Abstract
Probabilistic coherence spaces (PCoh) yield a semantics of higher-order probabilistic computation, interpreting types as convex sets and programs as power series. We prove that the equality of interpretations in PCoh characterizes the operational indistinguishability of programs in PCF with a random primitive.

This is the first result of full abstraction for a semantics of probabilistic PCF. The key ingredient relies on the regularity of power series and introduces, in denotational semantics, techniques coming from Calculus.

Along the way to the theorem, we design a weighted intersection type assignment system giving a logical presentation of PCoh.

Categories and Subject Descriptors CR-number [subcategory]:
third-level

1. Introduction
Probabilistic behaviors appear in many places in the study of programming languages, for instance if the environment of a program behaves randomly or if the program uses probabilistic constructs.

To understand how the introduction of probabilities changes the computational landscape, we use Semantics. Indeed, in the last decades [24, 27, 30], semantics has succeeded in giving insights on the way programs compute. More precisely, operational semantics allows one to formalize a program by the sequence of its execution steps, while denotational semantics represents programs by functions in some mathematical space relating the interpretations of inputs and outputs in a compositional way. If this last mathematical representation is correct and accurate enough, then denotational properties lead to computational features. A key example is full abstraction [23], stating that operational indistinguishability (i.e. behaving in the same way in any context) is characterized by denotational equivalence (i.e. having the same interpretation). So semantics is useful both to give a precise meaning to syntactical constructs and to separate non-equivalent programs.

Of course, probabilistic semantics have already been investigated. First, the domain-theoretic approach has led to a probabilistic powerdomain [19, 29] which is a sibling of the non-deterministic power domain [27]. This approach follows the computational monad method [25]: programs are interpreted as functions from the input domain to the powerdomain of the output domain. Intuitively, a program takes an input and returns a probability distribution on outputs. This line of work has been continued by the continuous random variable construction [15], introducing standard tools of probability theory into semantics. Secondly, the game-theoretic approach [1, 18] has been extended with probabilistic features [5]. Intuitively, probabilistic programs are interpreted as probabilistic strategies, that are stochastic processes on the plays of the games associated with the types of the programs.

Quantitative semantics follows another tradition stemming from Linear Logic models [10, 11]. From the very beginning, Linear Logic has been associated with intuitions coming from calculus and linear algebra [7, 8, 12]. Indeed, programs are interpreted as entire series between mathematical spaces or as analytic functors between sets [17], and programs that use their resources only once are interpreted as linear functions. This connection with resource consumption has been fruitful in the last decade with the introduction of differential nets and resource calculus [2, 9]. Recently, quantitative semantics has been explicitly related to the study of quantitative properties such as time or space consumed by a computation [22]. This illustrates a new paradigm where a semiring of scalars allows one to encode non-deterministic or probabilistic computations as opposed to the domain paradigm where monads are used. Another important tenet of Linear Logic is the perfect duality between programs and environments since a program can be seen as the environment of other programs. Therefore, representing probabilistic environments or programs will boil down to the same study. Probabilistic coherence spaces provide a quantitative semantics [4, 14] which model probabilistic computations.

The main contribution of the present work is to show that probabilistic coherence spaces provide a fully abstract model of PCF, a probabilistic extension of the functional programming language PCF [28]. Although the proof follows the general pattern that consists in finding a definable context that separates two terms, the key ingredient is based on Calculus (see Lemma 25) since programs are interpreted as power series.

To our knowledge, no known model of probabilistic PCF has yet been proved to be fully abstract. Games semantics provide fully abstract models of standard PCF via an extensional collapse [1, 18]. This technique has been adapted to probabilistic game semantics, providing a fully abstract model of probabilistic idealized ALGOL [5] (an extension of probabilistic PCF with references). Our result characterizes the operational indistinguishability without the
need of an extensional collapse and deals with a functional probabilistic language with no references.

Section 2 is devoted to an insight on the way programs and data are interpreted in probabilistic coherence spaces. Section 3 is devoted to the syntax and the operational semantics. Next, in Section 4, we describe the key notions of probabilistic coherence spaces that will be useful to prove, in Section 5, full abstraction. Along the way to the theorem, we define an intersection type assignment system (Figure 3) giving a logical presentation to the model. This system has an interest by its own, allowing one to turn a question of computing the semantics of a term (and hence its operational behavior) into a proof search problem. Finally, in Section 6, we show that inequational full abstraction fails, i.e. the semantic order does not coincide with the operational one. For this, we achieve a context lemma for probabilistic PCF (Proposition 31).

Notation 1. We write \( \mathbb{N} \) for the set of non negative integers, \( \mathbb{N}^* \) for the set of positive integers \((\mathbb{N}^* \triangleq \mathbb{N}\setminus\{0\})\). \( \mathbb{R}_+ \) for the set of non negative real numbers and \( \mathbb{R}_+^\infty \triangleq \mathbb{R}_+ \cup \{\infty\} \) for the completed real half line. Let \( S \) be a set, \( \mathfrak{p}S \) denotes its cardinality. Multisets of elements of \( S \) are identified with functions \( S \rightarrow \mathbb{N} \). If \( m \) is such a multiset, \( \text{Supp}(m) \) denotes its support set \( \{ a \in S \text{ s.t. } m(a) \neq 0 \} \). A finite multiset is a multiset with a finite support. We write \( M_t(S) \) for the set of all finite multisets of elements of \( S \). We enumerate \( m \) by using \((a,i) \in m \) to denote \( a \in \text{Supp}(m) \) and \( 1 \leq i \leq m(a) \). Whenever \((a_1, \ldots, a_n) \in S^n \), we write \([a_1, \ldots, a_n]\) for the finite multiset: \( a \in S \rightarrow \sharp \{i \text{ s.t. } a_i = a \} \). The empty multiset is \( \emptyset \) and \( \cup \) is the multiset union: \((m \cup p)(a) \triangleq m(a) + p(a) \). A vector \( v \in \mathbb{R}_+^S \) is given by its values \( v_a \) on any index \( a \in S \). Given a multiset \( m \in M_t(S) \), we define the power \( v^m \triangleq \prod_{a \in \text{Supp}(m)} v^m(a) \). For any \( a \in S \), let \( e_a \in \mathbb{R}^S_+ \) be the base vector \((e_a)_a \triangleq \delta_{a,b} \), with \( \delta_{a,b} \) denoting the Kronecker symbol.

Let us fix our typographic conventions: \( a, b, c \) range over web elements and \( m, p, q \) over multisets, \( v, w, u \) range over vectors, \( \phi, \psi, \xi \) over matrices, \( \alpha, \beta, \gamma \) over monomials. \( A, B, C \) range over simple types, \( \text{Int} \) being the integer type. \( x, y, z \) range over term variables. \( M, N, P \) range over terms. Finally, \( \kappa, \mu, \nu \) range over scalars in \( \mathbb{R}_+ \) and \( \kappa \) denotes a list of scalars and similarly for the other metavariabes.

2. Denoting Probabilistic Programming

Probabilistic programs use two levels of randomness: the first one on data (i.e. terms of ground type), the second one on programs (i.e. terms of higher-order type).

A probabilistic datum is a random variable whose outcome is given to the program. Thus, a probabilistic datum will be characterized by its law, that will be its interpretation. Then, a probabilistic program uses random instructions and behaves like a function from probabilistic data to probabilistic data. Moreover, a program can call several times its argument \( x \). Each of its occurrences is represented by the outcome of an independent random variable with the same distribution as \( x \).

We consider two examples of probabilistic data of type \( \text{Int} \). The first one, \( \text{Coin} \), is the toss of a \( 0/1 \) fair coin. The second one, \( \text{Rand}(n) \), follows the discrete uniform distribution with outcomes between \( 0 \) and \( n-1 \). If we are only interested in resulting values, then \( \text{Int} \) is interpreted by the set of non negative integers \([\text{Int}] = \mathbb{N} \). First, we consider \( \text{Coin} \) and \( \text{Rand}(n) \) as non deterministic data, and interpret them by the range of the corresponding random variables:

\[ |\text{Coin}| = \{0, 1\} \quad \text{and} \quad |\text{Rand}(n)| = \{0, \ldots, n-1\}. \]

Then, to take into account the randomized behavior of the datum, we associate a coefficient to each outcome: its probability to happen. The interpretation of a datum is now given by a sequence of non-negative integers indexed by the possible outcomes:

\[ [\text{Coin}] = \left\{ 0, \frac{1}{2}, \frac{1}{2}, 0, \ldots \right\} \quad \text{and} \quad [\text{Rand}(n)] = \left\{ \frac{1}{n}, \ldots, \frac{1}{n}, 0 \right\}. \]

In general, we will interpret the integer type by subprobability\(^1\) distributions over \( \mathbb{N} \):

\[ P(\text{Int}) = \left\{ (\kappa_n)_{n \in \mathbb{N}} \in \mathbb{R}_+^\mathbb{N} \text{ s.t. } \sum_{n \in \mathbb{N}} \kappa_n \leq 1 \right\}. \]

For probabilistic programs, we follow the same pattern as for probabilistic data. First, their non deterministic behavior is described. Then, coefficients are introduced in order to render their quantitative behavior.

As an example, let us consider the program \( \text{Rand} : \text{Int} \Rightarrow \text{Int} \) that takes an input value \( n \), and returns any non negative integer strictly less than \( n \) with equal chances and make it interact with probabilistic data (e.g. a probabilistic distribution over \( \mathbb{N} \)).

Focusing on the association between input and output values, its non deterministic behavior is described as a relation: \([\text{Rand}] \subseteq [\text{Int}] \times [\text{Int}]\):

\[ [\text{Rand}] = \left\{ (n, a) \text{ s.t. } n \in \mathbb{N}, a \in \{0, \ldots, n-1\} \right\}. \]

Then we associate a coefficient to each pair input-output. It represents the quantitative account of getting the given output knowing the input. The interpretation of the program is now turned into a matrix indexed by the values interpreting inputs (column indices) and outputs (row indices)\(^2\):

\[ [\text{Rand}] \in [\mathbb{R}_+]^{\mathbb{N} \times \mathbb{N}}. \]

The interaction between the program \( \text{Rand} \) and a probabilistic datum \( x \) is then given by the product of the matrix interpreting the program and of the sequence interpreting the datum. Besides, this probabilistic program preserves subprobability distributions.

Actually, this approach is valid only if the program uses exactly once its argument. Indeed, as a side effect of the call-by-name\(^3\) execution strategy, each occurrence of a probabilistic datum behaves as an independent sample of a random variable. So, if a program makes several calls to a given probabilistic datum, then the outcomes of the calls may differ due to the randomized setting. Thus, we gather the input values into a finite multiset: finite, since if the execution of a program terminates, then the number of resources effectively used is finite; multiset rather than a sequence since this model is not accurate enough to distinguish the order of the inputs.

To illustrate this point, we examine the probabilistic programs:

\[ \begin{align*}
\text{Once} & \triangleq \lambda x. \text{Int}. \text{if}(x, \text{Coin}, 42), \\
\text{Twice} & \triangleq \lambda x. \text{Int}. \text{if}(x, \text{Coin}, 42), \text{if}(x, 42, 0)),
\end{align*} \]

where \((x, \_\_\_)\) branches depending whether \( x \) evaluates to zero or not. Notice that the two programs uses, among others, probabilistic

\(^1\) Since a call to a datum can fail, the total probability distribution over the possible values may be less than 1.

\(^2\) \([\text{Rand}]^T\) is the transpose of a stochastic matrix and the image of a subprobability distribution is the matrix product \([\text{Rand}]^T \cdot v\).

\(^3\) This phenomenon will also appear in a call-by-value setting, if every probabilistic datum \( x \) is replaced by its CPS translation \( \lambda_0.\text{Int}.x \).
definitions the probability that 
0
by adding to standard
PCF
3. Probabilistic PCF

The value of \( P_{\text{ROBA}} \) is intuitively described as the probability of evolving from the state \( M \) to the state \( N \) in one step. A term is absorbing whenever \( P_{\text{ROBA}} = 1 \); the absorbing states are those which are invariant under the transition matrix. Notice that the normal forms are all absorbing, but the converse is false, e.g. \( \Omega \) is an absorbing term.

The \( n \)-th power \( P_{\text{ROBA}}^n \) of the matrix \( P_{\text{ROBA}} \) is a stochastic matrix on PPCF (in case \( n = 0 \), we have the identity matrix on PPCF). Intuitively, the value of \( P_{\text{ROBA}}^n \) is the probability of evolving from the state \( M \) to the state \( N \) in exactly \( n \) steps.

**Proposition 4** ([4, Lemma 32]). Let \( M \in \text{PPCF} \) and \( N \) absorbing, the sequence \( \{P_{\text{ROBA}}^n\}_{n \in \mathbb{N}} \) is monotonic.

We can thus define, for every program \( M \) and \( n \in \mathbb{N} \):

\[
\text{Prob}_{\text{ROBA}}^n \equiv \sup_{k=0}^{\infty} P_{\text{ROBA}}^k \quad (4)
\]

Intuitively, \( \text{Prob}_{\text{ROBA}}^n \) defines the probability that \( M \) reaches a numeral \( \sharp \) in an arbitrary number of steps.

In standard PCF the observational pre-order is defined with respect to the termination of a term in a context of type \( \text{Int} \). In a probabilistic framework like PPCF, one can refine such a pre-order...
Types $A, B, C ::= \text{Int} \mid A \Rightarrow B$  
Terms $M, N, P ::= x \mid \lambda x^A.M \mid (M) N \mid \text{fix}(M) \mid \emptyset \mid \text{s}(M) \mid \text{p}(M) \mid \text{if}(M, N, P) \mid \text{rand}$

(a) Grammar of types and terms. The constant Int is the base type of integers. Given $n \in \mathbb{N}$, $\mathbb{N}$ will denote its associated numeral, defined as $s^n(\emptyset)$.

(b) Simple type assignment system.

(c) Reduction rules. In case $n = 0$, (rand) $\emptyset$ is a normal form.

(d) Notational conventions. Also, we can use choose$(M_i)_{i=1}^n$ as a shortcut for choose$(M_1, \ldots, M_n)$.

Figure 1: Probabilistic extension of PCF.
express probabilistic data. At the ground type, \( \text{P}(\text{Int}) \) is in fact the convex set of the subprobabilistic distributions of the elements in \([\text{Int}]\). However, at higher-order types, this intuition is lost\(^5\) and the definition of \( \text{P}(A) \) depends on a duality condition describing the probability of having an interaction between a datum of type \( A \) (which is a program) and an environment.

Consider a program and an environment interacting on atomic data \( a \) included in a set \([A]\) such that their respective interpretations are positive real vectors: \( v, w \in \mathbb{R}^{[A]} \). Their pairing gives a quantitative estimation of the interaction success

\[
\langle v, w \rangle \triangleq \sum_{a \in [A]} v_a w_a \in \mathbb{R}_+.
\]

We say that their interaction is probabilistic whenever \( \langle v, w \rangle \leq 1 \). We then define a polar operation on sets of vectors \( P \subseteq \mathbb{R}_+^{[A]} \) as

\[
P^\perp \triangleq \{ w \in \mathbb{R}^{[A]} \mid \forall v \in P \langle v, w \rangle \leq 1 \}.
\]

The probabilistic duality environment/program is then enforced in our model by the closedness condition \( P^{\perp \perp} = P \).

**Definition 7** ([4, 14]). A probabilistic coherence space, or PCS for short, is a pair \( A = ([A], \text{P}(A)) \) where \([A]\) is a countable set called the web of \( A \) and \( \text{P}(A) \) is a subset of \( \mathbb{R}_+^{[A]} \) satisfying:

1. \( \text{P}(A)^{\perp \perp} = \text{P}(A) \).
2. \( \forall a \in [A], 3a > 0, \forall v \in \text{P}(A), v_a \leq a \).
3. \( \forall a \in [A], 3a > 0, 3\varepsilon_a \in \text{P}(A) \).

As a side effect, Condition 1 (forces \( \text{P}(A) \) to be a convex set. Condition 2 requires the projection of \( \text{P}(A) \) in any direction to be bounded, while 3 forces \( \text{P}(A) \) to cover every direction.\(^6\)

**Example 8.** The set \( \text{P}(\text{Int}) \) of subprobability distributions over \( \mathbb{N} \) yields a PCS. In particular, \( \text{P}(\text{Int})^{\perp \perp} = \text{P}(\text{Int}) \), its polar being \( \text{P}(\text{Int})^\perp = [0, 1]^\mathbb{N} \).

**4.2 The category PCoh**

The objects of \( \text{PCoh} \) are the PCSs and the set \( \text{PCoh}(A, B) \) of morphisms from \( A \) to \( B \) is the set of matrices \( \phi \in \mathbb{R}_+^{\text{M}([A]) \times [B]} \) such that \( \forall v \in \text{P}(A), (\phi(v)) \in \text{P}(B) \), where

\[
(\phi(v))_b \triangleq \sum_{m \in \text{M}([A])} \phi_{m,b} \cdot v^m,
\]

(see Notation 1 for the definition of \( v^m \)). Following Section 2, a morphism is presented as a matrix that gathers the coefficients of the power series described by the Equation (7).

The identity on \( A \) is given by the matrix

\[
\text{Id}^A_{m,a} \triangleq \begin{cases} 1 & \text{if } m = [a], \\ 0 & \text{otherwise.} \end{cases}
\]

In fact, we have \( \text{Id}^A = [0, 1]^{[A]} \).

Let \( \phi \in \text{PCoh}(A, B) \) and \( \psi \in \text{PCoh}(B, C) \), their composition must satisfy: \( \forall v \in \text{P}(A), (\psi \circ \phi)(v) = \psi(\phi(v)) \). In matricial terms, this comes out as: \( \forall v \in [C], \psi \in [C] \),

\[
\sum_{m \in \text{M}([A])} (\psi \circ \phi)_{m,c} \cdot v^m = \sum_{p \in \text{M}([B])} \psi_{p,c} \cdot \left( \sum_{q \in \text{M}([A])} \phi_{q,b} \cdot v^q \right)^{p(b)}.
\]

Now, to extract the coefficient of the monomial \( v^m \), we distribute product over sum. This amounts to choosing a partition of \( m = \bigcup(q(b,i))_{b,i} \in \text{M}([B]) \) matching the enumeration \( \{b,i\} \text{ s.t. } b \in \text{Sup}(p), i \leq p(b) \text{ of } p \).

Therefore, the composition \( \psi \circ \phi \) is defined\(^7\) as the matrix coefficients, for \( m \in \text{M}(A) \) and \( c \in [C] \):

\[
(\psi \circ \phi)_{m,c} \triangleq \sum_{p \in \text{M}(B)} \psi_{p,c} \prod_{(b,i) \in \text{M}(B)} \phi_{(b,i),b}.
\]

**4.3 PCoh is Cartesian Closed**

The cartesian product of any countable family \( (A_i)_{i \in I} \) of PCSs is:

\[
\prod_{i \in I} A_i \defeq \bigcup_{i \in I} \{ i \times A_i \},
\]

\[
P(\prod_{i \in I} A_i) \defeq \{ v \in \mathbb{R}_+^{\prod_{i \in I} A_i} \mid \forall i \in I, \pi_i(v) \in P(A_i) \},
\]

where \( \pi_i(v) \) is the vector in \( \mathbb{R}_+^{[A_i]} \) denoting the \( i \)-th component of \( v \), i.e. \( \{\pi_i(v)\}_{a} \triangleq v_{(i,a)} \).

The \( j \)-th projection \( P_j \in \text{PCoh}(\prod_{i \in I} A_i, A_j) \) and the product \( \langle \phi_i \rangle_{i \in I} \in \text{PCoh}(B, \prod_{i \in I} A_i) \) are given by:

\[
P_j \triangleq \begin{cases} 1 & \text{if } m \in \{j, [a]\}, \\ 0 & \text{otherwise.} \end{cases}
\]

The terminal object \( 1 \) is given by the empty product \( \{\emptyset, \emptyset\} \). Notice that the set of points of a PCS \( A \), i.e. the set \( \text{PCoh}(1, A) \), is isomorphic to the convex set \( \text{P}(A) \).

**Notation 9.** We write \( A_1 \times A_2 \) for the binary product: in the sequel, we present any \( v \in \text{P}(A_1 \times A_2) \) as the pair \( (\pi_1(v), \pi_2(v)) \) in \( \text{P}(A_1) \times \text{P}(A_2) \) of its components.

Notice that the set \( \mathcal{M}_I(\prod_{i \in I} A_i) \) is isomorphic to the set-theoretic cartesian product \( \prod_{i \in I} (\mathcal{M}_I([A_i])) \) via the map associating any \( m \in \mathcal{M}_I(\prod_{i \in I} A_i) \) with the \( I \)-indexed family \( (m_i)_{i \in I} \) defined as \( m_i(a) \triangleq m(i,a) \). This means that any morphism \( \phi \in \text{PCoh}(\prod_{i \in I} A_i, B) \) can be presented as a matrix indexed by sequences in \( \prod_{i \in I} (\mathcal{M}_I([A_i]) \times [B]) \).

The object of morphisms is defined as

\[
|A \Rightarrow B| \triangleq \mathcal{M}_I(\{A\} \times [B]), \quad \text{P}(A \Rightarrow B) \triangleq \text{PCoh}(A, B).
\]

\( \text{PCoh} \) is then turned into a cartesian closed category by the evaluation \( \text{Ev} \in \text{PCoh}(\{A \Rightarrow B \} \times A, B) \) and the currying \( \text{Cur}(\phi) \in \text{PCoh}(\{A \Rightarrow B \} \times A) \), for every \( \phi \in \text{PCoh}(A \times B, B) \), defined as:

\[
\text{Ev}(m,p,a) \triangleq \begin{cases} 1 & \text{if } m = \{p, [a]\}, \\ 0 & \text{otherwise.} \end{cases}
\]

\( \text{Cur}(\phi)_{m,p,b} \triangleq \phi(m,p,b) \).

Notice that the above equations use Notation 9, e.g. representing a multiset in \( \mathcal{M}_I(\{A \Rightarrow B \} \times A) \) as a pair \( (m, p) \) of multisets in \( \mathcal{M}_I(\{A \Rightarrow B \}) \times \mathcal{M}_I([A]) \).

Actually, the category \( \text{PCoh} \) is well-pointed in the sense that the equality of morphisms is extensional, i.e. given two matrices \( \phi, \psi \in \text{PCoh}(A, B) \), if for every \( v \in \text{P}(A), \phi(v) = \psi(v) \), then \( \phi = \psi \). To sum up, we have:

\(^5\)\( \text{P}(A) \) is not anymore a subset of \([0, 1]^{[A]} \), see discussion in Section 2.

\(^6\)These conditions are introduced in [4] for keeping finite all the scalars involved, yet they are not explicitly stated in the definition of PCS in [14].

\(^7\)The definition of Equation (8) is due to [17]. On a side note, remark that in [4], the authors use another formulation in which identical sums/parts produced by different partitions are gathered. This gives rise to multinomial coefficients that are hidden in the sums of the present formulation. However, the two definitions give rise to the same matrix.
Proposition 10 ([4, §1.6]). \( \text{PCoh} \) is a well pointed cartesian closed category.

4.4 Object of numerals and Cpo-Enrichment

The object of numerals of \( \text{PCoh} \) is the PCS \( \text{Int} \) \( \bowtie (\mathbb{N}, P(\text{Int})) \) equipped with the morphisms \( z \in \text{PCoh}(1, \text{Int}) \cong P(\text{Int}) \), \( \text{pred}, \text{succ} \in \text{PCoh}(\text{Int}, \text{Int}) \), and \( \text{ifz} \in \text{PCoh}(\text{Int} \times \text{Int} \times \text{Int}, \text{Int}) \) defined as

\[
z_a = \delta_{0,n}, \quad \text{pred}_{m,n} = \delta_{m,[n+1]}, \quad \text{succ}_{m,n} = \delta_{m,[n]},
\]

\[
\text{ifz}_{(m,p,q),n} = \begin{cases} 1 & \text{if } (m,p,q) = ([0],[n],[n]), \\ 0 & \text{otherwise.} \\
\end{cases}
\]

The natural order on \( \mathbb{R}_+ \) enriches \( \text{PCoh} \) with a cpo-structure, defined componentwise on morphisms: i.e., given \( \phi, \psi \in \text{PCoh}(A, B) \)

\[
\phi \leq \psi \iff \forall m \in M(A), \forall b \in M(B), \phi(m,b) \leq \psi(m,b).
\]

The matrix \( 0 \) is the minimum element and the lub of a directed net \( \{(\phi_d)_{d} \in D \) is then given by

\[
\left( \sup_{d \in D} (\phi_d)_{m,b} \right) \Delta = \sup_{d \in D} (\phi_d)_{m,b}.
\]

Remark 11. The componentwise order on matrices is not extensional, i.e. there are \( \phi, \psi \in \text{PCoh}(A, B) \) such that \( \forall v \in P(A), \phi(v) \leq \psi(v) \), but \( \phi \not\leq \psi \). For example, take \( \phi, \psi \in \text{PCoh}(\text{Int}, \text{Int}) \):

\[
\phi_{m,n} = \begin{cases} 1 & \text{if } m = [k], n = 0, \\ 0 & \text{otherwise.} \\
\end{cases}
\]

Although \( \phi \not\leq \psi \), for every subprobability distribution of natural numbers \( v \in P(\text{Int}) \), we get \( \phi(v) = (\sum_{1}^{n} v_k, 0, 0, \ldots) \leq (1, 0, 0, \ldots) = \psi(v) \). In fact, we will use this mismatch for disproving the inequality full abstraction in Section 6.

4.5 \( \text{PCoh} \) is an Adequate Model of PCF

The model of PCF is obtained by extending the usual categorical interpretation of PCF to \( \text{rand} \).

With a type \( A \), we associate a PCS \( A \), by induction on the type:

\[
\text{Int} \Rightarrow \text{Int} \quad A \Rightarrow B \Rightarrow A.
\]

Let \( \Gamma = x_1 : A_1, \ldots, x_n : A_n \). The interpretation of a judgment \( \Gamma \vdash M : B \) is a morphism \( [M]^{\Gamma} \in \text{PCoh}(\prod_{i=1}^{n} A_i, B) \), defined in Figure 2 by structural induction on the unique derivation of \( \Gamma \vdash M : B \).

The fix-point operator \( \text{fix}(M) \) is the lub of its approximants, given by induction by

\[
\text{fix}^0 \phi \Delta = 0, \quad \text{fix}^{n+1} \phi \Delta = \text{Ev} \circ \langle \phi, \text{fix}^n \phi \rangle.
\]

The operator \( \text{rand} \) is defined by (using a multiset variant of) the matrix \( \text{Rand} \) of Equation (1):

\[
\text{[rand]}_{m,k} \Delta = \begin{cases} 1 & \text{if } m = [n], k < n, \\ 0 & \text{otherwise.} \\
\end{cases}
\]

Together with the categorical interpretation of a term, we describe its action on the vectors in the convex set associated with its input type. Notice that, since the category is well pointed of the category, this action univocally determines the interpretation of the term. Notice also that the matrices interpreting the basic constructs in Figure 2 have 0, 1 coefficients, except for \( \text{[rand]} \) which is interpreted as the random function that introduces rational numbers in \([0, 1]\). Coefficients greater than 1 may be produced by composition of morphisms (Equation 8).

Thanks to Notation 9, \([M]^{\Gamma} \) can be described as a vector indexed by a tuple \((m, b) \) of \( m \in M(\Gamma) \) and a web element \( b \in B \). This convention will be used hereafter.

Example 12. If Coin is identified with \( \text{[rand]} \), then the interpretation of \( \text{Once} \) and \( \text{Twice} \) are the matrices given in Section 2.

Consider the term \( \Omega_A \Delta = \text{fix}^{\infty}(\lambda x. x . x) \). Notice that \( [\lambda x. x . x] \) is different from zero only on the web elements of the form \((m, a) \), so that \( \text{fix}^n [\lambda x. x . x](0) = 0 \) for any natural number \( n \). We conclude \( [\Omega_A] = 0 \), as expected.

Consider now the term \( P \Delta = \text{fix}^{\infty}(\lambda x. x \equiv 0 . x) \). We have that \( [\lambda x. x \equiv 0 . x](m, a) = 1 + 1 \), \( [\lambda x. x \equiv 0 . x](m, a) = 1 \), i.e. it is equal to \( 1 \) when \( m \neq a \), or when \( m = a \) and \( a = 0 \), otherwise it is equal to 0. This means that \( \text{fix}^n [\lambda x. x \equiv 0 . x](0) = 1 \) for any \( n \). We conclude that \([P] = \text{sup}_{n} \text{fix}^n [\lambda x. x \equiv 0 . x](0) = 1 \).

Proposition 13 (Soundness [4]). The semantics is invariant under reduction, i.e. for every \( \Gamma \vdash M : B \):

\[
[M]^{\Gamma} = \sum_{N} \text{PROBA}_{M,N} [M]^{\Gamma}.
\]

Proof (Sketch). The invariance under reduction rules of the standard PCF redexes follows by cartesian closedness and the cpo-enrichment of \( \text{PCoh} \). The soundness of the reduction of \( \text{[rand]} \) is straight from the definition of \( \text{[rand]} \). The soundness of the context rules depends on the fact that the interpretation of a context is linear in the argument associated with the fired redex. For example, the soundness of the context rule associated with application depends on the equality: \( \text{Ev} \circ \langle \phi + \mu \psi, \xi \rangle = \kappa(\text{Ev} \circ \langle \phi, \xi \rangle) + \mu(\text{Ev} \circ \langle \psi, \xi \rangle) \).

Theorem 14 (Adequacy [4]). Let \( M \) be a closed term of type \( \text{Int} \). Then, \( [M]^{\Gamma} = \text{sub-probability distribution on } N \) such that

\[
\forall n \in N, \ [M]^{\Gamma} \equiv \text{PROBA}_{M,N}.
\]

Remark 15. Adequacy allows one to prove that specific primitives are not definable in the language. A noteworthy example is the parallel or function [27]. In our setting, this should be a closed term \( \text{por} : \text{Int} \Rightarrow \text{Int} \Rightarrow \text{Int} \) such that \( \text{PROBA}_{\text{por} \equiv 0 \equiv 0}^{\text{por} \equiv 0 \equiv 0} = 1 \). By adequacy, \( \text{por} \equiv 0 \equiv 0 = \text{por} \equiv 0 \equiv 0 = \text{por} \equiv 0 \equiv 0 = 1 \). By definition of a morphism in \( \text{PCoh} \), \( [\text{por} \equiv 0 \equiv 0] \) must be a subprobability distribution, hence \( [\text{por} \equiv 0 \equiv 0] = 0 \). That implies \( [\text{por} \equiv 0 \equiv 0] < 0 \). On the other hand, \( [\text{por} \equiv 0 \equiv 0] \geq [\text{por} \equiv 0 \equiv 0] > [\text{por} \equiv 0 \equiv 0] = 2 \), which contradicts that \( [\text{por} \equiv 0 \equiv 0] > [\text{por} \equiv 0 \equiv 0] = 2 \). We conclude that \( \text{por} \equiv 0 \equiv 0 \) is not a term of \( \text{PCF} \).

However, let us mention that the Gustave function is a valid morphism of \( \text{PCoh} \) (see [13]).

Full abstraction extends to higher-order types the perfect matching syntax / semantics stated by adequacy on \( \text{Int} \). One direction of full abstraction is indeed a consequence of Theorem 14.

Corollary 16 (Abstraction). Given \( \Gamma \vdash M : A \), and \( \Gamma \vdash N : A \), we have that \([M] \leq [N]\) implies \( M \equiv \Gamma \ N \). In particular, \([M] = [N]\) implies \( M \equiv \Gamma \ N \).

Proof. By induction on \( C[\cdot] \), one proves that \([M] \leq [N]\) implies \( [C[M]] \leq [C[N]] \). Then the result follows from Theorem 14.

5. Full Abstraction

We prove equational full abstraction (Theorem 27), that is the converse of the part of Corollary 16 dealing with equality. This is a straightforward consequence of Lemma 26 stating that for any
closed terms \(M\) and \(N\) having different interpretations in \(\text{Pcoh}\), there is a testing term \(P\) such that \((P)\ M\) and \((P)\ N\) reduce to \(\bot\) with different probabilities. Let us outline the path to this result.

With any web element \(a\), we associate a testing term \(P(a)\) that is described in Figure 5 and that reminds the contexts used in [3]. \(P(a)\) is not an ordinary term of PPCF since its construction uses a random operator, weighted by a list \(X\) of formal parameters. However, a parameterized term becomes an ordinary PPCF term when we substitute \(X\) by a list \(\bar{\kappa}\) of rationals in \([0, 1]\). Following [6], we introduce in Figure 3 an intersection type system that defines semantics of parameterized terms. This interpretation is a formal power series over \(\bar{\kappa}\) (Definition 21). Moreover, parameterized semantics is compatible with \(\text{Pcoh}\) through substitution of parameters (Lemma 20).

Now, assume that \(M\) and \(N\) are interpreted by different matrices and pick a web element \(a\) such that \([M]\) \(\neq [N]\) a. Then, the semantics of the parameterized terms \((P(a))\ M\) and \((P(a))\ N\) are distinct formal power series. Indeed, Lemma 24 implies that they differ at least on one coefficient.

Finally, Lemma 25 ensures the existence of a list \(\vec{\kappa}\) of rationals in \([0, 1]\) on which these power series disagree. So, the substitution of \(\vec{X}\) by \(\vec{\kappa}\) in \(P(a)\) produces a testing context separating \(M\) and \(N\).

We first describe parameterized PPCF in Subsection 5.1. Then, testing terms are presented in Subsection 5.2. We conclude in Subsection 5.3 with the full abstraction theorem.

### 5.1 Parameterized PPCF

Let \(\mathfrak{Q}\) be a denumerable set of formal parameters. \(X, Y, Z\) range over parameters in \(\mathfrak{Q}\).

The **grammar** of parameterized PPCF is an extension of PPCF (Figure 1(a)) by multiplication of terms by parameters:

\[
\text{PPCF}_\mathfrak{Q} := \cdots \mid X \cdot M, \text{ where } X \in \mathfrak{Q}.
\]

The **simple type** of \(X \cdot M\) under a context \(\Gamma\) is \(A\) whenever \(\Gamma \vdash M : A\) in PPCF (Figure 1(b)).

The **substitution** of parameters by scalars let us recover ordinary terms from parameterized ones. More precisely, let \(M \in \text{PPCF}_\mathfrak{Q}\) and \(\vec{x} \in [0, 1]\) be a rational number. We define \(M(\vec{x}/X)\) as the term obtained by replacing in \(M\) with subterm of shape \(X \cdot N\) with

\[
\text{choose}(N^n, \Omega_{m-n}) \triangleq \text{choose}(N, \ldots, N, \Omega, \ldots, \Omega) \quad (9)
\]

(see Figure 1(d) for choose definition). Substitution is then generalized to lists \(\bar{X}\) of parameters and \(\vec{\kappa}\) of rationals as \(M(\vec{\kappa}/\bar{X})\).

**Fact 17.** If \(\bar{X}\) is the list of all parameters in \(M\) and \(\vec{\kappa}\) a list of rational numbers in \([0, 1]\), then \(M(\vec{\kappa}/\bar{X})\) is a term of PPCF.

The **semantics** of \(\text{PPCF}_\mathfrak{Q}\) is a refinement of PPCF semantics taking into account parameters. Yet, for the sake of the full abstraction proof, we give a different presentation and use a weighted intersection type system. Roughly speaking, types are web elements and with each type derivation \(\pi\), we associate a weight \(\omega(\pi)\) which is a positive monomial, i.e. a product of rationals in \([0, 1]\) and of finitely many parameters. Then, the interpretation \([M]\) of a \(\text{PPCF}_\mathfrak{Q}\) term \(M\) is a matrix indexed by web elements. For each web element, there can be several type derivations and the corresponding coefficient is the sum of their weights.

More precisely, Figure 3 describes the rules for constructing a derivation \(\pi : \Gamma^\ast \vdash_a M : a\) of what we call a web judgment \(\Gamma^\ast \vdash_a M : a\). Notice that \(\Gamma^\ast \vdash_a M : A\) is a valid simple type judgment, \(a \in [A]\) and \(a\) is a monomial. Besides, a web context \(\Gamma^\ast\) is defined as a function mapping any typed variable \(x^C\) occurring in \(\Gamma\) to a finite multisets \(m \in M_t([C])\) of web elements and mapping variables non-appearing in \(\Gamma\) to the empty multiset. For instance, for any \(\Gamma = x_1 : C_1, \ldots, x_n : C_n\) and \(\bar{m} \in M_t([\Gamma])\), \(\Gamma^{\#}\) denotes the web context \(x_i^{\#} \mapsto \bar{m}_i\), for \(1 \leq i \leq n\). Disjoint unions \(\Gamma^\ast \cup \Delta^\ast\) of web contexts are defined pointwise.

**Fact 18.** If \(\Gamma^\ast \vdash_a M : a\) is derivable and \(\Gamma^\ast(x)\) is a non-empty multiset, then \(x\) is free in \(M\).

Rules app and fix deserve some comments. Application of \(\text{PPCF}_\mathfrak{Q}\) terms is interpreted following Equation (8) that defines composition. Remark that indices of \(\text{app}\) mainly coincide with the indices of the sums in (8), with one more difficulty since we have to split contexts. Now, fix is derived from the rule app and from

---

**Figure 2:** The standard semantics of PPCF terms together with its action on \(\varepsilon \in P(\Gamma)\).

---

![Diagram](image-url)

(Insert diagram here if possible)
\[ x^A : [a] \vdash_1 x : a \quad \text{var} \quad \vdash_1 \mathbb{B} : n \quad \text{nat} \quad \vdash_1 \mathbb{I} : \text{rand} : ([n], k) \quad \text{rand} \quad \Gamma_* \vdash_1 \alpha : M : a \quad \text{abs} \quad \Gamma_* \vdash_1 \lambda x^A.M : (m, a) \]

\[ \Gamma^* \vdash_1 M : (m, b) \quad \forall (a, i) \in m, \quad \Gamma^* \vdash_1 \beta_{\langle a, i \rangle} N : a \quad \text{app} \quad \text{s.t.} \quad \begin{cases} m \in \mathcal{M}_A ([\mathcal{A}]) \\ \Gamma^* \cup \bigcup_{(a, i) \in m} \Gamma^*_{\langle a, i \rangle} = \Gamma^* \end{cases} \]

\[ \Gamma^* \vdash_1 M : (m, b) \quad \forall (a, i) \in m, \quad \Gamma^* \vdash_1 \beta_{\langle a, i \rangle} \text{fix}(M) : b \quad \text{fix} \quad \text{s.t.} \quad \begin{cases} m \in \mathcal{M}_A ([\mathcal{A}]) \\ \Gamma^* \cup \bigcup_{(a, i) \in m} \Gamma^*_{\langle a, i \rangle} = \Gamma^* \end{cases} \]

\[ \Gamma^* \vdash_1 M : n + 1 \quad \text{pred} \quad \Gamma^* \vdash_1 \mathbb{S}(M) : n + 1 \quad \text{succ} \quad \Gamma^* \vdash_1 \alpha : a \quad \text{par} \]

\[ \Gamma^* \vdash_1 M : 0 \quad \Delta^* \vdash_1 N : a \quad \text{if}_0 \quad \Gamma^* \vdash_1 M : n + 1 \quad \Delta^* \vdash_1 P : a \quad \text{if}_s \quad \Gamma^* \vdash_1 \mathbb{X} : a \quad \text{par} \]

**Lemma 20 (Soundness).** Let \( M \) be a term of PPCF such that \( \Gamma \vdash M : B \) and \( \bar{X} \) lists its parameters. For any \( \bar{p} \in \mathcal{M}_f ([\Gamma]) \), \( b \in \mathcal{B} \), and any list \( \bar{r} \) of rational numbers in \([0, 1] \),

\[ \left[ M \bar{r}/\bar{X} \right]^{\bar{p}, b} = \left( \sum_{\bar{x} : \mathbb{F}_b \vdash_1 M} \omega(\bar{x}) \right) (\bar{r}) \]  

(10)

**Proof.** We prove Equation (10) by structural induction on \( M \), using rules of Figure 3 and the definition of the categorical interpretation of a term in PPCF. We consider the term \( \text{fix}(M) \) greater than \( (MM) \) \( y \) for any natural number \( n > 0 \) and variable \( y \). In fact, we are using the induction on the ordinal \( \omega \).

Most cases follow directly from induction hypothesis. We only detail three cases: (i) multiplication with a parameter, (ii) application, (iii) fix-point.

(i) Assume \( M = X, N \) and \( \nu_i = \frac{r_i}{n} \). Then (using Equation (9)) we get:

\[ \left[ M \bar{r}/\bar{X} \right]^{\bar{p}, b} = \left( \sum_{\bar{x} : \mathbb{F}_b \vdash_1 M} \omega(\bar{x}) \right) (\bar{r}) \]

(ii) Assume \( M = X, N \) and \( \nu_i = \frac{r_i}{n} \). Then (using Equation (9)) we get:

\[ \left[ M \bar{r}/\bar{X} \right]^{\bar{p}, b} = \left( \sum_{\bar{x} : \mathbb{F}_b \vdash_1 M} \omega(\bar{x}) \right) (\bar{r}) \]

Now, any derivation \( \tau : \Gamma^* \vdash M \bar{r}/\bar{X} \) : \( b \) consists of a sequence of \( h \leq n \) rules if, with on top, the rule if _0 with a derivation \( \pi : \Gamma^* \vdash N \bar{r}/\bar{X} : b \) as right premise. Indeed, the subterm \( \Omega \) has no web type (see Example 23). When \( \pi \) is fixed, \( \omega(\tau) = \frac{\omega(\pi)}{m} \) and there are as many derivations \( \tau \) as many choices of \( \bar{h} \in \{1, \ldots, n\} \). So, we compute

\[ \sum_{\tau : \Gamma^* \vdash_1 M} \omega(\tau) \left[ \bar{r}/\bar{X} \right]^{\bar{p}, b} = \frac{m}{\sum_{\tau : \Gamma^* \vdash_1 M} \omega(\tau)} \left[ \sum_{\tau : \Gamma^* \vdash_1 M} \omega(\tau) \right] (\bar{r}) \]

which is equal to \( \frac{m}{\sum_{\tau : \Gamma^* \vdash_1 M} \omega(\tau)} \left[ \sum_{\tau : \Gamma^* \vdash_1 M} \omega(\tau) \right] (\bar{r}) \) by induction hypothesis.

(iii) Assume \( M = X, N \), \( \nu_i = \frac{r_i}{n} \). Then (using Equation (9)) we get:

\[ \left[ M \bar{r}/\bar{X} \right]^{\bar{p}, b} = \left( \sum_{\bar{x} : \mathbb{F}_b \vdash_1 M} \omega(\bar{x}) \right) (\bar{r}) \]

By induction hypothesis and distributing product over sum, we get

\[ \sum_{\tau : \Gamma^* \vdash_1 M} \omega(\tau) \left[ \bar{r}/\bar{X} \right]^{\bar{p}, b} = \left( \sum_{\tau : \Gamma^* \vdash_1 M} \omega(\tau) \right) (\bar{r}) \]

Although the indices of the sums might be frightening, they precisely describe all possible derivations of \( \Gamma^T \vdash \langle N \rangle \) : \( b \). Namely, the first sum defines the label of the terminal application rule, while the other sums give the choices of the derivations of their premises. The total weight is then the product of premise weights.

(iii) Assume \( M = \text{fix}(N) \). Any derivation \( \pi : \Gamma^T \vdash \text{fix}(N) : b \) with a cluster of \( \text{if} \) rules (see Example 23). For \( n \in \mathbb{N} \), let \( \Pi^n \) be the set of derivations whose cluster height is at most \( n \).

Now, remark that a derivation in \( \Pi^n \) can be transformed into a derivation \( \tau : \Gamma^T, y^B : [1] \vdash (N^n) : b \) (where \( y \) is fresh) by replacing \( \text{if} \) rules with app rules (keeping labels) and each occurrence of \( \text{fix}(N) \) with \( (N^n) \). When \( n \leq h \), besides this transformation preserves weights. Therefore:

\[ \sum_{\tau : \Gamma^T, y^B : [1] \vdash (N^n) : b} \omega(\tau) = \sum_{\tau : \Gamma^* \vdash_1 M} \omega(\tau) \]
Since \( \|\text{fix}(N)\|_{\Gamma, P}^f = \bigvee_{n \in \mathbb{N}} \|\text{fix}(N)\|_{\Gamma, P}^f \), and by induction hypothesis: \( \|\text{fix}(N)\|_{\Gamma, P}^f = \bigvee_{n \in \mathbb{N}} \|\text{fix}(N)\|_{\Gamma, P}^f \), we have:

\[
\|\text{fix}(N)\|_{\Gamma, P}^f = \bigvee_{n \in \mathbb{N}} \sum_{\pi \in \Pi^n} \omega(\pi) = \bigvee_{n \in \mathbb{N}} \sum_{\pi \in \Pi^n} \omega(\pi) = \sum_{\pi \in \Pi^n} \omega(\pi)
\]

According to this result, we generalize the semantics notation to parameterized PPCF:

**Definition 21.** For any term \( M \) in PPCF with \( n \) parameters, \( \|M\|_{\Theta}^{\bullet} \triangleq \sum_{\pi \in \Pi^n} \omega(\pi) \) (11)

It is a power series whose domain of convergence contains \([0, 1]^n\).

The weighted type assignment system of Figure 3 has an interest by its own. It turns computation of semantics of terms (and hence its observational behavior) into a proof search problem.

The reader can convince himself that this approach is useful by computing the semantics of the following examples by applying directly rules of Figure 2.

**Example 22.** Let \( k \) and \( k' \) be non negative integers. Let us consider the possible derivations of the application \( (N)\) with \( M \triangleq \lambda x. x \) of the term occurring with degree \( n \) depends on parameterized PPCF:

\[
\|M\|_{\Theta}^{\bullet} = \sum_{\pi \in \Pi^n} \omega(\pi)
\]

The derivable judgments on the term \( N \) are of the form \( y \vdash (n) : \Gamma, k \) with \( k \in \{k, k'\} \). In fact, once fixed \( \pi \), the derivation is unique and is given at the left-hand side of Figure 4. As for the term \( M \), the derivable judgments are of the form \( y \vdash (n) \vdash N :: (\pi) \vdash \xi \vdash M \) with \( \xi = \{(h, h'), 42\} \), where \( h, h' > 0 \). However, in this case we have two different derivations whenever \( h \neq h' \); one derivation is given at the right-hand side of Figure 4, while the other one is obtained by swapping the order between the var rules on \( h \) and \( h' \).

The weight of a derivation of \( (N) \) is \( 1 \). However, the number of derivation of a fix judgment \( y \vdash (n) \vdash (N) M :: (\pi) \vdash \xi \vdash M \) depends on \( n, n, k, k' \) following cases:

- if \( k = k' \) and \( n = n' \), then there is exactly one derivation;
- if \( k = k' \) and \( n 

\[
\begin{align*}
P(n) &= \lambda x. \text{if}(x = 0, \Omega, \text{fix}(\lambda x. x)) \\
N(n) &= \text{fix}(\lambda x. x)
\end{align*}
\]

\[
\begin{align*}
P(a) &= \alpha \mapsto (\Omega, (\text{fix}(\lambda x. x)(a)) \mapsto 0) \\
N(a) &= (\text{fix}(\lambda x. x)(a) \mapsto 0)
\end{align*}
\]

\[
\begin{align*}
P(\lambda x. \text{if}(x = 0, \Omega, \text{fix}(\lambda x. x))) &= \alpha \mapsto (\Omega, (\text{fix}(\lambda x. x)(\alpha) \mapsto 0)\mapsto 0) \\
N(\lambda x. \text{fix}(\lambda x. x)(\alpha)) &= (\text{fix}(\lambda x. x)(\alpha) \mapsto 0)
\end{align*}
\]

**Figure 5:** testing terms. In the higher-order case, i.e. \( A = B \Rightarrow C \), we suppose \( a \in \{(b_1, \ldots, b_n)\}, \) with \( b_i \in \{B\} \) and \( c \in \{C\} \). Sub-testing terms are supposed to have disjoint parameters and the \( X_i \)’s occurring in the definition of \( P(a) \) are assumed to be fresh.

\[
\begin{align*}
M &\triangleq \lambda x. \text{fix}(\lambda x. x)(a) \\
N &\triangleq \text{fix}(\lambda x. x)(a)
\end{align*}
\]

**5.2 Testing terms**

Figure 5 associates with every web element \( a \in \{A\} \) two closed terms of PPCF with one term \( P(a) \) of type \( \Omega \Rightarrow \text{fix}(\lambda x. x) \) and one term \( N(a) \) of type \( A \) defined by mutual induction on \( A \).

\[
\begin{align*}
P(a) &= \alpha \mapsto (\Omega, (\text{fix}(\lambda x. x)(a)) \mapsto 0) \\
N(a) &= (\text{fix}(\lambda x. x)(a) \mapsto 0)
\end{align*}
\]

**Lemma 23.** Any derivation of a fix-point term \( \text{fix}(M) \) ends with a cluster of \( \text{fix} \) rules, each rule has one premise typing \( M \) and a number of premises typing \( \text{fix}(M) \) and along which the cluster grows. Since the derivation must be finite, the cluster eventually ends with \( \text{fix} \) rules of label \((\), ). They have exactly one premise typing \( M \) with a web element of shape \((\), \( a \). As a consequence, the term \( \Omega = \text{fix}(\lambda x. x)(a) \) has no web type, as \( \lambda x. x \) cannot have a web type of shape \((\), \( a \). We find again \( \Omega = \text{fix}(\lambda x. x)(a) \).

On the other hand, any derivation of \( \text{fix}(\lambda x. x)(a) \mapsto 0 \) will end with a branch of \( n > 0 \) \( \text{fix} \) rules; \( n - 1 \) rules labelled by \((0, 0), \) and the rule at the top of the branch labelled by \((1, 1)\) and having as premise the unique derivation of \( \lambda x. x \mapsto 0 \) \( \Omega \). The whole derivation has conclusion \( \Omega \mapsto 0 \) \( \text{fix}(\lambda x. x)(a) \mapsto 0 \) and the sum of these weights for \( n > 0 \) yields \( \text{fix}(\lambda x. x)(a) \mapsto 0 \) \( \Omega \).

We find again the results of Examples 3 and 12.

**5.3 Testing terms**

In the higher-order case, i.e. \( A = B \Rightarrow C \), we suppose \( a \in \{(b_1, \ldots, b_n)\}, \) with \( b_i \in \{B\} \) and \( c \in \{C\} \). Sub-testing terms are supposed to have disjoint parameters and the \( X_i \)’s occurring in the definition of \( P(a) \) are assumed to be fresh.
The shape of the term forces us to use the application rule again: 

\[
\begin{align*}
\text{IH}(B): b_{ij} & = b'_{ij} \quad \text{(6)} \\
\vdash_{\beta} X_{ij} \cdot N(b_{ij}) : b'_{ij} \quad \text{(7)} \\
\vdash_{\beta} \text{choose}(X_{i} \cdot N(b_{j}))_{i=1}^{k} : b'_{j} \quad \text{(8)} \\
\end{align*}
\]

For the inductive case, let us assume that the result holds for the weight of the conclusion of \(P\). We conclude that the coefficient is non zero iff \(m = [n]\) and \(a' = n\).

For the inductive case, let us assume that the result holds for \(P\). We conclude that the coefficient is non zero iff \(m = [n]\) and \(a' = n\).

\[
\begin{align*}
\text{IH}(B): b_{ij} & = b'_{ij} \quad \text{(6)} \\
\vdash_{\beta} X_{ij} \cdot N(b_{ij}) : b'_{ij} \quad \text{(7)} \\
\vdash_{\beta} \text{choose}(X_{i} \cdot N(b_{j}))_{i=1}^{k} : b'_{j} \quad \text{(8)} \\
\end{align*}
\]

Combining all these results, we eventually get that \(m = [(p', c)] = \{(b_{1}, \ldots, b_{k}, c)\} = [a]\). We conclude that there is a derivation of \(\vdash_{\text{app}}(\text{a}) \cdot N'(a) : a'\), which must have the following form:

\[
\begin{align*}
\vdash_{\alpha} \lambda x.\exists c'. \text{if} (\Lambda_{i=1}^{k} \text{choose}(X_{i} \cdot N(b_{j}))_{i=1}^{k})_{i=1}^{k} : c' \\
\end{align*}
\]

We prove by recursion on \(k - j\) the property RH(j):

\[
\begin{align*}
\vdash_{\alpha} \lambda x.\exists c'. \text{if} (\Lambda_{i=1}^{k} \text{choose}(X_{i} \cdot N(b_{j}))_{i=1}^{k})_{i=1}^{k} : c' \\
\end{align*}
\]

Since by definition no parameter \(X_{i}\) occurs in \(N(b_{j})\), hence no \(X_{i}\) appears in \(\beta_{ij}\), we have that the mapping \(j \mapsto i_{j}\) is a bijective correspondence between \(\{1, \ldots, k\}\) and \(\{1, \ldots, k\}\), so \(k = k'\). Finally, since the sets of parameters occurring in \(N(b_{j})\) are pairwise disjoint, we deduce that \(\beta_{ij}\) is proportional to \(\text{app}(b_{j})\). We can then apply the induction hypothesis on \(\beta\) and deduce that \(b_{ij} = b_{ij}\) and so \(p' = \{b_{i}, \ldots, b_{k}\}\).

The base case \((j = k)\) is the induction hypothesis IH(C). Indeed, if \(x : \vdash_{\alpha} N'(a) : c'\) is derivable, then \(\alpha\) is proportional to \(\text{app}(b_{j})\) if and only if \(c = c'\). The inductive case is proved accordingly to Figure 7:

\[
\begin{align*}
\vdash_{\alpha} \lambda x.\exists c'. \text{if} (\Lambda_{i=1}^{k} \text{choose}(X_{i} \cdot N(b_{j}))_{i=1}^{k})_{i=1}^{k} : c' \\
\end{align*}
\]

(1) Since \(\Omega_{C}\) has no web type (Example 23), the only possible rule is \(\text{app}\). The choice of how to partition the context \(m_{1} \cup m_{2}\) will be deduced from the type derivation. Since by definition the set of parameters in \(\text{app}(b_{j})\) and the set of parameters in \(\text{if} (\Lambda_{i=1}^{k} \text{choose}(X_{i} \cdot N(b_{j}))_{i=1}^{k})_{i=1}^{k} : c'\) are disjoint, we have that \(\beta\) is proportional to \(\text{app}(b_{j})\) and \(\gamma\) is proportional to \(\prod_{i=1}^{k} \text{app}(b_{j})\).

(2) For the right premise we can then apply the hypothesis RH(j+1) and deduce that \(c = c'\) and \(m_{2} = \{b_{j+1}, \ldots, b_{k}\}\).

(3) For the left premise, the only possible rule is \(\text{app}\). We have then to choose the intermediate multiset \(m'\). As in the previous cases, one can argue that the left premise has empty context.
since \( P(b_j) \) is closed and then the induction hypothesis \( \text{IH}(C) \) gives \( m' = [b_j] \). This means that \((\text{3})\) has only one premise at the right-hand side, which is a conclusion of a \( \text{var} \) rule, so that \( m' = m_1 \).

We conclude by combining all these results: we get that the context \( m_1 \sqcup m_2 = [b_1, \ldots, b_k] \) and \( c = c' \).

We have proved that \( \text{RH}(1) \) holds: \( c = c' \) and \( p' = [b_1, \ldots, b_k] \), hence \( a' = a \).

5.3 Main Result

The theory of multi-variables analytic functions can be subtler than single variable ones (for instance zeros of functions are not isolated in general). In the following Lemma 25, we underline that the coefficients of a power series vanishing on a neighborhood of zero have to be null. Indeed, coefficients are computed by successive coefficients of a power series vanishing on a neighborhood of zero single variable ones (for instance zeros of functions are not isolated)

Lemma 25. \([16]\) Let \( f \) be a power series from \( \mathbb{R}^n \) to \( \mathbb{R} \) absolutely converging on \([0,1]^n\). If \( f \) vanishes on a dense subset of \([0,1]^n\), then the coefficients of the power series \( f \) are zero.

Lemma 26. Let \( M \) and \( N \) be two closed terms of PPCF with the same type \( A \). If \([M] \neq [N] \), then there is a PPCF term \( P \) of type \( A \Rightarrow \text{Int} \) such that \( \text{PROBA}^{P}(\tilde{\kappa}/\tilde{X}) \neq \text{PROBA}^{N}(\tilde{\kappa}/\tilde{X}) \).

Proof. Let \( a \in [A] \) such that \([M]_a \neq [N]_a \). Consider the testing term \( P(a) \) as defined in Figure 5.

Let \( f \) denote the series \([\langle P(a) \rangle M]_a \), which, by Definition 21, is a power series from \([0,1]^n \) to \([0,1]^n \), where \( n \) is the number of parameters in \( P(a) \). Let \( \text{cf}(\tilde{\kappa}(a),f) \) denote the coefficient of the monomial \( \tilde{\kappa}(a) \) in \( f \). By definition (Equation (11)) this coefficient is given by the sum of the weights of the possible derivations of \( \gamma P(a) \). The last rule of this derivation must be an app rule of shape:

\[
\vdash \gamma P(a) : (m,0) \quad \forall (a',i) \in m, \quad \vdash \beta_{(a',i)} M : a'
\]

Since \( M \) has no free parameter (it is a term of PPCF), each \( \beta_{(a',i)} \) is a positive real number. Hence, \( \alpha = \gamma \prod_{(a',i) \in m} \beta_{(a',i)} \) is proportional to \( \tilde{\kappa}(a) \) iff \( \gamma \) is proportional to \( \tilde{\kappa}(a) \). We can then apply Lemma 24 and get \( m = [a] \). Moreover, by Lemma 20, the sum of the weights of the derivations of \( \gamma P(a) : M : a \) is equal to the scalar \([M]_a \).

To sum up, we conclude:

\[
\text{cf}(\tilde{\kappa}(a),f) = \text{cf}(\tilde{\kappa}(a),[P(a)]_{(a,[0]),0}) [M]_a
\]

with \( \text{cf}(\tilde{\kappa}(a),[P(a)]_{(a,[0]),0}) \neq 0 \). By an analogous reasoning:

\[
\text{cf}(\tilde{\kappa}(a),g) = \text{cf}(\tilde{\kappa}(a),[P(a)]_{(a,[0]),0}) [N]_a
\]

where \( g \) denotes the power series \([P(a)]_N \). We can conclude that \( f \) and \( g \) have a different coefficient for the monomial \( \tilde{\kappa}(a) \) as soon as \([M]_a \neq [N]_a \).

Now, we apply Lemma 25 to the series \( f - g \), which is not the zero power series because of the coefficient of \( \tilde{\kappa}(a) \). Therefore, as rational numbers are dense in \([0,1] \), there is a list \( \tilde{n} \) of rational numbers in \([0,1] \) such that \( f(\tilde{n}) \neq g(\tilde{n}) \). By Lemma 20, this is equivalent to \( \left\| (P(a) \tilde{n}/X) M \right\|_0 \neq \left\| (P(a) \tilde{n}/X) N \right\|_0 \).

We conclude by setting \( P = P(a) \tilde{n}/X \). Actually, \( P \) is a well-defined PPCF term by Fact 17. So, by adequacy (Theorem 14), \( \text{PROBA}^{P}(\tilde{\kappa}/\tilde{X}) = \left\| (P) N \right\|_0 = \text{PROBA}^{N}(\tilde{\kappa}/\tilde{X}) \).

Theorem 27 (Full Abstraction). Given \( \Gamma \vdash M : A, \) and \( \Gamma \vdash N : A, \) we have:

\[ M \equiv \Gamma N \text{ iff } [M]_{\Gamma} = [N]_{\Gamma}. \]


As a consequence, the observational equivalence does not depend on the chosen probabilistic primitive (recall Remark 6):

Corollary 28. For any \( p \in \{\text{coin.rand, @x, @}x\text{rand}, (@x)_{h,n}\} \), let \( \equiv \) denote the observational equivalence induced by the extension of PPCF with the probabilistic primitive \( p \). The equivalence \( \equiv \) coincides with the equality of the interpretations in PPCoh. In particular, \( \equiv \) gives the same equivalence on PPCF terms, for any choice of \( p \).

Proof (Sketch). We have that \( \equiv_{\text{coin}} = \equiv_{\text{rand}} \) since the two primitives \( \text{coin} \) and \( \text{rand} \) are interdefinable by [5]. Similarly, by the discussion of Remark 6, we have: \( \equiv_{\text{rand}} \supseteq \equiv_{@x} \supseteq \equiv_{@x\text{rand}} \supseteq \equiv_{(@x)_{h,n}} \). In [4], the adequacy property of PPCoh is established for the language containing the \( (@x)_{h,n} \) primitive, hence \( \equiv_{(@x)_{h,n}} \) contains the equality \( \equiv_{\text{prob}} \) of the interpretations in PPCoh. Finally, by Theorem 27, we have \( \equiv_{\text{prob}} \supseteq \equiv_{\text{rand}} \) and we can conclude.

6. A counter-example to inequational FA

We prove that inequational full abstraction fails for PPCoh. The proof consists in: (i) showing that the morphisms of Remark 11, that proves the non-extensionality of the PPCoh order, are definable in PPCF (Equation 12); (ii) achieving a context lemma (Proposition 31) allowing us to infer the observational inequals by just observing the behavior of closed terms in applicative contexts.

If one would like to enrich the language in order to make observable also the componentwise order of PPCoh, one should add a kind of differential operator to PPCF, in the spirit of [9]. However, such an extension is not trivial, since PPCoh is known not to be sound for the whole differential \( \lambda \)-calculus.

Let us define \( M_1 \) and \( M_2 \) as follows:

\[ M_1 \triangleq \lambda x. \text{Int}.if(x, 0, 0), \quad M_2 \triangleq \lambda x. \text{Int}.0. \]

By using rules of Figure 3 and Lemma 20, one can check that the semantics of the two terms is given by the morphisms of Example 11, i.e. \( [M_1] = \phi \) and \( [M_2] = \psi \). Hence, the two matrices \([M_1]\) and \([M_2]\) are incomparable in PPCoh.
In order to prove $M_1 \sqsubseteq M_2$, we use a standard reasoning and introduce a logical relation (Definition 29) allowing us to shrink the set of contexts (Proposition 31). We conclude with Corollary 32.

**Definition 29.** By structural induction on a type $A$, we define the binary relation $\ll_A$ over closed PPCF terms of type $A$, as follows: $\ll_{\text{Int}} N' \iff \forall n \in \mathbb{N}, \left[ N' \right]_n \leq \left[ N' \right]_n$, and $M \ll_{A \to B} M' \iff \forall N \ll_A N', (M) N \ll_B (M') N'$. 

**Lemma 30.** Let $M$ be a term of type $x_1 : A_1, \ldots, x_n : A_n \vdash M : A$. If for all $i$, $N_i \ll_{A_i} N'_i$, then $M \left[ \bar{N} / \bar{x} \right] \ll_A M \left[ \bar{N}' / \bar{x} \right]$. 

**Proposition 31.** Let $M$ and $N$ be closed terms of type $A$. Then, $M \ll_{\text{Int}} N \iff M \sqsubseteq_{\text{Int}} A N$. 

**Corollary 32.** The terms $M_1$ and $M_2$ of Equation (12) are logically related: $M_1 \ll_{\text{Int} \to \text{Int}} M_2$. Hence, $M_1 \sqsubseteq M_2$.

### 7. Conclusion

We show that the observational equality over probabilistic programs is faithfully described by PCoh, a relatively abstract model based on convex sets and extensional functions. The proof uses innovative tools which might be useful to study probabilistic programming from a semantical viewpoint. In a language with random functions, two programs should be considered different not only when they give different results, but also when they give the same result but with different probabilities. Showing this difference can be much harder than in a deterministic language. Indeed, it requires a sharp control over coefficients expressing probabilities. PCoh denotes programs with power series, this allows us to use standard tools of Calculus for handling probabilities.

Although we chose one probabilistic primitive (see Remark 6), Corollary 28 shows that our result does not depend on this choice. Moreover, we focused on PPCF, a call-by-name functional language, but our result might be extended to other frameworks. In fact, PCoh comes from a model of linear logic: it is the co-kleisli category associated with the exponential comonad, that corresponds to the translation of the functional arrow $A \Rightarrow B$ into the linear logic formula $!A \multimap B$. Call-by-value can be obtained by using the Eilenberg-Moore construction, i.e. translating $A \Rightarrow B$ into $!(A \multimap B)$. Control operators can be introduced considering a polarized fragment of linear logic. Yet, extending this model to concurrent systems or references is certainly more challenging.

A crucial tool in the proof of full abstraction is the use of the type system of Figure 3, which is a kind of intersection type system. In fact, a web element $(\langle a_1, \ldots, a_n \rangle, b)$ can be seen as a type $(a_1 \land \cdots \land a_n) \to b$, where the intersection is non-idempotent. This system is a quantitative refinement of De Carvalho’s [6] and yields the first logical presentation of a vectorial based semantics. Clearly, both type inference and type checking are undecidable. However, one can look for interesting restrictions of PPCF where the system becomes decidable, in the spirit of [20].

Last, it should be noticed that, unlike most full abstraction models for PCF, our model has no simple PPCF-definability properties. Our full abstraction proof builds applicative contexts using terms which belong to a very small subset of the domains associated with types. Their discriminating power relies on a strong regularity property of power series (unlike smooth functions, a power series which vanishes on an open set must be equal to 0, see for example to the function defined by $e^{-1/x^2}$ for $x > 0$ and 0 for $x \leq 0$). In contrast, most full abstraction proofs build applicative contexts using terms which belong to a dense subset of the corresponding domains.

### References


