin{aligned}
&\approx 1 - \lim_{n\to\infty} \mathbf{P}\left(p_A(c) + \frac{\varepsilon}{2} \le p_A^n(c)\right) \\
&= 1 - \int_{a_0}^{\infty} \exp\left(\frac{-x^2}{2}\right)
\end{aligned}
$$

with

$$
a_0 = \frac{n\varepsilon}{2}\frac{1}{\sqrt{npq}} = \frac{\varepsilon\sqrt{n}}{2\sqrt{pq}} = \frac{\varepsilon\sqrt{n}}{2\sqrt{p_A(c)(1-p_A(c))}}.
$$

We see that the only way to increase the probability $\mathbf{P}(H)$, i.e. the confidence that we formulate the right hypothesis about the identity of X, is by minimising the integral. In order to do this we have to increase the lower bound a_0 of the integral. This can be achieved — as one would expect — by increasing the number n of experiments.

We can also see that for a smaller ε we have to perform more tests n to reach the same level of confidence, $\mathbf{P}(H)$: The smaller n the harder it is to distinguish A and B experimentally. Note that for $\varepsilon = 0$, the probability of correctly guessing which of the agents A and B is in the black box is $\frac{1}{2}$, which is the best blind guess we can make anyway. In other words: for $\varepsilon = 0$ we cannot distinguish between A and B.

Example 8. Consider the agents in Example 5. The problem is to determine from the experimentally obtained approximation of the observables $\mathcal{O}_n\left(\frac{1}{2} : X \parallel \frac{1}{2} : C\right)$ for $X = A$ or $X = B$ the true identity of X. If, for example, X is actually agent A and if we concentrate on the constraint $c \sqcup d \sqcup e$ we have

$$
\varepsilon = \frac{1}{12} \text{ and } p = p_A(c \sqcup d \sqcup e) = \frac{7}{12}
$$

The probability $\mathbf{P}(H)$ to formulate the correct hypothesis H depends on the lower bound a_0 of the above integral, i.e. the normal distribution $N(a_0, \infty)$:

$$
\mathbf{P}(H) = 1 - \int_{a_0(n)}^{\infty} \exp\left(\frac{-x^2}{2}\right) = 1 - N(a_0, \infty).
$$

The bound a_0 in turn depends on the number n of experiments we perform. The value of a_0 for 9 tests is:

$$
a_0(9) = \frac{\sqrt{9}}{24}\frac{1}{\sqrt{\frac{7}{12} - (\frac{7}{12})^2}} = \frac{1}{8}\frac{12}{\sqrt{35}} = \frac{3}{\sqrt{140}} \approx 0.25355
$$

while for 144 tests we get:

$$
a_0(144) = \frac{\sqrt{144}}{24}\frac{1}{\sqrt{\frac{7}{12} - (\frac{7}{12})^2}} = \frac{1}{2}\frac{12}{\sqrt{35}} = \frac{6}{\sqrt{35}} \approx 1.0142
$$

In other words, if we repeat the execution of $\frac{1}{2} : X \parallel \frac{1}{2} : C$ exactly 9 times, the probability of formulating a correct hypothesis H about the identity of X is about (using a normal distribution table, e.g. [GS97, p499]):

$$\mathbf{P}(H) = 1 - \int_{0.25}^{\infty} \exp\left(\frac{-x^2}{2}\right) \approx 0.5987,$$

but if we perform 144 test our confidence level will rise to

$$\mathbf{P}(H) = 1 - \int_{1.0}^{\infty} \exp\left(\frac{-x^2}{2}\right) \approx 0.8413.$$

For 9 tests the hypothesis formulated will be right with an about 60% chance, while for 144 tests it will be correct with about 85%.

3 Process Algebra Formulation of Noninterference

3.1 Probabilistic Process Algebra

Process algebras are specification languages (see, e.g., [BW90,BPS01]) that describe the behaviour of concurrent systems through actions, which in our setting are syntactically divided into output actions and input actions, and through algebraic operators, which in our setting are enriched with probabilistic information (see, e.g., [BBS95]). Here we consider a slight variant of the probabilistic process algebra introduced in [BA03] (a core algebra of the reacher calculus EMPA_{gr} [BB00]). The algebraic model of a system communicates with the environment through its inputs and outputs and performs internal computations through special, unobservable actions, termed τ actions. Formally, we denote with *AType* the set of visible action types, ranged over by $a, b, \ldots$. For each visible action type a, we distinguish the output action a and the input action a_*. The complete set of actions, termed *Act* and ranged over by $\pi, \pi', \ldots$, contains the input actions and the output actions with type in *AType* and the action τ. The set $\mathcal{L}$ of process terms, ranged over by $P, Q, \ldots$, is generated by the syntax:

$$P ::= \underline{0} \,|\, \pi.P \,|\, P +^p P \,|\, P \parallel^p_S P \,|\, P\backslash L \,|\, P/^p_a \,|\, A$$

where $S, L \subseteq AType$, $a \in AType$, and $p \in]0,1[$. $\underline{0}$ expresses the null, deadlocked term [1], and $.$, $+^p$, $\parallel^p_S$, $\backslash L$, and $/^p_a$ denote the prefix, alternative, parallel, restriction, and hiding operators, respectively. Constants A are used to specify recursive systems. In particular, we assume a set of constants defining equations of the form $A \stackrel{\Delta}{=} P$ to be given. In the rest of the paper, we restrict ourselves to the set $\mathcal{G}$ of finite state, closed, guarded terms of $\mathcal{L}$, which we call processes [Mil89].

Now, we informally describe the algebraic operators and the probabilistic model through an example. The reader interested in details and proofs should refer to [ABG03].

[1] We omit $\underline{0}$ when it is clear from the context.

Example 9. Consider the following abstraction of the Automatic Teller Machine interacting with a client (cf. Example 1 in Section 2.2):

$$Client \parallel^p_S ATM .$$

The communication interface between processes *Client* and ATM, defined by set $S = \{insert\_pin,\ cash,\ fail\}$, says that the two processes (i) interact by synchronously executing actions of type in S, and (ii) asynchronously and independently execute each other local action. Probability p is the parameter of a probabilistic scheduler that, in each system state, decides which of the two processes must be scheduled, i.e. *Client* with probability p and ATM with probability $1 - p$.

Now, let us detail each component in isolation. Process *Client* repeatedly tries to insert a pin until the right number allows it to withdraw the cash:

$$Client \stackrel{\Delta}{=} insert\_pin.Client' +^q \tau.Client.$$

The alternative choice operator "$\_ +^q \_$" says that process *Client* can either insert a pin (output action *insert_pin*) with probability q, and afterwards behaving as process *Client*$'$, or stay idle (action τ) with probability $1 - q$, and afterwards behaving as the same process *Client*. The actions *insert_pin* and τ follow the *generative* model of probabilities [GSS95], which is the same model adopted by PCCP (cf. Section 2). In essence, the process autonomously decides, on the basis of a probability distribution (guided by parameter q), which action will be executed and how to behave after such an event.

$$Client' \stackrel{\Delta}{=} cash_*.\underline{0} +^{q'} fail_*.Client.$$

Process *Client*$'$ waits for the answer provided by the environment, i.e., it can either withdraw cash in case the pin number is right (input action $cash_*$), and afterwards stopping its execution (see the null term $\underline{0}$), or receive an unsuccessful message (input action $fail_*$), and afterwards behaving as process *Client* again. In practice, process *Client*$'$ internally reacts to the choice of the action type, *cash* or *fail*, performed by the environment (i.e., the machine). Formally, the input actions $fail_*$ and $cash_*$ follow the *reactive* model of probabilities [GSS95]. In particular, if the machine decides to communicate the action of type *fail*, then the client performs with probability 1 the unique input action of that type, which leads to process *Client*. Similarly, if the machine outputs the action of type *cash*, then the client chooses the input action $cash_*$ and then stops its execution. As a consequence of such a behaviour, parameter q' is not considered or, equivalently, from the viewpoint of process *Client*$'$ in isolation, the choice between such actions is purely nondeterministic, because their execution is entirely guided by the external environment.

Process ATM, instead, is ready to accept new incoming pins or it stays idle:

$$ATM \stackrel{\Delta}{=} (insert\_pin_*.fail.ATM +^r insert\_pin_*.cash.ATM) +^{r'} \tau.ATM .$$

The two actions $insert\_pin_*$ model the internal reaction of process ATM to the choice of the action type $insert\_pin$ performed by its environment (i.e., the client). Such a reaction is guided by a probability distribution associated with the input actions of type $insert\_pin$ that process ATM can perform. More precisely, whenever the action type $insert\_pin$ is chosen by the client, process ATM reacts by choosing either the first action $insert\_pin_*$ with probability r and then refusing the pin (output action $fail$), or the second action $insert\_pin_*$ with probability $1 - r$ and then delivering cash (output action $cash$). Alternatively, if process ATM is not accepting pins from the environment, the internal action τ is repeatedly executed to model the idle periods of the machine. The choice between the input actions $insert\_pin_*$ and such an internal event is nondeterministic (parameter r' is not considered), because the execution of an action $insert\_pin_*$ is entirely guided by the external environment.

According to the considerations above, the processes interact in the composed system as follows. In the initial state of our example, the system executes a move of process $Client$ with probability p: it executes either the internal move τ with probability $p \cdot (1-q)$, or the synchronising move $insert\_pin$ with probability $p \cdot q$ (with probability $p \cdot q \cdot r$ it executes an $insert\_pin$ action synchronised with the first input action of process ATM and with probability $p \cdot q \cdot (1-r)$ an $insert\_pin$ action synchronised with the second input action of process ATM). Note that the result of a synchronisation between an output action $insert\_pin$ and an input action $insert\_pin_*$ is again an output action of type $insert\_pin$ (similarly as in CSP [Hoa85]). On the other hand, the system may schedule with probability $1 - p$ process ATM by executing its internal action τ, which gets the *entire* probability $1 - p$ associated to process ATM. Afterwards, if, e.g., the winning action is the action $insert\_pin$ leading to term $Client' \,\|_S^p\, cash.ATM$, then the system executes the synchronising action $cash$ with probability 1, because it is the unique action that can be performed by the composed system. In particular, note that action $fail_*$ of process $Client'$ is blocked, because the environment of process $Client'$, represented by process $cash.ATM$, is not available to output a corresponding output action $fail$. In Figure 8 we report the labeled transition systems that are associated with processes $Client$ and ATM in isolation and with the composed system.

The example above emphasises some features of the probabilistic process algebra that we now describe in more detail.

As far as the CSP-like communication policy is concerned, in any binary synchronisation at most one output action can be involved and, in such a case, the result is an output action of the same type. Instead, in case two input actions of type a synchronise, then the result is again an input action of type a. We recall that the actions belonging to the communication interface are constrained to synchronise, while all the other actions are locally and independently executed by the processes that compose the system.

As far as the probability model is concerned, we have seen that output and internal actions follow the generative model, while input actions follow the reactive model. Probabilistic choices among output/internal actions or among input

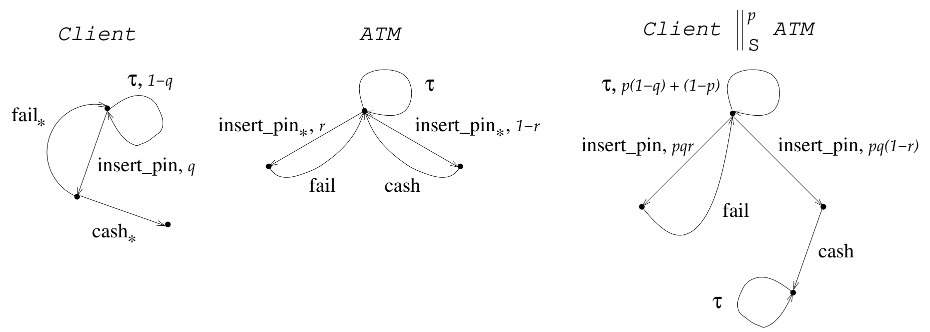


Fig. 8. Labeled transition systems associated to different process terms. Transitions are labeled with an action and a probability, which is equal to 1 if omitted.

actions of the same type are fully probabilistic, while in each other case the choice is purely nondeterministic. This is because input actions are underspecified, in the sense that their execution is guided by the environment behaviour. Hence, the parameters that probabilistically guide the choices come into play if and only if a probabilistic choice is really to be performed. Moreover, Example 9 has emphasised the following behaviors of the parallel operator:

- In case the execution of some output actions of P is prevented in $P \,||^p_S\, Q$ $(P \backslash L)$, the probabilities of executing the remaining output/internal actions of P are proportionally redistributed (similarly for Q). That is a standard approach when restricting actions in the generative model of probabilities [GSS95], as also seen in case of PCCP (cf. Section 2).
- In case of synchronising output actions a of P in $P \,||^p_S\, Q$, their probability is distributed among the multiple actions a obtained by synchronising with input actions a_* executable by Q, according to the probability the actions a_* are chosen in Q.

As a consequence of the policies specified above, we point out that in each system state of a process term, the sum of the probabilities of output and internal actions (input actions of a given type a), if there are any, is always equal to 1.

Now, we informally describe the behaviour of the hiding operator, which is needed to specify security properties. The hiding operator $P/^p_a$ turns output and input actions of type a into actions τ, by changing the probabilities according to the following rules.

- As far as output/internal actions executable by $P/^p_a$ are concerned, we distinguish the following cases:
  1. If P enables both some output/internal actions and some input actions a_*, then $P/^p_a$ chooses an action τ (obtained by hiding an action a_* of P) with probability p and an output/internal action previously enabled in P with probability $1 - p$. Such a rule guarantees that the hiding operator does not introduce nondeterminism among actions that follow the generative model of probability.

2. If either P does not enable output/internal actions, or P does not enable input actions a_*, then in $P/^p_a$ parameter p is not considered.

– As far as input actions are concerned, $P/^p_a$ enables the same input actions (with the same probability distribution) of type $b \neq a$ enabled in P.

Example 10. Consider process $P \stackrel{\Delta}{=} a_* +^{q'} (b +^q c)$, where the choice among a_* and the output actions is purely nondeterministic (parameter q' is not considered). The semantics of $P/^p_a$, which corresponds to process $\tau +^p (b +^q c)$, is a probabilistic choice between τ, executed with probability p, and the actions b and c, executed with probability $(1-p) \cdot q$ and $(1-p) \cdot (1-q)$, respectively. Hence, parameter p expresses the probability that the action τ obtained by hiding the input action a_* of P is executed with respect to the output actions previously enabled by P.

A goal of the hiding operator consists of turning open systems (i.e., systems enabling reactive choices due to input actions) into fully specified systems (i.e., fully generative systems, which do not include nondeterministic behaviours). In particular, the hiding operator resolves all the nondeterministic choices due to possible interactions with the environment by turning them into probabilistic choices. Intuitively, the effect of hiding an input action a_* corresponds to the execution of a synchronisation between a_* and an output action a offered by the environment. Such an interaction gives rise to an internal action τ whose probability distribution depends on parameter p of the hiding operator. When analysing security properties that employ the hiding operator, we will show that the low-level behaviour of a secure system does not depend on the choice of parameter p.

In the rest of the paper we use the following abbreviations. We assume parameter p to be equal to $\frac{1}{2}$ whenever it is omitted from any probabilistic operator. Moreover, when it is clear from the context, we use the abbreviation P/S, with $S = \{a_1, \ldots, a_n\} \subseteq AType$, to denote the expression $P/_{a_1} \ldots /_{a_n}$.

3.2 Operational Semantics and Equivalence

In this section, we provide a brief formal presentation of the semantics of the calculus. The reader not interested in such details can skip the rest of the section and proceed with the description of the security model.

The operational semantics of the probabilistic process algebra is given by the labeled transition system $(\mathcal{G}, Act, T)$, whose states are process terms and the transition relation T is the least multiset satisfying the operational rules reported in Table 3 and in Table 4. For a formal presentation of the semantics rules, the reader should refer to [BA03,ABG03], while here we just discuss some general aspects.

As far as the notation is concerned, we denote with $RAct$ and $GAct$ the sets of input actions, termed reactive actions, and of output and internal actions, termed generative actions, respectively. Then, we use the abbreviations $P \xrightarrow{\pi}$ to stand for $\exists p, P' : P \xrightarrow{\pi, p} P'$, denoting that P can execute action π with probability p and then behave as P', and $P \xrightarrow{G}$, with $G \subseteq GAct$, to stand for $\exists a \in G : P \xrightarrow{a}$, meaning that P can execute a generative action belonging to set G.

As far as the rules for $P +^p Q$ and $P \parallel_S^p Q$ are concerned, note that in addition to the reported rules, which refer to the local moves of the left-hand process P, we also consider the symmetric rules taking into account the local moves of the right-hand process Q. Such symmetric rules are obtained by exchanging the roles of terms P and Q in the premises and by replacing p with $1 - p$ in the label of the derived transitions. Moreover, we also point out that for both operators, parameter p comes into play if and only if a probabilistic choice between P and Q is really to be performed. For instance, in case of the alternative choice operator, if P enables at least a generative action and Q does not, then $P +^p Q$ performs a generative transition of P with probability 1. Otherwise, if both P and Q enable some generative actions, then $P +^p Q$ performs a generative transition of P with probability p.

Two important remarks are in order in case of the parallel operator. On the one hand, if both P and Q can execute some synchronising actions a_* in $P \parallel_S^p Q$, then the composed system can execute some actions a_*: the probability of each action a_* executable by $P \parallel_S^p Q$ is the product of the probabilities of the two actions a_* (one of P and one of Q) that are involved in the synchronisation. On the other hand, as also explained in the previous section, when considering $P \parallel_S^p Q$ we must pay attention to the computation of the probability distribution of its generative actions, whose overall probability must sum up to 1. To this aim, in semantics rules we employ the function $\nu_P(G_{S,Q}) : \mathcal{P}(AType \cup \{\tau\}) \longrightarrow]0, 1]$, which computes the sum of the probabilities of the generative transitions of P (executable by $P \parallel_S^p Q$) whose type belongs to set $G_{S,Q} \subseteq AType \cup \{\tau\}$. In particular, set $G_{S,Q} = \{a \in AType \cup \{\tau\} \mid a \notin S \vee (a \in S \wedge Q \xrightarrow{a_*})\}$ contains the action types not belonging to the synchronisation set S and the action types belonging to S for which an input action of Q can be performed. Hence, $\nu_P(G_{S,Q})$ computes the aggregate probability of the generative transitions of P that can be executed by $P \parallel_S^p Q$ and can be used to normalise the probabilities of the generative transitions of P.

Finally, note that the tables omit the rules for the restriction operator. This is because it can be easily derived from the parallel operator. Indeed, we have that $P \backslash L$ corresponds to process $P \parallel_L \underline{0}$.

Since the security model we are going to present is based on the semantics of processes (i.e., the security check considers the program behaviour), we need an equivalence relation allowing for a comparison among the observable behaviours of different systems. To this aim, we resort to a probabilistic variant of the weak bisimulation [BH97], which abstracts away from τ actions and is able to identify deadlock. More precisely, such a relation, termed $\approx_{\mathrm{PB}}$, is a probabilistic extension of the nondeterministic weak bisimulation ($\approx_{\mathrm{B}}$) of [Mil89]. In essence, $\approx_{\mathrm{PB}}$ replaces the classical weak transitions of $\approx_{\mathrm{B}}$ by the probability of reaching classes of equivalent states. The notion of weak probabilistic bisimulation is based on the following definitions (for more details, see [ABG03]). We use a function

$$\pi.P \xrightarrow{\pi,1} P$$

$$\frac{P \xrightarrow{a_*,q} P' \quad Q \xrightarrow{a_*}}{P +^p Q \xrightarrow{a_*,p\cdot q} P'} \qquad \frac{P \xrightarrow{a_*,q} P' \quad Q \stackrel{a_*}{\not\longrightarrow}}{P +^p Q \xrightarrow{a_*,q} P'}$$

$$\frac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{GAct}}{P +^p Q \xrightarrow{a,p\cdot q} P'} \qquad \frac{P \xrightarrow{a,q} P' \quad Q \stackrel{GAct}{\not\longrightarrow}}{P +^p Q \xrightarrow{a,q} P'}$$

$$\frac{P \xrightarrow{a_*,q} P' \quad P \xrightarrow{GAct}}{P/^p_a \xrightarrow{\tau,p\cdot q} P'/^p_a} \qquad \frac{P \xrightarrow{a_*,q} P' \quad P \stackrel{GAct}{\not\longrightarrow}}{P/^p_a \xrightarrow{\tau,q} P'/^p_a}$$

$$\frac{P \xrightarrow{b_*,q} P'}{P/^p_a \xrightarrow{b_*,q} P'/^p_a} \quad a \neq b$$

$$\frac{P \xrightarrow{b,q} P' \quad P \xrightarrow{a_*}}{P/^p_a \xrightarrow{b,(1-p)\cdot q} P'/^p_a} \quad a \neq b \quad \frac{P \xrightarrow{a,q} P' \quad P \xrightarrow{a_*}}{P/^p_a \xrightarrow{\tau,(1-p)\cdot q} P'/^p_a}$$

$$\frac{P \xrightarrow{b,q} P' \quad P \stackrel{a_*}{\not\longrightarrow}}{P/^p_a \xrightarrow{b,q} P'/^p_a} \quad a \neq b \quad \frac{P \xrightarrow{a,q} P' \quad P \stackrel{a_*}{\not\longrightarrow}}{P/^p_a \xrightarrow{\tau,q} P'/^p_a}$$

$$\frac{P \xrightarrow{\pi,q} P'}{A \xrightarrow{\pi,q} P'} \quad \text{if } A \stackrel{\Delta}{=} P$$

Table 3. Operational semantics (part I)

Prob such that $Prob(P, a_*, C)$ denotes the aggregate probability of going from P to a term in the class (of equivalent terms) C by executing an action a_*. Moreover, $Prob(P, \tau^*a, C)$ expresses the aggregate probability of going from P to a term in the equivalence class C via sequences of the form τ^*a (if $a \neq \tau$) or τ^* (if $a = \tau$). Formally:

$Prob(P, \tau^*a, C) =$

$$\begin{cases} 1 & \text{if } a = \tau \wedge P \in C \\ \sum_{Q\in\mathcal{G}} Prob(P,\tau,Q) \cdot Prob(Q,\tau^*,C) & \text{if } a = \tau \wedge P \notin C \\ \sum_{Q\in\mathcal{G}} Prob(P,\tau,Q) \cdot Prob(Q,\tau^*a,C) + Prob(P,a,C) & \text{if } a \neq \tau. \end{cases}$$

Definition 6. *An equivalence relation $R \subseteq \mathcal{G} \times \mathcal{G}$ is a weak probabilistic bisimulation if and only if, whenever $(P, Q) \in R$, then for all $C \in \mathcal{G}/R$:*

$$\frac{P \xrightarrow{a_*,q} P' \quad Q \xrightarrow{a_*}}{P \parallel_S^p Q \xrightarrow{a_*,p\cdot q} P' \parallel_S^p Q} \; a \notin S \qquad \frac{P \xrightarrow{a_*,q} P' \quad Q \not\xrightarrow{a_*}}{P \parallel_S^p Q \xrightarrow{a_*,q} P' \parallel_S^p Q} \; a \notin S$$

$$\frac{P \xrightarrow{a_*,q} P' \quad Q \xrightarrow{a_*,q'} Q'}{P \parallel_S^p Q \xrightarrow{a_*,q\cdot q'} P' \parallel_S^p Q'} \; a \in S$$

$$\frac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{G_{S,P}}}{P \parallel_S^p Q \xrightarrow{a,p\cdot q/\nu_P(G_{S,Q})} P' \parallel_S^p Q} \; a \notin S$$

$$\frac{P \xrightarrow{a,q} P' \quad Q \not\xrightarrow{G_{S,P}}}{P \parallel_S^p Q \xrightarrow{a,q/\nu_P(G_{S,Q})} P' \parallel_S^p Q} \; a \notin S$$

$$\frac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{a_*,q'} Q' \quad Q \xrightarrow{G_{S,P}}}{P \parallel_S^p Q \xrightarrow{a,p\cdot q'\cdot q/\nu_P(G_{S,Q})} P' \parallel_S^p Q'} \; a \in S$$

$$\frac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{a_*,q'} Q' \quad Q \not\xrightarrow{G_{S,P}}}{P \parallel_S^p Q \xrightarrow{a,q'\cdot q/\nu_P(G_{S,Q})} P' \parallel_S^p Q'} \; a \in S$$

Table 4. Operational semantics (part II)

– $Prob(P, \tau^* a, C) = Prob(Q, \tau^* a, C) \ \forall a \in GAct$
– $Prob(P, a_*, C) = Prob(Q, a_*, C) \ \forall a_* \in RAct.$

Two terms $P, Q \in \mathcal{G}$ are weakly probabilistically bisimulation equivalent, denoted $P \approx_{\mathrm{PB}} Q$, if there exists a weak probabilistic bisimulation R containing the pair (P, Q).

Note that such a definition requires two equivalent terms to be strongly equivalent in case of reactive actions and weakly equivalent in case of generative actions. This is because τ is a generative action, therefore the computation of the probability of executing a mixed trace of generative/reactive actions (like, e.g., $\tau^* a_*$) does not actually make sense.

Example 11. Consider the processes $P \stackrel{\Delta}{=} a +^{\frac{1}{2}} b$ and $Q \stackrel{\Delta}{=} \tau.Q +^{\frac{1}{3}} (a +^{\frac{1}{2}} b)$, which, from an external observer viewpoint, behave the same since they execute either an output action a or an output action b with equal probabilities. We now want to formally verify such an intuition, i.e. we show that P and Q are weakly probabilistically bisimulation equivalent. Let R be the relation that considers the classes $\{C, [\underline{0}]\}$, where $C = \{P, Q\}$ and $[\underline{0}] = \{\underline{0}\}$. The only interesting case is given by $Prob(P, \tau^*\pi, [\underline{0}]) = \frac{1}{2}$, where $\pi \in \{a, b\}$. In order to compute the probability $Prob(Q, \tau^*\pi, [\underline{0}])$ we must consider that Q can execute an arbitrary number of times the action τ before reaching state $\underline{0}$ via an action a (b). To this aim, we redistribute the probability $\frac{1}{3}$ associated with the outgoing internal transition of Q among the other outgoing transitions of Q. Formally, by applying the definition of function $Prob$, we obtain $Prob(Q, \tau^*a, [\underline{0}]) = \frac{1}{3} \cdot Prob(Q, \tau^*a, [\underline{0}]) + \frac{1}{3}$, from which we derive $Prob(Q, \tau^*a, [\underline{0}]) = \frac{1}{2}$ (similarly for b). Hence, R is a weak probabilistic bisimulation and $P \approx_{\mathrm{PB}} Q$.

3.3 Probabilistic Noninterference

Probabilistic noninterference extends the classical, possibilistic definition of noninterference by providing the means for:

1. capturing those covert channels that are not observable in a purely nondeterministic setting, and
2. measuring the information leakage in terms of probability of observing the related covert channel.

In this section, we show how to formalise probabilistic noninterference in our process algebraic framework, while in the next one we extend the same approach in order to deal with the problem of giving a quantitative estimation of the information leakage.

As usual in security models, in our process algebraic framework we distinguish among high-level visible actions and low-level visible actions by defining two disjoint sets $AType_H$ of high-level types and $AType_L$ of low-level types, which form a covering of $AType$, such that the output action a and the input action a_* are high- (low-) level actions if $a \in AType_H$ ($a \in AType_L$). Usually, we use $l, l', \ldots$ to denote low-level types and $h, h', \ldots$ to denote high-level types. Then, in such a setting, we provide a semantics-based approach to noninterference, i.e., an approach where different program behaviours are compared to analyse a security property. Roughly, we derive two models from the algebraic specification of the system at hand, and then check the semantic equivalence between such derived models. On the one hand, the definition of semantic equivalence between processes is based on the weak probabilistic bisimulation equivalence $\approx_{\mathrm{PB}}$. On the other hand, the choice of the sub-models to be compared depends on the definition of the security property. Here, we consider the noninterference property of [Ald01,ABG03], which in turn is the probabilistic version of the Strong Nondeterministic Noninterference property proposed in [FG95] to express the classical noninterference idea of [GM82]. In essence, in order to detect potential high-level interferences, we compare the low-level behaviours of the system model P that can be observed in two different scenarios differing in the high-level behaviours only. In the former scenario, P is isolated from the high-level environment, so that all its high-level interactions are prevented, while in the latter scenario, P interacts with any high-level user that enables all the high-level actions of P.

The definition of Probabilistic Noninterference, here termed PNI, is as follows. For the sake of conciseness, we denote with $h_1^P, \ldots, h_n^P$ the sequence (in alphabetic order) of high-level types that syntactically occur in the action prefix operators within P.

Definition 7. $P \in PNI \Leftrightarrow P \backslash AType_H \approx_{PB} P/^{p_1}_{h_1^P} \dots /^{p_n}_{h_n^P} \; \forall p_1, \dots, p_n \in]0,1[$.

Such a formulation also defines the particular class of adversaries (high-level users) with respect to which the probabilistic noninterference property is parameterised. Formally, according to the *PNI* definition, we can argue as follows.

On the one hand, $P \backslash AType_H$ expresses the low-level view of the system in isolation (without high-level interactions with the environment), since all the high-level actions are prevented.

On the other hand, $P/^{p_1}_{h_1^P} \dots /^{p_n}_{h_n^P} \; \forall p_1, \dots, p_n \in]0,1[$, where all the high-level actions are hidden, expresses the low-level view of P in case all the high-level interactions with the environment are enabled. In this formula, the hiding operator models the behaviour of any high-level user H that allows all the high-level actions enabled by P to be executed. More precisely, H allows the high-level output actions of P (turned into internal τ actions) to be executed with the probability distribution chosen by P itself. On the other hand, H allows the high-level input actions of P (turned into internal τ actions) to be executed with a probability distribution chosen by H itself according to parameters $p_1, \dots, p_n$.

The class of attackers considered by the *PNI* property, here called $\mathcal{A}_{PNI}$, contains active and memoryless high-level users. More precisely, they are active as they can affect the probabilistic behaviour of the system activities, and they are memoryless as they cannot alter their behaviour depending on the previous history. In particular, as stated by the hiding operators, the probability distributions for the high-level inputs are chosen a priori and do not change during the system execution.

Example 12. Consider a program that writes a low-level variable in two possible ways, only one being legal, and represented by the following system:

$$\begin{aligned} P \stackrel{\Delta}{=} \; & \tau \,.\, (copy\_secret\_PIN +^{0.001} copy\_random\_value) +^{p} \\ & high \,.\, (copy\_secret\_PIN +^{0.5} copy\_random\_value). \end{aligned}$$

If the high-level user interacts with the system (such a communication is modeled by the execution of the high-level action *high*), then the program assigns to the public variable either a confidential value (low-level action *copy_secret_PIN*) with probability 0.5 or a random value (low-level action *copy_random_value*) with equal probability 0.5. On the other hand, if the high-level user does not interfere, then the program performs an internal activity that leads to the execution of the illegal assignment with a negligible probability. The choice between the interaction with the high-level user and the internal action is left to the system, which performs it according to parameter p.

A nondeterministic approach to noninterference [2] does not reveal any covert channel. This is because independently of the high-level behaviour, the low-level view of the system is always the same. However, if an external observer

[2] [Ald02,ABG03] rephrase the approach of [FG95] in a nondeterministic simplification of our process algebra, thus obtaining the same security property taxonomy.

considers the outcomes of repeated executions of the system, then the relative frequency of such outcomes reveals the high-level interference. Formally, we have that $P \backslash AType_H$ and $P/AType_H$ are not weakly probabilistically bisimulation equivalent. For instance, we have that $P \backslash AType_H$ performs the action *copy_secret_PIN* (preceded by an invisible transition) with probability 0.001, while $P/AType_H$ executes the same observable action (preceded by an invisible transition) with probability $p \cdot 0.001 + (1-p) \cdot 0.5$. Therefore, the *PNI* property is more than enough to capture the probabilistic covert channel described above.

3.4 Approximate Noninterference

In this section, we show how the knowledge about the probabilistic behaviour of a system may help the modeler to give a quantitative estimation of each information leakage, thus overcoming the qualitative view according to which a system is or is not secure. More precisely, given a covert channel that is responsible for an illegal information flow (which, e.g., could be revealed also in the possibilistic setting), we can evaluate the effectiveness of such a covert channel, by measuring the probability for an external observer of detecting it.

From a practical standpoint, a quantitative (probabilistic) approach to information flow analysis is useful for the verification of the security level of systems for which probabilities play an important role. For instance, many problems can be solved by using deterministic algorithms that turn out to be secure and require exponential time. On the other hand, probabilistic algorithms are often implemented that solve the same problems in polynomial time (see, e.g., [CKV00,MR99]). In such a case, the price to pay for a computational gain is the possibility for the observer of detecting an illegal information flow. Because of such a possibility, a probabilistic algorithm cannot be secure in case we limit the information flow analysis to the nondeterministic case. Instead, if we resort to a probabilistic approach, we can formally prove that the same algorithm has an illegal information flow, which, however, occurs with probability close to 0 (see, e.g., [AG02]). Based on these considerations, we need a quantitative approach in order to estimate the difference between the non-secure system and a secure one.

In our process algebraic setting, we may try to analyse the labeled transition system underlying an algebraic specification, in order to compute the probability that an information flow (from high level to low level) really happens. Unfortunately, a solution to such a problem cannot be provided if the verification of the security properties depends on a behavioural equivalence relation like the weak probabilistic bisimulation considered in the previous sections. This is because any equivalence relation states whether or not two given transition systems behave exactly the same. From a security standpoint, such an approach simply provides a binary answer: the system suffers or does not suffer an information leakage. Hence, small fluctuations in the system behaviour cannot be tolerated. Instead, we need a relaxed relation, which cannot be an equivalence relation, allowing for *similar* processes to be related, where the term *similar* stands for "behave almost the same up to small fluctuations".

On the basis of the considerations above, we now introduce a quantitative notion of behavioural similarity for deciding if two probabilistic processes are confined or, more precisely, for measuring the distance between probabilistic transition systems.

Formally, we now introduce the definition of weak probabilistic bisimulation with ε-precision, which is a relaxed version of the weak probabilistic bisimulation $\approx_{\mathrm{PB}}$ presented in Section 3.2.

Definition 8. *A relation $R \subseteq \mathcal{G} \times \mathcal{G}$ is a weak probabilistic bisimulation with ε-precision, where $\varepsilon \in]0,1[$, if and only if, whenever $(P, Q) \in R$, then for all $C \in \mathcal{G}/R$:*

- $|\, Prob(P, \tau^* a, C) - Prob(Q, \tau^* a, C) \,| \leq \varepsilon \quad \forall a \in GAct$
- $|\, Prob(P, a_*, C) - Prob(Q, a_*, C) \,| \leq \varepsilon \quad \forall a_* \in RAct.$

We use the abbreviation $P \approx_{PB_\varepsilon} Q$ to denote that there exists a weak probabilistic bisimulation with ε-precision R containing the pair (P, Q); alternatively, we say that P (Q) is a ε-perturbation of Q (P). Note that $\approx_{PB_\varepsilon}$ is not a transitive relation and, therefore, it cannot be an equivalence relation.

Example 13. Let us consider the fully specified transition systems depicted in Figure 9, which enable generative transitions only. It is easy to see that they cannot be weakly probabilistically bisimulation equivalent according to the definition of $\approx_{PB}$. Indeed, we have that s_2 and s_4 belong to the same equivalence class, while s_0, s_1, and s_3 are in three separate classes, since they have different probabilities of reaching the class $[\underline{0}]$ of the null term by executing the sequence $\tau^* a$ $(\tau^* b)$. However, we can observe that the observable behaviours of such systems are almost the same up to a perturbation ε. More formally, if we tolerate a distance at most equal to ε, we can define a relation that is a weak probabilistic bisimulation with ε-precision as follows. First, we immediately obtain that s_1 and s_2 (s_4) are similar, i.e. they belong to the same class C. For the same reason, we have that s_0 is in C, since $Prob(s_0, \tau^* a, [\underline{0}]) = \frac{1}{2} \cdot (\frac{1}{2} + \varepsilon) + \frac{1}{4} = \frac{1}{2} + \frac{1}{2} \cdot \varepsilon$ (similarly, for b we obtain $\frac{1}{2} - \frac{1}{2} \cdot \varepsilon$). Finally, s_3 is in C too, since $Prob(s_3, \tau^* a, [\underline{0}]) = \varepsilon + \frac{1}{2} \cdot (1 - \varepsilon) = \frac{1}{2} + \frac{1}{2} \cdot \varepsilon$ and $Prob(s_3, \tau^* b, [\underline{0}]) = \frac{1}{2} \cdot (1 - \varepsilon) = \frac{1}{2} - \frac{1}{2} \cdot \varepsilon$. Therefore, we have obtained a weak probabilistic bisimulation with ε-precision including the pair (s_0, s_3), i.e. the two transition systems are a ε-perturbation of the same system.

3.5 Approximating *PNI*

The similarity relation can be used to approximate the noninterference property by simply replacing the equivalence relation in its formulation with such a similarity relation. In essence, instead of qualitatively asserting whether or not two sub-models of the system are equivalent, we just look at how much they differ. Since the sub-models to be compared express the low-level behaviour in case the system is isolated from the high environment and the low-level behaviour in

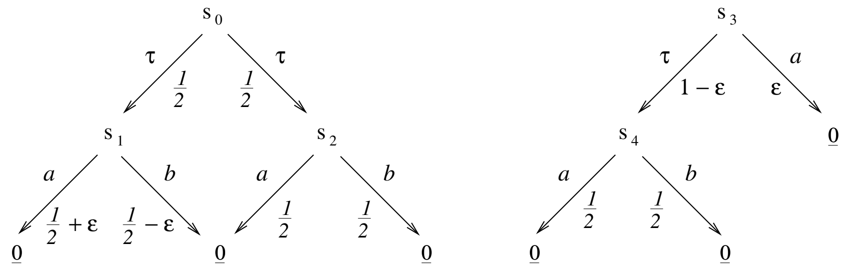


Fig. 9. Example of weak probabilistic bisimulation with ε-precision

case the system interacts with high users, respectively, an approximated noninterference property quantitatively states the capacity of a low-level observer of guessing the high environment behaviour by observing the system execution.

In our setting, the definition of process similarity is not parametric with respect to a specific set of adversaries (admissible spies, as termed in Section 2.4). Instead, the given security property is parameterised by a particular class of adversaries. Hence, security strictly depends on the definition of the property. In particular, here we show what happens when approximating the *PNI* property, which, as we have seen, is parameterised with respect to a particular class $\mathcal{A}_{PNI}$ of adversaries. In particular, if we replace in the definition of *PNI* the weak probabilistic bisimulation with the weak probabilistic bisimulation with ε-precision, we obtain a relaxed property that states if the behaviour of P in isolation is close (according to the *distance* ε) to that observed when P interacts with anyone of the high-level users in $\mathcal{A}_{PNI}$.

Example 14. Consider the system of Figure 10:

$$P \stackrel{\Delta}{=} h.l'.\underline{0} +^p \tau.(l.\underline{0} +^q h.l.\underline{0})$$

where it can be observed that:

- the left-hand component, which is chosen with probability p, is clearly non-secure, since the execution of the action l' reveals to the low-level observer that the action h occurred;
- the right-hand component, which is chosen with probability $1-p$, is secure, since independently of the (probabilistic) high behaviour a low-level observer always sees the action l with probability 1.

We point out that the probabilistic information is not necessary to capture a covert channel in P, which is easily revealed as a "1-bit covert channel" by the nondeterministic counterpart of the *PNI* property [ABG03]. In other words, the probabilistic information described in P is not responsible for the information leakage. In spite of this, such an information turns out to be useful to analyse the security level of P. Indeed, the observation of the frequency of the possible outcomes of repeated executions of the system reveals that the behaviour of P

is secure with probability $1-p$ and discloses an unwanted information flow with probability p. In practice, after a certain number, let us say n, of experiments during which the high-level user interacts with P, it turns out that the mean number of l' that have occurred is $n \cdot p$, while the mean number of l that have occurred is $n \cdot (1-p)$. Instead, after n experiments during which the high-level user does not interact with P, it turns out that the number of l that have occurred is n. Obviously, by observing the relative frequencies "on the long run" of the observable results, we have that $P \backslash AType_H$ and $P/AType_H$ will differ by exactly a factor p. That means if an external observer executes the system (under one of the two scenarios) "infinitely often", then it can determine whether or not the high-level user was interfering. However, in a realistic scenario, after a finite number n of experiments and in case p is a value very close to 0, it is very hard for an external observer to understand whether or not the system was interacting with the high-level user. In such a case, the covert channel occurs with a negligible probability and P may be considered as a good approximation of a secure system.

The standard interpretation of probabilities as relative frequencies also helps to give an estimation of the covert channel capacity. Indeed, if we assume, e.g., that the system above is executed n times per week, then we can conclude that such a system suffers an information leakage equal to $n \cdot p$ bits per week, since that is (on average) the number of experiments that reveals the high-level user behaviour.

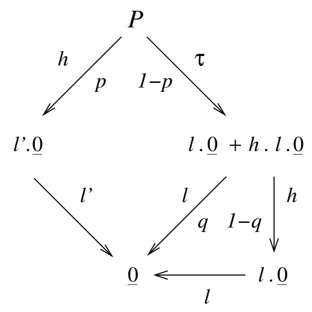


Fig. 10. Example of probabilistic information flow

Now, we formally show how the weak probabilistic bisimulation with ε-precision is able to determine the security level of P. According to the *PNI* definition, we have $P \backslash AType_H \not\approx_{PB} P/AType_H$. However, $P \backslash AType_H$ is a p-perturbation of $P/AType_H$, since $P \backslash AType_H \approx_{PB} \tau.l.\underline{0} \approx_{PB} \tau.(l.\underline{0} +^q \tau.l.\underline{0})$ $\approx_{PB_p} \tau.l'.\underline{0} +^p \tau.(l.\underline{0} +^q \tau.l.\underline{0}) \approx_{PB} P/AType_H$. Therefore, the system can be considered secure enough as p tends to the value 0. Note that, according to the definition of weak probabilistic bisimulation with ε-precision, if p is less than the threshold ε, then the subsystem P' reached from $P/AType_H$ by executing the hidden high-level action h is simply disregarded, since it expresses a behaviour of the system reachable with a negligible probability. Therefore P' is not to be related with any corresponding behaviour of the system $P \backslash AType_H$.

Example 15. As another example, consider the following probabilistic process:

$$P \stackrel{\Delta}{=} (l.\underline{0} +^p l.l'.\underline{0}) +^q l.h.l'.\underline{0}.$$

It is easy to see that P is not *PNI* secure. Formally, let us denote by C_1 the equivalence class of the null term $\underline{0}$ and by C_2 the equivalence class of term $l'.\underline{0}$. On the one hand, we have $Prob(P \backslash AType_H, \tau^* l, C_1) = q \cdot p + 1 - q$ and $Prob(P \backslash AType_H, \tau^* l, C_2) = q \cdot (1-p)$. On the other hand, we have $Prob(P / AType_H, \tau^* l, C_1) = q \cdot p$ and $Prob(P / AType_H, \tau^* l, C_2) = 1 - q \cdot p$. Therefore, $|Prob(P \backslash AType_H, \tau^* l, C_1) - Prob(P / AType_H, \tau^* l, C_1)| = 1 - q = |Prob(P \backslash AType_H, \tau^* l, C_2) - Prob(P / AType_H, \tau^* l, C_2)|$, from which we derive that (i) process P does not satisfy the *PNI* property, and (ii) $P \backslash AType_H \approx_{PB_\varepsilon} P / AType_H$ if $q \geq 1 - \varepsilon$. Intuitively, if q is close to 1, then the low view of P, with or without the interaction with the high-level user, changes according to a small ε-fluctuation. While on the long run such a difference can be precisely identified, for a finite number of experiments $P \backslash AType_H$ and $P / AType_H$ turn out to behave almost the same. That means if we observe the low-level outcome of repeated executions of the system we are not able to notice the behaviour of the high-level user, since the high interference changes the frequency associated with each possible low-level outcome according to small, negligible fluctuations.

3.6 Statistical Interpretation

In a realistic scenario, an external observer makes a guess about the high environment behaviour after a certain number of tests (system executions). That means we need a formal way to measure the difference (by a finite number of experiments) between the low view of P in isolation, modeled by process $P \backslash AType_H$, and the low view of P interacting with any high user in $\mathcal{A}_{PNI}$, expressed by process $P /^{p_1}_{h_1^P} \dots /^{p_n}_{h_n^P}$ for any sequence of probabilities $p_1, \dots, p_n \in]0,1[$. The capability of the observer of revealing the difference between such processes expresses a measure of the effectiveness of the covert channel from high level to low level.

As an expected result, we can rephrase in our setting the same approach described in Section 2.5 to evaluate the confidence we can have in our hypothesis about the identity of a process after a finite number of experiments. We omit the technical part concerning the statistical methods behind such an approach (see Section 2.5) and we directly proceed with some clarifying examples.

Example 16. Consider the system:

$$\begin{aligned} P &\stackrel{\Delta}{=} h_*.(l.\underline{0} +^{\frac{2}{3}} l'.\underline{0}) + (l.\underline{0} +^{\frac{7}{12}} l'.\underline{0}) \text{ such that} \\ P \backslash AType_H &\approx_{PB} (l.\underline{0} +^{\frac{7}{12}} l'.\underline{0}) \text{ and} \\ P /^p_h &\approx_{PB} \tau.(l.\underline{0} +^{\frac{2}{3}} l'.\underline{0}) +^p (l.\underline{0} +^{\frac{7}{12}} l'.\underline{0}). \end{aligned}$$

According to the low view of the system in isolation, expressed by term $P \backslash AType_H$, a low-level observer sees the action l with probability $\frac{7}{12}$ and the action l' with probability $\frac{5}{12}$. On the other hand, if P interacts with a high-level user that synchronises with the reactive action h_* with probability p, then the low view of the system changes. In particular, a low-level observer sees the action l with probability $\frac{7}{12} + \frac{1}{12} \cdot p$ and the action l' with probability $\frac{5}{12} - \frac{1}{12} \cdot p$. That means for $p \in]0, 1[$ the probability of observing the action l varies in the range $]\frac{7}{12}, \frac{2}{3}[$ and the probability of observing the action l' is in the range $]\frac{1}{3}, \frac{5}{12}[$. As a consequence, it turns out that $P/^p_h$ is a $\frac{1}{12}-$perturbation of $P \backslash AType_H$ for all $p \in]0, 1[$. Formally, it is easy to verify that $P \backslash AType_H \approx_{PB_{\frac{1}{12}}} P/^p_h$, for all $p \in]0, 1[$.

An external low-level observer tries to distinguish the case in which P is isolated from the high environment from the case in which P interacts with a high-level user. To this purpose, he observes the relative frequencies of the low-level outcomes that derive from repeated executions of the system. After a number n of experiments, he formulates a hypothesis about the scenario in which P has been executed. The confidence he can have in such a hypothesis can be determined as reported in Section 2.5. In particular, we know that an upper bound for the distance between processes $P \backslash AType_H$ and $P/^p_h$ is $\varepsilon = \frac{1}{12}$. If we consider the scenario in which P is isolated from the high environment and we concentrate on the low-level outcome l (whose probability is equal to $\frac{7}{12}$), we obtain the same results shown in Example 8. More precisely, if we assume $n = 9$, we have that the hypothesis formulated by the low-level observer will be right with an about 60% chance, while for $n = 144$ it will be correct with about 85%.

Example 17. Now, let us consider again the same process P of Example 15. We want to estimate the confidence an external observer can have in a hypothesis about the high environment behaviour after a finite number n of experiments. To this purpose, let us assume $p = 0.5$ and $q = 0.99$. Such a scenario expresses the fact that the two possible behaviours (i.e. the single output l and the sequence $l.l'$) are chosen by the system with equal probabilities except for a small fluctuation due to scarce interferences by the high-level user. Formally, in $P \backslash AType_H$ the probability of observing the sequence $l.l'$ is equal to 0.495, while in $P/AType_H$ such a probability is equal to 0.505. Symmetrically, we can compute the probability of observing a single l, which is equal to 0.505 for $P \backslash AType_H$ and equal to 0.495 for $P/AType_H$. According to what we have shown in Example 15, the distance between such processes is $\varepsilon = 0.01$. Now, we assume that the high-level user is interacting with the system and we concentrate on the sequence of events $l.l'$. The probability $\mathbf{P}$ for an external low-level observer to identify the correct high environment behaviour depends on the number n of experiments. In particular, for $n = 10$ we have (cf. Section 2.5):

$$a_0(10) = \frac{10 \cdot 0.01}{2} \frac{1}{\sqrt{10 \cdot 0.505 \cdot 0.495}} \approx 0.03$$

and

$$\mathbf{P} = 1 - \int_{0.03}^{\infty} \exp\left(\frac{-x^2}{2}\right) \approx 0.512$$

Hence, for 10 tests the hypothesis that the observer formulates will be right with about 51%. Note that the probability of the best blind guess the observer can make is exactly 50%. We also emphasise that if we want such a probability to reach about 90%, then the external observer should execute about 16640 experiments.

3.7 The ATM example

We present a simple but real example showing the need for a quantitative estimation of illegal information flows. In particular, we consider an Automatic Teller Machine (ATM), which gives cash if and only if the client inserts the unique, correct PIN number i (of m possible PINs) within a fixed number, say n, of attempts, after which the ATM retires the card:

$$ATM_k \stackrel{\Delta}{=} insPIN_{i_*}.cash.ATM_1 + \textstyle\sum_{j=1,j\neq i}^{m} insPIN_{j_*}.fail.ATM_{k+1} \quad 0 < k < n$$
$$ATM_n \stackrel{\Delta}{=} insPIN_{i_*}.cash.ATM_1 + \textstyle\sum_{j=1,j\neq i}^{m} insPIN_{j_*}.retire.ATM_1$$

An attacker that is in possession of the card (but not of the PIN) may try to illegally withdraw cash:

$$Spy \stackrel{\Delta}{=} insPIN_1.Spy' +^{p_1} (insPIN_2.Spy' +^{p_2} \ldots)$$
$$Spy' \stackrel{\Delta}{=} cash_*.spend.\underline{0} + fail_*.Spy + retire_*.flee.\underline{0}$$

We can assume that *cash* is the unique low-level action, since it expresses the tangible proof that a dishonest spy withdrew cash, while all the other events are considered to be high-level actions. If we take the composed system

$$ATMSys \stackrel{\Delta}{=} ATM_1 \parallel_{\{cash,retire,fail,insPIN_i,\, i=1,\ldots,m\}} Spy$$

and check the nondeterministic counterpart of *PNI* [ABG03], we observe that the system is clearly non-secure. Indeed, if we hide the high-level actions, expressing the fact that the attacker interacts with the machine, then the action *cash* is observable. On the contrary, if we purge the system of the high-level actions, modeling the lack of any interaction between the machine and the attacker, then the action *cash* is not executable. Obviously, a purely nondeterministic approach captures the fact that an illegal behaviour can be observed in case the spy guesses the right PIN. In a realistic scenario, such an event is possible but negligible. For instance, assume that for any attempt the spy randomly samples a PIN value according to a uniform distribution, and take two realistic parameters, i.e. $m = 100000$ and $n = 3$. Then, denoted C the equivalence class of the null term, we have that $Prob(ATMSys/AType_H, \tau^* cash, C) \approx 0.00003$. Formally, if we employ the weak probabilistic bisimulation with ε-precision ($\varepsilon = 0.00003$), then the system turns out to satisfy the approximated *PNI* property. This is because the probability of observing the illegal cash leakage is considered to be negligible.

4 Related Work and Conclusion

In this paper, we surveyed two techniques for approximating noninterference properties, thus enriching the intuition behind the definition of probabilistic noninterference, which appeared in the literature to overcome the limitations of classical possibilistic approaches to information flow analysis. Initially, a formulation of probabilistic covert channel was proposed in [McL90,Gra90], and later on in [Gra92] and in [GS92,SG95]. More recently, in [SS00] the same intuition has been rephrased in the setting of an imperative language with dynamic thread creation, where, as a novelty, a probabilistic notion of bisimulation is used to formalise a security condition. In [Smi01], a type system is presented that aims to ensure secure information flow in a simple multi threaded imperative programming language running under a uniform probabilistic scheduler. The same author also employs a definition of weak probabilistic bisimulation (inspired by [BH97]) in [Smi03].

In the first approach presented in this paper we have concentrated on a notion of observable behaviour for programs in the PCCP language, which corresponds to the probabilistic input/output observables. These can be described by probability distributions on the underlying space of constraints, and we used a vector norm to measure their similarity. By considering the observables of two processes executed in the context of a spy we were then able to measure their confinement. Different analyses can be constructed depending on the type of attacks we consider. For example, in [DHW03b,DHW02b] a control-flow analysis for the confinement property is presented which refers to *internal attacks*. This is the case where the attacker is part of the observed system and is therefore subject to the same scheduler as the host system. In another context one might be interested in *external attacks*, where the attacker is only allowed to observe the system from the outside and is thus scheduled in a different way, or one might impose other restrictions on the way a spy may observe the agents in question. In [DHW02a], an analysis is presented for the case of external attacks, which exploits information about the average store of an agent in some specified number of steps (the observation time).

In the second approach we described, the notion of observable behaviour for processes is formalised in the process algebraic calculus of [BA03], whose semantics is given in terms of a probabilistic version of the weak bisimulation equivalence [BH97]. In this setting, we have shown that the robustness of a system against a specified class of attackers (as defined by the probabilistic noninterference property) can be checked by following the same approach introduced in [FG95] in a purely nondeterministic framework. Along this line, in [ABG03] a complete taxonomy of probabilistic security properties is described. The expressiveness of the probabilistic process algebra and of the particular model of probability we adopted allow us to model and analyse real, complex systems. For example, in [AG02], a case study shows the adequacy of such an approach for analysing the security level (under any probabilistic adversary) of a probabilistic cryptographic protocol [MR99] implemented to achieve a fairness property.

In the literature, several papers propose a formal definition of approximated bisimilarity. For example, in [vBW01,DGJP99] different pseudometrics are introduced that quantify the similarity of the behavior of probabilistic transition systems that are not bisimilar. In particular, in [DGJP99] the authors consider a metric on partial labeled Markov chains, which are a generalization of the fully specified transition systems described in Sect. 3, in that for each state the sum of the probabilities of the outgoing transitions, if there are any, is less than (or equal to) 1, while in our case such a sum always sums up to 1. Moreover, they extend the same approach to the weak bisimulation case in [DGJP02]. With respect to the notion of approximated weak probabilistic bisimulation $\approx_{PB_\varepsilon}$, the approximated equality introduced in [DGJP02] is compositional. On the other hand, $\approx_{PB_\varepsilon}$ allows systems that can have largely different possible behaviours to be related under the condition that such behaviours are observable with a negligible probability. Another approach to the approximation of bisimilarity has been recently proposed in [DHW03c,DHW03a], which extends the approach presented in this paper to probabilistic transition systems and is based on the definition of bisimulation via a linear operator and the use of an operator norm for measuring noninterference.

Acknowledgement

This work has been partially funded by Progetto MEFISTO (Metodi Formali per la Sicurezza e il Tempo) and the EU FET open projects SecSafe and Degas.

References

[Ald01] A. Aldini. Probabilistic Information Flow in a Process Algebra. In Proc. of *12th Int. Conf. on Concurrency Theory (CONCUR'01)*, Springer LNCS 2154:152–168, 2001.

[Ald02] A. Aldini. On the Extension of Non-interference with Probabilities. In Proc. of *WITS'02 – 2nd Workshop on Issues in the Theory of Security* (J. Guttman, Ed.), Portland, OR (USA), 2002.

[ABG03] A. Aldini, M. Bravetti, and R. Gorrieri. A Process-algebraic Approach for the Analysis of Probabilistic Noninterference. *Journal of Computer Security*, to appear.

[AG02] A. Aldini and R. Gorrieri. Security Analysis of a Probabilistic Non-repudiation Protocol. In Proc. of *2nd Joint Int. Workshop on Process Algebra and Performance Modelling, Probabilistic Methods in Verification (PAPM-ProbMiV'02)*, Springer LNCS 2399:17–36, 2002.

[BW90] J.C.M. Baeten and W.P. Weijland. *"Process Algebra"*, Cambridge University Press, 1990.

[BBS95] J.C.M. Baeten, J.A. Bergstra, and S.A. Smolka. Axiomatizing Probabilistic Processes: ACP with Generative Probabilities. *Information and Computation* 121:234–255, 1995.

[BH97] C. Baier, and H. Hermanns. Weak Bisimulation for Fully Probabilistic Processes. In Proc. of *9th Int. Conf. on Computer Aided Verification (CAV'97)*, Springer LNCS 1254:119–130, 1997.

[BPS01] J.A. Bergstra, A. Ponse, and S.A. Smolka (Eds.) *Handbook of Process Algebra*, Elsevier Science Publishers B.V., Amsterdam, 2001.

[Ber99] M. Bernardo. Theory and Application of Extended Markovian Process Algebra. *Ph.D. Thesis*, University of Bologna, Italy, 1999. `ftp://ftp.cs.unibo.it/pub/techreports/99-13.ps.gz`

[BDG98] M. Bernardo, L. Donatiello, and R. Gorrieri. A Formal Approach to the Integration of Performance Aspects in the Modeling and Analysis of Concurrent Systems. *Information and Computation* 144(2):83–154, 1998.

[Bil86] P. Billingsley. *Probability and Measure.* Wiley & Sons, New York, 2nd edition, 1986.

[Bra02] M. Bravetti. Specification and Analysis of Stochastic Real-Time Systems. *Ph.D. Thesis*, University of Bologna (Italy), 2002. `http://www.cs.unibo.it/~bravetti`

[BA03] M. Bravetti and A. Aldini. Discrete Time Generative-reactive Probabilistic Processes with Different Advancing Speeds. *Theoretical Computer Science* 290(1):355–406, 2003.

[BB00] M. Bravetti and M. Bernardo. Compositional Asymmetric Cooperations for Process Algebras with Probabilities, Priorities, and Time. In Proc. of *1st Workshop on Models for Time-Critical Systems (MTCS'00)*, State College (PA), ENTCS 39(3), 2000.

[vBW01] F. van Breugel, and J. Worrell. Towards Quantitative Verification of Probabilistic Systems (extended abstract). In Proc. of *28th Int. Colloquium on Automata, Languages and Programming (ICALP'01)*, Springer LNCS 2076:421–432, 2001.

[BHK01] E. Brinksma, H. Hermanns, and J.-P. Katoen (Eds.) *Lectures on Formal Methods and Performance Analysis*, Springer LNCS 2090, 2001.

[CKV00] C. Cachin, K. Kursawe, and V. Shoup. Random Oracles in Constantipole: Practical Asynchronous Byzantine Agreement Using Cryptography (extended abstract). In Proc. of *19th Symposium on Principles of Distributed Computing*, ACM Press, pp. 123–132, 2000.

[CT02] M.C. Calzarossa and S. Tucci (Eds.) *Performance Evaluation of Complex Systems: Techniques and Tools*, Performance 2002 Tutorial Lectures, Springer LNCS 2459, 2002.

[CHM02] D. Clark, S. Hunt, and P. Malacaria. Quantitative Analysis of Leakage of Confidential Data. In *QAPL 2001 - First International Workshop on Quantitative Aspects of Programming Laguages*, volume 59 of *Electronic Notes in Theoretical Computer Science*, Elsevier, 2002.

[dDP95] F.S. de Boer, A. Di Pierro, and C. Palamidessi. Nondeterminism and Infinite Computations in Constraint Programming. *Theoretical Computer Science*, 151(1):37–78, 1995.

[DGJP99] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for Labeled Markov Processes. In Proc. of *10th Int. Conf. on Concurrency Theory (CONCUR'99)*, Springer LNCS 1664:258–273, 1999.

[DGJP02] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. The Metric Analogue of Weak Bisimulation for Probabilistic Processes. In Proc. of *17th Symposium on Logic in Computer Science (LICS)*, IEEE CS Press, pp. 413–422, 2002.

[DHW01] A. Di Pierro, C. Hankin, and H. Wiklicky. Probabilistic Confinement in a Declarative Framework. In *Declarative Programming – Selected Papers from AGP 2000 – La Havana, Cuba*, volume 48 of *Electronic Notes in Theoretical Computer Science*, Elsevier, 2001.

[DHW02a] A. Di Pierro, C. Hankin, and H. Wiklicky. Analysing Approximate Confinement under Uniform Attacks. In Proc. of *SAS 2002 - 9th Int. Symposium on Static Analysis*, Springer LNCS 2477:310–326, 2002.

[DHW02b] A. Di Pierro, C. Hankin, and H. Wiklicky. Approximate Non-interference. In Proc. of *15th IEEE Computer Security Foundations Workshop*, pages 3–17, Cape Breton, Nova Scotia, Canada, 2002.

[DHW03a] A. Di Pierro, , C. Hankin, and H. Wiklicky. Measuring the Confinement of Concurrent Probabilistic Systems. In Proc. of *WITS'03 – Workshop on Issues in the Theory of Security* (R. Gorrieri, Ed.), Warsaw, Poland, 2003. `http://www.dsi.unive.it/IFIPWG1_7/wits2003.html`.

[DHW03b] A. Di Pierro, C. Hankin, and H. Wiklicky. Approximate Non-interference. *Journal of Computer Security*, 2003. To appear.

[DHW03c] A. Di Pierro, C. Hankin, and H. Wiklicky. Quantitative Relations and Approximate Process Equivalences. In Proc. of *14th Int. Conf. on Concurrency Theory (CONCUR'03)*, Lecture Notes in Computer Science, Springer Verlag, 2003. To appear.

[DW98a] A. Di Pierro and H. Wiklicky. An Operational Semantics for Probabilistic Concurrent Constraint Programming. In Proc. of *ICCL'98 – Int. Conf. on Computer Languages*, P. Iyer, Y. Choo, and D. Schmidt, Eds., pp. 174–183, IEEE Computer Society Press, 1998.

[DW98b] A. Di Pierro and H. Wiklicky. Probabilistic Concurrent Constraint Programming: Towards a Fully Abstract Model. In Proc. of *MFCS'98 – Mathematical Foundations of Computer Science*, L. Brim, J. Gruska, and J. Zlatuska, Eds., Springer LNCS 1450:446–455, 1998.

[DW00] A. Di Pierro and H. Wiklicky. Quantitative Observables and Averages in Probabilistic Concurrent Constraint Programming. In *New Trends in Constraints – Selected Papers of the 1999 ERCIM/Compulog Workshop on Constraints*, K.R. Apt, T. Kakas, E. Monfroy, and F. Rossi, Eds., Springer LNCS 1865, 2000.

[GM82] J.A. Goguen and J. Meseguer. Security Policy and Security Models. In Proc. of *Symposium on Security and Privacy (SSP'82)*, IEEE CS Press, pp. 11–20, 1982.

[FG95] R. Focardi and R. Gorrieri. A Classification of Security Properties. *Journal of Computer Security* 3(1):5–33, 1995.

[FG01] R. Focardi and R. Gorrieri (Eds.) *Foundations of Security Analysis and Design - Tutorial Lectures*, Springer LNCS 2171, 2001.

[GSS95] R. J. van Glabbeek, S. A. Smolka, and B. Steffen. Reactive, Generative and Stratified Models of Probabilistic Processes. *Information and Computation* 121:59–80, 1995.

[Gra90] J. W. Gray III. Probabilistic Interference. In Proc. of *Symposium on Security and Privacy (SSP'90)*, IEEE CS Press, pp. 170–179, 1990.

[Gra92] J. W. Gray III. Toward a Mathematical Foundation for Information Flow Security. *Journal of Computer Security* 1:255–294, 1992.

[GS92] J. W. Gray III and P. F. Syverson. A Logical Approach to Multilevel Security of Probabilistic Systems. In Proc. of *Symposium on Security and Privacy (SSP'92)*, IEEE CS Press, pp. 164–176, 1992.

[GS97] C.M. Grinstead and J.L. Snell. *Introduction to Probability.* American Mathematical Society, Providence, Rhode Island, second revised edition, 1997.

[HS95] P. Harrison and B. Strulo. Stochastic Process Algebra for Discrete Event Simulation. In *Quantitative Methods in Parallel Systems, ESPRIT Basic Research Series*, pp. 18–37, Springer, 1995.

[HMT71] L. Henkin, J.D. Monk, and A. Tarski. *Cylindric Algebras (Part I)*. North-Holland, 1971.

[HHHMR94] H. Hermanns, U. Herzog, J. Hillston, V. Mertsiotakis, and M. Rettelbach. Stochastic Process Algebras: Integrating Qualitative and Quantitative Modelling. In *7th Conf. on Formal Description Techniques (FORTE'94)*, pp. 449–451, 1994.

[Hil96] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

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

[Koc95] P.C. Kocher. Cryptanalysis of Diffie-Hellman, RSA, DSS, and Other Cryptosystems Using Timing Attacks. In *Advances in Cryptology, CRYPTO'95: 15th Annual Int. Cryptology Conf.*, D. Coppersmith, Ed., Springer LNCS 963:171–183, 1995.

[LMMS98] P.D. Lincoln, J.C. Mitchell, M. Mitchell, and A. Scedrov. A Probabilistic Poly-time Framework for Protocol Analysis. In *ACM Conf. on Computer and Communications Security*, pp. 112-121, ACM Press, 1998.

[MR99] O. Markowitch and Y. Roggeman. Probabilistic Non-Repudiation Without Trusted Third Party. In *2nd Conf. on Security in Communication Networks*, Amalfi, Italy, 1999.

[McL90] J. McLean. Security Models and Information Flow. In Proc. of *Symposium on Security and Privacy (SSP'90)*, IEEE CS Press, pp. 180–189, 1990.

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

[RMMG01] P.Y.A. Ryan, J. McLean, J. Millen, and V. Gligor. Non-interference: Who needs It? In Proc. of *14th Computer Security Foundations Workshop (CSFW'01)*, IEEE CS Press, pp. 237–238, 2001.

[SS99] A. Sabelfeld and D. Sands. A Per Model of Secure Information Flow in Sequential Programs. In Proc. of *European Symp. on Programming (ESOP'99)*, Springer LNCS 1576:40–58, 1999.

[SS00] A. Sabelfeld and D. Sands. Probabilistic Noninterference for Multithreaded Programs. In Proc. of *13th Computer Security Foundations Workshop (CSFW'00)*, IEEE CS Press, pp. 200–215, 2000.

[SM03] A. Sabelfeld and A.C. Myers. Language-Based Information Flow Security. *IEEE Journal on Selected Areas in Communications* 21(1):5–19, 2003.

[SR90] V.A. Saraswat and M. Rinard. Concurrent constraint programming. In *Symposium on Principles of Programming Languages (POPL)*, pp. 232–245, ACM Press, 1990.

[SRP91] V.A. Saraswat, M. Rinard, and P. Panangaden. Semantics foundations of concurrent constraint programming. In *Symposium on Principles of Programming Languages (POPL)*, pp. 333–353, ACM Press, 1991.

[Sha99] J. Shao. *Mathematical Statistics*. Springer Texts in Statistics. Springer Verlag, 1999.

[Smi01] G. Smith. A new Type System for Secure Information Flow. In Proc. of *14th Computer Security Foundations Workshop (CSFW'01)*, IEEE CS Press, pp. 115–125, 2001.

[Smi03] G. Smith. Probabilistic Noninterference through Weak Probabilistic Bisimulation. In Proc. of *16th Computer Security Foundations Workshop (CSFW'03)*, IEEE CS Press, pp. 3–13, 2003.

[SG95] P. Syverson and J. W. Gray III. The Epistemic Representation of Information Flow Security in Probabilistic Systems. In Proc. of *8th Computer Security Foundations Workshop (CSFW'95)*, IEEE CS Press, pp. 152–166, 1995.