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,herbert}@doc.ic.ac.uk

Corresponding Author:
Alessandra Di Pierro
Dipartimento di Informatica - Università di Pisa
Via F. Buonarroti, 2 – 56127 Pisa (Italy)
Tel.: +39 0502212779
Fax: +39 0502212726
e-mail:dipierro@di.unipi.it

Abstract

We address the problem of characterising the security of a program against unauthorised information flows. Classical approaches are based on non-interference models which depend ultimately on the notion of process equivalence. In these models confidentiality is an absolute property stating the absence of any illegal information flow. We present a model in which the notion of non-interference is approximated in the sense that it allows for some exactly quantified leakage of information. This is characterised via a notion of process similarity which replaces the indistinguishability of processes by a quantitative measure of their behavioural difference. Such a quantity is related to the number of statistical tests needed to distinguish two behaviours. We also present two semantics-based analyses of approximate non-interference and we show that one is a correct abstraction of the other.

1 Introduction

Non-interference was introduced by Goguen and Meseguer in their seminal paper [14] 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 [27]. A classification of security properties based on different notions of behavioural equivalence is given in [12, 13]. For systems where nondeterminism 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 aspect 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 [37].

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 [15]. 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 [33, 38, 34] where a type-system based security analysis is developed. Another recent contribution to this problem is the work in [28, 29], 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 [26]. One of these 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" [39]. 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 [26].

The key idea of our approach is to replace *indistinguishability* by *similarity* in the basic formalisation of non-interference. As a result, two behaviours though distinguishable might still be considered 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 a discussion of related works and 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), which was introduced in [9, 10] as a probabilistic version of the Concurrent Constraint Programming (CCP) paradigm [31, 30]. 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 [31, 6] for more details), which encodes the information ordering. This is referred to as the *entailment* relation $\vdash$ and is sometimes denoted by $\sqsupseteq$. A cylindric constraint system includes constraints of the form $\exists_x c$ (cylindric elements) to model *hiding* of local variables, and constraints of the form d_{xy} (diagonal elements) to model *parameter passing*. The axioms of the constraint system include laws from the theory of cylindric algebras [17] which model the cylindrification operators $\exists_x$ as a kind of first-order existential quantifiers, and the diagonal elements d_{xy} as the equality between x and y.

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

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

2.2 Operational Semantics

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

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

The rules of Table 2 are closely related to the ones for nondeterministic CCP, and we refer to [6] 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). This means that we have to re-define the probability distribution so that only enabled/active agents have non-zero probabilities and the sum of these probabilities is one. The probability after normalisation is denoted by $\tilde{p}_j$. For example, in rule **R2** the normalised transition probability can be defined for all enabled agents by

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

where the sum $\sum_{\vdash c_j} p_j$ is over all enabled agents. When there are no enabled agents normalisation is not necessary. We treat a zero probability in the same way as a non-entailed guard, i.e. agents with zero probability are not enabled; this guarantees that normalisation never involves a division by a zero value. Analogous considerations apply to the normalisation of active agents in **R3**. It might be interesting to note that there are alternative ways to deal with the situation where $\sum_{\vdash c_j} p_j = 0$ (all enabled agents have probability zero). In [11] normalisation is defined in this case as the assignment of a uniform distribution on the enabled agents; such a normalisation procedure allows, for example, to introduce a quasi-sequential composition.

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

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

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

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:

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

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

Operationally, a distribution $\mathcal{O}(A, d)$ corresponds to the set of all pairs $\langle c, p\rangle$, where c is the result of a computation of A starting in store d and p is the probability of computing that result. For the purpose of this paper we will restrict to agents which only exhibit computations whose length is bounded. Note that since our programs are finitely branching this implies by the König's lemma that the vector space of constraints is finite-dimensional. We will also exclude those situations in which the final configuration does not correspond to successful termination and yet no transitions are possible, i.e. the case of *suspended computations*. These assumptions allow for a simpler presentation of the ideas and techniques at the base of the analysis proposed later on. The results presented can nevertheless be extended to the general case by considering an appropriate topology on the (possibly infinite-dimensional) vector space of constraints and appropriate adjustments to the analysis techniques so as to account for deadlock.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3 Identity Confinement

The original idea of non-interference as stated in [14] 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 [29] where the setting is that of imperative languages. This example also shows the difference between non-deterministic and probabilistic (identity) confinement.

Example 1 *In an imperative language, confinement — as formulated for example in [28, 29] — 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:

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

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:

- *For* $h \; mod \; 2 = 0$: $\{l = 0, l = 1\}$
- *For* $h \; mod \; 2 = 1$: $\{l = 1, l = 0\}$,

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:*

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

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

Example 2 *We can re-formulate the situation above in our declarative setting by considering the following agents:*

$$\begin{array}{rcl}
\texttt{hOn} & \equiv & \mathbf{ask}(true) \rightarrow \dfrac{1}{2} : \mathbf{tell}(\texttt{on}) \;\square\; \mathbf{ask}(true) \rightarrow \dfrac{1}{2} : \texttt{Rand} \\
\texttt{hOff} & \equiv & \mathbf{ask}(true) \rightarrow \dfrac{1}{2} : \mathbf{tell}(\texttt{off}) \;\square\; \mathbf{ask}(true) \rightarrow \dfrac{1}{2} : \texttt{Rand} \\
\texttt{Rand} & \equiv & \mathbf{ask}(true) \rightarrow \dfrac{1}{2} : \mathbf{tell}(\texttt{on}) \;\square\; \mathbf{ask}(true) \rightarrow \dfrac{1}{2} : \mathbf{tell}(\texttt{off})
\end{array}$$

The constraint system consists of four elements:

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

where $true \sqsubseteq \mathtt{on} \sqsubseteq false$ *and* $true \sqsubseteq \mathtt{off} \sqsubseteq false$.

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 $\mathcal{V}(\mathcal{C})$. Thus, the difference between two observables corresponds to the vector difference between the given observables and can be measured by means of a *norm*. We adopt here the supremum norm $\|\cdot\|_\infty$ formally defined as

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

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

Probabilistic identity confinement is then defined as follows [7]:

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

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

or equivalently,

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

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

Example 3 *It is easy to check that any context can distinguish between the agents* `hOn` *and* `hOff` *of Example 2. In fact, even if executed on their own their observables are different (cf Figure 1):*

$$\begin{aligned} \left\|\mathcal{O}(\mathtt{hOn}, true) - \mathcal{O}(\mathtt{hOff}.true)\right\| &= \\ \left\|\left\{\left\langle \mathtt{on}, \frac{3}{4}\right\rangle, \left\langle \mathtt{off}, \frac{1}{4}\right\rangle\right\} - \left\{\left\langle \mathtt{on}, \frac{1}{4}\right\rangle, \left\langle \mathtt{off}, \frac{3}{4}\right\rangle\right\}\right\| &= \frac{1}{2}. \end{aligned}$$

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

Example 4 *Consider the following two PCCP agents [7]:*

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

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

It is easy to see that in their non-deterministic versions 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 (cf. Figure 2), but they are not probabilistically confined. A context which reveals the identity of A and B is for example the agent:

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

as the executions of A and B in this context give different observables (cf. Figure 3 and Figure 4):

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

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

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

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

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

If neither c nor d entails g then D will never be executed, and the executions of $p : A \parallel q : D$ and $p : B \parallel q : D$ are essentially the same as for A and B alone (cf. Figure 2).

If only d entails g we obtain the derivations in Figure 5. The case where g is entailed by c alone is analogous. In all cases we end up with a single result $c \sqcup d \sqcup h$ with probability one.

The derivations of $p : A \parallel q : D$ and $p : B \parallel q : D$ in the case that both c and d entail g are depicted in Figure 6: Again we obtain the same result $c \sqcup d \sqcup h$ with probability one.

In general, identical behaviour in all contexts is hardly ever achievable. It therefore makes sense to ask for identical observables if A and B are executed in parallel with agents with only limited capabilities. Moreover, the power of a context can be evaluated in terms of its ability to distinguish the behaviours of two agents. It is also reasonable to think that its effectiveness will depend on the probabilities of the scheduling in the interleaving with the given agents. This leads to the definition of a weaker (and yet more practical) notion of probabilistic identity confinement which is parametric in the type of context S and the scheduling probability p. We will introduce such a notion, which we call *approximate identity confinement*, in Section 4.2.

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

Example 5 *[4] Consider an* ATM *(Automatic Teller Machine) accepting only a single* PIN *number n out of m possible* PIN*s, e.g. $m = 10000$:*

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

This agent simulates an ATM *which recognises* PINn*: if* PINn *has been told the machine dispenses cash, otherwise — for any incorrect* PINi *— it sounds an alarm. 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 ATMn_1 *and* ATMn_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 $\mathcal{O}(p : \mathrm{ATM}n_2 \parallel q : S)$:*

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

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

For most 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:*

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

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

The notion of approximate confinement we will define in the following is based on the idea of measuring how much the behaviour of two agents differs if we put them in a certain context. In the next section we will discuss different kinds of such contexts, which we will refer to as *spies* or *attackers*.

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 [22].

As an example, will discuss here a class of 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 agents are passive and memoryless attackers: They do not change the behaviour of the hosts, and are only allowed to interact with the store in one step. Nevertheless, they are sufficient for formalising quite powerful attacks such as the timing attacks in [20].

A generalisation of this class is to consider active spies (e.g. Example 6 and Example 5) and/or spies with memory such as $\mathbf{ask}(c) \rightarrow p : \mathbf{ask}(d) \rightarrow q : \mathbf{tell}(f)$.

Example 6 *Consider the two agents:*

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

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

4.2 Approximate Identity Confinement

We introduce a notion of approximate confinement which is a generalisation of the identity confinement introduced in [7] and defined in Section 3. The definition we give is parametric with respect to a set of admissible spies $\mathcal{S}$ and scheduling probabilities p and $q = 1 - p$. We say that two agents A and B are approximately confined with respect to a set of spies $\mathcal{S}$ iff there exists an $\varepsilon \geq 0$ such that for all $S \in \mathcal{S}$ the *distance* between the observables of $p : A \parallel q : S$ and $p : B \parallel q : S$ is smaller than ε. We consider as a measure for this distance the supremum norm $\|\cdot\|_\infty$ as in Definition 4. In this case, the choice of this norm is particularly appropriate because it allows us to identify a single constraint c for which the associated probabilities are maximally different. In the following we will usually omit the index ∞.

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

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

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

The number ε associated to a given class of spies $\mathcal{S}$ can be seen as a measure of the "power" of $\mathcal{S}$. In fact, it is strongly related to the number of tests a spy needs to perform in order to reveal the identity of the host agents. We will make this argument more precise in the next section. Note that this number depends on the scheduling probability. This is because the effectiveness of a spy can only be evaluated depending on the internal design of the host system which is in general not known to the spy.

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

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. [32]).

4.3.1 Identification by Testing

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

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

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

Unfortunately — as J.M. Keynes pointed out — we are all dead on the long run. The above described experimental setup is therefore only of theoretical value. For practically purposes we need a way to distinguish A and B by finite executions of $p : A \parallel q : S$ and $p : B \parallel q : S$. If we execute $p : A \parallel q : S$ and $p : B \parallel q : S$ only a finite number of — say n — times, we can observe a certain experimental frequency $p^n_A(c)$ and $p^n_B(c)$ for getting c as a result. Each time we repeat this finite sequence of 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 $p^n_X(c)$ for an unknown agent X by testing, i.e. by looking at n executions of $p : X \parallel q : S$. We can then try to compare $p^n_X(c)$ 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 $p : X \parallel q : S$, although the (long-run) probabilities of obtaining c by executing $p : A \parallel q : S$ or $p : B \parallel q : S$ are, let's say, $p_A = 0.1$ and $p_B = 0.5$. If we stop our experiment after $n = 1$ executions we get $p^1_X(c) = 0$. 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, on the basis of such experiments, how much the identification of X with A is "more correct" than identifying X with B on the basis of such experiments.

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

4.3.2 Hypothesis Testing

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

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

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

1. We execute $p : X \parallel q : S$ exactly n times in order to obtain an experimental

approximation, i.e. average, for its observables

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

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

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

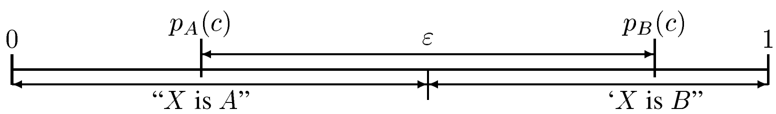
The question is now whether the guess expressed by the hypothesis H about the true identity of the black box X, which we formulate according to the above procedure, is correct; or more precisely: What is the probability that the hypothesis H holds? To do this we have to distinguish two cases or scenarios:

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

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

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

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



If the experimental value $p_X^n(c) = p_A^n(c)$ we obtained in our test is anywhere to the left of $p_A(c) + \varepsilon/2$ then the hypothesis H we formulate (based on $p_A^n(c)$) 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".

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

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

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

4.3.3 Confidence Estimation

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

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

for some $\varepsilon \geq 0$. These statistical methods are essentially all based on the *central limit theorem*, e.g. [3, 16, 32].

The type of tests we consider here to formulate a hypothesis about the identity of the unknown agent X are described in statistical terms by so called *Bernoulli Trials* which are parametric with respect to two probabilities p and $q = 1-p$ (which have nothing to do with the scheduling probabilities above). The central limit theorem for this type of tests [16, Thm 9.2] gives us an estimate for the probability that the experimental value $S_n = n \cdot X_n$ after n repetitions of the test will be in a certain interval $[a, b]$:

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

where

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

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

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

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

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

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

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

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

with

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

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

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

Example 7 *Consider the agents in Example 4. The problem is to determine from the experimentally obtained approximations of the observables* $\mathcal{O}_n\left(\frac{1}{2} : X \parallel \frac{1}{2} : C\right)$ *for* $X = A$ *or* $X = B$ *the true identity of* X*. If, for example,* X *is actually agent* A *and if we concentrate on the constraint* $c \sqcup d \sqcup e$ *we have*

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

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

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

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

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

while for 144 *tests we get:*

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

In other words, if we repeat the execution of $\frac{1}{2} : X \parallel \frac{1}{2} : C$ *exactly* 9 *times, the probability of formulating a correct hypothesis* H *about the identity of* X *is about (using a normal distribution table, e.g. [16, p499]):*

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

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

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

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

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 turns out to be more appropriate for the analysis we present. This semantics describes in a slightly more abstract way the same observables defined in Section 2.3. The analysis 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 the sequence of the "fronts" in the computational tree of a given agent A. Each front represents all the possible configurations which are reachable in one step from a node in the previous front of the tree, and can be described as a family of pairs of configurations and associated probabilities.

The sequence of fronts can be constructed via the transition relation $\longrightarrow$ defined by the rules in Table 3. These rules define an operator $\mathbf{G}$ on multi-sets of pairs $\langle\langle B, c\rangle, p\rangle$, where $\langle B, c\rangle \in \mathrm{Conf}$ is a configuration and p is the probability of reaching that configuration. The semantics of an agent A is obtained by iteratively applying the transition rules starting from an initial configuration $\langle A, d\rangle$. This yields the sequence

$$[\![A, d]\!]_{coll} = \{\Phi_i(A, d)\}_i = \{\mathbf{G}^i(\Phi_0(A, d))\}_i,$$

with $\Phi_0(A, d) = \langle\langle A, d\rangle, 1\rangle$ and $\Phi_{i-1}(A, d) \longrightarrow \Phi_i(A, d)$ for all $i \geq 1$.

When the store d is omitted it will be intended to be $true$. We will also omit the specification of A when it is clear from the context.

Example 8 *The collecting semantics for the two agents A and B in Example 4 is depicted in Table 4.*

The reason why $\mathbf{G}$ operates on multi-sets is that due to the probabilistic choice a configuration can occur more than once in a front with an associated possibly different probability. This semantics is correct with respect to the notion of observables $\mathcal{O}(A,d)$ defined in Section 2.3 in the sense stated by the following Propositions.

Proposition 1 *Let A be an agent with collecting semantics $[\![A,d]\!]_{coll} = \{\Phi_i(A,d)\}_i$, and let $\pi \in \mathrm{Comp}(A,d)$ be the computational path for A in store d*

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

Then we have that $\langle\langle A_0,c_0\rangle\,,1\rangle \in \Phi_0(A,d)$, and for all $1 \leq i \leq n$, $\langle\langle A_i,c_i\rangle\,,p_i\rangle \in \Phi_i(A,d)$. Moreover, $\langle\langle \mathbf{stop},c_n\rangle\,,p_n\rangle \in \Phi_j(A,d)$ for all $j \geq n$.

Proof
By induction on the length n of π.

$(n = 1)$: In this case we have that $A \equiv \mathbf{tell}(c)$ and

$$\pi \equiv \langle \mathbf{tell}(c),d\rangle = \langle A_0,c_0\rangle \longrightarrow_1 \langle \mathbf{stop}, d \sqcup c\rangle\,.$$

On the other hand, by applying $\mathbf{G}$ to $\Phi_0 = \langle\langle \mathbf{tell}(c),d\rangle\,,1\rangle$ we get

$$\Phi_1 = \langle\langle \mathbf{stop}, d\sqcup c\rangle\,,1\rangle = \Phi_j,$$

for all $j \geq 1$.

$(n > 1)$: The proof is by cases. We show the assertion for the choice and parallel constructs. The proof is similar in the remaining two cases of hiding and procedure call. We will use the notation $\pi \cdot \pi'$ to indicate the concatenation of two paths π and π'.

$A \equiv [\!]_{s=1}^m\ \mathbf{ask}(c_s) \to p_s : A_s$: A path $\pi \in \mathrm{Comp}(A,d)$ of length n is of the form

$$\pi \equiv \langle A,d\rangle = \langle A_0,c_0\rangle \longrightarrow_{\tilde{p_k}} \langle A_k,d\rangle \cdot \pi',$$

where $k \in [1,s]$, $d \vdash c_k$ and $\pi' \in \mathrm{Comp}(A_k,d)$ is of the form

$$\pi' \equiv \langle A_k,d\rangle \longrightarrow_{q_1} \left\langle A_k^1,d_k^1\right\rangle \longrightarrow_{q_2} \dots \longrightarrow_{q_{n_k}} \left\langle A_k^{n_k},d_k^{n_k}\right\rangle,$$

with $n_k = n-1$.
By applying $\mathbf{G}$ to $\Phi_0 = \langle\langle A,d\rangle\,,1\rangle$ we have that $\langle\langle A_k,d\rangle\,,1\cdot\tilde{p_k}\rangle \in \Phi_1$.
By the Induction Hypothesis, $\left\langle\left\langle A_k^j,d_k^j\right\rangle,q_j\right\rangle \in \Phi_{j+1}$, for all $1 \leq j \leq n_k$ and $\langle\langle \mathbf{stop},d_k^{n_k}\rangle\,,q_k\rangle \in \Phi_t$, for all $t > n_k$.

$A \equiv \|_{s=1}^{m}\ p_s : A_s$: We have that π is of the form

$$\pi \equiv \langle A, d\rangle = \langle A_0, c_0\rangle \longrightarrow_{p\cdot \tilde{p_k}} \left\langle \|_{k\neq s=1}^{m}\ p_s : A_s \ \|\ p_k : A'_k, d'\right\rangle \cdot \pi',$$

for some $k \in [1, s]$, if $\langle A_k, d\rangle \longrightarrow_p \langle A'_k, d'\rangle$ and $\pi' \in \mathrm{Comp}(\|_{k\neq s=1}^{m}\ p_s : A_s \ \|\ p_k : A'_k, d')$ is a computational path of length $n' = n - 1$.
By applying $\mathbf{G}$ to $\Phi_0 = \langle\langle A, d\rangle, 1\rangle$ we have that $\langle\langle B, d'\rangle\, p \cdot \tilde{p}_k\rangle \in \Phi_1$, where $B = \|_{k\neq s=1}^{m}\ p_s : A_s \ \|\ p_k : A'_k$.
Suppose that π' is of the form

$$\pi' \equiv \langle B, d'\rangle \longrightarrow_{q_1} \langle B_1, d'_1\rangle \longrightarrow_{q_2} \ldots \longrightarrow_{q_{n'}} \langle B_{n'}, d'_{n'}\rangle .$$

Then by the Induction Hypothesis, we have that $\left\langle \left\langle B_j, d'_j\right\rangle, q_j\right\rangle \in \Phi_{j+1}$ for all $1 \le j \le n'$, and $\langle\langle \mathbf{stop}, d'_{n'}\rangle, q_{n'}\rangle \in \Phi_t$, for all $t > n'$.

□

Since we consider only computations of bounded length, we can fix for each agent A the maximal number of iterations of $\mathbf{G}$:

$$l_A = max\{n \mid \pi \equiv \langle A_0, c_0\rangle \longrightarrow_{p_1} \langle A_1, c_1\rangle \ldots \longrightarrow_{p_n} \langle A_n, c_n\rangle \in \mathrm{Comp}(A, d)\}.$$

Moreover, we can "compactify" the last front by considering one only occurrence of each configuration with a probability given by the sum of all the probabilities associated to its different occurrences. This operation can be defined in general as follows.

Definition 6 *Let $\Phi = \{\langle\langle A_j, c_j\rangle, p_j\rangle\}_j$ be a front and let $\left\{\left\langle \langle A_j, c_j\rangle^k, p_j^k\right\rangle\right\}_k$ be all the occurrences of configuration $\langle A_j, c_j\rangle$ in Φ. We define the compactification of Φ as $\mathcal{K}(\Phi) = \{\langle\langle A_j, c_j\rangle, P_j\rangle\}_j$, where $P_j = \sum_k p_j^k$.*

Definition 7 *Let $\Phi = \{\langle\langle A_j, c_j\rangle, p_j\rangle\}_j$ be a front. We define the distribution on stores associated to Φ as*

$$\alpha(\Phi) = \{\langle c_j, p_j\rangle \mid \langle\langle A_j, c_j\rangle, p_j\rangle \in \mathcal{K}(\Phi) \text{ and } A_j = \mathbf{stop}\}.$$

Proposition 2 *Let A be a PCCP agent with collecting semantics*

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

and let l_A be the maximal length of a computational path for A in store d. Then

$$\mathcal{O}(A, d) \subseteq \alpha(\mathcal{K}(\Phi_{l_A}(A, d))).$$

Proof
Let $\langle c, p\rangle \in \mathcal{O}(A, d)$. Then there exists $\pi \in \mathrm{Comp}(A, d)$ such that $c = res([\pi])$ and $p = prob([\pi])$.

By Proposition 1 and the definition of l_A, for all $\pi' \in [\pi]$, $\langle\langle \mathbf{stop}, c\rangle, p_{\pi'}\rangle \in \Phi_{l_A}$, where $p_{\pi'}$ is the probability of c in path π'.

Thus, $\langle\langle \mathbf{stop}, c\rangle, p\rangle \in \mathcal{K}(\Phi_{l_A})$ and so $\langle c, p\rangle \in \alpha(\mathcal{K}(\Phi_{l_A}(A, d)))$. □

5.2 Security Analysis

Given two agents A and B, scheduling probabilities p and $q = 1 - p$ and a spy S, our aim is to calculate

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

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 $\mathcal{S}_1$ and $\mathcal{S}_2$. This will yield the information needed to calculate ε.

For the simpler spies in $\mathcal{S}_1$, i.e. spies of the form:

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

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 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:

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

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

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

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 front Φ_k where c is entailed the first time for some c_i^k, i.e.

$$\Phi_k = \{\ldots, \langle\langle A_i^k, c_i^k \rangle, p_i^k \rangle, \ldots\}$$

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

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

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

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

The monotonicity of PCCP guarantees that once c is entailed it will continue to be so, i.e. $c_i^k \vdash c$ implies $c_i^t \vdash c$, for all $t \geq k$.

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

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

$$\begin{array}{rcl} & = & \{\ldots, \\ & & \left\langle \left\langle p : A_i^{k+m} \parallel q : S, c_i^{k+m} \right\rangle, p_i^{k+m} \cdot p^m \right\rangle, \\ & & \left\langle \left\langle p : A_i^{k+m-1} \parallel q : \textbf{stop}, c_i^{k+m-1} \sqcup f \right\rangle \right\rangle p_i^{k+m-1} \cdot (1 - p^m) \\ & & \ldots\}\end{array}$$

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

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

$\square$

Proposition 3 *Let A be a PCCP agent, and let $S \in \mathcal{S}_2$ be a spy of the form*

$$S \equiv \textbf{ask}(c_1) \to q_1 : \textbf{tell}(f_1) \,[]\, \textbf{ask}(c_2) \to q_2 : \textbf{tell}(f_2),$$

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:

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

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.

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 $\mathcal{S}_1$. Thus by Lemma 1 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 the 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,

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

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

$j < k$: The fronts 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 an intermediate 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 front $\overline{\Phi}_{k+1}$ is given by:

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

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:

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

$j = k + m + 1$: We 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 $p : A_i^{k+m-1} \parallel q :$ **stop** has to continue quasi-deterministically with A_i^{k+m-1}.

$$\begin{array}{rcl} \overline{\Phi}_{k+m+1} & = & \{ \ldots, \\ & & \langle \langle p : A_i^{k+m+1} \parallel q : S, c_i^{k+m+1} \rangle, \\ & & \quad p_i^{k+m+1} \cdot p^{m+1} \rangle, \end{array}$$

$$
\begin{aligned}
&\langle\langle p : A_i^{k+m} \parallel q : \textbf{stop}, c_i^{k+m} \sqcup f_1\rangle\,, \\
&\quad p_i^{k+m} \cdot p^m q q_1\rangle, \\
&\langle\langle p : A_i^{k+m} \parallel q : \textbf{stop}, c_i^{k+m} \sqcup f_2\rangle\,, \\
&\quad p_i^{k+m} \cdot p^m q q_2\rangle, \\
&\langle\langle p : A_i^{k+m} \parallel q : \textbf{stop}, c_i^{k+m} \sqcup f_1\rangle\,, \\
&\quad p_i^{k+m} \cdot (1 - p^m)\rangle, \\
&\ldots\} \\
= \quad &\{\ldots, \\
&\langle\langle p : A_i^{k+m+1} \parallel q : S, c_i^{k+m+1}\rangle\,, \\
&\quad p_i^{k+m+1} \cdot p^{m+1}\rangle, \\
&\langle\langle p : A_i^{k+m} \parallel q : \textbf{stop}, c_i^{k+m} \sqcup f_1\rangle\,, \\
&\quad p_i^{k+m} \cdot (p^m q q_1 + (1 - p^m))\rangle, \\
&\langle\langle p : A_i^{k+m} \parallel q : \textbf{stop}, c_i^{k+m} \sqcup f_2\rangle\,, \\
&\quad p_i^{k+m} \cdot p^m q q_2\rangle, \\
&\ldots\}.
\end{aligned}
$$

$j = k + m + n$**:** After further n steps the agent A terminates.

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

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

$$
\begin{aligned}
&\overline{\Phi}_{k+m+n+1} = \\
&\quad = \quad \{\ldots, \\
&\qquad \langle\langle p : \textbf{stop} \parallel q : \textbf{stop}, c_i^{k+m+n} \sqcup f_1\rangle\,, \\
&\qquad\quad p_i^{k+m+n} \cdot p^{m+n} q_1\rangle,
\end{aligned}
$$

$$
\begin{array}{ll}
 & \langle\big\langle p:\mathbf{stop} \parallel q:\mathbf{stop}, c_i^{k+m+n} \sqcup f_2\big\rangle\,, \\
 & \quad p_i^{k+m+n}\cdot p^{m+n}q_2\rangle, \\
 & \langle\big\langle p:\mathbf{stop} \parallel q:\mathbf{stop}, c_i^{k+m+n} \sqcup f_1\big\rangle\,, \\
 & \quad p_i^{k+m+n}\cdot ((\sum_{l=0}^{n-1} p^{m+l})qq_1 + (1-p^m))\rangle, \\
 & \langle\big\langle p:\mathbf{stop} \parallel q:\mathbf{stop}, c_i^{k+m+m} \sqcup f_2\big\rangle\,, \\
 & \quad p_i^{k+m+n}\cdot (\sum_{l=0}^{n-1} p^{m+l})qq_2\rangle, \\
 & \ldots\} \\
= & \{\ldots, \\
 & \langle\big\langle p:\mathbf{stop} \parallel q:\mathbf{stop}, c_i^{k+m+n} \sqcup f_1\big\rangle\,, \\
 & \quad p_i^{k+m+n}\cdot (((\sum_{l=0}^{n-1} p^{m+l})q + p^{m+n})q_1 + (1-p^m))\rangle, \\
 & \langle\big\langle p:\mathbf{stop} \parallel q:\mathbf{stop}, c_i^{k+m+m} \sqcup f_2\big\rangle\,, \\
 & \quad p_i^{k+m+n}\cdot ((\sum_{l=0}^{n-1} p^{m+l})q + p^{m+n})q_2\rangle, \\
 & \ldots\}.
\end{array}
$$

□

Corollary 1 *With the hypothesis of Proposition 3 the following holds for all i:*

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

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 Equation 1 we have that:

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

□

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

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

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) (since $q = 1 - p$) we get for r_i^2 the following formula:

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

For the case where 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 $\mathcal{O}(p : A \parallel q : S)$ 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 4 *Given an agent $S \in \mathcal{S}_2$ of the form*

$$S \equiv \textbf{ask}(c_1) \rightarrow q_1 : \textbf{tell}(f_1) \, [] \, \textbf{ask}(c_2) \rightarrow q_2 : \textbf{tell}(f_2),$$

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_j^1 and r_j^2 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 closed expressions for r_j^1 and r_j^2, if $r_j^1(A) \neq r_j^1(B)$ 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 probabilities, 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 5 *Let S be a spy in $\mathcal{S}_2$ of the form*

$$S \equiv \textbf{ask}(c_1) \rightarrow q_1 : \textbf{tell}(f_1) \, [] \, \textbf{ask}(c_2) \rightarrow q_2 : \textbf{tell}(f_2),$$

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

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

Clearly, the effectiveness of a spy is a function of the scheduling distribution. In other words, it is determined not only by the intrinsic power of the spy but also by the internal design of the host system.

Example 9 *We compute the effectiveness of the spy C for the two agents A and B in Example 4. To this purpose, we first compute $r^c_{c\sqcup d}$ and $r^d_{c\sqcup d}$ using Proposition 3. 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:*

$$\begin{aligned} r^c_{c\sqcup d,1} &= \frac{1}{2}((0 + p^{1+0})q_1 + (1 - p^1)) \\ &= \frac{1}{2}(1 - p + pq_1) \\ r^d_{c\sqcup d,1} &= \frac{1}{2}(0 + p^{1+0})q_2 = \frac{1}{2}pq_2. \end{aligned}$$

We know that for the spy C, $q_1 = \frac{2}{3}$ and $q_2 = \frac{1}{3}$. 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:*

$$\begin{aligned} r^d_{c\sqcup d,2} &= \frac{1}{2}((0 + p^{1+0})q_2 + (1 - p^1)) \\ &= \frac{1}{2}(1 - p + pq_2) \\ r^c_{c\sqcup d,2} &= \frac{1}{2}(0 + p^{1+0})q_1 = \frac{1}{2}pq_1, \end{aligned}$$

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

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

Summing up over the two paths, we get the probabilities for $c\sqcup d\sqcup e$ and $c\sqcup d\sqcup f$:

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

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

$$\begin{aligned}
r^c_{c\sqcup d,1} &= ((0 + p^{0+0})q_1 + (1 - p^0)) \\
&= ((0 + 1)q_1 + (1 - 1)) = q_1 \\
r^d_{c\sqcup d,1} &= (0 + p^{0+0})q_2 = (0 + 1)q_2 = q_2
\end{aligned}$$

For q_1 and q_2 as the choice distribution in C and the uniform scheduling we obtain 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 with respect to the scheduling $p = q = \frac{1}{2}$, 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 $\mathcal{S} = \{C\}$ and $p = \frac{1}{2}$.

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 q_1 is very close to 0 and q_2 is very close to 1 or vice versa. In other words the "best" spy is the one where the probabilities are at the extreme opposite. Note that the number ε we calculate for the best spy may vary depending on the scheduling probabilities. However, although the actual measure of the effectiveness might be different, it will always be maximal compared to the other spies in the same class.

Example 10 *Using the limit analysis we now show that C is not the most effective spy for the agents A and B in Example 4. 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:*

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

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* $c \sqcup d \sqcup f$*:*

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

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* $\mathcal{S}_2$*:*

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

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.

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

The parallel agent $\|_{i=1}^{n} A_i$ is interpreted by the set of all possible interleavings of the constituent agents. Given two sets of tuples X and Y, the operation $\odot$ is defined as follows:

$$
\begin{array}{lcl}
X \odot Y & = & \bigcup_{\tau \in min(X)} \{\{\tau\} \cup P \mid P \in ((X \backslash min(X)) \uparrow \tau) \odot Y'_\tau\} \cup \\
 & & \bigcup_{\tau \in min(Y)} \{\{\tau\} \cup P \mid P \in X'_\tau \odot ((Y \backslash min(Y)) \uparrow \tau\} \\
\text{where} & & \\
X'_{\langle \_, e, \_, \_\rangle} & = & \{\langle c \sqcup e, d \sqcup e, t+1, p\rangle \mid \langle c, d, t, p\rangle \in X\} \\
Y'_{\langle \_, e, \_, \_\rangle} & = & \{\langle c \sqcup e, d \sqcup e, t+1, p\rangle \mid \langle c, d, t, p\rangle \in Y\} \\
X \uparrow \langle c, d, t, p\rangle & = & \{\langle c', d', t', p'\rangle \mid \langle c', d', t', p'\rangle \in X \wedge d \vdash c' \wedge t' > t\}
\end{array}
$$

The *min* function returns a set of tuples that have the smallest time stamp (this might not be a singleton because of the choice operator). The generalisation of $\odot$ to more than two agents is straightforward.

We use the operator $\nabla^{\delta,n}$ 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 $\nabla_0^{\delta,n}$ and continue by replacing the procedure call p in the term A by the previous approximation — denoted by $[\![A]\!][p \mapsto \nabla_i^{\delta,n}([\![A]\!])]$ — 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 $\nabla_\infty^{\delta,n}$ 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:

$$\nabla^{\delta,n}([\![A]\!]) = \lim_{i \to \infty} \nabla_i^{\delta,n}([\![A]\!])$$

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 $\nabla^{\delta,n}$ is the quantitative analogue of a widening operator in the standard approaches to abstract interpretation [5]; whilst the standard definition of a widening involves over-approximation (of an upper bound), in the quantitative setting we settle for "closeness". We use $\mathcal{C} \backslash \nabla_n^{\delta,n}([\![A]\!])$ to denote the set of constraint stores that do not appear in $\nabla_n^{\delta,n}([\![A]\!])$.

6.1 Abstract Security Analysis

Given the set of quadruples associated with an agent, we can extract the set of abstract paths of execution starting from some given constraint store e:

$$
\begin{array}{l}
\overline{Paths}_e([\![A]\!]) = \{q_0 \ldots q_n \mid \\
\quad \forall 1 \leq i \leq n. q_i \in \{\langle c \sqcup e, d \sqcup e, t, p\rangle \mid \langle c, d, t, p\rangle \in [\![A]\!]\} \wedge \\
\quad q_0 \equiv \langle e, e, 0, 1\rangle \wedge d_{q_i} \vdash c_{q_{i+1}} \wedge \\
\quad (t_{q_{i+1}} > t_{q_i} \ \vee \ t_{q_{i+1}} = t_{q_i} = \infty) \wedge \\
\quad \forall j \leq i. q_j \neq \langle c_{q_{i+1}}, d_{q_{i+1}}, t_{q_{i+1}}, p_{q_{i+1}}\rangle\}
\end{array}
$$

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 $\overline{Paths_e}(\llbracket A \rrbracket)$ may be smaller than in the concrete semantics.

Given a path $\sigma \in \overline{Paths_e}(\llbracket A \rrbracket)$ 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. If either of the time stamps is ∞, then the difference between times (i.e. the abstract number of steps) is taken to be ∞. For each pair of constraints c_1 and c_2, the abstract analysis of agent A gives us the set:

$$\overline{\mathcal{A}}^e_A(c_1, c_2) = \bigcup_{\sigma \in \overline{Paths_e}(\llbracket A \rrbracket)} \{\langle \overline{p}_\sigma, \overline{m}_\sigma \rangle\}.$$

6.2 Correctness of the Analysis

In order to state the correctness of the analysis, we need to define concrete computation paths. We use the following definition (note that we are only considering finite computations):

$$\begin{array}{rcl} Paths_e(A) & = & \{\langle\langle A, e\rangle, 1\rangle\, Q \mid \\ & & Q \in \bigcup_{\langle\langle A', e'\rangle, p\rangle \in \mathbf{G}(\langle\langle A, e\rangle, 1\rangle)} p.Paths_{e'}(A') \wedge \\ & & A \not\equiv \mathbf{stop}\} \\ & & \cup \\ & & \{\langle\langle \mathbf{stop}, e\rangle, 1\rangle \mid A \equiv \mathbf{stop}\} \end{array}$$

By analogy with the previous section, for each pair of constraints c_1 and c_2, we define:

$$\mathcal{A}^e_A(c_1, c_2) = \bigcup_{\pi \in Paths_e(A)} \{\langle p_\pi, m_\pi \rangle\}.$$

Given these definitions we can state the correctness of the analysis.

Proposition 6

$$\begin{array}{l} \forall c_1, c_2, e.(\forall \langle p, m\rangle \in \mathcal{A}^e_A(c_1, c_2). \\ \quad (\exists \langle \overline{p}, \overline{m}\rangle \in \overline{\mathcal{A}}^e_A(c_1, c_2). \overline{p} \le p \wedge \overline{m} \ge m)) \end{array}$$

Proof

Proof is by induction over the length of the longest paths in $Paths_e(A)$. We illustrate just three cases:

$A \equiv \mathbf{tell}(c)$: The only concrete path is

$$\langle\langle \mathbf{tell}(c), e\rangle, 1\rangle \langle\langle \mathbf{stop}, c \sqcup e\rangle, 1\rangle$$

and there is a corresponding abstract path:

$$\langle e, e, 0, 1\rangle \langle e, c \sqcup e, 1, 1\rangle$$

from which the result follows.

$A \equiv q_1 : A_1 \parallel q_2 : A_2$: First we note that

$$\begin{array}{rcl} Paths_e(A) & = & \{\langle\langle A, e\rangle, 1\rangle Q \mid \\ & & Q \in p_1\tilde{q}_1 Paths_{e_1}(q_1 : A_1' \parallel q_2 : A_2) \cup \\ & & p_2\tilde{q}_2 Paths_{e_2}(q_1 : A_1 \parallel q_2 : A_2')\} \end{array}$$

where

$$\langle A_1, e\rangle \rightarrow_{p_1} \langle A_1', e_1\rangle \text{ and } \langle A_2, e\rangle \rightarrow_{p_2} \langle A_2', e_2\rangle$$

It follows that

$$\mathcal{A}_A^e(c_1, c_2) = \begin{cases} p_1\tilde{q}_1\mathcal{A}^{e_1}_{A_1'\parallel A_2}(c_1, c_2) \cup p_2\tilde{q}_2\mathcal{A}^{e_2}_{A_1\parallel A_2'}(c_1, c_2) & \text{if } e \not\vdash c_1 \\ \{\langle p, m+1\rangle \mid \langle p, m\rangle \in p_1\tilde{q}_1\mathcal{A}^{e_1}_{A_1'\parallel A_2}(c_1, c_2) \cup p_2\tilde{q}_2\mathcal{A}^{e_2}_{A_1\parallel A_2'}(c_1, c_2)\} & \text{otherwise} \end{cases}$$

Now if $e \not\vdash c_1$ the result follows from the Induction Hypothesis (twice) and the fact that the abstract semantics does not normalise the probabilities – unnormalised probabilities are always smaller than the normalised versions.

Now suppose that $e \vdash c_1$. Note that

$$\begin{array}{l} q_1[\![A_1]\!] \odot q_2[\![A_2]\!] = \\ \bigcup_{\tau\in min([\![A_1]\!])}\{\{q_1\tau\} \cup P \mid P \in (([\![A_1]\!]\backslash min([\![A_1]\!])) \uparrow \tau) \odot [\![A_2]\!]'_\tau\} \cup \\ \bigcup_{\tau\in min([\![A_2]\!])}\{\{q_2\tau\} \cup P \mid P \in [\![A_1]\!]'_\tau \odot (([\![A_2]\!]\backslash min([\![A_2]\!])) \uparrow \tau\} \end{array}$$

and thus

$$\begin{array}{l} \overline{Paths_e}([\![A]\!]) = \\ \{\langle e, e, 0, 1\rangle \langle e, e_1, 1, p_1q_1\rangle P_1 \mid P_1 \in \overline{Paths_{e_1}}([\![q_1 : A_1' \parallel q_2 : A_2]\!])'\} \cup \\ \{\langle e, e, 0, 1\rangle \langle e, e_2, 1, p_2q_2\rangle P_2 \mid P_2 \in \overline{Paths_{e_2}}([\![q_1 : A_1 \parallel q_2 : A_2']\!])'\} \end{array}$$

and the result follows.

$A \equiv p(x)$: This result follows from a similar analysis to the previous case. The widening operator, ∇ introduces some additional subtlety. If both c_1 and c_2 are entailed within the first *n* un-foldings of the recursion, the estimate of $\overline{m}$ will be finite and an over-approximation of *m* (which follows from the Induction Hypothesis). Otherwise, $\overline{m}$ will be ∞ and $\overline{p}$ will be zero. These values trivially satisfy the proposition.

□

7 Conclusions and Related Work

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 respect 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 the approximate confinement notion. If we had to consider strict confinement — i.e. $\varepsilon = 0$ — 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 $\varepsilon \neq 0$.

In our recent work we have extended the approach presented in this paper to general probabilistic process algebras. In particular, we have concentrated on a simple probabilistic process calculus, namely PCCS [19], based on Milner's SCCS [25]. Like PCCP, this calculus is based on a generative model [36]. Such a model, where nondeterminism is completely replaced by probabilistic choices, is the appropriate base for investigating our notion of approximation which requires a purely statistical model.

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 a quantity, whereas equivalence relations such as observational equivalence or bisimilarity can only establish whether two objects can be identified or not. For example, in [22] the security of cryptographic protocols is specified via an observational equivalence relation which identifies protocols which differ asymptotically for a polynomial factor. Such a quantity is nevertheless neither used to quantify the similarity of the protocols nor to calculate a corresponding approximation level of the protocols security property. Analogously, the bisimulation through probabilistic testing of Larsen and Skou [21] 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 [35] 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. There is also a considerable body of work that concerns quantification of process behaviour. The most closely related is the work of Desharnais *et al* [18]. They study Labelled Markov Processes (LMP) from a domain-theoretic point of view. They establish that quantitative observations of a continuous-state LMP can be approximated by observations on finite-state Markov chains. They specify the approximation by two parameters: n – the number of successive transitions possible from the start state – and ϵ – a rational number that measures the accuracy with which the transition probabilities in the approximation reflect the transition probabilities of the original process. This measure is closely related to our ϵ but is used in a rather different way; there is no discussion about applications to security in the cited paper. We should also mention the work of Gavin Lowe [23] and Alessandro Aldini [1]. The former takes a rather different approach to quantifying information flow; Lowe gives a formal definition of the capacity of covert channels by measuring the number of different High behaviours that can be observed by a Low observer. The work is based on CSP and testing equivalence. Aldini extends Focardi's and Gorrieri's seminal work on the classification of security properties [12] with probabilities. He proposes probabilistic versions of bisimulation-based non-interference properties identified by Focardi and Gorrieri. The work does not incorporate any notion of approximation but this might be an interesting future development. Finally we mention the work of Leo Marcus [24], which takes a model-theoretic view of dependence and independence relations in computer security. This work is expressed in abstract terms for some given computational theory. Our approach is rather more concrete, the static analysis takes particular note of the algorithmic structure of the computation, and it remains a challenge for us to understand the precise relationship with Marcus' work.

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) [8].

References

[1] A. Aldini. On the Extension of Non-interference with Probabilities. In P. Syverson and J. Guttman, editors, *Proceedings of WITS'02 – Workshop on Issues in the Theory of Security, 14–15 January, Portland*, January 2002. `http://www.dsi.unive.it/IFIPWG1_7/WITS2002`.

[2] J.A. Bergstra, A. Ponse, and S.A. Smolka, editors. *Handbook of Process Algebra*. Elsevier Science, Amsterdam, 2001.

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

[4] D. Clark, S. Hunt, and P. Malacaria. Quantitative analysis of leakage of confidential data. In *QAPL 2001 - First International Workshop on Quantitative Aspects of Programming Laguages*, volume 59 of *Electronic Notes in Theoretical Computer*, November 2002.

[5] P. Cousot and R. 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.

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

[7] A. Di Pierro, C. Hankin, and H. Wiklicky. Probabilistic confinement in a declarative framework. In *Declarative Programming – Selected Papers from AGP 2000 – La Havana, Cuba*, volume 48 of *Electronic Notes in Theoretical Computer Science*, pages 1–23. Elsevier, 2001.

[8] A. Di Pierro, C. Hankin, and H. Wiklicky. Analysing approximate confinement under uniform attacks. In *SAS 2002 - 9th International Symposium on Static Analysis*, volume 2477 of *Lecture Notes in Computer Science*, pages 310–326. Springer, 2002.

[9] A. Di Pierro and H. 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.

[10] A. Di Pierro and H. 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.

[11] A. Di Pierro and H. 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.

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

[13] R. Focardi and R. Gorrieri. Classification of Security Properties (Part I: Information Flow). In *Foundations of Security Analysis and Design - Tutorial Lectures*, volume 2171 of *Lecture Notes in Computer Science*, pages 331–396. Springer, 2001.

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

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

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

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

[18] R. Jagadeesan J. Desharnais, V. Gupta and P. Panangaden. Approximating Labelled Markov Processes. In *Proceedings of LICS'2000*. IEEE Computer Society Press, 2000.

[19] B. Jonsson, W. Yi, and K.G. Larsen. *Probabilistic Extentions of Process Algebras*, chapter 11, pages 685–710. Elsevier Science, Amsterdam, 2001. see [2].

[20] P.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.

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

[22] P.D. Lincoln, J.C. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic polytime framework for protocol analysis. In *ACM Conference on Computer and Communication Security (CCS-5)*, 1998.

[23] G. Lowe. Quantifying Information Flow. In *Proceedings of CSFW15 – Computer Security Foundations Workshop*. IEEE Computer Society Press, 2002.

[24] L. Marcus. Formal Dependence and Independence Relations in Computer Security. In *Haifa Logic Colloquium*, 1995.

[25] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin – New York, 1980.

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

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

[28] A. Sabelfeld and D. 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.

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

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

[31] 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.

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

[33] G. Smith and D. 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.

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

[35] F. van Breugel and J. 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.

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

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

[38] D. Volpano and G. 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.

[39] 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.

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

Table 1: The Syntax of PCCP Agents

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

Table 2: The Transition System for PCCP

$$
\begin{array}{ll}
\textbf{R0} & \{\ldots,\langle\langle\textbf{stop},d\rangle,p\rangle,\ldots\} \longrightarrow \{\ldots,\langle\langle\textbf{stop},d\rangle,p\rangle,\ldots\} \\
\textbf{R1} & \{\ldots,\langle\langle\textbf{tell}(c),d\rangle,p\rangle,\ldots\} \longrightarrow \{\ldots,\langle\langle\textbf{stop},c\sqcup d\rangle,p\rangle,\ldots\} \\
\textbf{R2} & \{\ldots,\langle\langle\square_{i=1}^{n}\ \textbf{ask}(c_i)\rightarrow p_i : A_i,d\rangle,q\rangle,\ldots\} \longrightarrow \\
 & \qquad\qquad \longrightarrow \{\ldots,\langle\langle A_j,d\rangle,q\cdot\tilde{p}_j\rangle,\ldots\} \\
 & \qquad\qquad\qquad\qquad \text{for } j\in[1,n] \text{ and } d\vdash c_j \\
\textbf{R3} & \{\ldots,\langle\langle\|_{i=1}^{n}\ \ p_i : A_i,d\rangle,q\rangle,\ldots\} \longrightarrow \\
 & \qquad\qquad \longrightarrow \left\{\ldots,\left\langle\left\langle\|_{i\neq j}^{n}\ p_i : A_i \parallel p_j : A_j',d'\right\rangle,q\cdot\tilde{p}_j\right\rangle,\ldots\right\} \\
 & \qquad\qquad\qquad\qquad \text{for } j\in[1,n] \text{ and } \langle A_j,d\rangle \longrightarrow_p \left\langle A_j',d'\right\rangle \\
\textbf{R4} & \{\ldots,\langle\langle\exists_x A,d\rangle,q\rangle,\ldots\} \longrightarrow \{\ldots,\langle\langle\exists_x A',d\sqcup\exists_x d'\rangle,q\cdot p\rangle,\ldots\} \\
 & \qquad\qquad\qquad\qquad \text{for } \langle A,\exists_x d\rangle \longrightarrow_p \langle A',d'\rangle \\
\textbf{R5} & \{\ldots,\langle\langle p(y),d\rangle,q\rangle,\ldots\} \longrightarrow \{\ldots,\left\langle\left\langle\Delta_y^x A,d\right\rangle,q\right\rangle,\ldots\} \\
 & \qquad\qquad\qquad\qquad \text{for } p(x) : -A \in P
\end{array}
$$

Table 3: A Collecting Semantics for PCCP

$[\![A]\!]_{coll}$
$\Phi_0 = \{\langle\langle A, \mathit{true}\rangle,1\rangle\}$
$\Phi_1 = \left\{\left\langle\left\langle\frac{1}{2} : \textbf{stop} \parallel \frac{1}{2} : \textbf{tell}(d), c\right\rangle,\frac{1}{2}\right\rangle \left\langle\left\langle\frac{1}{2} : \textbf{tell}(d) \parallel \frac{1}{2} : \textbf{stop}, d\right\rangle,\frac{1}{2}\right\rangle\right\}$
$\Phi_2 = \{\langle\langle\textbf{stop}, c\sqcup d\rangle,1\rangle\}$

$[\![B]\!]_{coll}$
$\Phi_0 = \{\langle\langle B, \mathit{true}\rangle,1\rangle\}$
$\Phi_1 = \{\langle\langle\textbf{stop}, c\sqcup d\rangle,1\rangle\}$

Table 4: The Collecting Semantics for A and B

$$
\begin{array}{lcl}
[\![\mathbf{tell}(c)]\!] & = & \{\langle \mathit{true}, c, 1, 1\rangle\} \\
[\![\Box_{i=1}^{n}\ \mathbf{ask}(c_i) \rightarrow p_i : A_i]\!] & = & \bigoplus_i (p_i, c_i, [\![A_i]\!]) \\
[\![||_{i=i}^{n}\ q_i : A_i]\!] & = & \bigodot_i q_i [\![A_i]\!] \\
[\![\exists_x A]\!] & = & [\![A]\!] \\
[\![p(x)]\!] & = & \nabla^{\delta,n}([\![A]\!]) \ \text{ for } \ p(x) : -A \in P
\end{array}
$$

Table 5: The Analysis for PCCP Agents

$$
\begin{array}{lcl}
\nabla_0^{\delta,n}([\![A]\!]) & = & \{\langle \mathit{true}, \mathit{true}, 1, 1\rangle\} \\
\nabla_{i+1}^{\delta,n}([\![A]\!]) & = & \begin{cases}
[\![A]\!][p \mapsto \nabla_i^{\delta,n}([\![A]\!])] \\
\quad \text{if } ||[\![A]\!][p \mapsto \nabla_i^{\delta,n}([\![A]\!])] - \nabla_i^{\delta,n}([\![A]\!])|| > \delta \text{ and } i \leq n, \\
\nabla_i^{\delta,n}([\![A]\!]) \\
\quad \text{if } ||[\![A]\!][p \mapsto \nabla_i^{\delta,n}([\![A]\!])] - \nabla_i^{\delta,n}([\![A]\!])|| \leq \delta \text{ and } i \leq n, \\
\nabla_\infty^{\delta,n}([\![A]\!])) \\
\quad \text{otherwise}
\end{cases} \\
\nabla_\infty^{\delta,n}([\![A]\!]) & = & \nabla_n^{\delta,n}([\![A]\!]) \cup \bigcup_{c \in \mathcal{C} \setminus \nabla_n^{\delta,n}([\![A]\!])} \{\langle c, c, \infty, 0\rangle\}
\end{array}
$$

Table 6: Unwinding a Procedure Call

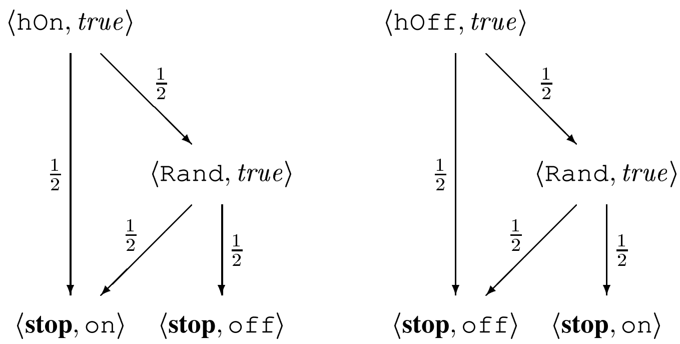


Figure 1: Transitions for `hOn` and `hOff`

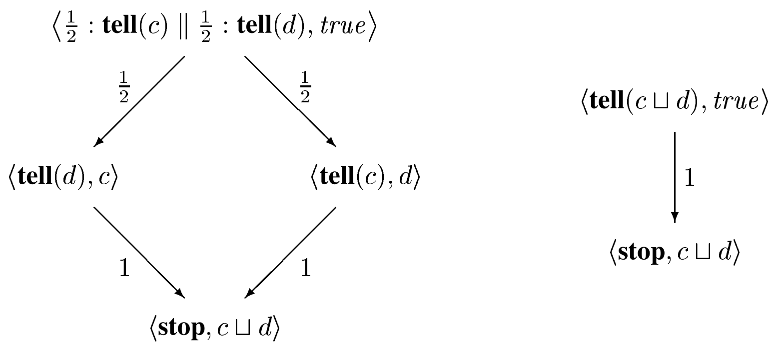


Figure 2: Independent Executions of A and B

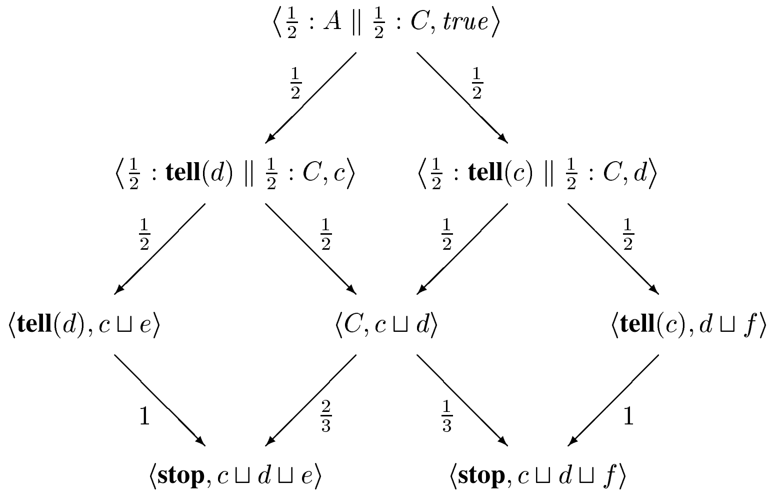


Figure 3: Executions of A in Context C

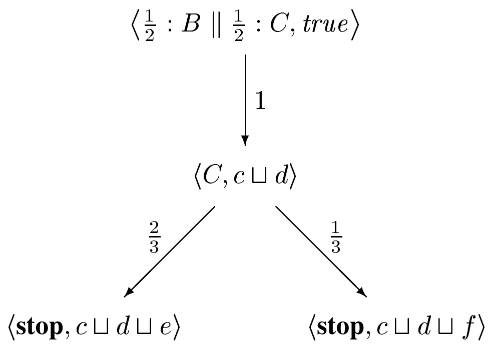


Figure 4: Executions of B in Context C

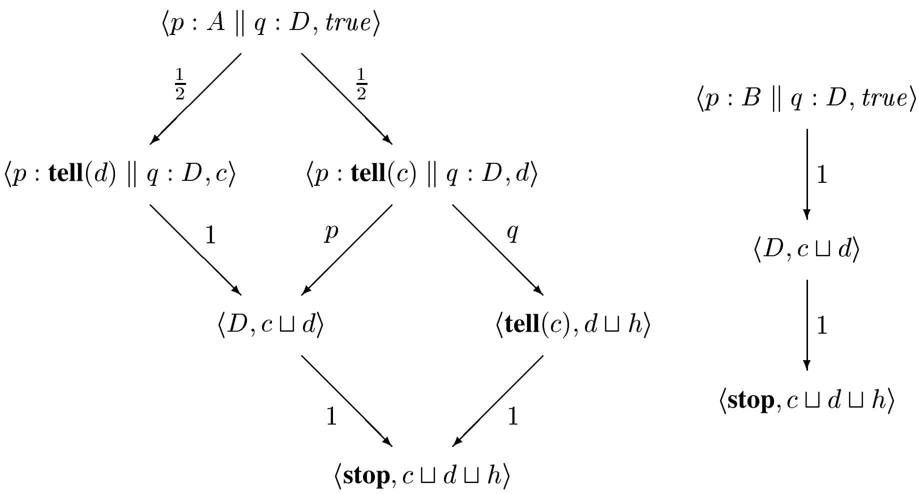


Figure 5: Executions in Context D when d entails g

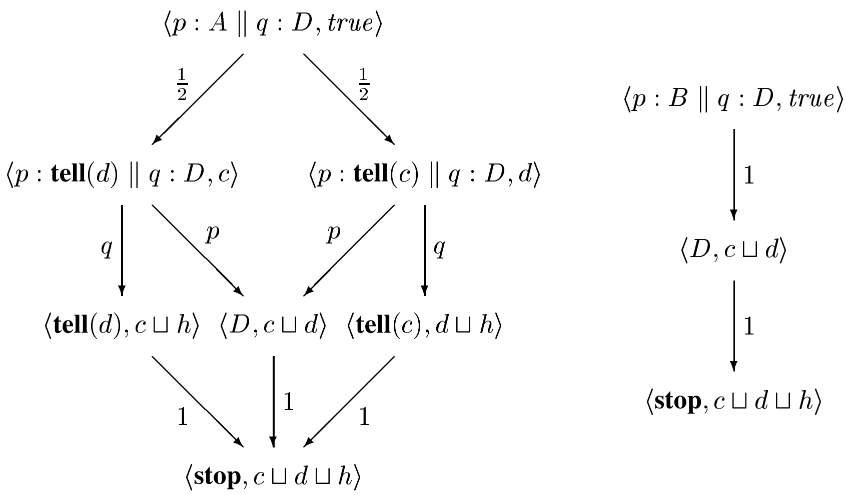


Figure 6: Executions in Context D when both c and d entail g