Approximate Probabilistic Confinement

Alessandra Di Pierro
Universitá di Pisa
dipierro@di.unipi.it

Chris Hankin
Imperial College
clh@doc.ic.ac.uk

Herbert Wiklicky
Imperial College
herbert@doc.ic.ac.uk

SecSafe Document	
Number:	SECSAFE-ICSTM-005
Version:	1.0
Classification:	Public
Status:	Work in Progress

1. INTRODUCTION

Non-interference models are an approach to security which focus on the absence of certain information flows in a system. In process algebra non-interference is often reduced to equivalence of certain processes [16] – this essentially reduces non-interference to *confinement*.

An information processing system has a *confinement* property if it protects some important — so-called high level — information from non-authorised — so-called low level — access. That means the information, e.g. the state or value of a high level variable, does not "leak" out, i.e. influence low level information. In a more complex implementation of the standard security model [8] one may have several levels ("confident", "secret", "top secret", etc) but we consider here only the simple two levels case.

For example, a computer operating system might try to protect certain user information, e.g. passwords, etc. (high level information), from being accessed by some applications, e.g. spreadsheets, browsers, etc (low level applications), though it might well allow these applications (the low level) access to other data, e.g. the system time, etc. If the low level applications are unable to access this high level information, e.g. the password of a user, we can say that it is confined.

Depending on the nature of the information flow between high and low level information, one can distinguish several types of confinement, namely *deterministic*, *nondeterministic*, and *probabilistic* confinement [24].

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 [10]. 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, 25, 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 never be achieved in real systems where high-level inputs interferes with low-level output all the time. 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).

We propose a notion of approximate confinement which overcomes this problem by allowing some leak of information provided that it is below a fixed approximation threshold ε. Thus we weaken equivalence to the notion of *similarity* measured by an appropriate measure of distance. This gives rise to the notion of the *most effective spy* which is the spy that is best able to distinguish between a pair of processes. This is closely related to the notion of hardest attacker in [14] and the time-bounded adversaries of [13].

We develop the approach in the context of a simple probabilistic declarative language which is presented in Section 2. Section 3 reviews the literature on probabilistic confinement. We introduce the notion of approximate confinement in Section 4. Sections 5, 6 and 7 concern analysis; in Section 5 we develop an exact analysis based on a collecting semantics, in Section 6 we give an application and Section 7 presents an abstract semantics. We conclude the paper in Section 8.

2. PROBABILISTIC CCP

To illustrate our approach we will consider throughout this paper a simple probabilistic declarative language, the Probabilistic Concurrent Constraint Programming (PCCP) language, which was introduced in [5, 6] as a probabilistic version of the Concurrent Constraint Programming (CCP) paradigm [20, 19].

2.1 Syntax of PCCP

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, 3] 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 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, $(\mathrm{Conf}, \longrightarrow_p)$, 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 [3] for a detailed description.

Rule **R1** describes the effect of $\mathbf{tell}(c)$. This agent always terminates successfully with probability one, and the new store is the least upper bound of the constraint c and the current store d, i.e. $c \sqcup d$. Note that the agent **stop** represents successful termination and is used to distinguish success from other forms of termination, e.g. deadlock (no guard is enabled) or in general situations in which agents get *stuck*.

Both rules **R2** and **R3** refer to *normalised* probabilities $\tilde{p}_i$. Normalisation can be defined for a generic set of real numbers $X = \{x_i\}$, as the process of replacing each x_i by $\tilde{x}_i$ defined as follows:

Definition 1. *Let $x = \sum_i x_i$, and n the cardinality of $X = \{x_i\}$. Then*

$$\tilde{x}_i = \begin{cases} \frac{x_i}{x} & \text{if } x \neq 0 \\ \frac{1}{n} & \text{otherwise} \end{cases}$$

Rule **R2** describes *probabilistic choice*. An agent A_i is called *enabled* iff its guard c_i is entailed by the store, i.e. $d \vdash c_i$. The normalisation process described above is applied only to those p_i's which are associated to enabled agents. The $\tilde{p}_i$'s are therefore normalised with respect to the set $\{p_i \mid d \vdash c_i\}$. Then the choice is made among the enabled agents only.

Rule **R3** describes a *prioritised interleaving*: Each time the scheduler has to select an agent to be executed, it will choose according to the probabilities $\tilde{p}_i$, i.e. the probabilities normalised with respect to the set $\{p_i \mid A_i \text{ is active }\}$. An agent A is called *active* if it can make a transition, i.e. there exists $\langle A', d'\rangle \in \mathrm{Conf}$ such that $\langle A, d\rangle \longrightarrow_p \langle A', d'\rangle$. Again, the normalisation and the successive choice is applied to active agents only. In this way the scheduling probability of an agent actually represents its *priority* [7]. Note that there is no rule for a transition from the **stop** agent to any other agent, i.e. **stop** is never active.

Rule **R4** in Table 2 deals with the introduction of local variables; we use the notation $\exists_x^d A$ for the agent A with local store d containing information on x which is hidden to the external store (see [19, 20, 3] for further details). Obviously, the transition probability p is not changed by hiding.

In the recursion rule **R5** for the procedure call $p(y)$, we assume that the link between the actual parameter y and the formal parameter x has been correctly established before $p(y)$ is replaced by the body of its definition in the program P. In [20], this link is elegantly expressed by using the hiding operation $\exists_x$ and only one fresh variable. As this is a deterministic operation the transition probability is one.

2.3 Observables

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 2. *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 $A_0 = A$, $c_0 = true$, $A_n = \mathbf{stop}$ 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*.

Definition 3. *Let π be a computational path for A*

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

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

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

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.

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

Because of nondeterminism, 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 4. *Let $S = \{\langle c_{ij}, p_{ij}\rangle\}_{i,j}$ be a (multi-)set of results, where c_{ij} denote the jth occurrence of the constraint c_i, and let $P_{c_i} = \sum_{c_i} p_{c_i}$ be the sum of all probabilities occurring in the set which are associated with c_i. The compactification of S is defined as follows:*

$$\mathcal{K}(S) \;=\; \{\langle c_i, P_{c_i}\rangle \mid P_{c_i} = \textstyle\sum_{c_i} p_{c_i} = \sum_j p_{ij}\}_i.$$

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

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

3. PROBABILISTIC CONFINEMENT

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

Table 1: The Syntax of PCCP Agents.

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

Table 2: The transition system for PCCP

3.1 Imperative Languages

In an imperative language, confinement — as formulated for example in [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 [18]:

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

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:

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

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

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

therefore, h is probabilistically not confined.

3.2 Declarative Languages

Because of the different role played by variables in imperative and declarative programming, in [4] the authors introduce the notion of *identity confinement*, which refers to the observables of the agents rather than to variables. According to this notion two agents are identity confined if their (probabilistic) observables are the same when run in the context of any spy. Consider the following two PCCP agents:

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

$$B \equiv \mathbf{tell}(c \sqcup d).$$

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:

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

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 notions we discussed above are *exact* in the sense that they all refer — even in their probabilistic version — to some *identity* notion among the agents' behaviour (probabilistic bisimulation , observational equivalence, etc.).

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 look at the distance between the distributions representing the agents' observables instead of their identity.

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, to which we 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:

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

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

A generalisation of this class is to consider active spies (e.g. Example 1 and Example 2) and/or spies with memory (e.g. $\mathbf{ask}(c) \rightarrow p : \mathbf{ask}(d) \rightarrow q : \mathbf{tell}(f)$).

Example 1. *Consider the two agents:*

$$\begin{array}{rcl} A & \equiv & \mathbf{ask}(c) \rightarrow 1 : \mathbf{tell}(d) \\ B & \equiv & \mathbf{stop}. \end{array}$$

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 $S \equiv \mathbf{tell}(c)$ it is easy to determine if it is being executed in parallel with A or B.*

4.2 Approximate Confinement

We introduce the notion of approximate confinement as a generalisation of the identity confinement defined in [4]. 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 $p : A \parallel q : S$ and $p : B \parallel q : S$ is smaller than ε. We consider as a measure for this distance the supremum norm $\|.\|_\infty$ formally defined as

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

where $(x_i)_{i\in I}$ represents a probability distribution.

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.

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

Example 2. *[2] Consider a cash dispenser ATM (Automatic Teller Machine) accepting only a single PIN (Personal Identification Number) n out of m possible PINs, e.g. $m = 10000$ in the UK.*

$$\begin{array}{rcl} ATMn & \equiv & \left(\square_{i=1, i\neq n}^{m} \mathbf{ask}(PINi) \rightarrow 1 : \mathbf{tell}(alarm) \right) \\ & & \square\ \mathbf{ask}(PINn) \rightarrow 1 : \mathbf{tell}(cash) \end{array}$$

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

The (active) spy S tries a randomly chosen PIN number i:

$$S \equiv \square_{i=1}^{m} \mathbf{ask}(true) \rightarrow 1 : \mathbf{tell}(PINi)$$

If we consider two such machines $ATMn_1$ and $ATMn_2$ for any $n_1 \neq n_2$ and execute them in context S we obtain the following observables:

$$\begin{array}{rl} \mathcal{O}(p : ATMn_1 \parallel q : S) = & \left\{ \left\langle PINn_1 \sqcup cash, \frac{1}{m} \right\rangle \right\} \\ \cup & \bigcup_{i=1, i\neq n_1}^{m} \left\{ \left\langle PINi \sqcup alarm, \frac{1}{m} \right\rangle \right\} \\ \mathcal{O}(p : ATMn_2 \parallel q : S) = & \left\{ \left\langle PINn_2 \sqcup cash, \frac{1}{m} \right\rangle \right\} \\ \cup & \bigcup_{i=1, i\neq n_2}^{m} \left\{ \left\langle PINi \sqcup alarm, \frac{1}{m} \right\rangle \right\} \end{array}$$

Clearly, $\mathcal{O}(p : ATMn_1 \parallel q : S)$ and $\mathcal{O}(p : ATMn_2 \parallel q : S)$ are not identical, thus $\{ATMn\}_n$ are not probabilistically confined. Nevertheless, the difference between the observables is

$$\|\mathcal{O}(p : ATMn_1 \parallel q : S) - \mathcal{O}(p : ATMn_1 \parallel q : S)\|_\infty = \frac{1}{m},$$

i.e. $\{ATMn\}_n$ are ε-confined with respect to $\mathcal{S} = \{S\}$ with $\varepsilon = \frac{1}{m}$. 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.

Example 3. *The two agents A and B in Example 3.2 are $\frac{1}{12}$-confined with respect to $\{C\}$.*

4.3 Statistical Interpretation of ε

Approximate confinement is strongly related to statistical concepts, in particular to so-called *hypothesis testing* [21]. Let us sketch this relation briefly.

The problem is to determine experimentally if an unknown agent X is either A or B. Assuming that A and B have the same observables if executed on their own, the only way we can obtain information about X is by running it in parallel with a spy S. In this way we obtain an experimental estimate for the observables of $X \parallel S$. We then compare this estimate with the actual observables of $A \parallel S$ and $B \parallel S$. The closer the observables of $A \parallel S$ and $B \parallel S$, i.e. the smaller the ε, the more experiments n we will have to undertake in order to discriminate between them with sufficient confidence. The relation between ε and n will be made more precise in the following.

4.3.1 Hypothesis Testing

The method to formulate a hypothesis about the identity of an unknown process X consists of two 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
$$H : \begin{cases} X = A \text{ if } \|\overline{\mathcal{O}}_n(X \parallel S) - \mathcal{O}(A \parallel S)\| \leq \\ \qquad\qquad \|\overline{\mathcal{O}}_n(X \parallel S) - \mathcal{O}(B \parallel S)\| \\ X = B \text{ otherwise} \end{cases}$$

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

To answer this question, let us assume that X is actually A. We then observe that the hypothesis H will be correct if and only if $\overline{\mathcal{O}}_n(X \parallel S) = \overline{\mathcal{O}}_n(A \parallel S)$ is closer to $\mathcal{O}(A \parallel S)$ than to $\mathcal{O}(B \parallel S)$, i.e.

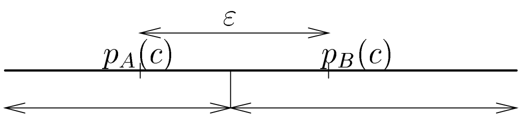
$$\begin{aligned} P(H) &= P(\|\overline{\mathcal{O}}_n(A \parallel S) - \mathcal{O}(A \parallel S)\| \\ &\qquad < \|\overline{\mathcal{O}}_n(A \parallel S) - \mathcal{O}(B \parallel S)\|) \end{aligned}$$

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

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 $\overline{\mathcal{O}}_n$ and $\mathcal{O}$ is maximal, i.e.

$$\begin{aligned} &\|\overline{\mathcal{O}}_n(A \parallel S) - \mathcal{O}(A \parallel S)\|_\infty = \\ &= \sup_d |\overline{p}^n_A(d) - p_A(d)| = |\overline{p}^n_A(c) - p_A(c)| \end{aligned}$$

We therefore can look at a simplified situation where we are considering only a 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 $\overline{p}^n_A(c)$ is anywhere to the left of $p_A(c)+\varepsilon/2$ then the hypothesis H is correct. The probability for this happening is:

$$P(\overline{p}^n_A(c) < p_A(c) + \varepsilon/2) = 1 - P(p_A(c) + \varepsilon/2 < \overline{p}^n_A(c)).$$

We aim in estimating this probability $P(\overline{p}^n_A(c) < p_A(c) + \varepsilon/2$. There are various statistical methods to estimate the probability that an experimental average is within a certain interval of the corresponding expectation value, i.e. $P(\|\overline{X}_n - \mathbf{E}(X)\|)$. These methods are essentially all based on the *central limit theorem*, e.g. [1, 9, 21]. In our case the expectation value $\mathbf{E}(X)$ corresponds to the observable $p_A(c)$ and the experimental average $\overline{X}_n$ is represented by $\overline{p}^n_A(c)$.

The type of tests we consider here are described by so called *Bernoulli Trials*. The central limit theorem for this type of test [9, Thm 9.2], gives us an estimate for the probability that an experimental value S_n is in an interval $[a, b]$:

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

where

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

Our situation is described by the following identifications:

$$\begin{aligned} p &= p_A(c) \\ q &= 1 - p_A(c) \\ S_n &= n\overline{p}^n_A \\ a &= p_A(c) + \varepsilon/2 \\ b &= \infty. \end{aligned}$$

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

$$\lim_{n\to\infty} P((p_A(c) + \varepsilon/2) \leq \overline{p}^n_A(c) \leq \infty)$$

The following approximation is common in statistics:

$$\begin{aligned} P(H) &= 1 - P((p_A(c) + \varepsilon/2) \leq \overline{p}^n_A(c)) \\ &\approx 1 - \lim_{n\to\infty} P((p_A(c) + \varepsilon/2) \leq \overline{p}^n_A(c)) \\ &= 1 - \int_{a_0}^{\infty} \exp(-x^2/2) \end{aligned}$$

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 $P(H)$ is by minimising the integral over the so called *standard normal density*. In order to do this we have to increase the lower bound a_0 of the integral which can be achieved 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, $P(H)$. 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

5.1 A Collecting Semantics

We will base our analysis on a collecting semantics consisting of distributions over configurations. Such distributions represent all the possible states which are reachable by a given program in one step. Formally they are defined as multi-sets of pairs:

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

where p_i represents the probability associated to the configuration $\langle A_i, c_i\rangle$.

The collecting semantics for PCCP is given in Table 3.

Given the collecting semantics of A as a sequence of distributions over configurations,

$$[\![A]\!]_{coll} = (\Phi_i)_i,$$

we can abstract the information about the agents to obtain a sequence of intermediate stores, or more precisely a sequence of distributions over stores $(\phi_i)_i$, where for all i:

$$\phi_i = \{\langle c_j, p_j\rangle\}_j^i.$$

The observables of A are then obtained as the limit of this sequence.

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

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

EXAMPLE 4. *For the two agents A and B in Example 3.2 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

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

The following analysis shows how to obtain 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 $\mathcal{S}_1$ and $\mathcal{S}_2$.

5.2.1 Spies in $\mathcal{S}_1$

Let us consider first spies in $\mathcal{S}_1$, i.e. spies of the form:

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

For spies of this form, we can actually ignore the intermediate states and concentrate only on the final one, i.e. the observables. The reason is the *monotonicity* of PCCP: if the guard of S, i.e. c is ever entailed by the store in an execution of A then it will be entailed until the final store is reached. The spy S therefore will be scheduled either with probability q each time after the step when c is first entailed, or eventually after A has reached the **stop** configuration. The fresh constraint f will thus be added to the store depending only on the fact whether c is entailed by the final store or not.

LEMMA 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:

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

5.2.2 Spies in $\mathcal{S}_2$

Let us now consider spies of the form

$$\begin{aligned}S \equiv\ & \mathbf{ask}(c_1) \rightarrow p_1 : \mathbf{tell}(f_1) \\ & [\!]\ \mathbf{ask}(c_2) \rightarrow p_2 : \mathbf{tell}(f_2),\end{aligned}$$

with $p_1 + p_2 = 1$.

In the construction of $\mathcal{O}(p : A \parallel q : S)$, we must now consider the intermediate states. We will use the following notations. Given a PCCP agent A and its observables $\mathcal{O}(A) = \{\langle d_i, p_i\rangle\}$ we denote by $\pi_{ik} \in \mathrm{Comp}(A)$ all the computational paths leading to d_i, and by p_{ik} their associated probability.

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

$$\begin{aligned}S \equiv\ & \mathbf{ask}(c_1) \rightarrow p_1 : \mathbf{tell}(f_1) \\ & [\!]\ \mathbf{ask}(c_2) \rightarrow p_2 : \mathbf{tell}(f_2).\end{aligned}$$

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:

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

where

$$\begin{aligned}r_i^1 &= p_i \cdot (((\sum_{l=0}^{n_i-1} p^{m_i+l})q + p^{m_i+n_i})p_1 + (1 - p^{m_i})) \\ r_i^2 &= p_i \cdot ((\sum_{l=0}^{n_i-1} p^{m_i+l})q + p^{m_i+n_i})p_2,\end{aligned}$$

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 Lemma 2 the following holds for all i:*

$$p_i = r_i^1 + r_1^2.$$

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 following proposition holds.

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

$$\begin{aligned}S \equiv\ & \mathbf{ask}(c_1) \rightarrow p_1 : \mathbf{tell}(f_1) \\ & [\!]\ \mathbf{ask}(c_2) \rightarrow p_2 : \mathbf{tell}(f_2).\end{aligned}$$

Suppose the observables of A are $\mathcal{O}(A) = \{\langle d_i, p_i\rangle\}_i$.

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	$\{\ldots, \langle\langle \square_{i=1}^{n} \mathbf{ask}(c_i) \rightarrow p_i : A_i, d\rangle, q\rangle, \ldots\} \longrightarrow \{\ldots, \langle\langle A_j, d\rangle, q \cdot \tilde{p}_j\rangle, \ldots\}$	$j \in [1,n]$ and $d \vdash c_j$
R3	$\{\ldots, \langle\langle \Vert_{i=1}^{n}\ p_i : A_i, d\rangle, q\rangle, \ldots\} \longrightarrow \{\ldots, \langle\langle A'_j, d'\rangle, q \cdot \tilde{p}_j\rangle, \ldots\}$	$j \in [1,n]$ 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

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

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

Table 4: The Collecting Semantics for A and B.

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

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

with

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

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}.

Obviously $p_i = r_i^1 + r_1^2$ holds in the general case too.

5.2.3 Limit Analysis

Starting from the formulas

$$r_i^1 = p_i \cdot (p_1(q \sum_{l=0}^{n_i - 1} p^{m_i + l} + p^{m_i + n_i}) + (1 - p^{m_i})),$$

and

$$r_i^2 = p_i \cdot p_2(q(\sum_{l=0}^{n_i - 1} p^{m_i + l} + p^{m_i + n_i})$$

consider the difference

$$|r_i^1 - r_i^2| = p_i \cdot [q \cdot \sum_{l=0}^{n_i} p^{m_i + l} \cdot |p_1 - p_2| + (1 - p^{m_i})].$$

This difference is maximal when $|p_1 - p_2|$ is maximal, namely when $|p_1 - p_2|$ tends to 1. Therefore the “best” spy is obtained by letting p_1 tend to 0 and p_2 tend to 1, or vice-versa.

In the case $\lim p_1 = 0$ and $\lim p_2 = 1$ 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 $\lim p_1 = 1$ and $\lim p_2 = 0$ 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.

6. APPLICATIONS AND EXAMPLES

The previous analysis allows us to address several questions such as: (1) Is an agent $S \in \mathcal{S}_n$ indeed able to distinguish between two agents A and B? (2) How effective is this distinction, i.e. how large is the corresponding ε? (3) What is the strongest, i.e. most effective, spy in $\mathcal{S}$ for A and B?

6.1 Properties of a Spy

An agent S is a spy for two agents A and B (or in general a set of more than two agents) if and only if the observables $\mathcal{O}(p : A \parallel q : S)$ are different from $\mathcal{O}(p : B \parallel q : S)$. 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*

$$\begin{aligned} S \quad \equiv \quad & \mathbf{ask}(c_1) \to p_1 : \mathbf{tell}(f_1) \\ & [\!] \ \mathbf{ask}(c_2) \to p_2 : \mathbf{tell}(f_2), \end{aligned}$$

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. $r^1_j(A) \neq r^1_j(B)$.

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

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 $0 < p < 1$.*

6.2 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. 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*

$$\begin{aligned} S \quad \equiv \quad & \mathbf{ask}(c_1) \to p_1 : \mathbf{tell}(f_1) \\ & [\!] \ \mathbf{ask}(c_2) \to p_2 : \mathbf{tell}(f_2). \end{aligned}$$

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

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

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

EXAMPLE 5. *We compute the effectiveness of the spy C for the two agents A and B in Example 3.2. To this purpose, we first compute $r^c_{c\sqcup d}$ and $r^d_{c\sqcup d}$, using Lemma 2 and Proposition 2. The notation $r^c_{c\sqcup d}$ ($r^d_{c\sqcup d}$) is the same as r^1_j (r^2_j), where we use the constraint in place of its index, the guard first entailed instead of 1, and the guard last entailed instead of 2.*

A: *For the agent A we have two computational paths from store true to $c \sqcup d$ each with probability $p_{c\sqcup d,i} = \frac{1}{2}$.*

1. *In the path true $\to c \to c \sqcup d$, c is told first, i.e. $c_1 = c$, $c_2 = d$. Since $m_1 = 1$ and $n_1 = 0$, we get:*

$$\begin{aligned} r^c_{c\sqcup d,1} &= \frac{1}{2}((0 + p^{1+0})p_1 + (1 - p^1)) \\ &= \frac{1}{2}(1 - p + pp_1) \\ r^d_{c\sqcup d,1} &= \frac{1}{2}(0 + p^{1+0})p_2 = \frac{1}{2}pp_2. \end{aligned}$$

We know that for the spy C, $p_1 = \frac{2}{3}$ and $p_2 = \frac{1}{3}$. Assuming the uniform scheduling $p = q = \frac{1}{2}$ we finally get:

$$r^c_{c\sqcup d,1} = \frac{1}{2}(1 - \frac{1}{2} + \frac{1}{2}\frac{2}{3}) = \frac{5}{12}$$

and

$$r^d_{c\sqcup d,1} = \frac{1}{2}\frac{1}{2}\frac{1}{3} = \frac{1}{12}.$$

2. *In path true $\to d \to c \sqcup d$, as d is told before c we have that $c_1 = d$ and $c_2 = c$. Since $m_1 = 1$ and $n_1 = 0$, we get:*

$$\begin{aligned} r^d_{c\sqcup d,2} &= \frac{1}{2}((0 + p^{1+0})p_2 + (1 - p^1)) \\ &= \frac{1}{2}(1 - p + pp_2) \\ r^c_{c\sqcup d,2} &= \frac{1}{2}(0 + p^{1+0})p_1 = \frac{1}{2}pp_1, \end{aligned}$$

and finally:

$$r^d_{c\sqcup d,2} = \frac{1}{2}(1 - \frac{1}{2} + \frac{1}{2}\frac{1}{3}) = \frac{4}{12}$$

and

$$r^c_{c\sqcup d,2} = \frac{1}{2}\frac{1}{2}\frac{2}{3} = \frac{2}{12}.$$

By Proposition 2 the probabilities for $c \sqcup d \sqcup e$ and $c \sqcup d \sqcup f$ are:

$$r^c_{c\sqcup d} = \frac{5}{12} + \frac{2}{12} = \frac{7}{12} \text{ and } r^d_{c\sqcup d} = \frac{1}{12} + \frac{4}{12} = \frac{5}{12}$$

B: *Here we have only one path: true $\to c \sqcup d$ where c and d appear simultaneously in the final store, i.e. $m = n = 0$ and $p_{c\sqcup d} = 1$. Therefore,*

$$\begin{aligned} r^c_{c\sqcup d,1} &= ((0 + p^{0+0})p_1 + (1 - p^0)) \\ &= ((0 + 1)p_1 + (1 - 1)) = p_1 \\ r^d_{c\sqcup d,1} &= (0 + p^{0+0})p_2 = (0 + 1)p_2 = p_2 \end{aligned}$$

For p_1 and p_2 as the choice distribution in C and the uniform scheduling priorities we obtain (as expected) the probabilities

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

We can now compute the effectiveness ε of the spy C, as:

$$\begin{aligned} \varepsilon &= |r^c_{c\sqcup d}(A) - r^c_{c\sqcup d}(B)| = |r^d_{c\sqcup d}(A) - r^d_{c\sqcup d}(B)| \\ &= |\frac{8}{12} - \frac{7}{12}| = |\frac{4}{12} - \frac{5}{12}| = \frac{1}{12}. \end{aligned}$$

This also shows that A and B are $\frac{1}{12}$-confined with respect to $\mathcal{S} = \{C\}$.

6.3 The Most Effective Spy

The limit analysis in Section 5.2.3 shows that the most effective spy for a fixed pair of guards (c_1, c_2) is obtained by considering a choice distribution where $\lim p_1 = 0$ and $\lim p_2 = 1$ 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 3.2. 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 $\lim p_1 = 0$ and $\lim p_2 = 1$. 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_i^1 and r_i^2 in Section 5.2.3.

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:*

$$\begin{aligned} r^c_{c\sqcup d,1} &= \frac{1}{2}(1-p) = \frac{1}{2}(1-\frac{1}{2}) = \frac{1}{4} \\ r^d_{c\sqcup d,1} &= \frac{1}{2}p = \frac{1}{4} \end{aligned}$$

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

$$\begin{aligned} r^d_{c\sqcup d,2} &= \frac{1}{2}1 = \frac{1}{2} \\ r^c_{c\sqcup d,2} &= \frac{1}{2}0 = 0. \end{aligned}$$

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:*

$$\begin{aligned} r^c_{c\sqcup d,1} &= 1(1-1) = 0 \\ r^d_{c\sqcup d,1} &= 1 \cdot 1 = 1 \end{aligned}$$

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 $\mathcal{S}_2$.

7. ANALYSIS: ABSTRACT SEMANTICS

The use of an exact (collecting) semantics makes the analysis presented in the previous section 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.

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

Table 5: The Analysis for PCCP Agents

7.1 Abstract Semantics

We associate to each agent a set of tuples. Each tuple $\langle c, d, p, t\rangle$ consists of two constraints, a probability and a time stamp. It represents a transition to store d, which takes place with probability p provided that the current store 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 includes the guard and the time is increased by 1.

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

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:

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

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

In general it is impossible to limit the recursion depth of a procedure call. We therefore model it by fixing $t = \infty$ for all procedure calls:

$$\infty([\![X]\!]) = \{\langle d, c, p, \infty\rangle \mid \langle d, c, p, t\rangle \in X\}.$$

This leads to a substantial over-approximation of t.

7.2 Security Analysis

By a further abstraction of the denotation of a program we eliminate from the tuples the dependency store (first component):

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

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:

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

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

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

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

$$\overline{\mathcal{A}}(c_1,c_2)=\bigcup_{s\in paths(c_1,c_2')}\{\langle\overline{p}_s,\overline{m}_s\rangle\}.$$

7.3 Correctness of the Analysis

For each pair of constraints c_1 and c_2, the concrete semantics allows us to calculate the set $\mathcal{A}(c_1,c_2)=\{\langle p_i,m_i\rangle\}_i$ of all pairs $\langle p_i,m_i\rangle$, such that for each computational path π_i, 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 † operation implies that

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

On the other hand, for each i, the operation max used in the definition of the sequential and the ∞ defining the procedure call in the abstract semantics imply that

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

EXAMPLE 7. *Consider the following agent:*

$$\begin{aligned}A \;\equiv\;& \frac{1}{2}:(\frac{1}{2}:(\frac{1}{2}:\mathbf{tell}(c)\parallel\frac{1}{2}:\mathbf{tell}(d)) \\ &[]\ \frac{1}{2}:(\mathbf{tell}(c\sqcup d)))\parallel\frac{1}{2}:(\mathbf{tell}(e)).\end{aligned}$$

Comp(A) *contains two paths from true to* $c\sqcup d\sqcup e$ *of* 3 *steps and two paths of* 4 *steps. The concrete analysis therefore gives us:*

$$\mathcal{A}(true,c\sqcup d\sqcup e)=\left\{\left\langle\frac{1}{4},3\right\rangle,\left\langle\frac{1}{4},4\right\rangle,\left\langle\frac{1}{4},3\right\rangle,\left\langle\frac{1}{4},4\right\rangle\right\},$$

while the abstract one overestimates the length of one of these paths when the last sequential composition is resolved:

$$\overline{\mathcal{A}}(true,c\sqcup d\sqcup e)=\left\{\left\langle\frac{1}{4},4\right\rangle,\left\langle\frac{1}{4},4\right\rangle,\left\langle\frac{1}{4},4\right\rangle,\left\langle\frac{1}{4},4\right\rangle\right\}.$$

From the discussion above, we can conclude that

$$r^2_{ik}=p_{ik}p^{m_{ik}}\;\le\;\overline{p}_{ik}p^{\overline{m}_{ik}}=\overline{r}^2_{ik}$$

and furthermore that

$$r^2_i=\sum_k r^2_{ik}\;\le\;\sum_k\overline{r}^2_{ik}=\overline{r}^2_i.$$

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:

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

and finally:

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

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

$$\begin{aligned}S\;\equiv\;&\mathbf{ask}(c_1)\to p_1:\mathbf{tell}(f_1)\\ &[]\ \mathbf{ask}(c_2)\to p_2:\mathbf{tell}(f_2).\end{aligned}$$

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

$$\varepsilon\le\overline{\varepsilon}.$$

PROOF.

$$\begin{aligned}\varepsilon &= \max_i|r^2_i(A)-r^2_i(B)|\\ &\le \max_i(|r^2_i(A)|+|r^2_i(B)|)\\ &= \max_i(r^2_i(A)+r^2_i(B))\\ &\le \max_i(\overline{r}^2_i(A)+\overline{r}^2_i(B))\\ &= \overline{\varepsilon}.\end{aligned}$$

□

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 $\overline{r}^2_i$.

8. CONCLUSIONS

We have introduced a new notion of confinement, namely ε-confinement, which measures the difference in a pair of processes with respect to some spies. The ε measure is closely related to the number of tests that are required to differentiate the processes. This is markedly different from other approaches in the literature which tend to be expressed in terms of equality or equivalence. We have illustrated this new notion by example and we have also introduced a static analysis for PCCP which approximates the property.

In this paper we have restricted our study to passive and memoryless attackers. Within this context, it is possible to develop the concept of most effective attacker. In future work, we will develop these ideas to consider more powerful classes of attacker.

The proofs of the main results in this paper exploit the fact that we are working with PCCP. In PCCP the constraint store changes monotonically. In more conventional languages, the store changes in an erratic way and we would need a more detailed analysis to account for this. Nevertheless, we are confident that the general principles behind our framework are applicable to such languages and we will confirm this in future work.

9. 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] 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.

[4] 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. ISBN: 044450916X.

[5] 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.

[6] 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.

[7] Alessandra Di Pierro and Herbert Wiklicky. Quantitative observables and averages in Probabilistic Concurrent Constraint Programming. In K.R. Apt, T. Kakas, E. Monfroy, and F. Rossi, editors, *New Trends in Constraints — Selected Papers of the ERCIM/Compulog Workshop on Constraints, October 1999, Paphos, Cyprus*, number 1865 in Lecture Notes in Computer Science, Berlin — Heidelberg — New York, 2000. Springer Verlag.

[8] Center for High Assurance Computing Systems. The Navy Handbook for the computer security certification of trusted systems, 1998. Draft version on http://chacs.nrl.navy.mil/main.html.

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

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

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

[12] K. Rustan M. Leino and Rajeev Joshi. A semantic approach to secure information flow. SRC Technical Note 1997-032, Digital Systems Research Center, Palo Alto CA, December 1997.

[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] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer Verlag, Berlin – Heidelberg, 1999.

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

[25] 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.

Appendix
Proof of Lemma 1

Proof. Given the collecting semantics of A, $[\![A]\!]_{coll} = (\Phi_j)_j$ with:

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

we construct the collecting semantics of $p : A \parallel q : S$, as a sequence $[\![p : A \parallel q : S]\!]_{coll} = (\overline{\Phi}_j)_j$.

As long as no intermediate store c_i^j entails c we have

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

Suppose now that there is a distribution Φ_k where c is entailed the first time for some c_i^k, i.e.

$$\Phi_k = \left\{ \dots, \left\langle \left\langle A_i^k, c_i^k \right\rangle, p_i^k \right\rangle, \dots \right\}$$

with $c_i^k \vdash c$ for some i. Then the next distribution in the execution of $p : A \parallel q : S$ is given by:

$$\begin{array}{rcl} \overline{\Phi}_{k+1} & = & \{ \dots, \\ & & \left\langle \left\langle p : A_i^{k+1} \parallel q : S, c_i^{k+1} \right\rangle, p_i^{k+1} \cdot p \right\rangle, \\ & & \left\langle \left\langle p : A_i^k \parallel q : \mathbf{stop}, c_i^k \sqcup f \right\rangle, p_i^k \cdot q \right\rangle, \\ & & \dots \} \end{array}$$

as we have now the choice of scheduling again A (with probability p) or the spy S (with probability q). The next distribution is then given by:

$$\begin{array}{rcl} \overline{\Phi}_{k+2} & = & \{ \dots, \\ & & \left\langle \left\langle p : A_i^{k+2} \parallel q : S, c_i^{k+2} \right\rangle, p_i^{k+2} \cdot p^2 \right\rangle, \\ & & \left\langle \left\langle p : A_i^{k+1} \parallel q : \mathbf{stop}, c_i^{k+1} \sqcup f \right\rangle, p_i^{k+1} \cdot pq \right\rangle, \\ & & \left\langle \left\langle p : A_i^{k+1} \parallel q : \mathbf{stop}, c_i^{k+1} \sqcup f \right\rangle, p_i^{k+1} \cdot q \right\rangle, \\ & & \dots \} \\ & = & \{ \dots, \\ & & \left\langle \left\langle p : A_i^{k+2} \parallel q : S, c_i^{k+2} \right\rangle, p_i^{k+2} \cdot p^2 \right\rangle, \\ & & \left\langle \left\langle p : A_i^{k+1} \parallel q : \mathbf{stop}, c_i^{k+1} \sqcup f \right\rangle, p_i^{k+1} \cdot (q + pq) \right\rangle, \\ & & \dots \} \end{array}$$

The monotonicity of PCCP guarantees that once c is entailed it will continue to be so, i.e. $c_i^k \vdash c$ implies $c_i^{k+1} \vdash c$, etc.

In general we get in m steps after c was first entailed:

$$\begin{array}{rcl} \overline{\Phi}_{k+m} & = & \{ \dots, \\ & & \left\langle \left\langle p : A_i^{k+m} \parallel q : S, c_i^{k+m} \right\rangle, p_i^{k+m} \cdot p^m \right\rangle, \\ & & \left\langle p : A_i^{k+m-1} \parallel q : \mathbf{stop}, c_i^{k+m-1} \sqcup f, \right. \\ & & \quad \left. p_i^{k+m-1} \cdot q \sum_{l=0}^{m-1} p^l \right\rangle, \\ & & \dots \} \\ & = & \{ \dots, \\ & & \left\langle \left\langle p : A_i^{k+m} \parallel q : S, c_i^{k+m} \right\rangle, p_i^{k+m} \cdot p^m \right\rangle, \\ & & \left\langle \left\langle p : A_i^{k+m-1} \parallel q : \mathbf{stop}, c_i^{k+m-1} \sqcup f \right\rangle, \right. \\ & & \quad \left. p_i^{k+m-1} \cdot (1 - p^m) \right\rangle \\ & & \dots \} \end{array}$$

as for $p = 1 - q$ the following identity holds:

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

□

Proof of Lemma 2

PROOF. The first three terms of the above 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 get **tell**ed. The final configuration in this case is thus $\langle p : \mathbf{stop} \parallel q : S, d_i \rangle$ 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 $\langle p : \mathbf{stop} \parallel \mathbf{stop}, d_i \sqcup f_1 \rangle$ and $\langle p : \mathbf{stop} \parallel \mathbf{stop}, d_i \sqcup f_2 \rangle$ 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 $\left\langle p : A_i^k \parallel q : S, c_i^k \right\rangle$ with non-zero probability p_i^k in Φ_k such that $c_i^k \vdash c_1$ (and not yet $c_i^k \vdash c_2$):

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

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

$$\begin{array}{rcl} \overline{\Phi}_{k+1} & = & \{ \dots, \\ & & \left\langle \left\langle p : A_i^{k+1} \parallel q : S, c_i^{k+1} \right\rangle, p_i^{k+1} \cdot p \right\rangle, \\ & & \left\langle \left\langle p : A_i^k \parallel q : \mathbf{stop}, c_i^k \sqcup f_1 \right\rangle, p_i^k \cdot q \right\rangle, \\ & & \dots \}. \end{array}$$

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:

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

$j = k+m+1$**:** 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 $\mathbf{tell}(f_1)$ or (2b) executing the second one, $\mathbf{tell}(f_1)$, as both guards are entailed. The probabilities that this is happening are p for the case (1), and qp_1 and qp_2 for cases (2a) and (2b), respectively. The second agent $p : A_i^{k+m-1} \parallel q : \mathbf{stop}$ has to continue quasi-deterministically with A_i^{k+m-1}.

$$\begin{aligned}\overline{\Phi}_{k+m+1} &= \{\ldots, \\ &\quad \left\langle \left\langle p : A_i^{k+m+1} \parallel q : S, c_i^{k+m+1} \right\rangle, \right. \\ &\quad\quad \left. p_i^{k+m+1} \cdot p^{m+1} \right\rangle, \\ &\quad \left\langle \left\langle p : A_i^{k+m} \parallel q : \mathbf{stop}, c_i^{k+m} \sqcup f_1 \right\rangle, \right. \\ &\quad\quad \left. p_i^{k+m} \cdot p^m qp_1 \right\rangle, \\ &\quad \left\langle \left\langle p : A_i^{k+m} \parallel q : \mathbf{stop}, c_i^{k+m} \sqcup f_2 \right\rangle, \right. \\ &\quad\quad \left. p_i^{k+m} \cdot p^m qp_2 \right\rangle, \\ &\quad \left\langle \left\langle p : A_i^{k+m} \parallel q : \mathbf{stop}, c_i^{k+m} \sqcup f_1 \right\rangle, \right. \\ &\quad\quad \left. p_i^{k+m} \cdot (1-p^m) \right\rangle, \\ &\quad \ldots\} \\ &= \{\ldots, \\ &\quad \left\langle \left\langle p : A_i^{k+m+1} \parallel q : S, c_i^{k+m+1} \right\rangle, \right. \\ &\quad\quad \left. p_i^{k+m+1} \cdot p^{m+1} \right\rangle, \\ &\quad \left\langle \left\langle p : A_i^{k+m} \parallel q : \mathbf{stop}, c_i^{k+m} \sqcup f_1 \right\rangle, \right. \\ &\quad\quad \left. p_i^{k+m} \cdot (p^m qp_1 + (1-p^m)) \right\rangle, \\ &\quad \left\langle \left\langle p : A_i^{k+m} \parallel q : \mathbf{stop}, c_i^{k+m} \sqcup f_2 \right\rangle, \right. \\ &\quad\quad \left. p_i^{k+m} \cdot p^m qp_2 \right\rangle, \\ &\quad \ldots\}.\end{aligned}$$

$j = k+m+n$**:** After further n steps the agent A is finished [we assume it **stop**ed although if it had deadlocked it would be pretty much the same].

$$\begin{aligned}&\overline{\Phi}_{k+m+n} = \\ &\quad = \{\ldots, \\ &\quad\quad \left\langle \left\langle p : \mathbf{stop} \parallel q : S, c_i^{k+m+n} \right\rangle, \right. \\ &\quad\quad\quad \left. p_i^{k+m+n} \cdot p^{m+n} \right\rangle, \\ &\quad\quad \left\langle \left\langle p : A_i^{k+m+n-1} \parallel q : \mathbf{stop}, c_i^{k+m+n-1} \sqcup f_1 \right\rangle, \right. \\ &\quad\quad\quad \left. p_i^{k+m+n-1} \cdot ((\sum_{l=0}^{n-1} p^{m+l})qp_1 + (1-p^m)) \right\rangle, \\ &\quad\quad \left\langle \left\langle p : A_i^{k+m+n-1} \parallel q : \mathbf{stop}, c_i^{k+m+m-1} \sqcup f_2 \right\rangle, \right. \\ &\quad\quad\quad \left. p_i^{k+m+n-1} \cdot (\sum_{l=0}^{n-1} p^{m+l})qp_2 \right\rangle, \\ &\quad\quad \ldots\}.\end{aligned}$$

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

$$\begin{aligned}&\overline{\Phi}_{k+m+n+1} = \\ &\quad = \{\ldots, \\ &\quad\quad \left\langle \left\langle p : \mathbf{stop} \parallel q : \mathbf{stop}, c_i^{k+m+n} \sqcup f_1 \right\rangle, \right. \\ &\quad\quad\quad \left. p_i^{k+m+n} \cdot p^{m+n} p_1 \right\rangle, \\ &\quad\quad \left\langle \left\langle p : \mathbf{stop} \parallel q : \mathbf{stop}, c_i^{k+m+n} \sqcup f_2 \right\rangle, \right. \\ &\quad\quad\quad \left. p_i^{k+m+n} \cdot p^{m+n} p_2 \right\rangle, \\ &\quad\quad \left\langle \left\langle p : \mathbf{stop} \parallel q : \mathbf{stop}, c_i^{k+m+n} \sqcup f_1 \right\rangle, \right. \\ &\quad\quad\quad \left. p_i^{k+m+n} \cdot ((\sum_{l=0}^{n-1} p^{m+l})qp_1 + (1-p^m)) \right\rangle, \\ &\quad\quad \left\langle \left\langle p : \mathbf{stop} \parallel q : \mathbf{stop}, c_i^{k+m+m} \sqcup f_2 \right\rangle, \right. \\ &\quad\quad\quad \left. p_i^{k+m+n} \cdot (\sum_{l=0}^{n-1} p^{m+l})qp_2 \right\rangle, \\ &\quad\quad \ldots\} \\ &\quad = \{\ldots, \\ &\quad\quad \left\langle \left\langle p : \mathbf{stop} \parallel q : \mathbf{stop}, c_i^{k+m+n} \sqcup f_1 \right\rangle, \right. \\ &\quad\quad\quad \left. p_i^{k+m+n} \cdot (((\sum_{l=0}^{n-1} p^{m+l})q + p^{m+n})p_1 + (1-p^m)) \right\rangle, \\ &\quad\quad \left\langle \left\langle p : \mathbf{stop} \parallel q : \mathbf{stop}, c_i^{k+m+m} \sqcup f_2 \right\rangle, \right. \\ &\quad\quad\quad \left. p_i^{k+m+n} \cdot ((\sum_{l=0}^{n-1} p^{m+l})q + p^{m+n})p_2 \right\rangle, \\ &\quad\quad \ldots\}.\end{aligned}$$

□

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, by (1) and since $p_2 = 1-p_1$, we have that:

$$\begin{aligned} r_i^1 + r_i^2 &= p_i \cdot [p_1(q\sum_{l=0}^{n-1} p^{m+l} + p^{m+n}) + \\ &\quad +1 - p^m + p_2(q\sum_{l=0}^{n-1} p^{m+l} + p^{m+n})] \\ &= p_i[1 - p^m + q\sum_{l=0}^{n-1} p^{m+l} + p^{m+n}] \\ &= p_i[1 - p^m + p^m - p^{m+n} + p^{m+n}] \\ &= p_i. \end{aligned}$$

□