Probabilistic Call by Push Value

joint work with Thomas Ehrhard

Christine Tasson
Christine.Tasson@pps.univ-paris-diderot.fr

Laboratoire IRIF - Université Paris Diderot
pCBPV is a language well suited for writing probabilistic algorithm. It is equipped with the semantics of PCoh which is fully abstract thanks to quantitative properties.

1. Probabilistic PCF and Pcoh
2. An implementation issue
3. Probabilistic Call by push Value and Pcoh
pPCF is a language well suited for writing probabilistic algorithm. It is equipped with the semantics of P\text{Coh} which is fully abstract thanks to quantitative properties.

1 Probabilistic PCF and Pcoh
2 An implementation issue
3 Probabilistic Call by push Value and Pcoh
**pPCF** is a language well suited for writing **probabilistic algorithm**. It is equipped with the semantics of **Pcoh** which is fully abstract thanks to quantitative properties.

1. Probabilistic PCF and Pcoh
2. An implementation issue
3. Probabilistic Call by push Value and Pcoh
Probabilistic algorithm:

A Las Vegas example.
An example of Randomized algorithm

**Input**: A 0/1 array of length $n \geq 2$ in which half cells are 0.

```
0 1 2 3 4 5
0 1 0 1 1 0
```

**f**: 0, 2, 5 $\mapsto$ 0, 1, 3, 4 $\mapsto$ 1

**Output**: Find the index of a cell containing 0.

```
let rec LasVegas (f: nat -> nat) (n: nat) =
    let k = random n in
    if (f k = 0) then k
    else LasVegas f n
```

This algorithm succeeds with probability one.

- Success in 1 step is: $\frac{1}{2}$.
- Success in 2 steps is: $\frac{1}{2^2}$.
- Success in $n$ steps is: $\frac{1}{2^n}$.

Success in any steps is:

$$\sum_{k=1}^{\infty} \frac{1}{2^k} = 1.$$
The Denotational Semantics Hammer

pPCF, Pcoh and Full abstraction

POPL’14 – ”Probabilistic Coherence Spaces are Fully Abstract for Probabilistic PCF” with T. Ehrhard and M Pagani
Types: \( \sigma, \tau = \text{nat} | \sigma \Rightarrow \tau \)

Syntax:

\[
N, P, Q := n \mid \text{pred}(N) \mid \text{succ}(N) \mid x \mid \lambda x^\sigma P \mid (P)Q \mid \text{fix}(M) \\
\mid \text{if } (N = 0) \text{ then } P \text{ else } Q \mid \text{coin}(p), \text{ when } 0 \leq p \leq 1
\]

Operational Semantics: \( P \xrightarrow{a} Q \)

\( P \) reduces to \( Q \) in one step with probability \( a \)

\[
\text{if } (0 = 0) \text{ then } P \text{ else } Q \xrightarrow{1} P \\
\text{if } (n + 1 = 0) \text{ then } P \text{ else } Q \xrightarrow{1} Q \\
\text{coin}(p) \xrightarrow{p} 0 \\
\text{coin}(p) \xrightarrow{1 - p} 1
\]

\[
\text{Proba}(P \xrightarrow{*} Q) = \sum_{\rho} w(\rho)
\]
PCoh, a model of pPCF

1999 – ”Between Logic and Quantic : A tract”. J. Y. Girard

2011 – ”Probabilistic coherence spaces as a model of higher order probabilistic computation.” V. Danos and T. Ehrhard

Probabilistic Coherent Spaces:

\[ \mathcal{X} = (|\mathcal{X}|, P(\mathcal{X})) \]

where \(|\mathcal{X}|\) is a countable set and \(P(\mathcal{X}) \subseteq (\mathbb{R}^+)^{|\mathcal{X}|}\)

To ensure PCoh to be a model, \(P(\mathcal{X})\) are subject to biorthogonality, covering and boundedness conditions.

Type Example:

\[ [\text{nat}] = (\mathbb{N}, P(\text{nat}) = \{(\lambda_n) \mid \sum_n \lambda_n \leq 1\}) \]

Data Example:

if \(M : \text{nat}\), then \([M] \in P(\text{nat}) \subseteq (\mathbb{R}^+)\mathbb{N}\)

is a subprobability distributions.

Fair coin:

\[ [\text{coin}(\frac{1}{2})] = (\frac{1}{2}, \frac{1}{2}, 0, \ldots) \]
Probabilistic coherent Maps:

\[ f : (|X|, P(X)) \to (|Y|, P(Y)) \]

defined as a matrix \( M(f) \in (\mathbb{R}^+)^{\mathcal{M}_{\text{fin}}(|X|) \times |Y|} \) with

\[
f(x) = \sum_{[a_1,\ldots,a_n] \in \mathcal{M}_{\text{fin}}(|\sigma|)} M(f)[a_1,\ldots,a_n] \cdot \prod_{1 \leq i \leq n} x_{a_i}
\]

so that, the analytic function \( f : (\mathbb{R}^+)^{|X|} \to (\mathbb{R}^+)^{|Y|} \) preserves probabilistic coherence:

\[ f(P(X)) \subseteq P(Y) \]

Example:

if \( P : \text{nat} \to \text{nat} \), then \( [P] : (\mathbb{R}^+)^{\mathbb{N}} \to (\mathbb{R}^+)^{\mathbb{N}} \) is an analytic function preserving subprobability distributions.
The semantics of Probabilistic Programs

Once: \( \text{nat} \rightarrow \text{nat} \)

Input: an integer \( n \)

Output: if \( n=0 \) then 42
        else rand 2

\[
\begin{pmatrix}
0 & 1 & \cdots & \\
0 & \frac{1}{2} & \frac{1}{2} & \cdots \\
0 & \frac{1}{2} & \frac{1}{2} & \cdots \\
0 & 0 & 0 & \cdots \\
\vdots & \vdots & \vdots & \ddots \\
1 & 0 & \cdots \\
\vdots & \vdots & \vdots & \ddots \\
\end{pmatrix}

\rightarrow 0
\rightarrow 1
\cdots
\rightarrow 42
\]

Twice: \( \text{nat} \rightarrow \text{nat} \)

Input: an integer \( n \)

Output: if \( n=0 \) then 42
        else rand \( n \)

\[
([0], 42) \mapsto 1 \\
([n_1, n_2], k) \mapsto \frac{1}{n_1+1} + \frac{1}{n_2+1} \\
\text{if } 0 \leq k \leq n_1 \leq n_2 \\
([n_1, n_2], k) \mapsto \frac{1}{n_2+1} \\
\text{if } n_1 < k \leq n_2 \\
\text{Otherwise } 0
\]
Probabilistic Data:
If $x : \text{nat}$, then $[x] = (x_n)_{n \in \mathbb{N}}$

where $x_n$ is the probability that $x$ is $n$.

Probabilistic Program: $P : \text{nat} \rightarrow \text{nat}$

where $[P \ x]_n$ is the probability that $P \ x$ computes $n$.

\[
[\text{Once}] \in (\mathbb{R}^+)_{\mathbb{N} \times \mathbb{N}}
\]

\[
[\text{Once \ x}]_n = [\text{Once}] \cdot [x]
= \sum_{k} [\text{Once}]_{(k,n)} [x]_k
\]

\[
[\text{Twice}] \in (\mathbb{R}^+)_{\mathcal{M}_{\text{fin}}(\mathbb{N}) \times \mathbb{N}}
\]

\[
[\text{Twice \ x}]_n = [\text{Twice}] \cdot [x]!
= \sum_{\mu} [\text{Twice}]_{(\mu,n)} \prod_{k \in \mu} [x]_k
\]
Operational semantics:
Proba\( (P \rightarrow^* v) \) is the probability that \( P \) computes \( v \).

Denotational semantics:
Programs can be seen as distribution transformers which are analytic functions, so that

Adequacy Lemma:
If \( \vdash M : \text{nat} \), then \( \forall n \in \mathbb{N}, [M]_n = \text{Proba}(M \rightarrow^* n) \)

Probabilistic Full Abstraction:

\[
\begin{array}{c|c}
\text{Pcoh} & \text{pPCF} \\
\hline
[P] = [Q] & P \simeq_o Q \\
\hline
\text{Adequacy} & \forall C[\ ], \forall v \\
\hline
\text{Full Completeness} & \text{Proba}(C[P] \rightarrow^* v) = \text{Proba}(C[Q] \rightarrow^* v)
\end{array}
\]
How to encode the LasVegas Algorithm?

**Input**: A 0/1 array of length \( n \geq 2 \) in which half cells are 0.

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 & 5 \\
0 & 1 & 0 & 1 & 1 & 0
\end{array}
\]

\( f : 0, 2, 5 \mapsto 0, \quad 1, 3, 4 \mapsto 1 \)

**Output**: Find the index of a cell containing 0.

**Caml encoding**:

```ocaml
let rec LasVegas (f: nat -> nat) (n:nat) =
    let k = random n in
    if (f k = 0) then k
    else LasVegas f n
```

**pPCF encoding**:

\[
\begin{align*}
\text{rand } 0 &= \Omega \\
\text{rand } n+1 &= \text{if } (\text{coin}(n+1) = 0) \text{ then } n+1 \text{ else } \text{rand } n
\end{align*}
\]

\[
\begin{align*}
\text{fix } (\lambda \text{LasVegas}^{\text{nat} \Rightarrow \text{nat}}^{\text{nat} \Rightarrow \text{nat}}^{\text{nat} \Rightarrow \text{nat}}) \\
&\quad \lambda f^{\text{nat} \Rightarrow \text{nat}} \lambda n^{\text{nat}} \lambda k^{\text{nat}} \\
&\quad \text{if } (f k = 0) \text{ then } k \\
&\quad \text{else } \text{LasVegas } f n \text{ rand } n
\end{align*}
\]
Call By Name reflecting semantics:
In pPCF, data are not computed once for all (CBV) but whenever
it appears in the program (CBN). If it is used several times, it is
re-evaluated each time it appears.

Probabilistic Data as Random variables:
Each time a data is computed, it randomly gives a new value.

pPCF encoding:
\[
\begin{align*}
\text{fix} & \left( \lambda \text{LasVegas}^{\text{nat} \Rightarrow \text{nat}} \Rightarrow \text{nat} \Rightarrow \text{nat} \quad \lambda f^{\text{nat} \Rightarrow \text{nat}} \lambda n^{\text{nat}} \\
& \quad \left( \lambda k^{\text{nat}} \text{if } (f \ k = 0) \text{ then } k \\
& \quad \quad \text{else LasVegas } f \ n \right) (\text{rand } n) \right) \\
\rightarrow & \quad \text{fix} \left( \lambda \text{LasVegas}^{\text{nat} \Rightarrow \text{nat}} \Rightarrow \text{nat} \Rightarrow \text{nat} \quad \lambda f^{\text{nat} \Rightarrow \text{nat}} \lambda n^{\text{nat}} \\
& \quad \left( \lambda k^{\text{nat}} \text{if } (f \ (\text{rand } n) = 0) \text{ then rand } n \\
& \quad \quad \text{else LasVegas } f \ n \right) \right)
\end{align*}
\]
Towards a solution?

**Twice**: \( \text{nat} \rightarrow \text{nat} \)

- **Input**: an integer \( n \)
- **Output**: if \( n=0 \) then 42 else \( \text{rand} \ n \)

**Memo**: \( \text{nat} \rightarrow \text{nat} \)

- **Input**: an integer \( n \)
- **Output**: Let \( z = n \) in
  - if \( z=0 \) then 42
  - else \( \text{rand} \ z \)
When Semantics give the Solution.

Integer semantics in Pcoh :

\[
[nat] = \left( \mathbb{N}, P(nat) = \{ (\lambda_n) \mid \sum_n \lambda_n \leq 1 \} \right)
\]

A coalgebric structure in the linear Pcoh :

- **Contraction** : \( c^{nat} : \text{nat} \rightarrow \text{nat} \otimes \text{nat} \)

\[
c^{nat}(\sum_n x_n e_n) = \sum_n x_n e_n \otimes e_n
\]

\[
c_{n,(p,q)}^{nat} = \begin{cases} 
1 & \text{if } n = p = q \\
0 & \text{otherwise}
\end{cases}
\]

- **Weakening** : \( w^{nat} : \text{nat} \rightarrow 1 \)

\[
w^{nat}(\sum_n x_n e_n) = \sum_n x_n e_*
\]

\[
w_{n,*}^{nat} = 1, \text{ for any } n
\]
Call by Value for $\texttt{nat}$ and Call by Name pPCF

New conditional:

$$
\frac{
\Gamma \vdash N : \texttt{nat} \quad \Gamma \vdash P : \sigma \quad \Gamma, z : \texttt{nat} \vdash Q : \sigma
}{\Gamma \vdash \text{if } (N = 0) \text{ then } P \text{ else } [z] Q : \sigma}
$$

$$
\text{if } (0 = 0) \text{ then } P \text{ else } [z] Q \xrightarrow{1} P
$$

$$
\text{if } (n + 1 = 0) \text{ then } P \text{ else } [z] Q \xrightarrow{1} Q[n/z]
$$

Memorization of the value of an expression of type $\texttt{nat}$.

$$
\text{let } x = M \text{ in } N = \text{if } (M = 0) \text{ then } N[0/x] \text{ else } [z] N[\texttt{succ}(z)/x]
$$

New pPCF encoding:

$$
\text{LasVegas} = \text{fix } (\lambda LV (\texttt{nat} \Rightarrow \texttt{nat}) \Rightarrow \texttt{nat} \Rightarrow \texttt{nat} \quad \lambda f \texttt{nat} \Rightarrow \texttt{nat} \quad \lambda n \texttt{nat} \\
\text{let } k = \text{rand } n \text{ in } \\
\text{if } (f \ k, k, (LV) f \ n)
$$
pCBPV and Pcoh

Convenient, Adequate and Fully abstract.

TLCA’99 – ”Call By Push Value : A Subsuming Paradigm”. P. B. Levy

ESOP’16 – ”Call-By-Push-Value from a Linear Logic point of view”. T. Ehrhard

2016 – ”Probabilistic Call By Push Value” with T. Ehrhard
Main Ideas behind pCBPV and Pcoh

**Linear Logic inspired:**
Linear application on **positive** types, Exponential,...

**Coalgebraic Structure:**
For `nat` but also for other **positive** types (list, streams,...).

**pCBPV:**
Allows to combine CBV (for data) and CBN.

**Values** are particular terms of **positive** type which are:
- freely discardable and duplicable,
- interpreted as morphisms of coalgebras.
Syntax:

pCBPV – Probabilistic Call By Push Value
Syntax of pCBPV

Types :

(positive) \( \phi, \psi, \ldots := 1 \mid !\sigma \mid \phi \otimes \psi \mid \phi \oplus \psi \mid \zeta \mid \text{Fix} \zeta \cdot \phi \)

(general) \( \sigma, \tau \ldots := \phi \mid \phi \rightarrow \sigma \)

Programs : (general type)

\( M, N \ldots := x \mid () \mid M^I \mid (M, N) \mid \text{in}_1 M \mid \text{in}_2 M \)

\( \mid \lambda x^\phi M \mid \langle M \rangle N \mid \text{case}(M, x_1 \cdot N_1, x_2 \cdot N_2) \)

\( \mid \text{pr}_1 M \mid \text{pr}_2 M \mid \text{der}(M) \mid \text{fix} x^I \sigma M \)

\( \mid \text{fold}(M) \mid \text{unfold}(M) \mid \text{coin}(p), \ p \in [0, 1] \cap \mathbb{Q} \)

Typing context : \( \mathcal{P} = (x_1 : \phi_1, \ldots, x_k : \phi_k) \)

Values : (positive type)

\( V, W \ldots := x \mid () \mid M^I \mid (V, W) \mid \text{in}_1 V \mid \text{in}_2 V \mid \text{fold}(V) \).
Types :

(positive) \[ \phi, \psi, \ldots := 1 \mid !\sigma \mid \phi \otimes \psi \mid \phi \oplus \psi \mid \zeta \mid \text{Fix} \zeta \cdot \phi \]

(general) \[ \sigma, \tau \ldots := \phi \mid \phi \ra \sigma \]

Programs :

\[ \lambda x^\sigma M = \lambda x^{!\sigma} M \text{ and } (M)N = \langle M \rangle N \]

\[ M, N \ldots := x \mid () \mid M^! \mid (M, N) \mid \text{in}_1 M \mid \text{in}_2 M \]

\[ \mid \lambda x^\phi M \mid \langle M \rangle N \mid \text{case}(M, x_1 \cdot N_1, x_2 \cdot N_2) \]

\[ \mid \text{pr}_1 M \mid \text{pr}_2 M \mid \text{der}(M) \mid \text{fix} x^{!\sigma} M \]

\[ \mid \text{fold}(M) \mid \text{unfold}(M) \mid \text{coin}(p), \ p \in [0, 1] \cap \mathbb{Q} \]

Typing context :

\[ \mathcal{P} = (x_1 : \phi_1, \ldots, x_k : \phi_k) \]

Values :

\[ V, W \ldots := x \mid () \mid M^! \mid (V, W) \mid \text{in}_1 V \mid \text{in}_2 V \mid \text{fold}(V). \]
Encoding pPCF in Syntax of pCBPV

Types:

\[ \text{nat} = \text{Fix} \zeta \cdot 1 \oplus \zeta \] and \[ \sigma \Rightarrow \tau = !\sigma \rightharpoonup \tau \]

(positive) \[ \phi, \psi, \ldots := 1 | !\sigma | \phi \otimes \psi | \phi \oplus \psi | \zeta | \text{Fix} \zeta \cdot \phi \]

(general) \[ \sigma, \tau \ldots := \phi | \phi \rightharpoonup \sigma \]

Programs:

\[ \text{succ}(M) = \text{in}_2 M \]

\[ M, N \ldots := x | () | M^! | (M, N) | \text{in}_1 M | \text{in}_2 M \]

| \lambda x^\phi M | \langle M \rangle N | \text{case}(M, x_1 \cdot N_1, x_2 \cdot N_2) |
| pr_1 M | pr_2 M | \text{der}(M) | \text{fix} x^{!\sigma} M |
| \text{fold}(M) | \text{unfold}(M) | \text{coin}(p), \ p \in [0, 1] \cap \mathbb{Q} |

Typing context: \[ \mathcal{P} = (x_1 : \phi_1, \ldots, x_k : \phi_k) \]

Values:

\[ 0 = \text{in}_1 () \] and \[ n + 1 = \text{in}_2 n \]

\[ V, W \ldots := x | () | M^! | (V, W) | \text{in}_1 V | \text{in}_2 V | \text{fold}(V) . \]
Encoding pPCF in Syntax of pCBPV

Types:
\[ \text{nat} = \text{Fix} \cdot 1 \oplus \zeta \quad \text{and} \quad \sigma \Rightarrow \tau = !\sigma \rightarrow \tau \]

(positive) \( \phi, \psi, \ldots \) := 1 | !\sigma | \phi \otimes \psi | \phi \oplus \psi | \zeta | \text{Fix} \zeta \cdot \phi 

(general) \( \sigma, \tau \ldots \) := \phi | \phi \rightarrow \sigma 

Programs:
\[ \text{pred}(M) = \text{case}(M, x \cdot 0, z \cdot z) \]

\( M, N \ldots \) := \( x \mid () \mid M^! \mid (M, N) \mid \text{in}_1 M \mid \text{in}_2 M \]
\[ \mid \lambda x^\phi M \mid \langle M \rangle N \mid \text{case}(M, x_1 \cdot N_1, x_2 \cdot N_2) \]
\[ \mid \text{pr}_1 M \mid \text{pr}_2 M \mid \text{der}(M) \mid \text{fix} x^! \sigma M \]
\[ \mid \text{fold}(M) \mid \text{unfold}(M) \mid \text{coin}(p), \ p \in [0, 1] \cap \mathbb{Q} \]

Typing context: \( \mathcal{P} = (x_1 : \phi_1, \ldots, x_k : \phi_k) \)

Values:
\[ 0 = \text{in}_1 () \quad \text{and} \quad n + 1 = \text{in}_2 n \]

\( V, W \ldots \) := \( x \mid () \mid M^! \mid (V, W) \mid \text{in}_1 V \mid \text{in}_2 V \mid \text{fold}(V) \).
Encoding pPCF in Syntax of pCBPV

Types:

\( \text{nat} = \text{Fix} \zeta \cdot 1 \oplus \zeta \) and \( \sigma \Rightarrow \tau = !\sigma \rightarrow \tau \)

(positive) \( \phi, \psi, \ldots := 1 | !\sigma | \phi \otimes \psi | \phi \oplus \psi | \zeta | \text{Fix} \zeta \cdot \phi \)

(general) \( \sigma, \tau \ldots := \phi | \phi \rightarrow \sigma \)

Programs:

\[ \text{if } (M = 0) \text{ then } P \text{ else } [z] Q = \text{case}(M, x \cdot P, z \cdot Q) \]

\( M, N \ldots := x | () | M^1 | (M, N) | \text{in}_1 M | \text{in}_2 M \)

\[ | \lambda x^\phi M | \langle M \rangle N | \text{case}(M, x_1 \cdot N_1, x_2 \cdot N_2) \]

\[ | \text{pr}_1 M | \text{pr}_2 M | \text{der}(M) | \text{fix} x!^\sigma M \]

\[ | \text{fold}(M) | \text{unfold}(M) | \text{coin}(p), p \in [0, 1] \cap \mathbb{Q} \]

Typing context: \( \mathcal{P} = (x_1 : \phi_1, \ldots, x_k : \phi_k) \)

Values:

\( 0 = \text{in}_1 () \) and \( n + 1 = \text{in}_2 n \)

\( V, W \ldots := x | () | M^1 | (V, W) | \text{in}_1 V | \text{in}_2 V | \text{fold}(V) . \)
Probabilistic Coherent Spaces (Pcoh) :
Linearity, Exponential and Coalgebras
Pcoh semantics

LL-based denotational semantics:
Smcc (⊗, →, ...), *-autonomous (⊥), Cartesian (⊔), Exponential comonad (!⊥) with a strong symmetric monoidal structure

Fixpoints for formulae: (Types)

\[ X \subseteq Y \text{ iff } |X| \subseteq |Y| \text{ and } \begin{cases} \text{if } u \in P(X), \text{ then } \text{ext}_{|Y|}(u) \in P(Y) \\ \text{if } v \in P(Y), \text{ then } \text{res}_{|X|}(v) \in P(X) \end{cases} \]

\((Pcoh, \subseteq)\) is a cpo and all connectives are Scott Continuous.

Fixpoints for morphisms: (Programs)

if \( f \in Pcoh(!Y \otimes !X, X) \), then there is \( f^\dagger \in Pcoh(!Y, X) \) as morphisms, seen as analytic functions are Scott Continuous.
The category of coalgebras: \( \mathbf{Pcoh}^! \)

\( P = (P, h_P) \) with \( P \in \mathbf{Pcoh} \) and \( h_P \in \mathbf{Pcoh}(P, !P) \) satisfies:

\[
\array{
P & \xrightarrow{h_P} & !P \\
\downarrow{\text{Id}} & & \downarrow{\text{der}_P} \\
P & & P
}
\]

\[
\array{
P & \xrightarrow{h_P} & !P \\
\downarrow{h_P} & & \downarrow{\text{dig}_P} \\
!P & \xrightarrow{!h_P} & !!P
}
\]

**Positive types are coalgebras**

\( (!X, \text{dig}_X) \in \mathbf{Pcoh}^! \) and coalgebras are stable by \( \otimes, \oplus, \) fixpoints.

**Values are morphisms of coalgebras**

\( [V] \in \mathbf{Pcoh}^!([\mathcal{P}]^!, [\phi]^!) \)

where \( V \) is a value and \( \mathcal{P} \vdash V : \phi \) where \( \mathcal{P} = (x_1 : \phi_1, \ldots, x_k : \phi_k) \)

and \( [\mathcal{P}]^! = [\phi_1]^! \otimes \cdots \otimes [\phi_k]^! \).

C. Tasson

Introduction  pCBPV  Coalgebras  Full Abstraction
The Eilenberg Moore Category $\mathbf{Pcoh}$

**A Cartesian Category:** (not closed)

$(\mathbf{Pcoh}, \otimes, 1)$ is cartesian and for any coalgebra $P$, there is a contraction $c_P : P \to P \otimes P$ and a weakening $w_P : P \to 1$.

**Linearization:**

A positive type has structural rules, a function of type $\phi \to \sigma$ has no linearity restriction on the use of its argument.

**Dense Coalgebras:**

If $\phi$ is positif, then $\lbrack \phi \rbrack^!$ is dense.

$P = (\underline{P}, h_P)$ is **dense** if coalgebraic points characterize morphisms:

$$\forall X \in \mathbf{Pcoh} \text{ and } \forall t, t' \in \mathbf{Pcoh}(\underline{P}, X),$$

if $\forall v \in \mathbf{Pcoh}^!(1, P)$, $t v = t' v$, then $\forall u \in \mathbf{Pcoh}(1, \underline{P})$, $t u = t' u$.

Already known for $t, t' \in \mathbf{Pcoh}(X, Y)$ as $t u^! = t' u^!$ implies $t = t'$.
Interpretation of CBPV in Pcoh

Positive types:
\[[\phi]^! \in \text{Pcoh}^!\]

General types:
\[[\sigma] \in \text{Pcoh} \text{ with } [\phi] = [\phi]^!\]

Values:
if $\mathcal{P} \vdash V : \phi$, then $[V] \in \text{Pcoh}^!([[\mathcal{P}]^!, [\phi]^!])$.

Terms:
if $\mathcal{P} \vdash M : \sigma$, then $[M] \in \text{Pcoh}([[\mathcal{P}],[\sigma]])$. 
Probabilistic Full Abstraction:

The Adequacy Lemma

Let \( P, Q : \sigma \) \( \forall \alpha \in |\sigma|, [P]_\alpha = [Q]_\alpha \)

\textbf{Adequacy} \ \downarrow \ \uparrow \textbf{Full Abstraction}

\( \forall C : !\sigma \to 1, \)
\[ \text{Proba}(\langle C \rangle P^! \xrightarrow{*} ()) = \text{Proba}(\langle C \rangle Q^! \xrightarrow{*} ()) \]
Adequacy

Adequacy Lemma :
If $M$ is closed and $\vdash M : 1$, then $\llbracket M \rrbracket = \text{Proba}(M \rightarrow^* ())$

Ingredients : Tricky point : $\llbracket M \rrbracket \leq \text{Proba}(M \rightarrow^* ())$
- Logical relation between terms and elements of the model
- Pitt’s Technique for recursive types with positive and negative occurrences of type variables (fixpoint of tuple of relations)
- Two kinds of relations for positive and general types
- Hidden Biorthogonality closure for positive types
- Hidden Step Indexed Logical relation techniques (fold/unfold)

Adequacy proof :
If $\llbracket P \rrbracket = \llbracket Q \rrbracket$ then, $\text{Proba}(\langle C \rangle P \rightarrow^* ()) = \text{Proba}(\langle C \rangle Q \rightarrow^* ())$

1. Apply Adequacy Lemma : $\text{Proba}(\langle C \rangle P \rightarrow^* ()) = \llbracket \langle C \rangle P \rrbracket$.
2. Apply Compositionality : $\llbracket \langle C \rangle P \rrbracket = \sum_{\mu} \llbracket C \rrbracket_{\mu} \prod_{\alpha \in \mu} \llbracket P \rrbracket_{\alpha}^{\mu(\alpha)} = \sum_{\mu} \llbracket C \rrbracket_{\mu} \prod_{\alpha \in \mu} \llbracket Q \rrbracket_{\alpha}^{\mu(\alpha)} = \llbracket \langle C \rangle Q \rrbracket$
Probabilistic Full Abstraction:

The Full Abstraction theorem

Let \( P, Q : \sigma \) \( \forall \alpha \in |\sigma|, \, [P]_\alpha = [Q]_\alpha \)

Adequacy \( \downarrow \uparrow \) Full Abstraction

\( \forall C : !\sigma \to 1, \)
\[ \text{Proba}(\langle C \rangle P^! \rightarrow ()\) = \text{Proba}(\langle C \rangle Q^! \rightarrow ()\) \]
Full Abstraction proof:

1. By contradiction: $\exists \alpha \in |\sigma|, \llbracket P \rrbracket_\alpha \neq \llbracket Q \rrbracket_\alpha$

2. Find testing context: $T_\alpha$ such that $\llbracket (T_\alpha)P \rrbracket \neq \llbracket (T_\alpha)Q \rrbracket$
   (context only depends on $T_\alpha$)

3. Prove definability: $T_\alpha \in PPCF$ (uses $\text{coin}(p)$ and analyticity)

4. Apply Adequacy Lemma:
   $\text{Proba}(\langle T_\alpha \rangle P^I \rightarrow ()) \neq \text{Proba}(\langle T_\alpha \rangle Q^I \rightarrow ())$.

Tricky Hidden part:

- Defining contexts for positive and general types
- Definability relies on dense coalgebras
To sum up:

*pCBPV is a language well suited for writing probabilistic algorithm, combining CBN and CBV. It is equipped with the semantics of P Coh which is fully abstract thanks to quantitative properties.*

Further directions:

- use this language to study various effects (non-determinism, states etc) using computational monads on the linear category.
- resource calculi for CBPV, Taylor expansion