A PROBABILISTIC AND NON-DETERMINISTIC CALL-BY-PUSH-VALUE LANGUAGE

Jean Goubault-Larrecq
- PCF, probabilistic choice, and the trouble with \( \boxplus \)
- Curing the trouble using call-by-push-value
- Semantics, adequacy, full abstraction
PLOTKIN’S PCF (1977)

- **Types** $\sigma, \tau, \ldots ::= \text{int} | \sigma \to \tau$
- **Terms** $M, N, \ldots ::= x_{\tau}$
  - $MN$
  - $\lambda x_{\sigma}. M$
  - $\text{rec } x_{\sigma}. M$
  - $n$
  - $\text{succ } M$
  - $\text{pred } M$
  - $\text{ifz } M \ N \ P$

(All terms are typed. Call by name.)
PLOTKIN’S PCF (1977)

- **Types** \( \sigma, \tau, \ldots ::= \text{int} \mid \sigma \rightarrow \tau \)
- **Terms** \( M, N, \ldots ::= x_\tau \mid MN \mid \lambda x_\sigma . M \mid \text{rec } x_\sigma . M \mid n \mid \text{succ } M \mid \text{pred } M \mid \text{ifz } M \ N \ P \)

(All terms are typed. Call by name.)

- An **operational** semantics:
  \[ M \rightarrow^* N \]
- A **denotational** semantics:
  \[ [M] \]
- **Adequacy:**
  for every ground \( M : \text{int} \),
  \[ [M] = n \text{ iff } M \rightarrow^* n \]
PLOTKIN’S PCF (1977)

- An operational semantics: \( M \rightarrow^* N \)
- A denotational semantics: \([M] = n\) iff \( M \rightarrow^* n \)
- Adequacy: for every ground \( M : \text{int} \), \([M] = n\) iff \( M \rightarrow^* n \)

- Contextual preordering: \( M \preceq N \) iff
  for every context \( C : \text{int} \),
  \( C[M] \rightarrow^* n \Rightarrow C[N] \rightarrow^* n \)
- Fact: if \([M] \preceq [N]\) then \( M \preceq N \)
- Converse is full abstraction. Fails for PCF, works for PCF+por
Every type $T$ interpreted as a \textbf{dcpo} $\llbracket T \rrbracket$...

$\triangleq$ poset in which every directed family $D$ has a supremum $\bigvee D$
Every type $\tau$ interpreted as a dcpo $\llbracket \tau \rrbracket$...

$= \text{poset in which every directed family } D \text{ has a supremum } \bigvee D$

$\llbracket \text{int} \rrbracket = \mathbb{Z}_\bot (\bot \leq n, \text{all } n \text{ incomparable})$
Every type $\tau$ interpreted as a **dcpo** $\langle \tau \rangle$...

= poset in which every directed family $D$ has a supremum $\bigvee D$

$\langle \text{int} \rangle = \mathbb{Z}_\bot$ (\bot \leq n, all $n$ incomparable)

$\langle \sigma \to \tau \rangle = \langle \langle \sigma \rangle \to \langle \tau \rangle \rangle$, dcpo of Scott-continuous maps $\langle \sigma \to \langle \tau \rangle \rangle$ (monotonic + preserves directed sups)
THE SEMANTICS OF PCF

- **Types** \( \sigma, \tau, \ldots ::= \text{int} \ | \ \sigma \to \tau \)
- **Terms** \( M, N, \ldots ::= x_\tau \)
  - \( MN \)
  - \( \lambda x_\sigma.M \)
  - \( \text{rec } x_\sigma.M \)
  - \( n \)
  - \( \text{succ } M \)
  - \( \text{pred } M \)
  - \( \text{ifz } M N P \)

\[
\llbracket MN \rrbracket = \llbracket M \rrbracket(\llbracket N \rrbracket)
\]
\[
\llbracket \lambda x_\sigma.M \rrbracket = (V \mapsto \llbracket M \rrbracket[x_\sigma := V])
\]

- Meaningful since \textbf{Dcpo} is a Cartesian-closed category
CARTESIAN-CLOSEDNESS

- \([MN] = [M](N)\)
- \([\lambda x_{\sigma}. M] = (V \mapsto [M][x_{\sigma}:=V])\)
- Meaningful since **Dcpo** is a Cartesian-closed category

- In order to prove full abstraction (with por), we require to be able to **approximate** elements of \([T]\) by **definable** elements \([M]\).

- In the case of PCF, each \([T]\) is an **algebraic bc-domain**, making that possible.

- Cartesian-closed… good.
In order to prove full abstraction (with por), we require to be able to approximate elements of $\llbracket \tau \rrbracket$ by definable elements $\llbracket M \rrbracket$.

In the case of PCF, each $\llbracket \tau \rrbracket$ is an algebraic bc-domain, making that possible.

Cartesian-closed… good.
In order to prove full abstraction (with por), we require to be able to approximate elements of $\llbracket T \rrbracket$ by definable elements $\llbracket M \rrbracket$.

In the case of PCF, each $\llbracket T \rrbracket$ is an algebraic bc-domain, making that possible.

Cartesian-closed… good.

Many other CCCs would fit.
CONTINUOUS DCPOs

- **Approximation (way-below):**
  \[ x \ll y \iff \text{for every directed } D \text{ such that } y \leq \bigvee D, \]
  \[ x \text{ is already below some element of } D \]
CONTINUOUS DCPOs

- **Approximation (way-below):**
  \[ x \ll y \text{ iff for every directed } D \text{ such that } y \leq \bigvee D, \]
  \[ x \text{ is already below some element of } D \]
Approximation (way-below): $x \ll y$ iff for every directed $D$ such that $y \leq \bigvee D$, $x$ is already below some element of $D$. 
Approximation (way-below):

$x \ll y$ iff for every directed $D$ such that $y \leq \bigvee D$, $x$ is already below some element of $D$.

A basis $B$ of a dcpo $X$ iff for every $x$, 

$\{ b \in B \mid b \ll x \}$ directed and has $x$ as sup.

A dcpo $X$ is continuous iff has a basis.
CONTINUOUS DCPPOS

- **Approximation (way-below):**
  \( x \ll y \) iff for every directed \( D \) such that \( y \leq \vee D \),
  \( x \) is already below some element of \( D \)

- A **basis** \( B \) of a dcpo \( X \) iff for every \( x \),
  \( \{b \in B \mid b \ll x\} \) directed and has \( x \) as sup

- A dcpo \( X \) is **continuous** iff has a basis

- Ex: the finite subsets of \( A \) form a basis of \( \mathcal{P}(A) \) with inclusion
  \( \mathbb{N} \) forms a basis of \( \mathbb{N} \cup \{\infty\} \)
  \( \mathbb{Q}^+ \) forms a basis of \( \mathbb{R}^+ \cup \{\infty\} \) (\( x \ll y \) iff \( x=0 \) or \( x<y \) here)
ADDING PROBABILITIES

- **Types**
  \[ \sigma, \tau, \ldots ::= \text{int} \mid \sigma \rightarrow \tau \mid \nu \tau \]

- **Terms**
  \[ M, N, \ldots ::= \ldots \]
  \[ \mid M \oplus N \]
  \[ \mid \text{ret } M \]
  \[ \mid \text{do } x_\sigma \leftarrow M; N \]
Monadic type of subprobability valuations over $T$
ADDING PROBABILITIES

- Types
  \( \sigma, \tau, \ldots ::= \text{int} \mid \sigma \rightarrow \tau \mid \mathbf{v}_\tau \)

- Terms
  \( M, N, \ldots ::= \ldots \mid M \oplus N \mid \text{ret } M \mid \text{do } x_\sigma \leftarrow M ; N \)

Monadic type of subprobability valuations over \( \tau \) with \( M, N: \mathbf{v}_\tau \), choose between \( M \) and \( N \) with probability 1/2
**ADDING PROBABILITIES**

- **Types**
  \[ \sigma, \tau, \ldots ::= \text{int} \mid \sigma \rightarrow \tau \mid \mathbf{V}_\tau \]

- **Terms**
  \[ M, N, \ldots ::= \ldots \]
  \[ \mid M \oplus N \]
  \[ \mid \text{ret } M \]
  \[ \mid \text{do } x_\sigma \leftarrow M; N \]

---

Monadic type of subprobability valuations over \( \tau \)

with \( M, N: \mathbf{V}_\tau \), choose between \( M \) and \( N \) with probability \( 1/2 \)

monadic constructions:

\[ M: \mathbf{V}_\sigma \quad N: \mathbf{V}_\tau \Rightarrow \text{do } x_\sigma \leftarrow M; N : \mathbf{V}_\tau \]

(Moggi 1991)
THE TROUBLE WITH VALGEBRIC

bc-domains

algebraic complete lattices

Look for a category of continuous dcpos that is...

(Fung, Tix 1998)
Look for a category of continuous dcpos that is... 

- Cartesian-closed
THE TROUBLE WITH $\mathbf{V}$

- Look for a category of continuous dcpos that is...
  - Cartesian-closed
  - closed under $\mathbf{V}$

(Jung, Tix 1998)
MORE POSITIVELY:

- Look for a category of continuous dcpos that is:
  - Cartesian-closed
  - closed under $\mathbf{V}$

(Jung, Tix 1998)
MORE POSITIVELY:

- Look for a category of continuous dcpoś that is:
  - Cartesian-closed
  - closed under $\mathbf{V}$

(Jung, Tix 1998)
OTHER SOLUTIONS (1)

- Change categories entirely. E.g., reason in **probabilistic coherence spaces**

- Equationally **fully abstract** semantics (Ehrhard, Pagani, Tasson 14)

- also for call-by-push-value (Ehrhard, Tasson 19)

- probabilistic choice ‘built-in’
OTHER SOLUTIONS (2)

- Change categories, and opt for **QCB spaces/predomains** (Battenfeld 06)
  
  … Cartesian-closed, and has a probabilistic choice monad
OTHER SOLUTIONS (2)

- Change categories, and opt for **QCB spaces/predomains** (Battenfeld 06)
  
  ... Cartesian-closed, and has a probabilistic choice monad

- Changes categories, and opt for **quasi-Borel spaces/domains**
  
  (Heunen, Kammar, Staton, Yang 17; Vákár, Kammar, Staton 19)

  ... Cartesian-closed,

  and closed under a ‘laws of random variables’ functor
There is no need to leave domain theory after all.

An easy solution using call-by-push-value will also handle the mix with demonic non-determinism.

MORE POSITIVELY:

- Look for a category of continuous dcpos that is
  - Cartesian-closed
  - closed under $\vee$
- PCF, probabilistic choice, and the trouble with $\mathbb{V}$
- Curing the trouble using call-by-push-value
- Semantics, adequacy, full abstraction
- PCF, probabilistic choice, and the trouble with $V$
- Curing the trouble using call-by-push-value
- Semantics, adequacy, full abstraction
TWO KINDS OF TYPES?

- No such problem with two kinds of types:
  \[ \sigma, \tau, \ldots ::= \text{int} \mid \ldots \mid \sigma \times \tau \mid \bigvee \tau \]
  \[ \sigma, \imath, \ldots ::= \ldots \mid \sigma \to \imath \]

- continuous (coherent) dcpos
- bc-domains/continuous lattices
No such problem with two kinds of types:

\[ \sigma, \tau, \ldots ::= \text{int} | \text{unit} | U\sigma | \sigma \times \tau | V\tau \]

\[ \sigma, \tau, \ldots ::= F\sigma | \sigma \rightarrow \tau \]

This is the type structure of Paul B. Levy’s call-by-push-value (except for the \( V \) construction)

---

**Call-By-Push-Value: A Subsuming Paradigm (extended abstract)**

Paul Blain Levy

Department of Computer Science, Queen Mary and Westfield College
LONDON E1 4NS pbl@cs.qmw.ac.uk

Abstract. Call-by-push-value is a new paradigm that subsumes the call-by-name and call-by-value paradigms, in the following sense: both operational and denotational semantics for those paradigms can be seen as arising, via translations that we will provide, from similar semantics for call-by-push-value.

To explain call-by-push-value, we first discuss general operational ideas, especially the distinction between values and computations, using the principle that “is value, is a computation done”. Using an example program, we see that the lambda-calculus primitives can be understood as push/pop commands for an operand stack.

We provide operational and denotational semantics for a range of computational effects and show their agreement. We hence obtain semantics for call-by-name and call-by-value, of which some are familiar, some are new and some were known but previously appeared mysterious.

(Levy 1999)
CALL-BY-PUSH-VALUE

- No such problem with two kinds of types:
  \[ \sigma, \tau, \ldots ::= \text{int} \mid \text{unit} \mid U\sigma \mid \sigma \times \tau \mid V\tau \]
  \[ \sigma, \tau, \ldots ::= F\sigma \mid \sigma \rightarrow \tau \]

- This is the type structure of Paul B. Levy’s call-by-push-value (except for the V construction)

Call-By-Push-Value: A Subsuming Paradigm
(extended abstract)

Paul Blain Levy*
Department of Computer Science, Queen Mary and Westfield College
LONDON E1 4NS paul@cs.qmw.ac.uk

Abstract. Call-by-push-value is a new paradigm that subsumes the call-by-name and call-by-value paradigms, in the following sense: both operational and denotational semantics for those paradigms can be seen as arising, via translations that we will provide, from similar semantics for call-by-push-value.

To explain call-by-push-value, we first discuss general operational ideas, especially the distinction between values and computations, using the principle that “a value is a computation done”. Using an example program, we see that the lambda-calculus primitives can be understood as push/pop commands for an operand stack.

We provide operational and denotational semantics for a range of computational effects and show their agreement. We hence obtain semantics for call-by-name and call-by-value, of which some are familiar, some are new and some were known but previously appeared mysterious.

(Levy 1999)
No such problem with two kinds of types:

\[ \sigma, \tau, \ldots ::= \text{int} \mid \text{unit} \mid U\sigma \mid \sigma \times \tau \mid V\tau \]

\[ \sigma, \tau, \ldots ::= F\sigma \mid \sigma \rightarrow \tau \]

This is the type structure of Paul B. Levy’s **call-by-push-value** (except for the V construction)

---

**Call-By-Push-Value: A Subsuming Paradigm**

*(extended abstract)*

Paul Blain Levy*

Department of Computer Science, Queen Mary and Westfield College,

LONDON E1 4NS

*phl@cs.qmw.ac.uk*

**Abstract.** Call-by-push-value is a new paradigm that subsumes the call-by-name and call-by-value paradigms, in the following sense: both operational and denotational semantics for those paradigms can be seen as arising, via translations that we will provide, from similar semantics for call-by-push-value.

To explain call-by-push-value, we first discuss general operational ideas, especially the distinction between values and computations, using the principle that “a value is a computation done”. Using an example program, we see that the lambda-calculus primitives can be understood as push/pop commands for an operand stack.

We provide operational and denotational semantics for a range of computational effects and show their agreement. We hence obtain semantics for call-by-name and call-by-value, of which some are familiar, some are new and some were known but previously appeared mysterious.

(Levy 1999)
\[ \sigma, \tau, \ldots ::= \text{int} | \text{unit} | \sigma \times \tau | V\tau \]

- Continuous (coherent) dcpos
- bc-domains/continuous lattices
$\sigma, \tau, \ldots ::= \text{int} | \text{unit} | U\sigma | \sigma \times \tau | V\tau$

$\sigma, \tau, \ldots ::= \sigma \rightarrow \tau$

- $U$ converts from bc-domains to continuous coherent dcpos
- ... semantically the identity: $[U\sigma] = [\sigma]$
U AND F

- \( \sigma, \tau, \ldots ::= \text{int} \mid \text{unit} \mid U\sigma \mid \sigma \times \tau \mid V\tau \)

- \( \sigma, \tau, \ldots ::= \sigma \rightarrow \tau \)

- **U** converts from bc-domains to continuous coherent dcpos
  - ... semantically the identity: \([U\sigma] = [\sigma]\)

- \( M, N, \ldots ::= \ldots \)
  - | **force** \( M \) \((U\sigma \rightarrow \sigma)\)
  - | **thunk** \( M \) \((\sigma \rightarrow U\sigma)\)

- continuous (coherent) dcpos
- bc-domains/continuous lattices
**U AND F**

- \( \sigma, \tau, \ldots ::= \textbf{int} | \textbf{unit} | U\sigma | \sigma \times \tau | V\tau \)

- \( \sigma, \tau, \ldots ::= \sigma \rightarrow \tau \)

- **U** converts from bc-domains to continuous coherent dcpos
  
  … semantically the identity: \([U\sigma]=[\sigma]\)

- \( M, N, \ldots ::= \ldots \)
  
  \( \quad \mid \textbf{force}~M \quad (U\sigma \rightarrow \sigma) \)
  
  \( \quad \mid \textbf{thunk}~M \quad (\sigma \rightarrow U\sigma) \)

- \([\textbf{force}~M]=[M]\)
  
  \([\textbf{thunk}~M]=[M]\)

- \textbf{force thunk} \( M \rightarrow M \)
U AND F

\[\sigma, \tau, \ldots ::= \text{int} \mid \text{unit} \mid U\sigma \mid \sigma \times \tau \mid V\tau\]

\[\sigma, \tau, \ldots ::= F\sigma \mid \sigma \to \tau\]

- **U** converts from bc-domains to continuous coherent dcpos
  ... semantically the identity: \([U\sigma] = [\sigma]\)

- **F** converts from continuous coherent dcpos to bc-domains
  ... Ershov’s bounded complete hull would be the canonical choice

(but is too intricate for our purposes.)
THE SMYTH POWERDOMAIN

- $Q^X = \{\text{compact saturated subsets of } X\}$, reverse inclusion $\supseteq$

- **Fact.** For $X$ continuous coherent dcpo, Ershov’s bc-hull of $X$ is a subspace of $Q^X$.

- $Q^X$ is itself a bc-domain (even a continuous complete lattice), and is much easier to use.

- Serves as a model of **demonic non-determinism**.
THE SMYTH POWERDOMAIN

- $\mathcal{Q}X = \{\text{compact saturated subsets of } X\}$, reverse inclusion $\supseteq$ defines a(nother) monad on the cat. of cont. coh. dcpos.

- **Unit:** $\eta : X \to \mathcal{Q}X : x \mapsto \uparrow x$ (continuous)

- **Extension:** for $f : X \to L$ where $L$ continuous complete lattice, let $f^* : \mathcal{Q}X \to L : Q \mapsto \inf \{f(x) \mid x \in Q\}$
  - if $f$ is continuous then $f^*$ is continuous
  - $f^* \circ \eta = f$
  - $f^* \circ g^* = (f^* \circ g)^*$
THE SMYTH\(_{\perp}\) POWERDOMAIN

- \(\mathbb{Q}_{\perp}X = \mathbb{Q}X\) plus a fresh bottom \(\perp\)
defines a(nother) **monad** on the cat. of cont. coh. dcpos.

- **Unit:** \(\eta : X \to \mathbb{Q}_{\perp}X : x \mapsto \uparrow x\) (continuous)

- **Extension:** for \(f : X \to L\) where \(L\) continuous complete lattice,
  let \(f^* : \mathbb{Q}_{\perp}X \to L : Q \mapsto \inf \{f(x) \mid x \in Q\}, \perp \mapsto \perp\)
  — if \(f\) is continuous then \(f^*\) is continuous — and \(f^*\) is **strict** now
  — \(f^* \circ \eta = f\)
  — \(f^* \circ g^* = (f^* \circ g)^*\)
**U AND F**

- $\sigma, \tau, \ldots ::= \text{int} | \text{unit} | U\sigma | \sigma \times \tau | V\tau$
- $\sigma, \tau, \ldots ::= F\sigma | \sigma \rightarrow \tau$

**continuous (coherent) dcpo**

**bc-domains/continuous lattices**

- **U** converts from bc-domains to continuous coherent dcpo: $[U\sigma] = [\sigma]$
- **F** converts from continuous coherent dcpo to bc-domains: $[F\sigma] = Q \perp [\sigma]$
**U AND F**

- $\sigma, \tau, \ldots ::= \text{int} \mid \text{unit} \mid U\sigma \mid \sigma \times \tau \mid V\tau$
- $\sigma, \tau, \ldots ::= F\sigma \mid \sigma \to \tau$

**U** converts from bc-domains to continuous coherent dcpos: $[U\sigma] = [\sigma]$

**F** converts from continuous coherent dcpos to bc-domains: $[F\sigma] = Q_\bot [\sigma]$

- $M, N, \ldots ::= \ldots$
  - $\text{abort}_{F\sigma}$
  - $M \Diamond N$
  - $\text{produce} \ M$ (\(\sigma \to F\sigma\))
  - $M \text{ to } x_\sigma \text{ in } N$

**Continuous (coherent) dcpos**

**bc-domains/continuous lattices**
\( \sigma, \tau, \ldots ::= \text{int} \mid \text{unit} \mid U\sigma \mid \sigma \times \tau \mid V\tau \)

\( \sigma, \tau, \ldots ::= F\sigma \mid \sigma \to \tau \)

- **U** converts from bc-domains to continuous coherent dcpos: \([U\sigma] = [\sigma] \)
- **F** converts from continuous coherent dcpos to bc-domains: \([F\sigma] = \mathbb{Q}_\bot [\sigma] \)

\begin{align*}
M, N, \ldots & ::= \ldots \\
| \text{abort}_{F\sigma} \\
| M \odot N \\
| \text{produce} M \quad (\sigma \to F\sigma) \\
| M \text{ to } x_\sigma \text{ in } N
\end{align*}

\begin{align*}
[\text{abort}_{F\sigma}] & = \emptyset \\
[M \odot N] & = [M] \land [N] \\
[\text{produce } M] & = \eta([M]) \\
[M \text{ to } x_\sigma \text{ in } N] & = (V \mapsto [N][x_\sigma := V]^*)([M])
\end{align*}
\[ \begin{align*} 
\sigma, \tau, \ldots &::= \text{int} \mid \text{unit} \mid U\sigma \mid \sigma \times \tau \mid V\tau \\
\sigma, \tau, \ldots &::= F\sigma \mid \sigma \rightarrow \tau \\
U &\text{ converts from bc-domains to continuous coherent dcpos: } [U\sigma] = [\sigma] \\
F &\text{ converts from continuous coherent dcpos to bc-domains: } [F\sigma] = Q_{\bot} [\sigma] \\
M, N, \ldots &::= \ldots \\
&| \text{abort}_{F\sigma} \\
&| M \otimes N \\
&| \text{produce } M \quad (\sigma \rightarrow F\sigma) \\
&| M \text{ to } x_{\sigma} \text{ in } N \\
\end{align*} \]
PCF, probabilistic choice, and the trouble with $\mathbf{V}$

Curing the trouble using call-by-push-value

Semantics, adequacy, full abstraction
PCF, probabilistic choice, and the trouble with $\forall$

Curing the trouble using call-by-push-value

Semantics, adequacy, full abstraction
A Krivine machine for deterministic operations, working on configurations $C \cdot M$

\[
\begin{align*}
C \cdot E[M] & \rightarrow CE \cdot M & C[N] \cdot \lambda x. M & \rightarrow C \cdot M[x := N] \\
C[\text{to } x \text{ in } N] \cdot \text{produce } M & \rightarrow C \cdot N[x := M] & C[\text{force }] \cdot \text{thunk } M & \rightarrow C \cdot M \\
\text{let } M & \rightarrow \text{produce } \text{let } M & C[x := y] \cdot \text{produce } M & \rightarrow C \cdot \text{produce } x := y \\
C[\text{if } x \text{ then } M \text{ else } N] & \rightarrow C \cdot M & C[\text{if } \text{let } x := M \text{ then } N \text{ else } P] & \rightarrow C \cdot P \\
C[\text{let } x := M \text{ in } N] & \rightarrow C \cdot N & C[\text{let } x := N \text{ in } P] & \rightarrow C \cdot P \\
C[x := y] & \rightarrow C \cdot M & C[\text{let } x := y \text{ in } N] & \rightarrow C \cdot N \\
C[\text{let } x := y \text{ in } N] & \rightarrow C \cdot M & C[\text{let } x := y \text{ in } M] & \rightarrow C \cdot M \\
C[\text{do } x \rightarrow y \text{ in } N] & \rightarrow C \cdot N & C[\text{do } x \rightarrow y \text{ in } M] & \rightarrow C \cdot M \\
C[\text{rec } x, M] & \rightarrow C \cdot M[x := \text{rec } x, M] & C[\text{rec } x, M] & \rightarrow C \cdot M[x := \text{rec } x, M]
\end{align*}
\]
A Krivine machine for deterministic operations, working on configurations $C \cdot M$

Prob. must-termination judgments $C \cdot M \downarrow a$

(« whichever way you resolve the demonic non-deterministic choices, the probability that $C \cdot M$ terminates is $> a$. »)
OPERATIONAL SEMANTICS

- A Krivine machine for deterministic operations, working on configurations $C \cdot M$

- Prob. must-termination judgments $C \cdot M \downarrow a$
  (« whichever way you resolve the demonic non-deterministic choices, the probability that $C \cdot M$ terminates is $> a$. »)

- Let $\Pr(C \cdot M \downarrow) = \sup \{a \mid C \cdot M \downarrow a\}$, $\Pr(M \downarrow) = \Pr([\Box]. M \downarrow)$
Prop (adequacy).

For every $M : \mathbf{FVunit}$,
- $\llbracket M \rrbracket = \bot$ and $\Pr(M \downarrow) = 0$, or
- $\llbracket M \rrbracket = \emptyset$ and $\Pr(M \downarrow) = 1$, or else
- $\Pr(M \downarrow) = \min \{ \forall (\mathbf{T}) \mid \forall \in \llbracket M \rrbracket \}$


\begin{align*}
\llbracket x_r \rrbracket & = \rho(x_r) \\
\llbracket \lambda x_r. M \rrbracket & = V \in [x_r] \rightarrow [M] (\rho(x_r \mapsto V)) \\
\llbracket \mu x_r. M \rrbracket & = \llbracket M \rrbracket \rho = [M] \rho([M] \rho)
\end{align*}

\begin{align*}
\text{produce } M \rho & = \eta^0 ([M] \rho) \\
\text{to } x_r, \text{in } N \rho & = (V \in [x_r] \mapsto [N] \rho(x_r \mapsto V)) \ast ([M] \rho) \\
\text{think } M \rho & = [M] \rho \\
\text{force } M \rho & = [M] \rho
\end{align*}

\begin{align*}
\llbracket x_r \rrbracket & = \rho(x_r) \\
\llbracket \lambda x_r. M \rrbracket & = V \in [x_r] \rightarrow [M] (\rho(x_r \mapsto V)) \\
\llbracket \mu x_r. M \rrbracket & = \llbracket M \rrbracket \rho = [M] \rho([M] \rho)
\end{align*}

\begin{align*}
\text{produce } M \rho & = \eta^0 ([M] \rho) \\
\text{to } x_r, \text{in } N \rho & = (V \in [x_r] \mapsto [N] \rho(x_r \mapsto V)) \ast ([M] \rho) \\
\text{think } M \rho & = [M] \rho \\
\text{force } M \rho & = [M] \rho
\end{align*}
ADEQUACY

- **Prop (adequacy).**
  
  For every $M : \text{FVunit}$,
  
  - $\llbracket M \rrbracket = \bot$ and $\Pr(M \downarrow) = 0$, or
  
  - $\llbracket M \rrbracket = \emptyset$ and $\Pr(M \downarrow) = 1$, or else
  
  - $\Pr(M \downarrow) = \min \{ v(\{ \top \}) \mid v \in \llbracket M \rrbracket \}$

- I.e., $\Pr(M \downarrow) = h^*(\llbracket M \rrbracket)$
  where $h(v) = v(\{ \top \})$
ADEQUACY

- **Prop (adequacy).**
  - For every $M : \text{FVunit}$,
    - $\llbracket M \rrbracket = \bot$ and $\Pr(M \downarrow ) = 0$, or
    - $\llbracket M \rrbracket = \emptyset$ and $\Pr(M \downarrow ) = 1$, or else
    - $\Pr(M \downarrow ) = \min \{ \forall (\{ T \}) \mid \forall \in \llbracket M \rrbracket \}$
  
  I.e., $\Pr(M \downarrow ) = h^*(\llbracket M \rrbracket)$
  
  where $h(\forall ) = \forall (\{ T \})$

- **Proof:** by suitable logical relations.
None of that yet requires CCCs of continuous (or algebraic) domains

MORE POSITIVELY:
- Look for a category of continuous dcpo's that is
  - Cartesian-closed
  - closed under $\mathbf{V}$
None of that yet requires CCCs of **continuous** (or algebraic) domains.

Soundness/adequacy works even for non-call-by-push-value probabilistic languages, working in the CCC **Dcpo**.
None of that yet requires CCCs of *continuous* (or algebraic) domains.

Soundness/adequacy works even for non-call-by-push-value probabilistic languages, working in the CCC **Dcpo**.

Continuity is only needed for more advanced applications:
- full abstraction (next)
- commutativity of the **V** monad (Fubini) at higher types.
THE CONTEXTUAL PREORDER

- Let $M \preceq N$ iff for every context $C$ of output type $\text{FVunit}$,
  $\Pr(C \cdot M\downarrow) \leq \Pr(C \cdot N\downarrow)$
THE CONTEXTUAL PREORDER

- Let \( M \preceq N \) iff for every context \( C \) of output type \( \text{FVunit} \),
  \[ \Pr(C \cdot M \downarrow) \leq \Pr(C \cdot N \downarrow) \]

- \( M \preceq N \) iff for every context \( C \) of output type \( \text{FVunit} \),
  \[ h^*(\llbracket C[M]\rrbracket) \leq h^*(\llbracket C[N]\rrbracket) \]
  (adequacy)
THE CONTEXTUAL PREORDER

- Let \( M \preceq N \) iff for every context \( C \) of output type \texttt{FVunit},
  \[
  \Pr(C . M \downarrow) \leq \Pr(C . N \downarrow)
  \]

- \( M \preceq N \) iff for every context \( C \) of output type \texttt{FVunit},
  \[
  h^*(\llbracket C[M] \rrbracket) \leq h^*(\llbracket C[N] \rrbracket)
  \] (adequacy)

- **Corollary.** If \( \llbracket M \rrbracket \leq \llbracket N \rrbracket \) then \( M \preceq N \).
THE CONTEXTUAL PREORDER

Let $M \preceq N$ iff for every context $C$ of output type $\text{FVunit}$,
\[ \Pr(C . M \downarrow) \leq \Pr(C . N \downarrow) \]

$M \preceq N$ iff for every context $C$ of output type $\text{FVunit}$,
\[ h^*(\llbracket C[M] \rrbracket) \leq h^*(\llbracket C[N] \rrbracket) \] (adequacy)

**Corollary.** If $\llbracket M \rrbracket \leq \llbracket N \rrbracket$ then $M \preceq N$.

**Proof.** $\llbracket C[M] \rrbracket = \llbracket C \rrbracket (\llbracket M \rrbracket) \leq \llbracket C \rrbracket (\llbracket N \rrbracket) = \llbracket C[N] \rrbracket$

since $\llbracket C \rrbracket (= \llbracket \lambda x . C[x] \rrbracket)$ is Scott-continuous hence monotonic. Then apply $h^*$, which is monotonic as well. □
THE APPLICATIVE PREORDER

- Let $M \preceq N$ iff for every context $C$ of output type $\text{FVunit}$,
  $\Pr(C . M \downarrow) \leq \Pr(C . N \downarrow)$

- Let $M \preceq_{\text{app}} N$ iff for every term $P : \tau \to \text{FVunit}$,
  $\Pr(PM \downarrow) \leq \Pr(PN \downarrow)$
THE APPLICATIVE PREORDER

- Let $M \leq N$ iff for every context $C$ of output type $\text{FVunit}$,
  $\Pr(C \cdot M \downarrow) \leq \Pr(C \cdot N \downarrow)$

- Let $M \leq_{\text{app}} N$ iff for every term $P : \tau \rightarrow \text{FVunit}$,
  $\Pr(PM \downarrow) \leq \Pr(PN \downarrow)$

- Proposition (« Milner’s context lemma » in PCF):
  $M \leq N$ iff $M \leq_{\text{app}} N$. 
THE APPLICATIVE PREORDER

- Let $M \preceq N$ iff for every context $C$ of output type $\text{FVunit}$,
  $\Pr(C . M) \leq \Pr(C . N)$

- Let $M \preceq_{\text{app}} N$ iff for every term $P : \tau \rightarrow \text{FVunit}$,
  $\Pr(\text{PM}) \leq \Pr(\text{PN})$

- **Proposition** (« Milner’s context lemma » in PCF):
  $M \preceq N$ iff $M \preceq_{\text{app}} N$.

- Proof: based on an idea of A. Jung (Streicher 06), reusing our previous logical relation.
FULL ABstraction?

- Conjecture (full abstraction): $\left[ M \right] \leq \left[ N \right]$ iff $M \leq N$. 
**FULL ABSTRACTION?**

- **Conjecture (full abstraction):** \[M\] ≤ \[N\] iff \(M \leq N\).

- **Wrong.** (As in (Plotkin 77).) Using another logical relation, for every \(P : \text{int} \to \text{int} \to \text{Fint}\), if \(\llbracket P \rrbracket(\bot)(0) = \llbracket P \rrbracket(0)(\bot) = \{0\}\) then \(\llbracket P \rrbracket(\bot)(\bot) = \{0\}\).
FULL ABSTRACTION?

- **Conjecture (full abstraction):** $\llbracket M \rrbracket \leq \llbracket N \rrbracket$ iff $M \trianglerighteq N$.

- **Wrong.** (As in (Plotkin 77).) Using another logical relation, for every $P : \text{int} \rightarrow \text{int} \rightarrow \text{Fint}$, if $\llbracket P \rrbracket(\bot)(0)=\llbracket P \rrbracket(0)(\bot)=\{0\}$ then $\llbracket P \rrbracket(\bot)(\bot)=\{0\}$

- Hence with $M = \lambda P . P\Omega0==0 \land P0\Omega==0$
  $N = \lambda P . MP \land P\Omega\Omega==0$
  (and some syntactic sugar)
  we have $M \leq N$. 
FULL ABSTRACTION?

- **Conjecture (full abstraction):** $\llbracket M \rrbracket \leq \llbracket N \rrbracket$ iff $M \preceq N$.

- **Wrong.** (As in (Plotkin 77).) Using another logical relation, for every $P : \text{int} \to \text{int} \to \text{Fint}$, if $\llbracket P \rrbracket(\bot)(0)=\llbracket P \rrbracket(0)(\bot)={0}$ then $\llbracket P \rrbracket(\bot)(\bot)={0}$

- Hence with $M = \lambda P . \ P\Omega\Omega==0 && P\Omega==0$ \\
  $N = \lambda P . \ MP && P\Omega\Omega==0$ \\
  we have $M \preceq N$.

- But $\llbracket M \rrbracket \not\approx \llbracket N \rrbracket$ since $\llbracket M \rrbracket(\text{por})=\top$, $\llbracket N \rrbracket(\text{por})=\bot$. 
Add parallel if \textbf{pifz}:

\[
\left\lceil \textbf{pifz} \ M \ N \ P \right\rceil = \left\lceil N \right\rceil \text{ if } \left\lceil M \right\rceil = 0
\]

\[
\left\lceil P \right\rceil \text{ if } \left\lceil M \right\rceil \neq 0, \bot
\]

\[
\left\lceil N \right\rceil \wedge \left\lceil P \right\rceil \text{ (not } \bot!) \text{ if } \left\lceil M \right\rceil = \bot
\]
FULL ABSTRACTION?

- Add parallel if pifz:
  \[
  \langle \text{pifz } M N P \rangle = \lbrack N \rbrack \text{ if } \lbrack M \rbrack = 0
  \]
  \[
  \lbrack P \rbrack \text{ if } \lbrack M \rbrack \neq 0, \bot
  \]
  \[
  \lbrack N \rbrack \land \lbrack P \rbrack \text{ (not } \bot \text{!)} \text{ if } \lbrack M \rbrack = \bot
  \]

- Conjecture (full abstraction): \( \lbrack M \rbrack \leq \lbrack N \rbrack \text{ iff } M \preceq N. \)
FULL ABSTRACTION?

- Add parallel if \textbf{pifz}:
  \[\llbracket \text{pifz} \ M \ N \ P \rrbracket = \llbracket N \rrbracket \text{ if } \llbracket M \rrbracket = 0\]
  \[\llbracket P \rrbracket \text{ if } \llbracket M \rrbracket \neq 0, \bot\]
  \[\llbracket N \rrbracket \land \llbracket P \rrbracket \text{ (not } \bot!) \text{ if } \llbracket M \rrbracket = \bot\]

- **Conjecture (full abstraction):** \[\llbracket M \rrbracket \leq \llbracket N \rrbracket \text{ iff } M \preceq N.\]

- **Still wrong.** As in (GL 15), missing statistical termination testers.
Let $M = \lambda P . P(\Omega \oplus \text{ret} \, *)$

$N = \lambda P . P(\Omega) \oplus \text{ret} \, *$

Then $M \preceq N$, even with pifz

(modulo some missing force, produce, etc.)
Let \( M = \lambda P . P(\Omega \oplus \text{ret } *) \) 
\( N = \lambda P . P(\Omega) \oplus \text{ret } * \)  
(modulo some missing \text{force, produce}, etc.)

Then \( M \preceq N \), even with \text{pifz}

But \( \llbracket M \rrbracket \not\equiv \llbracket N \rrbracket \) since \( \llbracket M \rrbracket ([>b]) = \uparrow \delta_\top \), \( \llbracket N \rrbracket ([>b]) = \perp \) for all \( b < 1/2 \),
where \([>b] : [\text{Vunit}] \rightarrow [\text{FVunit}]\)
maps every \( \nu \) to ‘termination’ \( (\uparrow \delta_\top) \) if \( \nu(\{\top\}) > b \),
to \( \perp \) otherwise.
Let $M = \lambda P . P(\Omega \oplus \text{ret } *)$

$N = \lambda P . P(\Omega) \oplus \text{ret } *$

Then $M \preceq N$, even with \text{pifz}

But $\llbracket M \rrbracket \not\equiv \llbracket N \rrbracket$ since $\llbracket M \rrbracket([>b]) = \uparrow \delta \top$, $\llbracket N \rrbracket([>b]) = \bot$ for all $b < 1/2$, where $[>b] : \llbracket \text{Vunit} \rrbracket \rightarrow \llbracket \text{FVunit} \rrbracket$

maps every $\nu$ to ‘termination’ $(\uparrow \delta \top)$ if $\nu(\{\top\}) > b$

to $\bot$ otherwise.

$[>b]$ tests whether the prob. that its argument terminates is $> b$. 
FULL ABSTRACTION

- Add $\textbf{pifz} + \bigcirc_{>b}$ (with the semantics of $[>b], 0<b<1$ dyadic)
FULL ABSTRACTION

- Add $\text{pifz} + \bigcirc_{>b}$ (with the semantics of $[>b]$, $0<b<1$ dyadic)

- **Theorem (full abstraction):** with $\text{pifz}$ and $\bigcirc_{>b}$,
  $\llbracket M \rrbracket \leq \llbracket N \rrbracket$ iff $M \leq N$. 
FULL ABSTRACTION

- Add $\text{pifz} + \bigcirc_{>b}$ (with the semantics of $[>b], 0< b < 1$ dyadic)

- **Theorem (full abstraction):** with $\text{pifz}$ and $\bigcirc_{>b}$,
  \[
  \llbracket M \rrbracket \leq \llbracket N \rrbracket \text{ iff } M \preceq N.
  \]

- And now a glimpse of the argument…
We assume $[M] \not\equiv [N]$, and we wish to prove not $(M \leq N)$. 
We assume $\llbracket M \rrbracket \not\in \llbracket N \rrbracket$, and we wish to prove not $(M \leq N)$.

There is a subbasic open set $U / \llbracket M \rrbracket \in U, \llbracket N \rrbracket \not\in U$

(because $\leq$ is the specialization ordering of the Scott topology)
FULL ABSTRACTION: PROOF

- We assume $\llbracket M \rrbracket \not\in \llbracket N \rrbracket$, and we wish to prove not $(M \leq N)$.

- There is a subbasic open set $U / \llbracket M \rrbracket \in U, \llbracket N \rrbracket \not\in U$
  (because $\leq$ is the specialization ordering of the Scott topology)

- If $U$ is definable by a term $P$
  (i.e., $\llbracket P \rrbracket$ maps each $x \in U$ to $\uparrow\delta_T$ and each $x \not\in U$ to $\bot$)
  then $\llbracket PM \rrbracket = \uparrow\delta_T, \llbracket PN \rrbracket = \bot$, so not $(M \leq^{app} N)$.
  Conclude by Milner's context lemma.
FULL ABSTRACTION: PROOF

- We assume $\llbracket M \rrbracket \not\subseteq \llbracket N \rrbracket$, and we wish to prove not $(M \leq N)$.

- There is a subbasic open set $U / \llbracket M \rrbracket \in U, \llbracket N \rrbracket \notin U$
  (because $\leq$ is the specialization ordering of the Scott topology)

- If $U$ is **definable** by a term $P$
  (i.e., $\llbracket P \rrbracket$ maps each $x \in U$ to $\uparrow \delta_T$ and each $x \notin U$ to $\bot$)
  then $\llbracket PM \rrbracket = \uparrow \delta_T, \llbracket PN \rrbracket = \bot$, so not $(M \leq_{\text{app}} N)$.
  Conclude by Milner’s context lemma.

- Challenge: show that each $\llbracket T \rrbracket$ has a subbase of definable opens.
PROBABILISTIC TYPES

- $\mathbb{V}_\sigma$: subbasic opens $[U > b]$
  where $U$ is a basic open of $[\sigma]$, $b$ dyadic in $(0,1)$
PROBABILISTIC TYPES

- $\mathbf{V}\sigma$: subbasic opens $[U>b]$
  where $U$ is a basic open of $[\sigma]$, $b$ dyadic in $(0,1)$

- i.e. (requires $[\sigma]$ continuous!) Scott=weak topology (Kirch 93)
PROBABILISTIC TYPES

- $[\mathbf{V}\sigma]$: subbasic opens $[U>b]$
  where $U$ is a basic open of $[\sigma]$, $b$ dyadic in $(0,1)$

- I.e. (requires $[\sigma]$ **continuous**) Scott=weak topology (Kirch 93)

- test $\nu \in [U>b]$: iff $\nu(U) > b$, implemented through
  
  \[
  \text{‘} (\quad) > b (\textbf{do } x_\sigma \leftarrow \nu; \langle \text{test } x_\sigma \in U \rangle) \text{’}
  \]
PROBABILISTIC TYPES

- $\mathcal{V}_\sigma$: subbasic opens $[U>b]$
  where $U$ is a basic open of $[\sigma]$, $b$ dyadic in $(0,1)$

- I.e. (requires $[\sigma]$ continuous!) Scott=weak topology (Kirch 93)

- test $\nu \in [U>b]$: iff $\nu(U)>b$, implemented through
  
  \[
  \nu > b \ (\text{do } x_\sigma \leftarrow \nu; \langle \text{test } x_\sigma \in U \rangle)
  \]

- Note: $\mathcal{V}_\sigma$ also has a basis of
  
  $\sum_x a_x \delta_x, a_x$ dyadic, each $x$ from a basis of $[\sigma]$
  implementable using $\text{ret}$ and $\oplus$
[Fσ]: subbasic opens □U
where U is a basic open of [σ]
F TYPES

- $\langle F\sigma \rangle$: subbasic opens $\Box U$
  where $U$ is a basic open of $\sigma$

- I.e. (requires $\sigma$ continuous!) Scott=upper Vietoris topology
F TYPES

- \([F\sigma]\): subbasic opens \(\Box U\)
  where \(U\) is a basic open of \([\sigma]\)

- i.e. (requires \([\sigma]\) continuous!) Scott=upper Vietoris topology

- test \(Q \in \Box U\): iff \(Q \subseteq U\), implemented through
  \(Q \text{ to } x_\sigma \text{ in } \langle\text{test } x_\sigma \in U\rangle\)
**F TYPES**

- $[F\sigma]$: subbasic opens $\square U$
  
  where $U$ is a basic open of $[\sigma]$

- I.e. (requires $[\sigma]$ continuous!) Scott=upper Vietoris topology

- test $Q \in \square U$: iff $Q \subseteq U$, implemented through
  
  $'Q$ to $x_\sigma$ in $\langle$test $x_\sigma \in U'\rangle'$

- Note: $[F\sigma]$ also has a basis of $\uparrow\{x_1, \ldots, x_n\}$,
  
  (each $x_i$ from a basis of $[\sigma]$), plus $\perp$,
  
  implementable using produce, abort, and $\otimes$
FUNCTION TYPES (1/2)

- $\left[ \sigma \rightarrow \tau \right]$: subbasic opens $[x \mapsto V]$, where $x$ is from some basis of $\left[ \sigma \right]$, $V$ is a subbasic open of $\left[ \tau \right]$
FUNCTION TYPES (1/2)

- \([\sigma \rightarrow \tau]\): subbasic opens \([x \mapsto V]\),
  where \(x\) is from some basis of \([\sigma]\),
  \(V\) is a subbasic open of \([\tau]\)

- I.e. (uses \([\sigma]\) bc-domain!) Scott=top. of pointwise convergence
FUNCTION TYPES (1/2)

- $[\sigma \to \tau]$: subbasic opens $[x \mapsto V]$, where $x$ is from some basis of $[\sigma]$, $V$ is a subbasic open of $[\tau]$

- I.e. (uses $[\sigma]$ bc-domain!) Scott=top. of pointwise convergence

- test $f$ in $[x \mapsto V]$: iff $f(x) \in V$, implemented straightforwardly
FUNCTION TYPES (1/2)

- $[\sigma \to \tau]$: subbasic opens $[x \mapsto V]$, where $x$ is from some basis of $[\sigma]$, $V$ is a subbasic open of $[\tau]$

- I.e. (uses $[\sigma]$ bc-domain!) Scott=top. of pointwise convergence

- Test $f$ in $[x \mapsto V]$: iff $f(x) \in V$, implemented straightforwardly

- Note we need to also define a basis of each type $[\sigma]$ now. We have them for $\mathbf{V}$ and $\mathbf{F}$ types, and the difficult case is for $\to$. 
FUNCTION TYPES (2/2)

- Basis for \( \sigma \rightarrow \tau \): step functions \( \bigvee_{1 \leq i \leq n} U_i \downarrow y_i \)
  mapping each \( x \) to \( \bigvee \{ y_i \mid x \in U_i \} \)…
  but that sup is hard to implement — we only have infs.
FUNCTION TYPES (2/2)

- Basis for $[\sigma \rightarrow \tau]$:
  - Step functions $\bigvee_{1 \leq i \leq n} U_i \downarrow y_i$
  - Mapping each $x$ to $\bigvee\{y_i | x \in U_i\}$
  - But that sup is hard to implement — we only have infs.

- Trick: for each subset $I$ of $\{1, \ldots, n\}$,
  - Let $U_I = \text{intersection of } U_i, i \in I$,
  - $y_I = \text{sup of } y_i, i \in I$.
  - Then $\bigvee_{1 \leq i \leq n} U_i \downarrow y_i (x) = \inf\{y_I | I \supseteq l_x\}$ where $l_x = \{i | x \in U_i\}$
FUNCTION TYPES (2/2)

- Basis for $\sigma \rightarrow \tau$: step functions $\bigvee_{1 \leq i \leq n} U_i \downarrow y_i$
  mapping each $x$ to $\bigvee \{y_i \mid x \in U_i\}$
  but that sup is hard to implement — we only have infs.

- Trick: for each subset $I$ of $\{1, \ldots, n\}$,
  let $U_I = \text{intersection of } U_i, i \in I$,
  $y_I = \sup \text{ of } y_i, i \in I$.
  Then $\bigvee_{1 \leq i \leq n} U_i \downarrow y_i (x) = \inf \{y_I \mid I \supseteq l_x\}$ where $l_x = \{i \mid x \in U_i\}$

- Additional difficulties (need for pifz notably)… omitted. Done!
SUMMARY

- Circumventing the trouble with $\mathbf{V}$ by using two classes of types, as provided by call-by-push-value

- We obtain (inequational) full abstraction with prob. choice + demonic non-determinism

- Questions?

MORE POSITIVELY:

- Look for a category of continuous dcpo's that is
  - Cartesian-closed
  - closed under $\mathbf{V}$

CALL-BY-PUSH-VALUE

- No such problem with two kinds of types:
  \[
  \sigma, \tau, \ldots \colon \mathsf{int} | \mathsf{unit} | \mathsf{U}\sigma | \sigma \times \tau | \mathbf{V}\tau \\
  \sigma, \tau, \ldots \colon \mathsf{F}\sigma | \sigma \rightarrow \tau
  \]

- This is the type structure of Paul B. Levy's call-by-push-value

(Call-By-Push-Value: A Semining Paradigm (extended abstract))

(Levy 1999)