Approximate Non-Interference

Alessandra Di Pierro
Dipartimento di Informatica
Università di Pisa, Italy
dipierro@di.unipi.it

Chris Hankin, Herbert Wiklicky
Department of Computing
Imperial College London, UK
clh@doc.ic.ac.uk
herbert@doc.ic.ac.uk

1. Introduction

Non-interference was introduced by Goguen and Meseguer in their seminal paper [8] in order to provide an appropriate formalism for the specification of security policies. In its original formulation it 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".

This notion has been widely used to model various security properties. One such property is *confidentiality* which is concerned with how information is allowed to flow through a computer system. In recent years, there has been a proliferation of definitions of confidentiality, all based on the central 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 [16]. 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.

Another common aspects of these various formulations of confidentiality is that they all treat information flows 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. Depending on the nature of the information flow one can characterise different confinement properties, namely *deterministic*, *nondeterministic*, and *probabilistic* confinement [25].

It is important to notice that nondeterministic confinement is weaker than probabilistic confinement, as it is not able to capture those situations in which the probabilistic nature of an implementation may allow for the detection of the confidential information, e.g. by running the program a sufficient number of times [9]. In the context of imperative programming languages, confinement properties with respect to the value of high and low level variables, have been recently discussed in [22, 26, 23] where a type-system based security analysis is developed. Another recent contribution to this problem is the work in [17, 18], where the use of probabilistic power-domains is proposed, which allows for a compositional specification of the non-interference property underlying a type-based security analysis.

Although non-interference is the simplest characterisation of confidentiality, it has several problems [15]. One is that absolute non-interference can hardly ever be achieved in real systems. On the other hand, often computer systems "are not intended to be completely secure" [27]. As a consequence, notions of non-interference such as confinement turn out to be too strong as characterisation of the non-interference criterion effectively used in practice (especially in their non-deterministic version).

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 a notion of non-interference which is *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 [15].

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 dis-

A	$::=$	**stop**	successful termination
	$\mid$	**tell**(c)	adding a constraint
	$\mid$	$\square_{i=1}^{n}$ **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.

tinguishable might still be considered as effectively non-interfering provided that their difference is below a threshold ϵ. A similarity relation can be defined by means of an appropriate notion of distance and provides information (the ϵ) on "how much" two behaviours differ from each other. This information is not relevant in equivalence relations such as observational equivalence or bisimilarity, where the comparison between two behaviours is aimed to establish whether they can be identified or not.

We will formalise our approach in a particular process algebraic framework including probabilistic constructs which allow us to deal with probabilistic information flows. Such a framework is Probabilistic Concurrent Constraint Programming (PCCP) and will be presented in Section 2. The notion of identity confinement expressing confidentiality in PCCP is then defined in Section 3. In Section 4 we introduce an approximate version of the identity confinement and give a statistical interpretation of the quantity ϵ measuring the approximation. Finally, we will propose two analyses of the approximate confinement property based respectively on a concrete (Section 5) and an abstract (Section 6) semantics, and show the correctness of the abstract analysis with respect to the concrete semantics. We conclude with a summary and an outline of some further research directions in Section 7.

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 [6, 7] as a probabilistic version of the Concurrent Constraint Programming (CCP) paradigm [20, 19]. This language can be seen as a kind of "process algebra" enhanced with a notion of "computational state".

2.1. Syntax of Agents

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 [20, 4] 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.

2.2. Operational Semantics

The operational semantics of PCCP is defined in terms of a probabilistic transition system, $(\text{Conf}, \longrightarrow_p)$, 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 [4] 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 $\tilde{p}_j$.

2.3. Observables

The notion of observables we consider in this paper refers to 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:

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

R1	$\langle \textbf{tell}(c), d\rangle \longrightarrow_1 \langle \textbf{stop}, c \sqcup d\rangle$	
R2	$\langle \Box_{i=1}^{n} \textbf{ask}(c_i) \rightarrow p_i : A_i, d\rangle \longrightarrow_{\tilde{p}_j} \langle A_j, d\rangle$	$j \in [1,n]$ and $d \vdash c_j$
R3	$\dfrac{\langle A_j, c\rangle \longrightarrow_p \langle A'_j, c'\rangle}{\langle \parallel_{i=1}^{n} \; p_i : A_i, c\rangle \longrightarrow_{p\cdot\tilde{p}_j} \langle \parallel_{j\neq i=1}^{n} p_i : A_i \parallel p_j : A'_j, c'\rangle}$	$j \in [1,n]$
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

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 finite computation of A starting in store d and p is the probability of computing that result. We define formally such a set of results as follows.

Definition 1 *Let A be a PCCP agent. A computational path π for A in store d is defined by $\pi \equiv \langle A_0, c_0\rangle \longrightarrow_{p_1} \langle A_1, c_1\rangle \longrightarrow_{p_2} \dots \longrightarrow_{p_n} \langle A_n, c_n\rangle$,where $A_0 = A$, $c_0 = d$, $A_n =$* **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 $\mathrm{Comp}(A,d)$ 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 $\pi \equiv \langle A,d\rangle = \langle A_0, c_0\rangle \longrightarrow_{p_1} \langle A_1, c_1\rangle \longrightarrow_{p_2} \dots \longrightarrow_{p_n} \langle A_n, c_n\rangle$.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. $\mathcal{R}(A,d) = \{\langle c,p\rangle \mid \text{there exists } \pi \in \mathrm{Comp}(A) : c = res(\pi) \text{ and } p = prob(\pi)\}$.

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.

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

We can now define the observables of an agent A with respect to store d as:

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

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

3. Identity Confinement

The original idea of non-interference as stated in [8] 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.

We illustrate the notion of identity confinement via an example borrowed from [18] where the setting is that of imperative languages. This example also show the difference between non-deterministic and probabilistic (identity) confinement.

Example 1 *In an imperative language, confinement — as formulated for example in [17, 18] — 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 non-deterministic and probabilistic confinement:

$$h := h\, mod\ 2; (l := h \ {}_{\frac{1}{2}}[\!]_{\frac{1}{2}}\ (l := 0 \ {}_{\frac{1}{2}}[\!]_{\frac{1}{2}}\ l := 1))$$

The value of l clearly depends "somehow" on h. However, if we resolve the choice non-deterministically 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 $h\, mod\ 2 = 0$ we have $\{l = 0, l = 1\}$ and for $h\, mod\ 2 = 1$ we get $\{l = 1, l = 0\}$, 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 non-deterministically 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 $h\, mod\ 2 = 0$ we get $\left\{\langle \frac{3}{4}, l = 0\rangle, \langle \frac{1}{4}, l = 1\rangle\right\}$, and for $h\ mod\ 2 = 1$ we have $\left\{\langle \frac{1}{4}, l = 0\rangle, \langle \frac{3}{4}, l = 1\rangle\right\}$. Therefore, the probabilities to get $l = 0$ and $l = 1$ reveal if h is even or odd, i.e. h is probabilistically not *confined.*

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

$$\begin{array}{lcl}
\texttt{hOn} & \equiv & \mathit{true} \to \frac{1}{2} : \mathbf{tell}(\texttt{on}) \ [\!]\ \mathit{true} \to \frac{1}{2} : \texttt{Rand} \\
\texttt{hOff} & \equiv & \mathit{true} \to \frac{1}{2} : \mathbf{tell}(\texttt{off}) \ [\!]\ \mathit{true} \to \frac{1}{2} : \texttt{Rand} \\
\texttt{Rand} & \equiv & \mathit{true} \to \frac{1}{2} : \mathbf{tell}(\texttt{on}) \ [\!]\ \mathit{true} \to \frac{1}{2} : \mathbf{tell}(\texttt{off})
\end{array}$$

The constraint system consists of four elements:

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

where true $\leq$ `on` $\leq$ false and true $\leq$ `off` $\leq$ 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 2.3, the observables of a PCCP agent correspond to a distribution on the constraint system, that is a vector in the space $\mathcal{V}(\mathcal{C})$. The difference between two observables can then 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. In the following we will sometimes omit the index ∞.

Probabilistic identity confinement is then defined as follows [5]:

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, i.e. $\mathcal{O}(p : A \parallel q : S) = \mathcal{O}(p : B \parallel q : S)$ or equivalently,*

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

for some fixed scheduling priorities p and $q = 1 - p$.

It is easy to check that for the agents `hOn` and `hOff` of the previous example the simple agent $S \equiv \mathbf{ask}(\texttt{on}) \to 1/3 : \mathbf{tell}(f)[\!]\mathbf{ask}(\texttt{off}) \to 2/3 : \mathbf{tell}(e)$ is such that

$$\|\mathcal{O}(p : \texttt{hOn} \parallel q : S) - \mathcal{O}(p : \texttt{hOff} \parallel q : S)\| = 1/2.$$

Example 2 *As another example consider the following two PCCP agents:*

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

If we consider their non-deterministic versions we see that A and B executed in any context give the same observables. 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, but they are not probabilistically confined. A spy which reveals the identity of A and B is the following agent:

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

as the following observables are different.

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

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

4. Approximate Confinement

The confinement notion discussed above is *exact* in the sense that it refers to the equivalence of the agents' behaviour.

However, sometimes it is practically more useful to base confinement on some *similarity* 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 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 $\varepsilon \geq 0$.

Example 3 *[2] Consider an* ATM *(Automatic Teller Machine) accepting only a single* PIN *number n out of m possible* PIN*s, e.g. $m = 10000$:*

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

This agent simulates an ATM *which recognises* PIN*n: if* PIN*n has been told the machine dispenses cash, otherwise — for any incorrect* PIN*i — it sounds an alarm. The (active) spy S tries a random* PIN *number i:*

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

If we consider two such machines ATM*n_1 and* ATM*n_2 for $n_1 \neq n_2$ and execute them in context S we obtain two slightly different observables $\mathcal{O}(p : \mathrm{ATM}n_1 \parallel q : S)$ and $\mathcal{O}(p : \mathrm{ATM}n_2 \parallel q : S)$. For most* PIN*s both machines will sound an alarm in most cases, but if we are lucky, the spy will use the correct* PIN*s 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:*

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

The set $\{\mathrm{ATM}n\}_n$ is ε-confined with respect to $\mathcal{S} = \{S\}$ 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 confinement we will discuss 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 discuss first different kinds of such contexts, which we will refer to as *spies*.

4.1. Admissible Spies

Security depends on the quality of the possible attacker. Clearly, no system is secure against an omnipotent attacker. Therefore, it makes sense to restrict our consideration to particular classes of spies [13].

We will consider simple attackers expressed in PCCP by:

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

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 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. Nevertheless, they are sufficient for formalising quite powerful timing attacks as described for example in [11].

A generalisation of this class is to consider active spies (e.g. Example 4 and Example 3) and/or spies with memory (e.g. $\textbf{ask}(c) \rightarrow p : \textbf{ask}(d) \rightarrow q : \textbf{tell}(f)$).

Example 4 *Consider the two agents:*

$$\begin{aligned} A \quad &\equiv \quad \textbf{ask}(c) \rightarrow 1 : \textbf{tell}(d) \\ B \quad &\equiv \quad \textbf{stop}. \end{aligned}$$

A and B are obviously confined with respect to any passive *spy: In store $true$ they both do nothing, and it is therefore impossible to distinguish them by just observing. However, for an* active *spy like $S \equiv \textbf{tell}(c)$ it is easy to determine if it is being executed in parallel with A or B.*

4.2. Approximate Identity Confinement

We introduce the notion of approximate confinement as a generalisation of the identity confinement introduced in [5] and defined in Section 3. 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 $\mathcal{S}$ 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 $||.||_\infty$ as in Definition 4.

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

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

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

Obviously, if two agents A and B are ε-confined with $\varepsilon = 0$ then they are probabilistically identity confined.

The number ε depends on the particular class of spies $\mathcal{S}$ and in some sense 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.

4.3. Statistical Interpretation of ε

The notion of approximate confinement is strongly related to statistical concepts, in particular to so-called *hypothesis testing*, see e.g. [21].

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 $\mathcal{O}(A \parallel S)$ and $\mathcal{O}(B \parallel S)$ are ε-different (we ignore in this argument the concrete scheduling probabilities here, i.e. we write $A \parallel S$ instead of $p : A \parallel q : S$). In particular, we can identify some constraint 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 $A \parallel S$ and $p_B(c)$ is the probability that c is a result of $B \parallel S$. 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 $A \parallel S$ and $B \parallel S$ will differ "on the long run" by exactly a factor ε.

For an unknown agent X we can try do determine $p_X(c)$ experimentally by executing $X \parallel S$ 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 $\varepsilon \neq 0$).

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 practically purposes we need a way to distinguish A and B by finite executions of $A \parallel S$ and $B \parallel S$. If we execute $A \parallel S$ and $B \parallel S$ a finite number of — say n — times, we can observe a certain experimental frequency $p^n_A(c)$ and $p^n_B(c)$ for getting c as a result. Each time we repeat this finite sequence of some n executions we may get different values for $p^n_A(c)$ and $p^n_B(c)$ (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 $p^n_X(c)$ by testing, i.e. by looking at n executions of $X \parallel S$. We could now try to compare $p^n_X(c)$ with $p^n_A(c)$ and $p^n_B(c)$ 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 $p^n_X(c)$, $p^n_A(c)$ or $p^n_B(c)$ (each experiment may give us different values) nor can we test if $p^n_X(c) = p^n_A(c)$ or $p^n_X(c) = p^n_B(c)$ nor if $p^n_X(c) = p_A(c)$ or $p^n_X(c) = p_B(c)$.

For example, it is possible that c is (coincidental) not the result of the first execution of $X \parallel S$, although the (long-run) probabilities of obtaining c by executing $A \parallel S$ or $B \parallel S$ are, let's say, $p_A = 0.1$ and $p_B = 0.5$. If we stop our experiment after $n = 1$ executions we get $p^1_X(c) = 0$. We know that $X = A$ or $X = B$ but the observed $p^1_X(c)$ is different from both p_A and p_B. Nevertheless, we could argue that it is more likely that $X = A$ as the observed $p^1_X(c) = 0$ is closer to $p_A = 0.1$ than to $p_B = 0.5$. The problem is now to determine how much the identification of X with A is "more correct" than identifying X with B.

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

Hypothesis Formulation. The problem is to determine experimentally if an 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 $X \parallel S$. We then can compare this estimate with the observables of $A \parallel S$ and $B \parallel S$. We can then formulate a hypothesis H about the identity of X, namely either that "X is A" or that "X is B" depending on the fact that this estimate is closer to $\mathcal{O}(A \parallel S)$ or $\mathcal{O}(B \parallel S)$. More precisely, the method to formulate the hypothesis H about the identity of the unknown process X consists of two following steps:

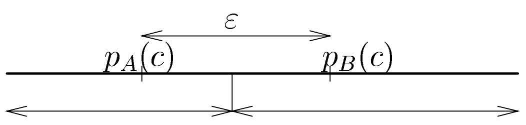
1. We execute $X \parallel S$ exactly n times in order to obtain an experimental approximation, i.e. average, for its observables
$$\overline{\mathcal{O}}_n(X \parallel S) = \left\{ \left\langle c, \frac{\text{\# of times } c \text{ is the result}}{n} \right\rangle \right\}_{c \in \mathcal{C}}$$
2. Depending if $\overline{\mathcal{O}}_n(X \parallel S)$ is closer to the observables $\mathcal{O}_n(A \parallel S)$ or $\mathcal{O}_n(B \parallel S)$ we formulate the hypothesis
$$H : \begin{cases} X = A & \text{if } \|\mathcal{O}_n(X \parallel S) - \mathcal{O}(A \parallel S)\| \leq \\ & \leq \|\mathcal{O}_n(X \parallel S) - \mathcal{O}(B \parallel S)\| \\ X = B & \text{otherwise} \end{cases}$$

Hypothesis Testing. The question is now whether the guess expressed by H about the true identity of the black box X is correct, or more precisely: What is the probability that the hypothesis H indeed holds?

In the following we use the notation $p_X(d)$ and $p^n_X(d)$ to indicate the probability assigned to d in the distribution representing the observables $\mathcal{O}(X \parallel S)$ and the experimental average $\mathcal{O}_n(X \parallel S)$ respectively.

If we assume that there are only finitely many constraints with non-zero probability in the observables then there exists a constraint c where the difference between $\mathcal{O}_n$ and $\mathcal{O}$ is maximal.

We therefore can look at a simplified situation where we are considering only this single constraint c. Let us assume without loss of generality that $p_A(c) < p_B(c)$ as in the diagram below:



If the experimental value $p_A^n(c)$ we obtained in our test is anywhere to the left of $p_A(c) + \varepsilon/2$ then the hypothesis H we formulate (based in $p_A^n(c)$) 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". Now assume that X is A. Then the probability $P(H)$ that we will formulate the correct hypothesis is:

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

To estimate $P(H)$ we therefore have just to estimate the probability $P(p_A^n(c) < p_A(c) + \varepsilon/2)$ that the experimental value $p_A^n(c)$ is left of $p_A(c) + \varepsilon/2$.

Confidence Estimation. The confidence we can have in the hypothesis we formulated can be determined by various statistical methods. We consider here methods which allow us to estimate the probability that an experimental average X_n, like $p_A^n(c)$, is within a certain distance from the corresponding expectation value $\mathbf{E}(X)$, like $p_A^n(c)$, i.e. the probability $P(||X_n - \mathbf{E}(X)|| \leq \varepsilon)$, for some $\varepsilon \geq 0$. These methods are essentially all based on the *central limit theorem*, see e.g. [1, 10, 21].

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*. The central limit theorem for this type of tests [10, Thm 9.2], gives us an estimate for the probability that the experimental value $S_n = n \cdot X_n$ is in a certain interval $[a, b]$:

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

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 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 $P(H)$.

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

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

This allows us (in principle) to compute the probability:

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

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

$$\begin{array}{rcl} P(H) & = & 1 - P\left(p_A(c) + \frac{\varepsilon}{2} \leq p_A^n(c)\right) \\ & \approx & 1 - \lim\limits_{n\to\infty} P\left(p_A(c) + \frac{\varepsilon}{2} \leq p_A^n(c)\right) \\ & = & 1 - \int_{a_0}^{\infty} \exp\left(\frac{-x^2}{2}\right) \end{array}$$

with

$$a_0 = \frac{n\varepsilon}{2} \frac{1}{\sqrt{npq}} = \frac{\varepsilon\sqrt{n}}{2\sqrt{pq}}.$$

We see that the only way to increase the probability $P(H)$, 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, $P(H)$: 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.

5. Analysis: Concrete Semantics

The ε-confinement property of a PCCP program can be checked by means of a semantics-based static analysis of the program. In this section we will consider a concrete collecting semantics which describes the same observables defined in Section 2.3 in a slightly more abstract way than the transition system in Table 2. The analysis we present will allow us to calculate an exact ε for measuring the confinement of a set of agents. In Section 6 we will present a compositional semantics for PCCP which will allow us to calculate a correct approximation of the ε in a more abstract and simplified way.

5.1. A Collecting Semantics

We will base our analysis on a collecting semantics consisting of distributions over the set Conf of configurations. Such distributions represent all the possible configurations which are reachable by a given program in one step. They are vectors in the space $\mathcal{V}(\text{Conf})$ obtained by means of a free construction similar to the one described for the space of constraints $\mathcal{V}(\mathcal{C})$, and can be represented as multi-sets of pairs:

$$\Phi = \{\langle\langle A_i, c_i\rangle, p_i\rangle\}_i,$$

where p_i represents the probability associated to the configuration $\langle A_i, c_i \rangle$. Note that one configuration can occur more than once in a multi-set Φ. In this case the corresponding distribution will be obtained by compactifying the set as described in Section 2.3 and then normalising the resulting set.

The collecting semantics for PCCP is defined by the rules given in Table 3. These rules defines a linear operator $\mathbf{G}$ on the space $\mathcal{V}(\mathrm{Conf})$. The semantics of an agent A is obtained by iteratively applying $\mathbf{G}$ starting from the initial configuration $\langle A, \mathit{true} \rangle$. This yields a sequence

$$[\![A]\!]_{coll} = (\Phi_i)_i = (\mathbf{G}^i(\langle A, \mathit{true} \rangle))_i.$$

From this semantics we can retrieve the observables $\mathcal{O}(A)$ as stated in the following proposition. To this purpose we first have to transform a distribution over configurations $\Phi = \{\langle \langle A_j, c_j \rangle, p_j \rangle\}_j$, into a distribution over stores $\phi = \{\langle c_j, p_j \rangle\}_j$ by abstracting the information about the agent.

Proposition 1 *Let A be an agent with collecting semantics $[\![A]\!]_{coll} = (\Phi_i)_i$ and let $(\phi_i)_i$ be the corresponding sequence of distributions over stores. Then:*

$$\mathcal{O}(A) = \lim_i \phi_i.$$

The proof of this proposition relies on a result on the existence of the limit of the sequence $(\mathbf{G}^i(\langle A, \mathit{true} \rangle))_i$ which we will omit for lack of space.

Example 5 *For the two agents A and B in Example 2 their collecting semantics is depicted in Table 4.*

5.2. Security Analysis

Given two agents A and B, scheduling priorities p and $q = 1 - p$ and a spy S, our aim is to calculate the ε such that

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

We will show how we can construct the observables $\mathcal{O}(p : A \parallel q : S)$ and $\mathcal{O}(p : B \parallel q : S)$ from the collecting semantics of A and B respectively, and for spies S in $\mathcal{S}_2$.

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

$$\begin{array}{rcl} S & \equiv & \mathbf{ask}(c_1) \to q_1 : \mathbf{tell}(f_1) \\ & & [\!]\ \mathbf{ask}(c_2) \to q_2 : \mathbf{tell}(f_2), \end{array}$$

with $q_1 + q_2 = 1$. Suppose the observables of A are $\mathcal{O}(A) = \{\langle d_i, p_i \rangle\}_i$ and there is only one computational path $\pi_i \in \mathrm{Comp}(A)$ leading to d_i for all i.

Then the observables of $p : A \parallel q : S$ are given by:

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

where

$$\begin{array}{rcl} r_i^1 & = & p_i \cdot (((\sum_{l=0}^{n_i-1} p^{m_i+l})q + p^{m_i+n_i})q_1 + (1 - p^{m_i})) \\ r_i^2 & = & p_i \cdot ((\sum_{l=0}^{n_i-1} p^{m_i+l})q + p^{m_i+n_i})q_2, \end{array}$$

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.

Corollary 1 *With the hypothesis of Proposition 2 the following holds for all i:*

$$p_i = r_i^1 + r_i^2.$$

For the general case, where there is more than one path leading to the same constraint d_i in the observables $\mathcal{O}(A)$ the probabilities r_i^1 and r_i^2 are given by

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

where the sum is over all computational paths $\pi_{ik} \in \mathrm{Comp}(A)$ 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}. Obviously $p_i = r_i^1 + r_i^2$ holds in the general case too.

5.3. Limit Analysis

Starting from the formulas

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

and

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

consider the difference

$$|r_i^1 - r_i^2| = p_i \cdot [q \cdot \sum_{l=0}^{n_i} p^{m_i+l} \cdot |q_1 - q_2| + (1 - p^{m_i})].$$

R0	$\{\ldots,\langle\langle\mathbf{stop},d\rangle,p\rangle,\ldots\}\longrightarrow\{\ldots,\langle\langle\mathbf{stop},d\rangle,p\rangle,\ldots\}$	
R1	$\{\ldots,\langle\langle\mathbf{tell}(c),d\rangle,p\rangle,\ldots\}\longrightarrow\{\ldots,\langle\langle\mathbf{stop},c\sqcup d\rangle,p\rangle,\ldots\}$	
R2	$\{\ldots,\langle\langle\Box_{i=1}^{n}\ \mathbf{ask}(c_i)\to p_i:A_i,d\rangle,q\rangle,\ldots\}\longrightarrow\{\ldots,\langle\langle A_j,d\rangle,q\cdot\tilde{p}_j\rangle,\ldots\}$	$j\in[1,n]$ and $d\vdash c_j$
R3	$\{\ldots,\langle\langle\|_{i=1}^{n}\ p_i:A_i,d\rangle,q\rangle,\ldots\}\longrightarrow\{\ldots,\langle\langle A'_j,d'\rangle,q\cdot\tilde{p}_j\rangle,\ldots\}$	$j\in[1,n]$ and $\langle A_j,c\rangle\longrightarrow_p\langle A'_j,c'\rangle$
R4	$\{\ldots,\langle\langle\exists_x A,d\rangle,q\rangle,\ldots\}\longrightarrow\{\ldots,\langle\langle\exists_x A',d\sqcup\exists_x d'\rangle,q\cdot p\rangle,\ldots\}$	$\langle A,\exists_x d\rangle\longrightarrow_p\langle A',d'\rangle$
R5	$\{\ldots,\langle\langle p(y),d\rangle,q\rangle,\ldots\}\longrightarrow\{\ldots,\langle\langle A,d\rangle,q\rangle,\ldots\}$	$p(x):-A\in P$

Table 3. A Collecting Semantics for PCCP

$$[\![A]\!]_{coll}=\left\{\begin{array}{c}\{\langle 1,A,true\rangle\}\\ \{\langle 1/2,1/2:\mathbf{stop}\ \|\ 1/2:\mathbf{tell}(d),c\rangle,\langle 1/2,1/2:\mathbf{tell}(d)\ \|\ 1/2:\mathbf{stop},d\rangle\}\\ \{\langle 1,\mathbf{stop},c\sqcup d\rangle\}\end{array}\right.$$

$$[\![B]\!]_{coll}=\left\{\begin{array}{c}\{\langle 1,B,true\rangle\}\\ \{\langle 1,\mathbf{stop},c\sqcup d\rangle\}\end{array}\right.$$

Table 4. The Collecting Semantics for A and B.

This difference 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 case q_1 goes to 0 and q_2 approaches 1 we obtain the following limit formulas for r_i^1 and r_i^2:

$$r_i^1=p_i\cdot(1-p^{m_i}),$$

and

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

By the equation

$$1-p^m=q\sum_{l=0}^{m-1}p^l,$$

we get for r_i^2 the following formula:

$$r_i^2=p_i\cdot p^{m_i}.$$

For the case that q_1 tends to 1 and q_2 goes 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.

5.4. Properties of a Spy S

An agent S is a spy for two agents A and B if and only if the observables $\mathcal{O}(p:A\ \|\ q:S)$ are different from $\mathcal{O}(p:B\ \|\ q:S)$. The discussion in Section 5.2 provides us with a useful criterion to decide whether an agent in $\mathcal{S}_2$ is a spy for A and B.

Proposition 3 *Given an agent $S\in\mathcal{S}_2$ of the form*

$$\begin{array}{rcl}S&\equiv&\mathbf{ask}(c_1)\to q_1:\mathbf{tell}(f_1)\\ &&\Box\ \mathbf{ask}(c_2)\to q_2:\mathbf{tell}(f_2),\end{array}$$

and two agents A and B with identical (probabilistic input/output) observables $\mathcal{O}(A)=\mathcal{O}(B)$, then S is a spy for A and B if there exists a constraint c_j such that

1. *$\langle c_j,p_j\rangle\in\mathcal{O}(A)=\mathcal{O}(B)$,*
2. *$c_j\vdash c_1$ and $c_j\vdash c_2$,*
3. *$r_j^1(A)\neq r_j^1(B)$.*

Note that the last condition is equivalent to $r_j^2(A)\neq r_j^2(B)$ as we always have $r_j^1(A)+r_j^2(A)=p_j=r_j^1(B)+r_j^2(B)$. For the same reason we also have: $|r_j^1(A)-r_j^1(B)|=|r_j^2(A)-r_j^2(B)|$.

The values of r^1_j and r^2_j depend on what scheduling between A or B and S we have chosen, i.e. on the concrete values of p and q. But as we can see from the the closed expressions for r^1_j and r^2_j, if $r^1_j(A) \neq r^1_j(B)$ for one scheduling then this is true also for any other (non-trivial) scheduling. The following holds therefore:

Corollary 2 *An agent S is a spy for A and B independently of the scheduling priorities, as long as $0 < p < 1$.*

5.5. Effectiveness of a Spy S

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

Proposition 4 *Let S be a spy in $\mathcal{S}_2$ of the form*

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

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

$$\begin{array}{rcl} \varepsilon & = & \max_{c_j} \{ |r^1_j(A) - r^1_j(B)| \} \\ & = & \max_{c_j} \{ |r^2_j(A) - r^2_j(B)| \}. \end{array}$$

Corollary 3 *The effectiveness ε of a spy S depends on the scheduling priorities p and $q = 1 - p$.*

5.6 The Most Effective Spy

The limit analysis in Section 5.3 shows that the most effective spy for a fixed pair of guards (c_1, c_2) is obtained by considering a choice distribution where $\lim q_1 = 0$ and $\lim q_2 = 1$ or vice versa. In other words the "best" spy is the one where the probabilities are at the extreme opposite.

Example 6 *Using the limit analysis we now show that C is not the most effective spy for the agents A and B in Example 2. 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 fix $q_1 = 0$ and $q_2 = 1$. Assuming again the uniform scheduling $p = q = \frac{1}{2}$, we now calculate the corresponding ε. To this purpose, we consider the expressions for r^1_i and r^2_i in Section 5.3.*

A**:** *For agent A and its two computational paths we get*

1. *For the first path with $c_1 = c, c_2 = d$ and $m_1 = 1$ we get:*

$$r^c_{c\sqcup d,1} = \frac{1}{2}(1-p) = \frac{1}{4} \text{ and } r^d_{c\sqcup d,1} = \frac{1}{2}p = \frac{1}{4}$$

2. *For the second path with $c_1 = d$, $c_2 = c$ and $m_1 = 1$ we get:*

$$r^d_{c\sqcup d,2} = \frac{1}{2}1 = \frac{1}{2} \text{ and } r^c_{c\sqcup d,2} = \frac{1}{2}0 = 0$$

Summing up we get the extreme probabilities for $c \sqcup d \sqcup e$ and $c \sqcup d \sqcup f$:

$$r^c_{c\sqcup d} = \frac{1}{4} + 0 = \frac{1}{4} \text{ and } r^d_{c\sqcup d} = \frac{1}{4} + \frac{1}{2} = \frac{3}{4}$$

B**:** *We have only one path with $c_1 = c, c_2 = d$ and $m_1 = 0$. Thus:*

$$r^c_{c\sqcup d,1} = 1(1-1) = 0 \text{ and } r^d_{c\sqcup d,1} = 1 \cdot 1 = 1$$

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 $\mathcal{S}_2$:

$$\varepsilon = |r^c_{c\sqcup d}(A) - r^c_{c\sqcup d}(B)| = |\frac{1}{4} - 0| = \frac{1}{4}.$$

We can therefore conclude that A and B are $\frac{1}{4}$-confined with respect to all agents in $\mathcal{S}_2$ (thus C was not the most effective spy).

6. Analysis: 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 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 at step t (at the earliest in a particular path) 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.

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

$[\![\textbf{stop}]\!]$	=	$\emptyset$
$[\![\textbf{tell}(c)]\!]$	=	$\{\langle true, c, 1, 1\rangle\}$
$[\![\,[\!]_{i=1}^{n}\ \textbf{ask}(c_i) \rightarrow p_i : A_i]\!]$	=	$\bigoplus_i (p_i, c_i, [\![A_i]\!])$
$[\![A; B]\!]$	=	$[\![A]\!] \odot [\![B]\!]$
$[\![\exists_x A]\!]$	=	$[\![A]\!]$
$[\![p(x)]\!]$	=	$\nabla_n^\delta([\![A]\!])$ for $p(x) : -A \in P$

Table 5. The Analysis for PCCP Agents

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 6: We start by a trivial approximation ∇_0^δ and continue by replacing the procedure call p in the term A by the previous approximation — denoted by $[\![A]\!][p \mapsto \nabla_i^\delta([\![A]\!])]$ — until the difference between the current and previous approximation becomes small enough (less or equal to δ) or we reach the maximal recursion depth n. In this case we take an approximation ∇_∞^δ in place of further unwindings. The difference between two approximations 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:

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

which is effectively always reached after a finite number of unwindings. This can lead to a substantial over-approximation of t for recursive agents. The operator ∇_n^δ is the quantitative analogue of a widening operator in the standard approaches to abstract interpretation [3]; whilst the standard definition of a widening involves over-approximation (of an upper bound), in the quantitative setting we settle for "closeness".

6.1. Abstract Security Analysis

Given the set of quadruples associated with an agent, we can extract the set of abstract paths of execution:

$$\begin{aligned} \mathit{paths}([\![A]\!]) = \{q_0 \ldots q_n | \forall 0 \le i \le n. q_i \in [\![A]\!] \wedge \\ q_0 \equiv \langle \mathit{true}, ?, 1, ?\rangle \wedge d_{q_i} \vdash c_{q_{i+1}} \wedge \\ (t_{q_{i+1}} > t_{q_i} \ \vee\ t_{q_{i+1}} = t_{q_i} = \infty) \wedge \\ \forall j \le i. q_j \ne \langle c_{q_{i+1}}, d_{q_{i+1}}, t_{q_{i+1}}, p_{q_{i+1}}\rangle\} \end{aligned}$$

where $q_i \equiv \langle c_{q_i}, d_{q_i}, t_{q_i}, p_{q_i}\rangle$.

Since the analysis of choice does not normalise the associated probabilities, the probabilities in $\mathit{paths}([\![A]\!])$ may be smaller than in the concrete semantics.

Given a path $\sigma \in \mathit{paths}([\![A]\!])$ with c first entailed in $q_{\sigma i}$ and c' first entailed in $q_{\sigma j}$, the difference $t_{q_{\sigma j}} - t_{q_{\sigma i}}$ defines the (abstract) number of steps $\overline{m}_\sigma$ between the store entailing c and the store c' in path σ, while $\overline{p}_\sigma$, which is the product of the probabilities, is the abstract probability associated to σ. Therefore, for each pair of constraints c_1 and c_2, the abstract analysis of program P gives us the set:

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

6.2. 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. 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 analysis of choice implies that

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

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

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

$$
\begin{array}{lcl}
\nabla^\delta_0([\![A]\!]) & = & \{\langle \mathit{true}, \mathit{true}, 1, 1\rangle\} \\
\nabla^\delta_{i+1}([\![A]\!]) & = & \begin{cases} [\![A]\!][p \mapsto \nabla^\delta_i([\![A]\!])] & \text{if } \|[\![A]\!][p \mapsto \nabla^\delta_i([\![A]\!])] - \nabla^\delta_i([\![A]\!])\| > \delta \text{ and } i \leq n, \\ \nabla^\delta_i([\![A]\!]) & \text{if } \|[\![A]\!][p \mapsto \nabla^\delta_i([\![A]\!])] - \nabla^\delta_i([\![A]\!])\| \leq \delta \text{ and } i \leq n, \\ \nabla^\delta_\infty([\![A]\!])) & \text{otherwise} \end{cases} \\
\nabla^\delta_\infty([\![A]\!]) & = & \{\langle d, c, \infty, p\rangle \mid \langle d, c, t, p\rangle \in [\![A]\!]\}
\end{array}
$$

Table 6. Unwinding a Procedure Call

7. Conclusions

We introduced a *quantitative measure* describing the vulnerability of a set of agents against some kind of attacks aimed at revealing their identity. Based on this measure we then defined the notion of ε-confinement. This notion differs from strict confinement — which aims in determining if agents are absolutely invulnerable — by allowing for some exactly quantified weaknesses. The confinement measure can be interpreted in statistical terms as the probability of guessing the right hypothesis about the identity of the host agent after a given number of tests. For a smaller ε a larger number of experiments must be performed to reach the same level of confidence.

In a second step we identified for each agent and an admissible spy two numbers, m and n, which forecast the observables of the agent in the presence of the spy. The collection of all m's and n's characterises an agent with respects to attacks by any admissible spy. We showed that for the most effective attackers the collection of m's alone is sufficient to determine the corresponding observables. The information on the m's is therefore all we need to know of a set of agents in order to compute their ε-confinement.

Finally, we observed that if we are able to determine some range for the m's — instead of their exact values — we can still compute the range of possible observables and compare them to get a correct approximation of the ε. Following this argument we formulated an abstract semantics which produces estimates — i.e. bounds — of the m's.

It is important to note that this abstract analysis only makes sense for approximate confinement notion. If we had to consider strict confinement any non-exact estimation of the m's would fail to give a meaningful result: only if we know the m's exactly can we tell if $\varepsilon = 0$ or $\varepsilon \neq 0$.

The notion of ε-confinement we introduced 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. A similarity relation provides information about such quantity, whereas equivalence relations such as observational equivalence or bisimilarity can only establish whether two objects can be identified or not. For example, in [13] the security of cryptographic protocols is specified via an observational equivalence relation which identifies protocols which differ asymptotically for a polynomial factor. Such quantity is nevertheless neither used to quantify the similarity of the protocols nor to calculate a correspondent approximation level of the protocols security property. Analogously, the bisimulation through probabilistic testing of Larsen and Skou [12] allows to state the indistinguishability of two processes with respect to so-called testable properties. These are properties that can be tested up to a given level δ of significance which gives an upper bound of making the wrong decision. Again such a quantity is not intended to provide a quantitative measure of the behavioural difference between two processes.

The quantity measuring the similarity of two objects 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 the probabilistic input/output observables of PCCP programs which can be described by probability distributions on the underlying space of constraints, and we used a vector norm to measure their similarity. In [24] 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.

The type of attacks we considered in this paper are *internal attacks* where the attacker is in some sense part of the observed system: in particular it is scheduled like any other agent. 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. It is obvious that for different types of attacks we need different types of quantitative information for our analysis. For external attacks, for example, a useful information is the average store of an agent in some specified number of steps (the observation time) [14].

References

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

[2] David Clark, Sebastian Hunt, and Pasquale Malacaria. private communication. July 2001.

[3] Patrick Cousot and Radhia Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Symposium on Principles of Programming Languages (POPL)*, pages 238–252, Los Angeles, 1977.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] Charles M. Grinstead and J. Laurie Snell. *Introduction to Probability*. American Mathematical Society, Providence, Rhode Island, second revised edition, 1997.

[11] Paul C. Kocher. Cryptanalysis of Diffie-Hellman, RSA, DSS, and other cryptosystems using timing attacks. In D. Coppersmith, editor, *Advances in Cryptology, CRYPTO '95: 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27–31, 1995*, volume 963 of *Lecture Notes in Computer Science*, pages 171–183, Berlin — Heidelberg — London, 1995. Springer-Verlag.

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

[13] 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.

[14] Alessandra Di Pierro, Chris Hankin, and Herbert Wiklicky. Approximate confinement under uniform attacks. submitted for publication.

[15] P. Rayan, 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.

[16] 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.

[17] Andrei Sabelfeld and David Sands. A per model of secure information flow in sequential programs. In *ESOP'99*, number 1576 in Lecture Notes in Computer Science, pages 40–58. Springer Verlag, 1999.

[18] 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.

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

[20] 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.

[21] Jun Shao. *Mathematical Statistics*. Springer Texts in Statistics. Springer Verlag, New York – Berlin – Heidelberg, 1999.

[22] Geoffrey Smith and Dennis Volpano. Secure information flow in a multi-threaded imperative language. In *Symposium on Principles of Programming Languages (POPL'98)*, pages 355–364, San Diego, California, 1998. ACM.

[23] Geoffrey Smith and Dennis Volpano. Verifying secrets and relative secrecy. In *Symposium on Principles of Programming Languages (POPL'00)*, pages 368–276, Boston, Massachusetts, 2000. ACM.

[24] 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.

[25] Dennis Volpano and Geoffrey Smith. Confinement properties for programming languages. *SIGACT News*, 29(3):33–42, September 1998.

[26] 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.

[27] 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.

Appendix

Proof of Proposition 2

Proof The first three terms of the expression are quite straight forward:

- If a constraint d_i in the final observables does neither entail c_1 nor c_2, then all along the computational path leading to d_i this was the case, the execution therefore can never schedule the spy and neither f_1 nor f_2 are added to the store. The final configuration in this case is thus $\langle p : \textbf{stop} \parallel q : S, d_i\rangle$ which we obtain with exactly the same probability as for A.

- If only one of the two guards are ever entailed, then the agent S acts just like a spy of class $\mathcal{S}_1$ discussed before. In this case we get $\langle p : \textbf{stop} \parallel \textbf{stop}, d_i \sqcup f_1\rangle$ and $\langle p : \textbf{stop} \parallel \textbf{stop}, d_i \sqcup f_2\rangle$ as final configurations together with their inherited probability p_j.

The general case in which both constraints c_1 and c_2 are entailed needs a more careful analysis. Let us assume without loss of generality that c_1 is told first, after k steps and that it needs $m \geq 0$ steps until also c_2 is entailed by the store, and that finally it takes n further steps until $p : A \parallel q : S$ terminates. Based on the collecting semantics of A,

$$\Phi_j = \left\{\left\langle\left\langle A_i^j, c_i^j\right\rangle, p_i^j\right\rangle\right\},$$

the collecting semantics $\overline{\Phi}$ of $p : A \parallel q : S$ is then of the following form:

$j < k$**:** The distributions for A and $p : A \parallel q : S$ are essentially identical:

$$\overline{\Phi}_j = \left\{\left\langle\left\langle p : A_i^j \parallel q : S, c_i^j\right\rangle, p_i^j\right\rangle\right\}_i.$$

$j = k$**:** The constraint c_1 is first entailed, i.e. there exists now a configuration $\langle p : A_i^k \parallel q : S, c_i^k\rangle$ with non-zero probability p_i^k in Φ_k such that $c_i^k \vdash c_1$ (and not yet $c_i^k \vdash c_2$):

$$\overline{\Phi}_k = \{\ldots, \langle\langle p : A_i^k \parallel q : S, c_i^k\rangle, p_i^k\rangle, \ldots\}.$$

$k < j < k + m$**:** Then the distribution $\overline{\Phi}_{k+1}$ is given by:

$$\begin{aligned} \overline{\Phi}_{k+1} = \{&\ldots, \\ &\langle\langle p : A_i^{k+1} \parallel q : S, c_i^{k+1}\rangle, p_i^{k+1} \cdot p\rangle, \\ &\langle\langle p : A_i^k \parallel q : \textbf{stop}, c_i^k \sqcup f_1\rangle, p_i^k \cdot q\rangle, \\ &\ldots\}. \end{aligned}$$

Only f_1 can be added to the store.

$j = k + m$**:** We get a similar iteration of $\overline{\Phi}_j$ as in the case of $\mathcal{S}_1$ spies, up to the moment when also c_2 gets entailed. In this case we have:

$$\begin{aligned} \overline{\Phi}_{k+m} = \{&\ldots, \\ &\langle\langle p : A_i^{k+m} \parallel q : S, c_i^{k+m}\rangle, p_i^{k+m} \cdot p^m\rangle, \\ &\langle\langle p : A_i^{k+m-1} \parallel q : \textbf{stop}, c_i^{k+m-1} \sqcup f_1\rangle, \\ &\quad p_i^{k+m-1} \cdot (1 - p^m)\rangle, \\ &\ldots\}. \end{aligned}$$

$j = k + m + 1$**:** We now have three possible continuations for the first agent $p : A_i^{k+m} \parallel q : S$: (1) we can ignore the spy and schedule A_i^{k+m}, or we can schedule the spy, in which case we have a choice between (2a) executing the first branch **tell**(f_1) or (2b) executing the second one, **tell**(f_1), as both guards are entailed. The probabilities that this is happening are p for the case

(1), and qq_1 and qq_2 for cases (2a) and (2b), respectively. The second agent $p : A_i^{k+m-1} \parallel q : \textbf{stop}$ has to continue quasi-deterministically with A_i^{k+m-1}.

$$\begin{aligned}
\overline{\Phi}_{k+m+1} \;=\; & \{\ldots, \\
& \langle\langle p : A_i^{k+m+1} \parallel q : S, c_i^{k+m+1}\rangle, \\
& \quad p_i^{k+m+1} \cdot p^{m+1}\rangle, \\
& \langle\langle p : A_i^{k+m} \parallel q : \textbf{stop}, c_i^{k+m} \sqcup f_1\rangle, \\
& \quad p_i^{k+m} \cdot p^m qq_1\rangle, \\
& \langle\langle p : A_i^{k+m} \parallel q : \textbf{stop}, c_i^{k+m} \sqcup f_2\rangle, \\
& \quad p_i^{k+m} \cdot p^m qq_2\rangle, \\
& \langle\langle p : A_i^{k+m} \parallel q : \textbf{stop}, c_i^{k+m} \sqcup f_1\rangle, \\
& \quad p_i^{k+m} \cdot (1-p^m)\rangle, \\
& \ldots\} \\
= \; & \{\ldots, \\
& \langle\langle p : A_i^{k+m+1} \parallel q : S, c_i^{k+m+1}\rangle, \\
& \quad p_i^{k+m+1} \cdot p^{m+1}\rangle, \\
& \langle\langle p : A_i^{k+m} \parallel q : \textbf{stop}, c_i^{k+m} \sqcup f_1\rangle, \\
& \quad p_i^{k+m} \cdot (p^m qq_1 + (1-p^m))\rangle, \\
& \langle\langle p : A_i^{k+m} \parallel q : \textbf{stop}, c_i^{k+m} \sqcup f_2\rangle, \\
& \quad p_i^{k+m} \cdot p^m qq_2\rangle, \\
& \ldots\}.
\end{aligned}$$

$j = k+m+n$: After further n steps the agent A terminates [either successfully or because of deadlock].

$$\begin{aligned}
& \overline{\Phi}_{k+m+n} = \\
& = \{\ldots, \\
& \quad \langle\langle p : \textbf{stop} \parallel q : S, c_i^{k+m+n}\rangle, \\
& \qquad p_i^{k+m+n} \cdot p^{m+n}\rangle, \\
& \quad \langle\langle p : A_i^{k+m+n-1} \parallel q : \textbf{stop}, c_i^{k+m+n-1} \sqcup f_1\rangle, \\
& \qquad p_i^{k+m+n-1} \cdot ((\sum_{l=0}^{n-1} p^{m+l})qq_1 + (1-p^m))\Big\rangle, \\
& \quad \langle\langle p : A_i^{k+m+n-1} \parallel q : \textbf{stop}, c_i^{k+m+m-1} \sqcup f_2\rangle, \\
& \qquad p_i^{k+m+n-1} \cdot (\sum_{l=0}^{n-1} p^{m+l})qq_2\Big\rangle, \\
& \quad \ldots\}.
\end{aligned}$$

$j = k+m+n+1$: The perhaps still not scheduled spy S must now finally be executed and we get finally:

$$\begin{aligned}
& \overline{\Phi}_{k+m+n+1} = \\
& = \{\ldots, \\
& \quad \langle\langle p : \textbf{stop} \parallel q : \textbf{stop}, c_i^{k+m+n} \sqcup f_1\rangle, \\
& \qquad p_i^{k+m+n} \cdot p^{m+n} q_1\rangle, \\
& \quad \langle\langle p : \textbf{stop} \parallel q : \textbf{stop}, c_i^{k+m+n} \sqcup f_2\rangle, \\
& \qquad p_i^{k+m+n} \cdot p^{m+n} q_2\rangle, \\
& \quad \langle\langle p : \textbf{stop} \parallel q : \textbf{stop}, c_i^{k+m+n} \sqcup f_1\rangle, \\
& \qquad p_i^{k+m+n} \cdot ((\sum_{l=0}^{n-1} p^{m+l})qq_1 + (1-p^m))\Big\rangle, \\
& \quad \langle\langle p : \textbf{stop} \parallel q : \textbf{stop}, c_i^{k+m+m} \sqcup f_2\rangle, \\
& \qquad p_i^{k+m+n} \cdot (\sum_{l=0}^{n-1} p^{m+l})qq_2\Big\rangle, \\
& \quad \ldots\} \\
& = \{\ldots, \\
& \quad \langle\langle p : \textbf{stop} \parallel q : \textbf{stop}, c_i^{k+m+n} \sqcup f_1\rangle, \\
& \qquad p_i^{k+m+n} \cdot (((\sum_{l=0}^{n-1} p^{m+l})q + p^{m+n})q_1 + (1-p^m))\Big\rangle, \\
& \quad \langle\langle p : \textbf{stop} \parallel q : \textbf{stop}, c_i^{k+m+m} \sqcup f_2\rangle, \\
& \qquad p_i^{k+m+n} \cdot ((\sum_{l=0}^{n-1} p^{m+l})q + p^{m+n})q_2\Big\rangle, \\
& \quad \ldots\}.
\end{aligned}$$

□

Proof of Corollary 1

Proof The original probability of d_i in the observables for A is $p_i^{k+m+n} = p_i$. Moreover, since $q_2 = 1 - q_1$ and by the following identity (for p=1-q):

$$1 - p^m = q \sum_{l=0}^{m-1} p^l, \qquad (1)$$

we have that:

$$\begin{aligned}
r_i^1 + r_i^2 \;=\; & p_i \cdot [q_1(q \sum_{l=0}^{n-1} p^{m+l} + p^{m+n}) + \\
& +1 - p^m + q_2(q \sum_{l=0}^{n-1} p^{m+l} + p^{m+n})] \\
= \; & p_i[1 - p^m + q \sum_{l=0}^{n-1} p^{m+l} + p^{m+n}] \\
= \; & p_i[1 - p^m + p^m - p^{m+n} + p^{m+n}] \\
= \; & p_i.
\end{aligned}$$

□