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

Jean Goubault-Larrecq
PCF, probabilistic choice, and the trouble with $\mathbb{V}$

- Curing the trouble using call-by-push-value
- Semantics, adequacy, full abstraction
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.)
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 \( M : \text{int}, \)\n  \([M]=n\) iff \( M \rightarrow^* n \)
PLOTKIN’S PCF (1977)

- An **operational** semantics:
  \[ M \rightarrow^* N \]

- A **denotational** semantics:
  \[ [M] \]

- **Adequacy:**
  for every ground \( M : \text{int}, \)
  \[ [M] = n \text{ iff } M \rightarrow^* n \]

- **Contextual preordering:**
  \( M \preceq N \text{ iff} \)
  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 $\tau$ interpreted as a **dcpo** $\llbracket \tau \rrbracket$

A directed family $D$. In a dcpo, every directed family $D$ has a supremum $\mathbin{\lor} D$. 
Every type $\tau$ interpreted as a dcpo $[\tau]$

$[\text{int}] = \mathbb{Z}_\bot$ ($\bot \leq n$, all $n$ incomparable)
DCPOS

- Every type $\tau$ interpreted as a dcpo $\llbracket \tau \rrbracket$
- $\llbracket \mathbf{int} \rrbracket = \mathbb{Z}_\bot$ ($\bot \leq n$, all $n$ incomparable)
- $\llbracket \sigma \rightarrow \tau \rrbracket = \llbracket \llbracket \sigma \rrbracket \rightarrow \llbracket \tau \rrbracket \rrbracket$, dcpo of Scott-continuous maps $\llbracket \sigma \rrbracket \rightarrow \llbracket \tau \rrbracket$ (monotonic + preserves directed sups)

A directed family $D$. In a dcpo, every directed family $D$ has a supremum $\lor D$. 

![Diagram showing a directed family $D$ with a supremum $\lor D$.]
THE SEMANTICS OF PCF

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

\[
\begin{align*}
\llbracket MN \rrbracket &= \llbracket M \rrbracket(\llbracket N \rrbracket) \\
\llbracket \lambda x_{\sigma}.M \rrbracket &= (V \mapsto \llbracket M \rrbracket[x_{\sigma}:=V]) \\
\end{align*}
\]

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

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

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

Cartesian-closed... good.
CCCS OF CONTINUOUS DCPPOS

- 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 $[[T]]$ by definable elements $[[M]]$.

In the case of PCF, each $[[T]]$ is an \textbf{algebraic bc-domain}, making that possible.

Cartesian-closed… good.

Many other CCCs would fit, provided they consist of \textbf{continuous dcpos}.
ADDING PROBABILITIES

- **Types**
  \[ \sigma, \tau, \ldots ::= \textbf{int} | \sigma \rightarrow \tau | \mathbf{V}\tau \]

- **Terms**
  \[ M, N, \ldots ::= \ldots \]
  \[ | M \oplus N \]
  \[ | \textbf{ret} \ M \]
  \[ | \textbf{do} \ x_{\sigma} \leftarrow M; N \]
ADDING PROBABILITIES

- Types
  \( \sigma, \tau, \ldots ::= \texttt{int} \mid \sigma \rightarrow \tau \mid V_\tau \)

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

Monadic type of subprobability valuations over \( \tau \)
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**
  \[ \sigma, \tau, \ldots ::= \text{int} \mid \sigma \rightarrow \tau \mid V_\tau \]

- **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 \( M, N : V_\tau \),
choose between \( M \) and \( N \)
with probability 1/2

monadic constructions:
\[ M : T \Rightarrow \text{ret } M : V_\tau \]
\[ M : V_\sigma \mid N : V_\tau \Rightarrow \text{do } x_\sigma \leftarrow M; N : V_\tau \]

(Moggi 1991)
THE TROUBLE WITH $V$

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

(Jung, Tix 1998)
THE TROUBLE WITH \( \mathbf{V} \)

Look for a category of continuous dcpos that is…

- Cartesian-closed

(Jung, Tix 1998)
THE TROUBLE WITH $\nabla$

Look for a category of continuous dcpo's that is...

- Cartesian-closed
- closed under $\nabla$

(Jung, Tix 1998)
MORE POSITIVELY:

Look for a category of continuous dcpos that is:

- Cartesian-closed
- closed under $\bigvee$

(Jung, Tix 1998)
MORE POSITIVELY:

- Look for a category of continuous dcpos that is:
  - Cartesian-closed
  - closed under $\forall$

(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)
  \(\ldots\) 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)
  \(\ldots\) 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**
PCF, probabilistic choice, and the trouble with $V$

- Curing the trouble using call-by-push-value
- Semantics, adequacy, full abstraction
PCF, probabilistic choice, and the trouble with \( \nabla \)

- Curing the trouble using call-by-push-value
- Semantics, adequacy, full abstraction
TWO KINDS OF TYPES?

- No such problem with **two kinds of types**: 
  \( \sigma, \tau, \ldots ::= \text{int} \mid \ldots \mid \sigma \times \tau \mid \forall \tau \)
  \( \sigma, \tau, \ldots ::= \ldots \mid \sigma \rightarrow \tau \)

  continuous (coherent) dcpos

  bc-domains/continuous lattices
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 phil@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)
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)
CALL-BY-PUSH-VALUE

- 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
phone: \( +44(0)20 \) 7882 5911
email: 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 "a value is, a computation does". 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)*
\[ \sigma, \tau, \ldots ::= \text{int} \mid \text{unit} \mid \sigma \times \tau \mid \nu \tau \]

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

- continuous (coherent) dcpos
- bc-domains/continuous lattices
\( U \) 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: \( \llbracket U\sigma \rrbracket = \llbracket \sigma \rrbracket \)

continuous (coherent) dcpos

bc-domains/continuous lattices
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: \([U\sigma]=[\sigma]\)

- \( M, N, \ldots ::= \ldots \)

  \[ \mid \text{force} \ M \quad (U\sigma \rightarrow \sigma) \]

  \[ \mid \text{thunk} \ M \quad (\sigma \rightarrow U\sigma) \]
\[ U \text{ AND } F \]

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

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

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

- \( M, N, \ldots ::= \ldots \)
  - \( \text{force } M \mid (U\sigma \rightarrow \sigma) \)
  - \( \text{thunk } M \mid (\sigma \rightarrow U\sigma) \)
  - \([\text{force } M] = [M]\)
  - \([\text{thunk } M] = [M]\)
  - \(\text{force thunk } M \rightarrow M\)

- 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
  … semantically the identity: \( \llbracket U\sigma \rrbracket = \llbracket \sigma \rrbracket \)

- \( F \) converts from continuous coherent dcpos to bc-domains
  … we take \( \llbracket F\sigma \rrbracket = \) (lifted) Smyth powerdomain of \( \llbracket \sigma \rrbracket \)
THE SMYTH POWERDOMAIN

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

- $Q_X$ is a continuous complete lattice for every continuous coherent dcpo $X$

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

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

- **Unit:** \(\eta : X \to QX : x \mapsto \uparrow x\) (continuous)

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

- Technically, we use $\mathbb{Q}_\bot X = \mathbb{Q}X$ plus a fresh bottom $\bot$
  ... allows $f^*$ to be strict now (needed for adequacy)
\( U \) 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: \( [F\sigma] = Q_{\perp} [\sigma] \)
U AND F

\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
\end{align*}

- **U** converts from bc-domains to continuous coherent dcpos: \([U\sigma] = [\sigma]\)
- **F** converts from continuous coherent dcpos to bc-domains: \([F\sigma] = Q_\perp [\sigma]\)

\begin{align*}
M, N, \ldots &:= \ldots \\
&\mid \text{abort}_{F\sigma} \\
&\mid M \land N \\
&\mid \text{produce } M \quad (\sigma \rightarrow F\sigma) \\
&\mid M \text{ to } x_{\sigma} \text{ in } N
\end{align*}
U AND F

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

- **U** converts from bc-domains to continuous coherent dcpos: \([U\sigma] = [\sigma]\)
- **F** converts from continuous coherent dcpos to bc-domains: \([F\sigma] = Q_\perp [\sigma]\)

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

- \([\text{abort}_{F\sigma}] = \emptyset\)
- \([M \otimes N] = [M] \land [N]\)
- \([\text{produce } M] = \eta([M])\)
- \([M \text{ to } x_\sigma \text{ in } N] = (V \mapsto [N][x_\sigma := V])^* ([M])\)
\[ \text{U干什么F} \]

\begin{itemize}
  \item \( \sigma, \tau, \ldots ::= \text{int} \mid \text{unit} \mid \text{U}\sigma \mid \sigma \times \tau \mid \text{V}\tau \)
  \item \( \sigma, \tau, \ldots ::= \text{F}\sigma \mid \sigma \rightarrow \tau \)
  \item \text{U} \text{ converts from bc-domains to continuous coherent dcpo}s: \([\text{U}\sigma]=\lbrack \sigma \rbrack\)
  \item \text{F} \text{ converts from continuous coherent dcpo}s to bc-domains: \([\text{F}\sigma]=[\text{Q}_\bot][\sigma] \)
  \item \(M, N, \ldots ::= \ldots\)
    \begin{align*}
      &\mid \text{abort}_{\text{F}\sigma} \\
      &\mid M \odot N \\
      &\mid \text{produce } M \quad (\sigma \rightarrow \text{F}\sigma) \\
      &\mid M \text{ to } x_\sigma \text{ in } N
    \end{align*}
    \begin{align*}
      &\mid [\text{abort}_{\text{F}\sigma}] = \emptyset \\
      &\mid [M \odot N] = [M] \wedge [N] \\
      &\mid [\text{produce } M] = \eta([M]) \\
      &\mid [M \text{ to } x_\sigma \text{ in } N] = \\
      &\quad (V \mapsto [N][x_\sigma:=V])^* ([M]) \\
      &\quad (\text{produce } M) \text{ to } x_\sigma \text{ in } N \rightarrow N[x_\sigma:=M] + \text{ etc.}
    \end{align*}
\end{itemize}
- PCF, probabilistic choice, and the trouble with \( \oplus \)
- Curing the trouble using call-by-push-value
- Semantics, adequacy, full abstraction
PCF, probabilistic choice, and the trouble with $\text{V}$

Curing the trouble using call-by-push-value

Semantics, adequacy, full abstraction
A Krivine machine for deterministic operations, working on configurations $C \cdot M$
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 $\Pr(C \cdot M \downarrow) = \sup \{a \mid C \cdot M \downarrow a\}$, $\Pr(M \downarrow) = \Pr([\_] \cdot M \downarrow)$
ADEQUACY

- **Prop (adequacy).**
  - $[M] = \perp$ and $\Pr(M \downarrow) = 0$, or
  - $[M] = \emptyset$ and $\Pr(M \downarrow) = 1$, or else
  - $\Pr(M \downarrow) = \min \{ \forall (\{\top\}) \mid \forall \in [M] \}$

\[
[x] \rho = \rho(x) \\
[\lambda x.M] \rho = V \in [x] \rightarrow [M] (\rho(x \rightarrow V)) \\
[M \downarrow] \rho = [M] \rho \rho [N] \rho \\
[\text{produce } M] \rho = \eta^0([M] \rho) \\
[M \text{ to } x, \text{ in } N] \rho = (V \in [x] \rightarrow [N] \rho(x \rightarrow V))'(\{M\} \rho) \\
[\text{think } M] \rho = [M] \rho \\
[\text{force } M] \rho = [M] \rho \\
[\text{succ } M] \rho = \begin{cases} n + 1 & \text{if } n = [M] \rho \neq \perp \\ \perp & \text{otherwise} \end{cases} \\
[\text{pred } M] \rho = \begin{cases} n - 1 & \text{if } n = [M] \rho \neq \perp \\ \perp & \text{otherwise} \end{cases} \\
[\text{if } M \text{ N } P] \rho = \begin{cases} [N] \rho & \text{if } [M] \rho = 0 \\ [P] \rho & \text{if } [M] \rho \neq 0, \perp \\ \perp & \text{otherwise} \end{cases} \\
[M; N] \rho = \begin{cases} [N] \rho & \text{if } [M] \rho = \top \\ \perp & \text{otherwise} \end{cases} \\
[\text{let } M] \rho = \delta_{[x]_\rho} \\
[\text{do } x_r \leftarrow M; N] \rho = (V \in [x] \rightarrow [N] \rho(x_r \rightarrow V))'(\{M\} \rho) \\
[M \oplus N] \rho = \frac{1}{2} ([M] \rho \oplus [N] \rho) \\
[M \otimes N] \rho = [M] \rho \otimes [N] \rho \\
[\text{abort}_r] \rho = \emptyset \\
[\text{rec } x_r; M] \rho = \text{fix}(V \in [x] \rightarrow [M] \rho(x_r \rightarrow V))
\]
Prop (adequacy).

For every $M : \text{FVunit}$,

- $\denot{M} = \bot$ and $\Pr(M \downarrow) = 0$, or
- $\denot{M} = \emptyset$ and $\Pr(M \downarrow) = 1$, or else
- $\Pr(M \downarrow) = \min \{ \forall (\{\top\}) \mid \forall \in \denot{M} \}$

\[ \text{i.e., } \Pr(M \downarrow) = h^*(\denot{M}) \]
\[ \text{where } h(V) = V(\{\top\}) \]
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
  - \( \Pr(M \downarrow) = \min \{ \nu(\{ \top \}) \mid \nu \in \llbracket M \rrbracket \} \)

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

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

MORE POSITIVELY:
- Look for a category of continuous dcpos that is
  - Cartesian-closed
  - closed under $V$
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**

MORE POSITIVELY:

- Look for a category of continuous dcpo that is...
- Cartesian-closed
- closed under $\forall$
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 abstractions (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 $\text{FVunit}$,
  $\Pr(C . M\downarrow) \leq \Pr(C . N\downarrow)$
THE CONTEXTUAL PREORDER

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

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

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

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

- **Corollary.** If $\sem{M} \preceq \sem{N}$ 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. □
FULL ABSTRACTION?

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

- Conjecture (full abstraction): $[M] \leq [N]$ iff $M \preceq N$.

- Wrong.
  — missing parallel if (pifz), as in (Plotkin77)
  — even with pifz, missing statistical termination testers $\bigcirc>_{b}$, as in (GL15):
    $\bigcirc>_{b} M$ terminates if $M$ terminates with prob. $>b$, otherwise does not terminate.
FULL ABSTRACTION

- Adding $\text{pifz} + \bigcirc > b$,

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

- For the argument, see the paper. Uses the deep structure of continuous coherent dcpos and continuous complete lattices. Core: theorems on (effective) coincidence of topologies.
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?