Call-by-push-value: the fine-grain structure of call-by-value and call-by-name

Paul Blain Levy

University of Birmingham

February 20, 2012

Call-By-Push-Value

Call-by-push-value is a form of λ-calculus with computational effects.
The goal of this talk is to explain the sense in which call-by-push-value is a direct description of the fine-grained structure of call-by-value and call-by-name typed λ-calculus.

Simply Typed λ-calculus

A pure functional language. The types are:

$$A ::= \quad 0 \mid A + A \mid 1 \mid A \times A \mid A \to A \mid \sum_{i \in \mathbb{N}} A_i \mid \prod_{i \in \mathbb{N}} A_i$$

$\beta\eta$-laws for all connectives, hence numerous type isomorphisms.

It has denotational semantics in any countably bicartesian closed category, in particular $\mathbf{Set}$.

CBV and CBN operational semantics give the same answer for a boolean term $\vdash M : 1 + 1$.

Call-by-name definitional interpreter

In CBN the terminals are $\texttt{inl}\ M, \texttt{inr}\ M, \lambda\texttt{x}.M, \ldots$
To evaluate

- $\lambda\texttt{x}.M$: return $\lambda\texttt{x}.M$.
- $\texttt{inl}\ M$: return $\texttt{inl}\ M$.
- $\texttt{match}\ M\ \texttt{as}\ \{\texttt{inl x.}\ N,\ \texttt{inr x.}\ N'\}$: evaluate M. If it returns $\texttt{inl}\ P$, evaluate $N[P/\texttt{x}]$, but if it returns $\texttt{inr}\ P$, evaluate $N'[P/\texttt{x}]$.
- MN: evaluate M. If it returns $\lambda\texttt{x}.P$, evaluate $P[N/\texttt{x}]$.

Call-by-value definitional interpeter

CBV terminals $T ::= \ \texttt{inl}\ T \mid \texttt{inr}\ T \mid \lambda\mathtt{x}.M \mid \cdots$

To evaluate

- $\lambda\mathtt{x}.M$: return $\lambda\mathtt{x}.M$.
- $\texttt{inl}\ M$: evaluate M. If it returns T, return $\texttt{inl}\ T$.
- $\texttt{match}\ M\ \texttt{as}\ \{\texttt{inl x}.\ N,\ \texttt{inr x}.\ N'\}$: evaluate M. If it returns $\texttt{inl}\ T$, evaluate $N[T/\mathtt{x}]$, but if it returns $\texttt{inr}\ T$, evaluate $N'[T/\mathtt{x}]$.
- MN: evaluate M. If it returns $\lambda\mathtt{x}.P$, evaluate N. If that returns T, evaluate $P[T/\mathtt{x}]$.

Variants of Simply Typed λ-calculus

There are many variants, e.g.

- we could include n-ary sum types $+(A, B, C)$
- we could include n-ary function types $(A, B, C) \to D$
- we could include either a pattern-match product $A \times B$, or a projection product $A \,\Pi\, B$, or both.

These variants are all equivalent, because of the type isomorphisms.
The largest of these variants is called jumbo λ-calculus.

The Experiment

We add computational effects (aka imperative features, aka Moggi's notions of computation) to our pure language: errors, I/O, divergence, nondeterminism, reading and assigning to memory cells, generating memory cells, callcc.

$E \stackrel{\text{def}}{=} \{\text{CRASH}, \text{BANG}\}$	$\mathcal{A} \stackrel{\text{def}}{=} \{\text{a}, \text{b}, \text{c}, \text{d}, \text{e}\}$
$\dfrac{}{\Gamma \vdash \texttt{error}\ e : B}\ e \in E$	$\dfrac{\Gamma \vdash M : B}{\Gamma \vdash \texttt{print}\ c.\ M : B}\ c \in \mathcal{A}$
To evaluate $\texttt{error}\ e$ halt with error message e	To evaluate $\texttt{print}\ c.\ M$ print c and then evaluate M

CBV/CBN Exercises

1. Evaluate

$$(\lambda \mathrm{x}.(\mathrm{x}+\mathrm{x}))(\texttt{print "hello"}.\ 4)$$

   in CBV and CBN.

2. Evaluate

$$\begin{array}{l}\texttt{match}\ (\texttt{print "hello"}.\ \texttt{inr error}\ \mathrm{CRASH})\ \texttt{as} \\ \quad \{\texttt{inl}\ \mathrm{x}.\ \mathrm{x}+1,\ \texttt{inr}\ \mathrm{y}.\ 5\}\end{array}$$

   in CBV and CBN.

What happens?

In each of these effectful languages:

- what equations survive as contextual equivalences?
- what type isomorphisms survive?
- can we give a denotational semantics?

Analyzing the denotational models for different effects,

- what patterns do we see?

What happens?

In each of these effectful languages:

- what equations survive as contextual equivalences?
- what type isomorphisms survive?
- can we give a denotational semantics?

Analyzing the denotational models for different effects,

- what patterns do we see?

We could carry out this project for any version of the λ-calculus. By choosing jumbo λ-calculus, we cover all possibilities.

Equations and Isomorphisms

Assuming $\mathtt{x} \notin \Gamma$.

Valid in CBV, but not in CBN

$$\Gamma, \mathtt{z} : A + B \vdash M \quad = \quad \mathtt{match\ z\ as} \left\{ \begin{array}{ll} \mathtt{inl\ x.} & M[\mathtt{inl\ x/z}] \\ \mathtt{inr\ x.} & M[\mathtt{inr\ x/z}] \end{array} \right\} : C$$

$$(A + B) + C \quad \cong \quad A + (B + C)$$

Valid in CBN, but not in CBV

$$\Gamma \vdash M \quad = \quad \lambda \mathtt{x}. (M\, \mathtt{x}) \ : A \to B$$

$$(A \,\Pi\, B) \to C \quad \cong \quad A \to (B \to C)$$

Thunks in CBV

In call-by-value, the type $() \to A$ is often called a "thunk" type.

$$\begin{array}{rcl} TA & \stackrel{\text{def}}{=} & () \to A \\ \texttt{thunk}\ M & \stackrel{\text{def}}{=} & \lambda().\, M \\ \texttt{force}\ M & \stackrel{\text{def}}{=} & M\,() \end{array}$$

Thunks can be used to delay evaluation.

CBV: Semantics In Outline

A term $\Gamma \vdash M : A$ denotes

$$
\begin{array}{rl}
[\![M]\!] : [\![\Gamma]\!] \longrightarrow [\![A]\!] + E & \qquad \text{(errors)} \\
[\![M]\!] : S \times [\![\Gamma]\!] \longrightarrow S \times [\![A]\!] & \qquad \text{(state)} \\
[\![M]\!] : [\![\Gamma]\!] \times ([\![A]\!] \to R) \longrightarrow R & \qquad \text{(continuations)}
\end{array}
$$

CBV: Semantics In Outline

A term $\Gamma \vdash M : A$ denotes

$$\begin{array}{rl} [\![M]\!] : [\![\Gamma]\!] \longrightarrow [\![A]\!] + E & \quad \text{(errors)} \\ [\![M]\!] : S \times [\![\Gamma]\!] \longrightarrow S \times [\![A]\!] & \quad \text{(state)} \\ [\![M]\!] : [\![\Gamma]\!] \times ([\![A]\!] \to R) \longrightarrow R & \quad \text{(continuations)} \end{array}$$

Why doesn't Paul just say it's the Kleisli category for the monad?

CBV: Semantics of Values

Values are given by

$$V ::= \quad \mathtt{x} \mid \mathtt{inl}\ V \mid \mathtt{inr}\ V \mid \lambda\mathtt{x}.M \mid \cdots$$

A value denotes $[\![V]\!]^{\mathsf{val}} : [\![\Gamma]\!] \longrightarrow [\![A]\!]$

Substitution lemma

We can obtain $[\![M[V/\mathtt{x}]]\!]$ in terms of $[\![M]\!]$ and $[\![V]\!]^{\mathsf{val}}$.

Dynamically Generated, Globally Visible Cells (CSL '02)

We have a poset of worlds $\mathcal{W}$, saying what cells are generated.

Sm is the set of states in world $m \in \mathcal{W}$.

A type denotes a functor $\mathcal{W} \longrightarrow \mathbf{Set}$.

$$[\![A \to B]\!]m = \prod_{n \geqslant m} (Sn \to [\![A]\!]n \to \sum_{p \geqslant n} (Sp \times [\![B]\!]p))$$

A value $\Gamma \vdash^{\mathsf{v}} V : A$ denotes a natural transformation $[\![\Gamma]\!] \xrightarrow{[\![V]\!]^{\mathsf{val}}} [\![A]\!]$.

Dynamically Generated, Globally Visible Cells (CSL '02)

We have a poset of worlds $\mathcal{W}$, saying what cells are generated.

Sm is the set of states in world $m \in \mathcal{W}$.

A type denotes a functor $\mathcal{W} \longrightarrow \mathbf{Set}$.

$$[\![A \to B]\!]m = \prod_{n \geqslant m} (Sn \to [\![A]\!]n \to \sum_{p \geqslant n} (Sp \times [\![B]\!]p))$$

A value $\Gamma \vdash^{\mathsf{v}} V : A$ denotes a natural transformation $[\![\Gamma]\!] \xrightarrow{[\![V]\!]^{\mathsf{val}}} [\![A]\!]$.

A term $\Gamma \vdash M : A$ denotes a function

$$Sm \times [\![\Gamma]\!]m \xrightarrow{[\![M]\!]m} \textstyle\sum_{n \geqslant m} (Sn \times [\![A]\!]n) \quad \text{for all } m \in \mathbb{N}.$$

Thunking Matters

A term $\Gamma \vdash M : A$ corresponds to a value $\Gamma \vdash V : TA$.

But they're not the same thing.

CBV: spot the pattern

Semantics of $\texttt{inl}\ M$, for errors, state, control

$$
\begin{array}{rcl}
[\![\texttt{inl}\ M]\!]\rho & = & \text{match}[\![M]\!]\rho \text{ as } \left\{ \begin{array}{l} \text{inl } x.\ \text{inl inl } x \\ \text{inr } e.\text{inr } e \end{array} \right. \\
[\![\texttt{inl}\ M]\!](s,\rho) & = & \text{match}[\![M]\!](s,\rho) \text{ as } (s',x).\ (s',\text{inl } x) \\
[\![\texttt{inl}\ M]\!](\rho,k) & = & [\![M]\!](\rho,\lambda x.k(\text{inl } x))
\end{array}
$$

The red part represents sequencing.
The blue part represents returning a value.

Fine-Grain Call-By-Value

Same types as before.

Judgements of Fine-Grain CBV

Values $\Gamma \vdash^{\mathsf{v}} V : A$ Computations $\Gamma \vdash^{\mathsf{c}} M : A$.

Terms:

$$\begin{array}{rcl} V & ::= & \mathtt{x} \mid \mathtt{inl}\ V \mid \mathtt{inr}\ V \mid \lambda\mathtt{x}.M \mid \cdots \\ M & ::= & \mathtt{return}\ V \mid M\ \mathtt{to}\ \mathtt{x}.\ N \mid V\,V \mid \cdots \end{array}$$

$$\frac{\Gamma \vdash^{\mathsf{v}} V : A}{\Gamma \vdash^{\mathsf{c}} \mathtt{return}\ V : A} \qquad \frac{\Gamma \vdash^{\mathsf{c}} M : A \quad \Gamma, \mathtt{x} : A \vdash^{\mathsf{c}} N : B}{\Gamma \vdash^{\mathsf{c}} M\ \mathtt{to}\ \mathtt{x}.\ N : B}$$

Closed distributive Freyd category (Power, Robinson, Thielecke).

Fine-grain call-by-value: definitional interpreter

Evaluates a closed computation to a closed value.

To evaluate

- $\texttt{return}\ V$: return V.
- $M\ \texttt{to x.}\ N$: evaluate M. If it returns V, then evaluate $N[V/\texttt{x}]$.
- $(\lambda\texttt{x}.M)V$: evaluate $M[V/\texttt{x}]$.
- $\texttt{match inl}\ V\ \texttt{as}\ \{\texttt{inl x.}\ N,\ \texttt{inr x.}\ N'\}$: evaluate $N[V/\texttt{x}]$.

Complex values

Fine-grain CBV allows pattern-matching into computations.

$$\texttt{match } V \texttt{ as } \{\texttt{inl x. } M,\ \texttt{inr x. } M'\}$$

Why not allow pattern-matching into values?

$$\texttt{match } V \texttt{ as } \{\texttt{inl x. } W,\ \texttt{inr x. } W'\}$$

+ Present in all denotational models.
- Doesn't appear in the semantics of CBV terms.
- Changes the operational character of the language: values have to be evaluated.
- Doesn't work well with recursive types.

CBN, What Doesn't Work (1): Semantics From CBV

The thunking transform CBN $\longrightarrow$ CBV (Danvy and Hatcliff).

Arguments of functions and components of tuples get thunked.

It doesn't preserve η-law for functions.

We obtain semantics for CBN not validating that η-law.

CBN, What Doesn't Work (2): Carrier Semantics

Frequently found in the semantics literature:

$$\begin{array}{rcl} [\![\texttt{bool}]\!] & = & T\mathbb{B} \\ [\![A+B]\!] & = & T([\![A]\!]+[\![B]\!]) \\ [\![A \to B]\!] & = & [\![A]\!] \to [\![B]\!] \end{array}$$

But we can't interpret `error` or `match`.

CBN Semantics With Algebras

A type denotes (not just a set but) an E-pointed set.
More generally, a T-algebra.

What's a T-algebra?

- a set X (the carrier)
- a function $TX \xrightarrow{\theta} X$ (the structure)

satisfying

$$\begin{tikzcd} X \arrow[r, "\eta X"] \arrow[dr, "\mathsf{id}"'] & TX \arrow[d, "\theta"] & T^2X \arrow[l, "\mu X"'] \arrow[d, "T\theta"] \\ & X & TX \arrow[l, "\theta"] \end{tikzcd}$$

Semantics of CBN Types

$$\begin{array}{rcl} [\![\texttt{bool}]\!] & = & F^T(1+1) \\ [\![A+B]\!] & = & F^T(U^T[\![A]\!] + U^T[\![B]\!]) \\ [\![A \to B]\!] & = & U^T[\![A]\!] \to [\![B]\!] \\ [\![A \mathbin{\Pi} B]\!] & = & [\![A]\!] \mathbin{\Pi} [\![B]\!] \end{array}$$

where we write

- F^T for the free T-algebra
- U^T for the carrier
- $\to$ for the exponential algebra
- Π for the product algebra.

Algebra Semantics: A Panacea?

Algebra semantics works well for errors and I/O, where but is more awkward for other effects. Have to prove soundness wrt operational semantics.

O'Hearn semantics of state

Big-step evaluation $s, M \Downarrow s, T$

A type denotes a set corresponding to configurations s, M.

Streicher-Reus semantics of control

CK-machine transition $M, K \leadsto M', K'$.

A type denotes a set corresponding to stacks K.

Summary

We have a denotational semantics for all of our effects and have shown correctness.

Summary

We have a denotational semantics for all of our effects and have shown correctness.

In algebra semantics,

- a CBV type denotes a set
- a CBN type denotes a T-algebra.

In semantics of dynamically generated, globally visible cells,

- a CBV type denotes a functor $\mathcal{W} \longrightarrow \mathbf{Set}$
- a CBN type denotes a functor $\mathcal{W}^{\mathrm{op}} \longrightarrow \mathbf{Set}$.

Summary

We have a denotational semantics for all of our effects and have shown correctness.

In algebra semantics,

- a CBV type denotes a set
- a CBN type denotes a T-algebra.

In semantics of dynamically generated, globally visible cells,

- a CBV type denotes a functor $\mathcal{W} \longrightarrow \mathbf{Set}$
- a CBN type denotes a functor $\mathcal{W}^{\mathsf{op}} \longrightarrow \mathbf{Set}$.

CBV types and CBN types are fundamentally different things.

Types: spot the pattern

CBN	$[\![A+B]\!]$	$[\![A \to B]\!]$
monad	$F^T(U^T[\![A]\!] + U^T[\![B]\!])$	$U^T[\![A]\!] \to [\![B]\!]$
state	$S \times ((S \to [\![A]\!]) + (S \to [\![B]\!]))$	$(S \to [\![A]\!]) \to [\![B]\!]$
control	$(([\![A]\!] \to R) + ([\![B]\!] \to R)) \to R$	$([\![A]\!] \to R) \times [\![B]\!]$

CBV	$[\![A+B]\!]$	$[\![A \to B]\!]$
monad	$[\![A]\!] + [\![B]\!]$	$U^T([\![A]\!] \to F^T[\![B]\!])$
state	$[\![A]\!] + [\![B]\!]$	$S \to ([\![A]\!] \to (S \times [\![B]\!]))$
control	$[\![A]\!] + [\![B]\!]$	$([\![A]\!] \times ([\![B]\!] \to R)) \to R$

We call these particles U, F, $\to$, $+$.

Summary: The Types of Call-By-Push-Value

value type	$A ::= \; U\underline{B} \mid 1 \mid A \times A \mid 0 \mid A + A \mid \sum_{i\in\mathbb{N}} A_i$
computation type	$\underline{B} ::= \; FA \mid A \to \underline{B} \mid 1_\Pi \mid \underline{B} \,\Pi\, \underline{B} \mid \prod_{i\in\mathbb{N}} \underline{B}_i$

As yet we do not know

- what U and F mean computationally
- why a function type is a “computation type”.

Judgements: spot the pattern

Fine-Grain CBV	$[\![\mathtt{x} : A, \mathtt{y} : B \vdash^{\mathsf{v}} V : C]\!]$	$[\![\mathtt{x} : A, \mathtt{y} : B \vdash^{\mathsf{c}} M : C]\!]$
monad	$[\![A]\!] \times [\![B]\!] \longrightarrow [\![C]\!]$	$[\![A]\!] \times [\![B]\!] \longrightarrow U^T F^T [\![C]\!]$
state	$[\![A]\!] \times [\![B]\!] \longrightarrow [\![C]\!]$	$S \times [\![A]\!] \times [\![B]\!] \longrightarrow S \times [\![C]\!]$
control	$[\![A]\!] \times [\![B]\!] \longrightarrow [\![C]\!]$	$[\![A]\!] \times [\![B]\!] \times ([\![C]\!] \rightarrow R) \longrightarrow R$

CBN	$[\![\mathtt{x} : A, \mathtt{y} : B \vdash M : C]\!]$
monad	$U^T [\![A]\!] \times U^T [\![B]\!] \longrightarrow U^T [\![C]\!]$
state	$S \times (S \rightarrow [\![A]\!]) \times (S \rightarrow [\![B]\!]) \longrightarrow [\![C]\!]$
control	$([\![A]\!] \rightarrow R) \times ([\![B]\!] \rightarrow R) \times [\![C]\!] \longrightarrow [\![C]\!]$

We obtain judgements for values $\Gamma \vdash^{\mathsf{v}} V : A$
and for computations $\Gamma \vdash^{\mathsf{c}} M : \underline{B}$
where all identifiers in Γ have value type.

Terms: spot the pattern

Fine-Grain CBV	$[\![\texttt{thunk}\ M]\!]$	$[\![\texttt{force}\ V]\!]$
monad	$[\![M]\!]$	$[\![V]\!]$
state	$\rho \mapsto \lambda s.[\![M]\!](s,\rho)$	$(s,\rho) \mapsto ([\![V]\!]\rho)s$
control	$\rho \mapsto \lambda k.[\![M]\!](\rho,k)$	$(\rho,k) \mapsto ([\![V]\!]\rho)k$

CBN	$[\![MN]\!]$	$[\![\mathtt{x}]\!]$
monad	$\rho \mapsto ([\![M]\!]\rho)'([\![N]\!]\rho)$	$\rho \mapsto \rho(x)$
state	$(s,\rho) \mapsto ([\![M]\!](s,\rho))'(\lambda s.[\![N]\!](s,\rho))$	$(s,\rho) \mapsto (\rho(x))s$
control	$(\rho,k) \mapsto [\![M]\!](\rho,\langle\lambda k.\ [\![N]\!](\rho,k),k\rangle)$	$(\rho,k) \mapsto (\rho(x))k$

We obtain particles `thunk`, `force`, `return`, sequencing, λ and application.

Typing Rules for F and U

The type FA

A computation in FA returns a value in A.

$$\frac{\Gamma \vdash^{\mathsf{v}} V : A}{\Gamma \vdash^{\mathsf{c}} \texttt{return}\ V : FA} \qquad \frac{\Gamma \vdash^{\mathsf{c}} M : FA \quad \Gamma, \mathtt{x} : A \vdash^{\mathsf{c}} N : \underline{B}}{\Gamma \vdash^{\mathsf{c}} M\ \texttt{to}\ \mathtt{x}.\ N : \underline{B}}$$

The type $U\underline{B}$

A value in $U\underline{B}$ is a thunk of a computation in $\underline{B}$.

$$\frac{\Gamma \vdash^{\mathsf{c}} M : \underline{B}}{\Gamma \vdash^{\mathsf{v}} \texttt{thunk}\ M : U\underline{B}} \qquad \frac{\Gamma \vdash^{\mathsf{v}} V : U\underline{B}}{\Gamma \vdash^{\mathsf{c}} \texttt{force}\ V : \underline{B}}$$

The constructs `thunk` and `force` are inverse.

Call-by-push-value definitional interpreter

The terminals are computations:

$$\texttt{return}\ V \qquad \lambda\mathtt{x}.M \qquad \lambda\{i.M_i\}_{i \in I}$$

Call-by-push-value definitional interpreter

The terminals are computations:

$$\texttt{return}\ V \qquad \lambda\texttt{x}.M \qquad \lambda\{i.M_i\}_{i\in I}$$

To evaluate

- $\texttt{return}\ V$: return $\texttt{return}\ V$.
- $M\ \texttt{to}\ \texttt{x}.\ N$: evaluate M. If it returns $\texttt{return}\ V$, then evaluate $N[V/\texttt{x}]$.
- $\lambda\texttt{x}.N$: return $\lambda\texttt{x}.N$.
- MV: evaluate M. If it returns $\lambda\texttt{x}.N$, evaluate $N[V/\texttt{x}]$.
- $\lambda\{i.N_i\}_{i\in I}$: return $\lambda\{i.N_i\}_{i\in I}$.
- $M\hat{\imath}$: evaluate M. If it returns $\lambda\{i.N_i\}_{i\in I}$, evaluate $N_{\hat{\imath}}$.
- $\texttt{let}\ V\ \texttt{be}\ \texttt{x}.\ M$: evaluate $M[V/\texttt{x}]$.
- $\texttt{force}\ \texttt{thunk}\ M$: evaluate M.
- $\texttt{match}\ \langle\hat{\imath}, V\rangle\ \texttt{as}\ \{\langle i, \texttt{x}\rangle.M_i\}_{i\in I}$: evaluate $M_{\hat{\imath}}[V/\texttt{x}]$.
- $\texttt{match}\ \langle V, V'\rangle\ \texttt{as}\ \langle \texttt{x}, \texttt{y}\rangle.M$: evaluate $M[V/\texttt{x}, V'/\texttt{y}]$.

Conclusions

There are many other calculi that contain call-by-name and call-by-value.

- Effect-PCF (Filinski)
- SFPL (Marz)
- $\lambda^{\mu\nu\perp}$-calculus (Howard)
- Various CPS calculi
- Polarized Linear Logic (Laurent)
- Polarized Intuitionistic Logic (Harper, Licata, Zeilberger)
- Effect Calculus (Egger, Møgelberg, Simpson)
- Call-by-push-value with complex values (Levy)

The distinctive feature of call-by-push-value is that it consists precisely of the particles that make up call-by-value and call-by-name.