On Approximate Non-Interference

Alessandra Di Pierro, Chris Hankin and Herbert Wiklicky

Dipartimento di Informatica, Universitá di Pisa, Italy
Department of Computing, Imperial College, London, UK

1 Introduction and Motivation

Confidentiality is that aspect of computer security concerned with how information is allowed to flow through a computer system. Information flows are treated in a binary fashion: they are either allowed to flow or not. Models for confidentiality typically characterise the absence of information flow between objects (across interfaces or along channels) by essentially reducing non-interference to confinement. The first attempt to formalise such an absence of information flow is the the notion of non-interference proposed by Goguen and Meseguer in their seminal paper [6]. Central to this notion is the idea of *indistinguishability* of behaviours: In order to establish that there is no information flow between two objects A and B, it is sufficient to establish that for any pair of behaviours of the system that differ only in A's behaviour, B's observations cannot distinguish these two behaviours [13]. For systems where non-determinism is present, the problem of characterising the equality of two behaviours is not a trivial one. In fact, there is no notion of system equivalence which everybody agrees upon; which notion is appropriate among , for example, trace or failure equivalence, (various forms of) bisimulation and (various forms of) testing equivalence, depends on the context and application in question.

In this work we approach the problem of confidentiality by looking at models which are able to give a *quantitative* estimate of the information flowing through a system. Such models abandon the purely qualitative binary view of the information flow by characterising how much information is actually "leaking" from the system rather than the complete absence of any flow. This allows us to define notions of non-interference which are *approximate* and yet able to capture the security properties of a system in a more "realistic" way: in real systems high-level input interferes with low-level output all the time [12].

The key idea of our approach is to replace *indistinguishability* by *similarity* in the basic formalisation of non-interference. As a result, two behaviours though distinguishable might still be considered non-interfering provided that their difference is below a threshold ϵ.

2 Probabilistic CCP

We illustrate our approach by referring to a probabilistic declarative language, namely Probabilistic Concurrent Constraint Programming (PCCP) language, which was introduced in [4, 5] as a probabilistic version of the Concurrent Constraint Programming (CCP) paradigm [16, 15]. This language can be seen as a kind of "process algebra" enhanced with a notion of "computational state".

The syntax and the basic execution model of PCCP are very similar to CCP. Both languages are based on the notion of a generic *constraint system* $\mathcal{C}$, defined as a cylindric algebraic complete partial order (see [16, 1] for more details), which encodes the information ordering. In PCCP probability is introduced via a probabilistic choice which replaces the nondeterministic choice of CCP, and a form of probabilistic parallelism, which replaces the pure nondeterminism in the interleaving semantics of CCP by introducing priorities.

The 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.

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

Table 1: The Syntax of PCCP Agents.

2.1 Operational Semantics

The operational semantics of PCCP is 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. The rules are closely related to the ones for nondeterministic CCP, and we refer to [1] 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 **ask**(c_i) succeed) or active (i.e. able to make a transition). The probability after normalisation is denoted by $\tilde{p}_j$.

Based on these transition rules we now introduce a notion of observables which captures the results of *finite* executions of a given agent together with their probabilities. Let us first introduce some basic definitions.

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

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

*where $A_0 = A$, $c_0 = true$, $A_n =$ **stop** and $n < \infty$.*

Note that this definition only accounts for *successful termination*. Our observables will not include infinite computation nor those situations in which the agent in the final configuration is not the **stop** agent and yet is unable to make a transition, i.e. the case of *suspended computations*. We denote by $\mathrm{Comp}(A)$ the set of all computational paths for A.

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

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

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

Given a PCCP program, the set $\mathcal{R}$ of the results of an agent A is the (multi-)set of all pairs $\langle c, p\rangle$, where c is the final store corresponding to the least upper bound of the partial constraints accumulated during a computational path, and p is the probability of reaching that result.

$$\mathcal{R}(A) = \{\langle c, p\rangle \ \mid \ \text{there exists } \pi \in \mathrm{Comp}(A) : c = res(\pi) \text{ and } p = prob(\pi)\}.$$

There might be different computational paths leading to the same result. Thus, we need to "compactify" the results so as to identify all those pairs with the same constraint as a first component. This operation is formally defined as follows.

Definition 3 *Let $S = \{\langle c_{ij}, p_{ij}\rangle\}_{i,j}$ be a (multi-)set of results, where c_{ij} denotes the jth occurrence of the constraint c_i, and let $P_{c_i} = \sum_j p_{ij}$ be the sum of all probabilities occurring in the set which are associated with c_i. The compactification of S is defined as follows:*

$$\mathcal{K}(S) \quad = \quad \{\langle c_i, P_{c_i}\rangle \ \mid \ \langle c_{ij}, p_{ij}\rangle \in S\}.$$

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

Table 2: The transition system for PCCP

We can now define the observables associated to an agent A as:

$$\mathcal{O}(A) = \mathcal{K}(\mathcal{R}(A)).$$

3 Similarity vs Equivalence

The notion of non-interference depends ultimately on the notion of process equivalence [13]. However, absolute non-interference can never be achieved in real systems. Moreover, by quoting [19], often computer systems "are not intended to be completely secure". For these reasons, equivalence-based formulations of non-interference may be too strong properties for a real system to satisfy. Thus, we introduce approximate non-interference, which weakens the classical notion by allowing some leak of information provided that it is below a fixed approximation threshold ϵ. To this purpose, we replace equivalence by the notion of *similarity* measured by an appropriate measure of distance.

Definition 4 *A* similarity *on a set $\mathcal{S}$ is a real-valued function $s : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}$.*

The intuitive idea behind this general definition is that of a (usually positive) number which indicates how much two objects are different. In the examples we discuss in the following, we will consider such a number as given by a *norm*.

3.1 Approximating Observational Equivalence

As an example illustrating our approach we consider here the notion of *identity confinement* introduced in [3, 2] in the context of PCCP, where indistinguishability is checked via the identity of the agents' behaviours in all contexts (aka *spies*). An approximate version of this notion can be defined by looking at *how much* the behaviours of two agents differ, instead of qualitatively asserting whether they are identical or not. In particular, as we refer to probabilistic observables, we look at the distance between the distributions representing the agents' observables instead of their identity.

The definition we give is parametric with respect to a set of admissible spies $\mathcal{S}$ and scheduling priorities p and $q = 1 - p$. We say that two agents A and B are approximately confined with respect to a set of spies $\mathcal{S}$ if there exists an $\varepsilon \geq 0$ such that for all $S \in \mathcal{S}$ the *distance* between the observables of agent $p : A \parallel q : S$ and the observables of agent $p : B \parallel q : S$ is smaller than ε. We consider as a measure for this distance 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. In the following we will omit the index and denote the supremum norm by $\|\cdot\|$.

Definition 5 *Given a set of admissible spies $\mathcal{S}$, $\varepsilon \geq 0$, and fixed scheduling priorities p and $q = 1 - p$, we call two agents A and B ε-confined iff:*

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

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

It easy to verify that if two agents A and B are ε-confined with $\varepsilon = 0$ then they are probabilistically confined [18, 14, 3].

Example 1 *Consider the following two PCCP agents:*

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

The non-deterministic versions of A and B exhibit the same observables when executed in any context. A and B are thus non-deterministically identity confined. *Treating the choice probabilistically still gives us the same observables for A and B if they are executed on their own. However, A and B are not probabilistically identity confined. A possible spy C which reveals the identity of A and B is given by the following agent:*

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

as the probabilistic observables

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

and

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

are different. The two agents A and B are in fact $\frac{1}{12}$-confined with respect to $\{C\}$.

One can ask for the hardest or most effective spy S which distinguishes A and B, i.e. an attacker for which ε is maximal. We will consider here a class of simple attackers which are expressed in PCCP as:

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

where $f_i \in \mathcal{C}$ are *fresh* constraints, that is constraints which never appear in the execution of the host agents. These spies 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.

4 Static Analysis

We introduce an analysis which will allow us to check whether PCCP agents are approximately confined by estimating an appropriate (upper bound for) ε.

4.1 A Collecting Semantics

We first re-formulate the operational semantics for PCCP so as to get a more suitable semantics for our analysis. The resulting collecting semantics is given in Table 3.

These rules define the transition from one level to next one in the computational tree of an agent. Each level is described by a multi-set of triples $\langle C_1, C_2, p\rangle$ where C_1 and C_2 are configurations such that there is a transition $C_1 \longrightarrow_p C_2$ in the transition system of Table 2. Such triples allow us to collect for each configuration occurring at level $i + 1$ the information about the generating

$$
\begin{array}{ll}
\textbf{R1} & \{\ldots, \langle\langle A', c'\rangle, \langle \textbf{tell}(c), d\rangle, p'\rangle, \ldots\} \\
& \longrightarrow \{\ldots, \langle\langle \textbf{tell}(c), d\rangle, \langle \textbf{stop}, c \sqcup d\rangle, 1\rangle, \ldots\} \\
\\
\textbf{R2} & \{\ldots, \langle\langle A', c'\rangle, \langle \square_{i=1}^{n} \textbf{ask}(c_i) \to p_i : A_i, d\rangle, p'\rangle, \ldots\} \\
& \longrightarrow \{\ldots, \langle\langle \square_{i=1}^{n} \textbf{ask}(c_i) \to p_i : A_i, d\rangle \langle A_j, d\rangle, \tilde{p}_j\rangle, \ldots\} \\
& \qquad\qquad j \in [1,n] \text{ and } d \vdash c_j \\
\\
\textbf{R3} & \{\ldots, \langle\langle A', c'\rangle, \langle \|_{i=1}^{n} \ p_i : A_i, d\rangle, p'\rangle, \ldots\} \\
& \longrightarrow \{\ldots, \langle\langle \|_{i=1}^{n} \ p_i : A_i, d\rangle, \langle A'_j, d'\rangle, \tilde{p}_j\rangle, \ldots\} \\
& \qquad\qquad j \in [1,n] \text{ and } \langle A_j, c\rangle \longrightarrow_p \langle A'_j, c'\rangle \\
\\
\textbf{R4} & \{\ldots, \langle\langle A', c'\rangle, \langle \exists_x A, d\rangle, p'\rangle, \ldots\} \\
& \longrightarrow \{\ldots, \langle\langle \exists_x A, d\rangle, \langle \exists_x A', d \sqcup \exists_x d'\rangle, p\rangle, \ldots\} \\
& \qquad\qquad \langle A, \exists_x d\rangle \longrightarrow_p \langle A', d'\rangle \\
\\
\textbf{R5} & \{\ldots, \langle\langle \exists_x A, d\rangle, \langle p(y), d\rangle, p'\rangle, \ldots\} \\
& \longrightarrow \{\ldots, \langle\langle p(y), d\rangle, \langle A, d\rangle, q\rangle, \ldots\} \\
& \qquad\qquad p(x) : -A \in P
\end{array}
$$

Table 3: A Collecting Semantics for PCCP

configuration at level i. Given a program P, these multi-sets of triples can be seen as non-normalised distributions over configurations formally defined inductively by:

$$
\begin{array}{rcl}
\Omega_0(P) & = & \{\langle\langle P, true\rangle, \langle P, true\rangle, 1\rangle\} \\
\Omega_{i+1}(P) & = & \{\langle C_{i_1}, C_{i_2}, p_i\rangle \mid \ C_{i_1}, C_{i_2} \in \text{Conf and } C_{i_1} \longrightarrow_{p_i} C_{i_2} \text{ and} \\
& & \qquad \text{there exists } \langle C, C', p\rangle \in \Omega_i \text{ s.t. } C' = C_{i_1}\}.
\end{array}
$$

Example 2 *Consider the agents A and B from Example 1. The collecting semantics of the agent B is given by a one step sequence, while the agent A generates a two steps sequence. More precisely:*

$$
\begin{array}{rcl}
\Omega_0(B) & = & \{\langle\langle B, true\rangle, \langle B, true\rangle, 1\rangle\} \\
\Omega_1(B) & = & \{\langle\langle B, true\rangle, \langle \textbf{stop}, c \sqcup d\rangle, 1\rangle\}
\end{array}
$$

indicating that for B there is only one possible transition (with probability $p = 1$). For the agent A we get:

$$
\begin{array}{rcl}
\Omega_0(A) & = & \{\langle\langle A, true\rangle, \langle A, true\rangle, 1\rangle\} \\
\Omega_1(A) & = & \left\{\left\langle \langle A, true\rangle, \langle \textbf{tell}(d), c\rangle, \frac{1}{2}\right\rangle, \left\langle \langle A, true\rangle, \langle \textbf{tell}(c), d\rangle, \frac{1}{2}\right\rangle\right\} \\
\Omega_2(A) & = & \left\{\left\langle \langle \textbf{tell}(d), c\rangle, \langle \textbf{stop}, c \sqcup d\rangle, \frac{1}{2}\right\rangle, \left\langle \langle \textbf{tell}(c), d\rangle, \langle \textbf{stop}, c \sqcup d\rangle, \frac{1}{2}\right\rangle\right\}
\end{array}
$$

After that we have:

$$\Omega_i(B) = \Omega_j(A) = \{\langle\langle \textbf{stop}, c \sqcup d\rangle, \langle \textbf{stop}, c \sqcup d\rangle, 1\rangle\}$$

for $i > 1$ and $j > 2$.

4.2 Security Analysis

Given two agents A and B, scheduling priorities p and $q = 1 - p$ and a spy $S \in \mathcal{S}_n$, our aim is to calculate ε_S such that

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

To this purpose we will use the collecting semantics of A and B to construct the observables $\mathcal{O}(p : A \parallel q : S)$ and $\mathcal{O}(p : B \parallel q : S)$ respectively, for spies S in $\mathcal{S}_1$ and $\mathcal{S}_2$. Concretely we will identify a few critical parameters which we obtain from analysing the execution of A and B in isolation which allow us to construct the observables of both agents when executed in the context S (without considering neither the execution of agent $p : A \parallel q : S$ nor the execution of agent $p : B \parallel q : S$). The number $\sup_{S \in \mathcal{S}_n} \varepsilon_S$ will then give us a quantitative estimate of the confinement of A and B.

Let us consider first the simple spies in $\mathcal{S}_1$. For these spies, we can ignore the intermediate states and concentrate only on the final one, i.e. the observables.

Proposition 1 *Let A be a PCCP agent and let $S \in \mathcal{S}_1$ be of the form*

$$S \equiv \mathbf{ask}(c) \rightarrow 1 : \mathbf{tell}(f).$$

Suppose the observables of A are $\mathcal{O}(A) = \{\langle d_i, p_i \rangle\}_i$. Then the observables of $p : A \parallel q : S$ are given by:

$$\begin{array}{rcl} \mathcal{O}(p : A \parallel q : S) & = & \{\langle d_i, p_i \rangle \mid \textit{if } d_i \not\vdash c\} \\ & \cup & \{\langle d_i \sqcup f, p_i \rangle \mid \textit{if } d_i \vdash c\}. \end{array}$$

For the more "sophisticated" spies in $S \in \mathcal{S}_2$ the construction of the observables of A (B) executed in the context S requires the consideration of the intermediate steps in the execution of A (B).

Proposition 2 *Let A be a PCCP agent, and let $S \in \mathcal{S}_2$ be a spy of the form*

$$\begin{array}{rcl} S & \equiv & \mathbf{ask}(c_1) \rightarrow q_1 : \mathbf{tell}(f_1) \\ & & \square\ \mathbf{ask}(c_2) \rightarrow q_2 : \mathbf{tell}(f_2). \end{array}$$

Suppose the observables of A are $\mathcal{O}(A) = \{\langle d_i, p_i \rangle\}_i$ and there is only one computational path π_i leading to d_i for all i. Then the observables of $p : A \parallel q : S$ are given by:

$$\begin{array}{rcl} \mathcal{O}(p : A \parallel q : S) & = & \{\langle d_i, p_i \rangle \mid \textit{if } d_i \not\vdash c_1 \wedge d_i \not\vdash c_2\} \\ & \cup & \{\langle d_i \sqcup f_1, p_i \rangle \mid \textit{if } d_i \vdash c_1 \wedge d_i \not\vdash c_2\} \\ & \cup & \{\langle d_i \sqcup f_2, p_i \rangle \mid \textit{if } d_i \not\vdash c_1 \wedge d_i \vdash c_2\} \\ & \cup & \{\langle d_i \sqcup f_1, r_i^1 \rangle \mid \textit{if } d_i \vdash c_1 \wedge d_i \vdash c_2\} \\ & \cup & \{\langle d_i \sqcup f_2, r_i^2 \rangle \mid \textit{if } d_i \vdash c_1 \wedge d_i \vdash c_2\}, \end{array}$$

where

$$r_i^1 = p_i \cdot (((\sum_{l=0}^{n_i - 1} p^{m_i + l}) q + p^{m_i + n_i}) q_1 + (1 - p^{m_i}))$$

$$r_i^2 = p_i \cdot ((\sum_{l=0}^{n_i - 1} p^{m_i + l}) q + p^{m_i + n_i}) q_2,$$

with m_i the number of steps in π_i needed to go from the store where c_1 is first entailed to the store where also c_2 is entailed, and n_i the number of the remaining steps until termination.

This result can be generalised to the case where there are several paths leading to the same final result c_i by defining the probabilities r_i^1 and r_i^2 as:

$$r_i^1 = \sum_k r_{ik}^1 \text{ and } r_i^2 = \sum_k r_{ik}^2,$$

where the sum is over all computational paths π_{ik} of A leading to c_i, and r_{ik}^1 and r_{ik}^2 are the probabilities of $d_i \sqcup f_1$ and $d_i \sqcup f_2$ via path π_{ik}.

It is immediate to see that r_i^1 and r_i^2 "split" the probability p_i that c_i occurs in the observables of A in the probabilities that $c_i \sqcup f_1$ or $c_i \sqcup f_2$ are in the observables of $p : A \parallel q : S$, i.e. $p_i = r_i^1 + r_i^2$. In principle this result can be generalised also to spies in $\mathcal{S}_n$ with $n > 2$.

4.3 Effectiveness of S

In Definition 5, the number ε measures *how confined* A and B are, or equivalently *how effective* the class $\mathcal{S}$ of spies is. Thus, an ε such that A and B are ε-confined with respect to $\{S\}$ gives us an estimate of the *effectiveness* of a single spy S, i.e. its ability to distinguish between the two agents A and B. The analysis of A and B provides a means to calculate how large ε is.

Proposition 3 *Let S be a spy in $\mathcal{S}_2$ of the form*

$$\begin{aligned} S \quad \equiv \quad & \mathbf{ask}(c_1) \to q_1 : \mathbf{tell}(f_1) \\ & [] \ \mathbf{ask}(c_2) \to q_2 : \mathbf{tell}(f_2). \end{aligned}$$

and let A and B be two agents with identical probabilistic observables $\mathcal{O}(A) = \mathcal{O}(B) = \{\langle c_j, p_j \rangle\}_j$. The effectiveness of S is

$$\begin{aligned} \varepsilon &= \max_{c_j} \{|r_j^1(A) - r_j^1(B)|\} \\ &= \max_{c_j} \{|r_j^2(A) - r_j^2(B)|\}. \end{aligned}$$

Note that the effectiveness ε of a spy S depends on the scheduling priorities p and $q = 1 - p$.

4.4 The Most Effective Spy

Some easy calculations show that the difference $|r_j^1(A) - r_j^1(B)|$ (or equivalently $|r_j^2(A) - r_j^2(B)|$) is maximal when $|q_1 - q_2|$ is maximal, namely when $|q_1 - q_2|$ tends to 1. Therefore, the "best" spy is obtained by letting q_1 tend to 0 and q_2 tend to 1, or vice-versa.

In the first case we obtain the following (limit) formulas for r_i^1 and r_i^2:

$$r_i^1 = p_i \cdot (1 - p^{m_i}) \text{ and } r_i^2 = p_i \cdot p^{m_i}.$$

For the case where q_1 tends to 1 and q_2 tends to 0 we get:

$$r_i^1 = 1 \text{ and } r_i^2 = 0.$$

As a result a spy S with a limit choice distribution effectively counts only the number of steps m_i between c_1 and c_2 along each path π_i. The r_is are then independent of the n_i steps till the end of the computation.

Example 3 *Using the limit analysis we now show that C is not the most effective spy for the agents A and B in Example 1. Clearly, the most effective spy must have the same guards c and d as C, since no other intermediate constraints exist for A and B. In order to determine the best spy, we therefore only need to consider a q_1 very close to 0 and a q_2 very close to 1. Assuming the uniform scheduling $p = q = \frac{1}{2}$, we now calculate the corresponding ε.*

To this purpose, we consider the expressions for r_i^1 and r_i^2:

A: *For agent A and its two computational paths the limit probabilities for $c \sqcup d \sqcup e$ and $c \sqcup d \sqcup f$ are:*

$$r^c_{c \sqcup d}(A) = \frac{1}{4} \text{ and } r^d_{c \sqcup d}(A) = \frac{3}{4}$$

B: *We have only one computational path for which we get:*

$$r^c_{c \sqcup d}(B) = 0 \text{ and } r^d_{c \sqcup d}(B) = 1.$$

The difference $|r^c_{c \sqcup d}(A) - r^c_{c \sqcup d}(B)|$ or equivalently $|r^d_{c \sqcup d}(A) - r^d_{c \sqcup d}(B)|$ gives us the largest ε for A and B and any spy in $\mathcal{S}_2$:

$$\varepsilon = |r^c_{c \sqcup d}(A) - r^c_{c \sqcup d}(B)| = \frac{1}{4}.$$

We can therefore conclude that A and B are $\frac{1}{4}$-confined with respect to all agents in $\mathcal{S}_2$.

$$
\begin{array}{lcl}
[\![\mathbf{stop}]\!] & = & \emptyset \\
[\![\mathbf{tell}(c)]\!] & = & \{\langle true, c, 1, 1\rangle\} \\
[\![\prod_{i=1}^{n} \mathbf{ask}(c_i) \to p_i : A_i]\!] & = & \bigoplus_i (p_i, c_i, [\![A_i]\!]) \\
[\![A; B]\!] & = & [\![A]\!] \odot [\![B]\!] \\
[\![\exists_x A]\!] & = & [\![A]\!] \\
[\![p(x)]\!] & = & \nabla_n^\delta([\![A]\!]) \quad \text{for} \quad p(x) : -A \in P
\end{array}
$$

Table 4: The Analysis for PCCP Agents

5 Static Analysis

5.1 Abstract Semantics

The use of an exact (collecting) semantics makes the analysis presented in the previous sections precise: no approximation is introduced in the calculation of ϵ.

We introduce here a semantics which is more abstract than the collecting semantics but still allows us for a useful though approximated analysis. We associate to each agent a set of tuples. Each tuple $\langle c, d, t, p\rangle$ consists of two constraints c and d, a time stamp t and a probability p. It represents a transition from a store c' to store d, which takes place (earliest) at step t with probability p, provided that the current store c' entails c. The time stamp t is interpreted as a step counter and will be used to extract information about the number m of the previous section.

The choice agent is modelled by the union of all tuples in the semantics of sub-agents A_i where the first constraint entails the guard and the time is increased by 1.

$$\bigoplus_i (p_i, c_i, [\![A_i]\!]) = \bigcup_i \{\langle c_i \sqcup d, c, t+1, p_i \cdot p\rangle \mid \langle d, c, t, p\rangle \in [\![A_i]\!]\}\,.$$

The parallel agent $\|_{i=1}^n A_i$ is interpreted as a choice among all possible sequential compositions (permutations) of agents A_i, representing all possible interleavings. The sequential composition of A and B is described by the set of all tuples denoting the agent last executed (B) where the time is updated so as to consider the previous execution of A. In general, given two sets of tuples X and Y, the operation $\odot$ is defined as follows:

$$[\![X]\!] \odot [\![Y]\!] = X \cup \{\langle d, c, t + \max(X), p\rangle \mid \langle d, c, t, p\rangle \in Y\}\,,$$

where $\max(X)$ stands for the maximum value in the set $\{t \mid \langle d, c, t, p\rangle \in X\}$.

We use the operator ∇_n^δ to approximate the semantics of a procedure call by "unwinding" it until the probability of a further continuation gets smaller than δ or until we reach a maximal recursion depth n. The unwinding is defined in Table 5: We start by a trivial approximation ∇_0^δ and continue by replacing the procedure call p in the term X by the previous approximation — denoted by $[\![X]\!][p \mapsto \nabla_i^\delta([\![X]\!])]$ — until either the difference between the current and previous approximation becomes zero (i.e. the recursion terminates and our approximation is exact) or this difference becomes small enough (less or equal to δ) or we reach the maximal recursion depth n. In this case we take an infinite, non-terminating approximation ∇_∞^δ in place of further unwindings. The difference between two approximation is the difference of the two sets of tuples seen as vector distributions $\{\langle\langle c, d, t\rangle, p\rangle\}$. The final approximation of a procedure call is then given as:

$$\nabla_n^\delta([\![X]\!]) = \lim_{i \to \infty} \nabla_i^\delta([\![X]\!])$$

which is effectively always reached after a finite number of unwindings. This can lead to a substantial over-approximation of t for recursive agents.

5.2 Abstract Security Analysis

By a further abstraction of the denotation of a program P we eliminate from the tuples of $[\![P]\!]$ the dependency store (first component):

$$
\begin{array}{lcl}
\nabla_0^\delta([\![X]\!]) & = & \{\langle true, true, 1, 1\rangle\} \\
\nabla_{i+1}^\delta([\![X]\!]) & = & \begin{cases} [\![X]\!][p \mapsto \nabla_i^\delta([\![X]\!])] & \text{if } \|[\![X]\!][p \mapsto \nabla_i^\delta([\![X]\!])] - \nabla_i^\delta([\![X]\!])\| > \delta \text{ and } i \leq n, \\ \nabla_i^\delta([\![X]\!]) & \text{if } \|[\![X]\!][p \mapsto \nabla_i^\delta([\![X]\!])] - \nabla_i^\delta([\![X]\!])\| = 0 \text{ and } i \leq n, \\ \nabla_\infty^\delta([\![X]\!])) & \text{otherwise} \end{cases} \\
\nabla_\infty^\delta([\![X]\!]) & = & \{\langle d, c, \infty, p\rangle \mid \langle d, c, t, p\rangle \in X\}
\end{array}
$$

Table 5: Unwinding a Procedure Call

$$
\begin{array}{rcl}
\dagger\, X & = & \{\langle c, t, p\rangle \mid init \vdash d \text{ and } \langle d, c, t, p\rangle \in X\} \\
& \cup & \{\langle c, t, p\rangle \mid \langle d, c, t, p\rangle \in X \text{ and } \langle c', t', p'\rangle \in \dagger\, X \text{ and } d \vdash c' \text{ and } t' < t\}\,.
\end{array}
$$

The $\dagger$ operation eliminates all those transitions or tuples where the dependency is not fulfilled in time, but does not normalise the associated probabilities. Therefore, the probabilities in $\dagger X$ may be smaller than in the concrete semantics.

In order to define a *path* in the denotation of a program we introduce the following relation:

$$\langle c, t, p\rangle\ \tilde{\rho}\ \langle c', t', p'\rangle \text{ iff } t' = t + 1 \text{ or } t' = \infty.$$

The paths leading from store c to store c' are defined as follows:

$$
\begin{array}{rcl}
paths(c, c') & = & \{\langle c_1, t_1, p_1\rangle \ldots \langle c_n, t_n, p_n\rangle \mid c_1 \vdash c \text{ and } c_n \vdash c' \\
& & \quad \text{and for all } i \in [1, \ldots, n-1] : \langle c_i, t_i, p_i\rangle\ \tilde{\rho}\ \langle c_{i+1}, t_{i+1}, p_{i+1}\rangle\}.
\end{array}
$$

Given a path $\sigma \in paths(c, c')$, the difference $t_{n_\sigma} - t_1$ defines the (abstract) number of steps $\overline{m}_\sigma$ between the store entailing c and the store c' in path σ, while p_{n_σ} defines the (abstract) probability $\overline{p}_\sigma$ associated to σ. Therefore, for each pair of constraints c_1 and c_2, the abstract analysis of program P gives us the set:

$$\overline{\mathcal{A}}_P(c_1, c_2) = \bigcup_{\sigma \in paths(c_1, c_2)} \{\langle \overline{p}_\sigma, \overline{m}_\sigma\rangle\}\,.$$

5.3 Correctness of the Analysis

For each pair of constraints c_1 and c_2, the concrete semantics allows us to calculate the set $\mathcal{A}_P(c_1, c_2) = \{\langle p_i, m_i\rangle\}_i$ of all pairs $\langle p_i, m_i\rangle$, such that for each computational path π_i fo P: p_i is the probability associated to π_i and m_i is the number of steps needed to go from the store which first entails c_1 to the store which entails also c_2. We show how the abstract analysis approximates p_i and m_i by $\overline{p_i}$ and $\overline{m_i}$. As already discussed in the previous section, the $\dagger$ operation implies that

$$\overline{p}_i \leq p_i.$$

On the other hand, for each i, the operation max used in the sequential operator and the definition of ∇_n^δ defining the procedure call in the abstract semantics imply that

$$\overline{m}_i \geq m_i.$$

We can then conclude that

$$r_{ik}^2 = p_{ik} p^{m_{ik}} \ \leq\ \overline{p}_{ik} p^{\overline{m}_{ik}} = \overline{r}_{ik}^2$$

and

$$r_i^2 = \sum_k r_{ik}^2 \ \leq\ \sum_k \overline{r}_{ik}^2 = \overline{r}_i^2.$$

We know that $r^1_{ik} = p_{ik} - r^2_{ik}$ and similarly $\overline{r}^1_{ik} = \overline{p}_{ik} - \overline{r}^2_{ik}$. Therefore we get:

$$r^1_{ik} = p_i - r^2_{ik} \geq \overline{p}_{ik} - r^2_{ik} \geq \overline{p}_{ik} - \overline{r}^2_{ik} = \overline{r}^1_{ik}$$

and finally:

$$r^1_i = \sum_k r^1_{ik} \;\geq\; \sum_k \overline{r}^1_{ik} = \overline{r}^1_i.$$

In oder to get an approximation for $\varepsilon = \max_i |r^1_i(A) - r^1_i(B)| = \max_i |r^2_i(A) - r^2_i(B)|$ we define

$$\overline{\varepsilon} \;\; = \;\; \max_i(\overline{r}^2_i(A) + \overline{r}^2_i(B)).$$

Proposition 4 *Let A and B be two PCCP agents, and let $S \in \mathcal{S}_2$ be a spy of the form:*

$$\begin{array}{rcl} S & \equiv & \mathbf{ask}(c_1) \to p_1 : \mathbf{tell}(f_1) \\ & & \square\ \mathbf{ask}(c_2) \to p_2 : \mathbf{tell}(f_2). \end{array}$$

Let ε and $\overline{\varepsilon}$ be defined as before. Then

$$\varepsilon \leq \overline{\varepsilon}.$$

One can use the abstract analysis to define better approximations for ε by combining the information provided by $\overline{r}^1_i$ as well as $\overline{r}^2_i$.

6 Conclusions and Related Work

The problem of characterising the security of a system can be addressed from an *approximate* viewpoint by gaining in effectiveness. Approximation is introduced by relaxing the notion of non-interference so as to allow some quantifiable amount of interference. This relaxation is introduced in the very definition of the property by replacing equivalence by a notion of ϵ-similarity at the base of the characterisation of the information flowing through a system: while equivalence implies the absence of any information flow, ϵ-similarity allows a limited amount of such an information flow. This amount corresponds to the number of tests that are required to differentiate the processes.

The use of experimental testing for *comparing* concurrent processes is discussed by Larsen and Skou in [9], where it is presented as a more "realistic", "practical", "effective", way to decide the equivalence of processes: Testing is seen as an *alternative* to a formal analysis. In our setting testing represents in the formal analysis the way a "spy", "intruder", or "attacker" can obtain information about a certain system. This leads to a statistical interpretation of the similarity notions we can define to approximate equivalence notions. In fact, we can use statistical concepts and results to quantify the number of "experiments" ("attacks" or "tests") we (or a spy) need to perform in order to distinguish between two processes. This depends on two parameters:

- the error margin within which the identification is correct (i.e. the significance level we aim for), and
- the desired confidence level, i.e. the probability with which the identification is correct.

As Larsen and Skou point out in [9] the "significance level" δ is directly related to the the number of tests n required to obtain a desired level of certainty about the behavioural equivalence of two processes. However, their aim is to minimise the error margin, i.e. to increase the level of significance, since they are interested in investigating *testable* properties, i.e. properties which can be tested with respect to any fixed level of significance. In contrast, our aim is to *quantify* the error margin as we are interested in estimating the level of significance achievable. This gives us a measure for dissimilarity, and thus for the level of non-interference. Our usage of the approximating ε as a measure of non-interference is based on a simple statistical argument: The more similar two processes are the more difficult it is for external observers, i.e. spies or attackers, to reveal information about them. For example: Suppose a process behaviour depends on the value of a "high variable", i.e. for two different values of this variable the process exhibits behaviours which are not equivalent according to some notion of process equivalence. The common

verdict about such a process is that the variable is not "confined" ("protected" or "secure"). However, this approach ignores the "size" of the behavioural difference: There are cases were strictly speaking non-interference can not be guaranteed, and yet a statistical attacker would need such an enormous number of experiments to reach any conclusions about the high variable that the system is effectively secure. Although statistical attacks will succeed on the long run, in practical terms they will nevertheless fail.

A first relaxation of the original notion of non-interference was introduced at the beginning of the nineties in order to overcome the problem with Trojan horses exploiting probabilistic interference to reliably leak information to unauthorised users [7, 8]. Work has been done since then on the definition of notions, models and proof systems for probabilistic non-interference [18, 14, 3]. All of these cited papers still consider equivalence-based non-interference notions.

In [10, 11] the authors introduce an asymptotic notion of observational equivalence for a probabilistic polynomial-time process language which takes into account the running time a spy needs to distinguish two processes via test runs. This is closely related to our notion of ϵ-similarity. The critical difference is that Lincoln *et al*'s notion is a contextual relation (quantified over all contexts) and defined in terms of the absolute value of the difference between probabilities rather than the norm-based approach that we advocate here. Furthermore, Lincoln *et al* have a rather different motivation form ours: they are concerned with the analysis of cryptographic protocols and they give a complexity theoretic analysis of their equivalence which contrasts with our semantic treatment.

The notion of ε-confinement introduced in this paper requires that the behaviour of an agent is described by some object — i.e. observables — and that we have a way to measure the similarity of such objects. This measure could be formalised mathematically by a norm, a metric, or some other appropriate notion of distance, depending on the domain of objects used to describe the behaviour of programs. In this paper we concentrated on input/output observables of PCCP programs which can be described by vector distributions and we used a vector norm to measure their similarity. In [17] van Breugel and Worrell consider instead derivation trees together with a pseudo-metric to achieve a similar weakening of the concept of behavioural equivalence of concurrent processes.

A problem arises when a behavioural (or observational) "equivalence" is defined directly without reference to some underlying concept of observables, as for example in various notions of bisimilarity[1]. In this case we have neither a simple domain of objects describing the behaviour of a term nor a canonical similarity measure and it is therefore difficult to apply our approach. Attempts, for example, to introduce a notion of "ε-bisimilarity" relation can easily run into problems with transitivity [10]. We plan to investigate such problems in our future work.

References

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

[2] Alessandra Di Pierro, , Chris Hankin, and Herbert Wiklicky. Probabilistic security analysis in a declarative framework. In P. Degano, editor, *Proceedings of WITS'00 – Workshop on Issues in the Theory of Security, 7-8 July,Geneva, Switzerland*, pages 83–86, July 2000.

[3] Alessandra Di Pierro, Chris Hankin, and Herbert 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*, pages 1–23. Elsevier, 2001.

[4] Alessandra Di Pierro and Herbert Wiklicky. An operational semantics for Probabilistic Concurrent Constraint Programming. In P. Iyer, Y. Choo, and D. Schmidt, editors, *ICCL'98 – International Conference on Computer Languages*, pages 174–183. IEEE Computer Society Press, 1998.

[5] Alessandra Di Pierro and Herbert Wiklicky. Probabilistic Concurrent Constraint Programming: Towards a fully abstract model. In L. Brim, J. Gruska, and J. Zlatuska, editors, *MFCS'98 – Mathematical Foundations of Computer Science*, volume 1450 of *Lecture Notes in Computer Science*, pages 446–455, Berlin – New York, August 1998. Springer Verlag.

[6] Joseph Goguen and José Meseguer. Security Policies and Security Models. In *IEEE Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, 1982.

[1] Note that these notions refer to a concept of "simulation" rather than of "similarity".

[7] J. W. Gray, III. Probabilistic interference. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 170–179. IEEE Computer Society Press, 1990.

[8] J. W. Gray, III. Towards a mathematical foundation for information flow security. In *Proceedings of the 1991 Symposium on Research in Security and Privacy*, pages 21–34, Oakland, California, May 1991. IEEE Computer Society.

[9] Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, September 1991.

[10] Patrick D. Lincoln, John C. Mitchell, Mark Mitchell, and Andre Scedrov. A probabilistic poly-time framework for protocol analysis. In *ACM Conference on Computer and Communication Security (CCS-5)*, 1998.

[11] J. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. A probabilistic polynomial-time calculus for analysis of cryptographic protocols (preliminary report). In *17-th Annual Conference on the Mathematical Foundations of Programming Semantics, Arhus, Denmark, May, 2001*, volume 45 of *Electronic Notes in Theoretical Computer Science*, pages 1–25. Elsevier, 2001.

[12] P. Ryan, J. McLean, J. Millen, and V. Gilgor. Non-interference, who needs it? In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pages 237–238, Cape Breton, Nova Scotia, Canada, June 2001. IEEE.

[13] Paul Y. A. Ryan and Steve A. Schneider. Process algebra and non-interference. *Journal of Computer Security*, 9(1/2):75–103, 2001. Special Issue on CSFW-12.

[14] Andrei Sabelfeld and David Sands. Probabilistic noninterference for multi-threaded programs. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop*, pages 200–214, 2000.

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

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

[17] Franck van Breugel and James Worrell. Towards quantitative verification of probabilistic transition systems. In *ICALP: 28th International Colloquium on Automata, Languages and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 421–432. Springer Verlag, 2001.

[18] Dennis Volpano and Geoffrey Smith. Probabilistic noninterference in a concurrent language. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop (CSFW '98)*, pages 34–43, Washington - Brussels - Tokyo, June 1998. IEEE.

[19] D. G. Weber. Quantitative hook-up security for covert channel analysis. In *Proceedings of the 1988 Workshop on the Foundations of Computer Security*, Franconia, New Hampshire, 1988.