Lecture 10: A Primer on Call-by-Push-Value

Luis Garcia

October 27, 2025

1 Outline

- Example program run through Call-by-Name and Call-by-Value
- Call-by-Push-Value Type System and Operational Semantics
- Encoding of CBV/CBN into CBPV

2 Background

Paul Levy came up with Call-by-Push-Value (CBPV) to come up with a mathematical account of program semantics in the presence of effects. As will be illustrated in the next section, such accounts of semantics fall apart when effects are involved: there was not a clear "story" for effects in program that would precisely characterize when and where they would occur in a program's run.

Before CBPV, operational semantics were broadly from two camps: call-by-value (CBV) or call-by-name (CBN). In the former, you cannot substitute an argument to a lambda until that argument is a value. In the latter, you can substitute the argument even if it is not a value. CBPV provides a unified view of these semantics that is backed up by category theory.

Now, why are we discussing CBPV in a class dedicated to focusing and polarity? One way to look at CBPV is as a "bizarro" form of a focused logical system. They're not exactly the same system, but there are enough similarities that Paul Levy and Noam Zeilberger (a preeminent researcher on focusing) had to discuss their work in order to detangle them. We will see some loose correspondences arise between a CBPV system and a polarized system as we go along.

3 An Example Program in CBV and CBN

Let's begin our discussion on CBPV with an example program that has effects. We will see that running through the program with our standard notions of

CBV	**CBN**
$>$ hello	$>$ world
$>$ world	$>$ hello
$>$ world	$>$ world
$> \langle 3, 4 \rangle$	$>$ hello
	$\langle 3, 4 \rangle$

Table 1: The presence of effects makes the semantics ambiguous.

operational semantics (CBV and CBN) introduces a bit of ambiguity about the order of effects.

$$\mathsf{let}\ y = \mathsf{print}\text{“hello”}; 3\ \mathsf{in}\ (\mathsf{let}\ f = \lambda x.\mathsf{print}\text{“world”}; x\ \mathsf{in}\ \langle f\ y, f\ (y+1)\rangle)$$

We would expect the statement to behave differently under CBV and CBN semantics. In fact, it prints the statements in different orders, as seen in Table 1. Not only is the ordering of effects unintuitive, but the fact that they happen in different orders with different multiplicities shows that there is some ambiguity to the semantics in the presence of effects.

4 CBPV Type-System and Operational Semantics

In CBPV, we discriminate between expressions which are values and those which are computations. For the most part, the expressions are what you would expect in a small lambda calculus, but with some additions that allow us to shift between the two syntactic domains.

$$\begin{array}{rrcl} \textit{Values} & v & ::= & x \mid (v_1, v_2) \mid () \mid \mathsf{inl}\ v \mid \mathsf{inr}\ v \mid \mathsf{susp}\ e \\ \textit{Computations} & e & ::= & \lambda x.e \mid e\ v \mid \mathsf{split}(v,\ x.y.e) \\ & & & \mid \mathsf{case}(v,\ x.e_1,\ y.e_2) \mid \mathsf{abort}\ v \mid \mathsf{return}\ v \\ & & & \mid \mathsf{force}\ v \mid e\ \mathsf{to}\ x.e' \end{array}$$

Note that susp allows us to mix expressions into our values. The expressions abort , return , and force allow us to do the same with values. abort is a special form that we use for error handling.

Types can be presented uniformly, but we will choose to present them in a stratified, polarized manner. This will allow us to more immediately see a possible link between polarity and CBPV.

$$\begin{array}{rrcl} \textit{Value Types} & A^+, B^+ & ::= & A^+ \times B^+ \mid \mathbf{1} \mid \mathbf{0} \mid A^+ + B^+ \mid UA^- \\ \textit{Computation Types} & A^-, B^- & ::= & A^+ \to B^- \mid FA^+ \end{array}$$

The types UA^- and FA^+ are *shifts* that allow us to go between the two syntactic categories.

We may now go over the typing rules. They will be stratified, as well. First, we have our typing rules for values. Note that the introduction rules for value types exactly correspond to positive types (we are treating products as positive).

$$\frac{v_1 : A^+ \qquad v_2 : B^+}{(v_1, v_2) : A^+ \times B^+} \times\mathrm{I} \qquad \frac{x : A \in \Gamma}{\Gamma \vdash x : A}\ \mathrm{hyp} \qquad \frac{}{\Gamma \vdash \langle\rangle : \mathbf{1}}\ \mathbf{1}\mathrm{I}$$

$$\frac{\Gamma \vdash v : A^+}{\mathsf{inl}\ v : A^+ + B^+} +\mathrm{I}_1 \qquad \frac{\Gamma \vdash v : B^+}{\mathsf{inl}\ v : A^+ + B^+} +\mathrm{I}_2$$

Computation types correspond to negative introduction rules (just for implication) and elimination forms.

$$\frac{\Gamma, x : A^+ \vdash e : B^-}{\Gamma \vdash \lambda x.e : A^+ \to B^-} \to\mathrm{I} \qquad \frac{\Gamma \vdash e : A^+ \to B^- \qquad \Gamma \vdash v : A^+}{\Gamma \vdash e\ v : B^-}\ \mathrm{hyp}$$

$$\frac{\Gamma \vdash v : A^+ \times B^+ \qquad \Gamma, x : A^+, y : B^+ \vdash e : C}{\mathsf{split}(v,\ x.y.e) : C} \times\mathrm{E}$$

$$\frac{\Gamma, v : \mathbf{0}}{\mathsf{abort}\ v : A}\ \mathbf{0}\mathrm{E}$$

Finally, we have our rules for shifts.

$$\frac{\Gamma \vdash v : A^+}{\Gamma \vdash \mathsf{return}\ v : FA^+}\ \mathrm{F\ I} \qquad \frac{\Gamma \vdash e : A^-}{\mathsf{susp}\ e : UA^-}\ \mathrm{U\ I}$$

$$\frac{\Gamma \vdash e : FA^+ \qquad \Gamma, x : A^+ \vdash e' : C}{\Gamma \vdash e\ \mathsf{to}\ x.e' : C}\ \mathrm{F\ E} \qquad \frac{\Gamma \vdash v : UA^-}{\Gamma \vdash \mathsf{force}\ v : A^-}\ \mathrm{U\ E}$$

Now, to discuss semantics, we must include the notion of *terminal computations*. These are just expressions that we single out as being "value-like"—that is, there is nothing more to do with them.

$$\textit{Terminal Computations} \quad T \quad ::= \quad \mathsf{return}\ v \mid \lambda x.e$$

Our big-step semantics are thus:

$$\frac{e \Downarrow \mathsf{return}\ v \qquad [v/x]e' \Downarrow T}{e\ \mathsf{to}\ x.e' \Downarrow T} \qquad \frac{e \Downarrow T}{\mathsf{force}\ \mathsf{susp}\ e \Downarrow T}$$

$$\frac{e \Downarrow \lambda x.e' \qquad [v/x]e' \Downarrow T}{e\ v \Downarrow T}$$

See Levy [1] for small-step and stack semantics for CBPV.

CBV	CBPV
$\ulcorner A^+ + B^+ \urcorner$	$\ulcorner A^+ \urcorner + \ulcorner B^+ \urcorner$
$\ulcorner A^+ \to B^+ \urcorner$	$U(\ulcorner A^+ \urcorner \to F \ulcorner B^+ \urcorner)$
$\ulcorner x : A^+ \urcorner$	$\mathsf{return}\ x : F \ulcorner A^+ \urcorner$
$\ulcorner \mathsf{inl}\ M : A^+ + B^+ \urcorner$	$\ulcorner M \urcorner\ \mathsf{to}\ x.\mathsf{return}\ x : F(\ulcorner A^+ \urcorner + \ulcorner B^+ \urcorner)$
$\ulcorner \mathsf{inr}\ M : A^+ + B^+ \urcorner$	$\ulcorner M \urcorner\ \mathsf{to}\ x.\mathsf{return}\ x : F(\ulcorner A^+ \urcorner + \ulcorner B^+ \urcorner)$
$\ulcorner \lambda x.M : A^+ \to B^+ \urcorner$	$\mathsf{return}\ (\mathsf{susp}\ (\lambda x. \ulcorner M \urcorner)) : F(U(\ulcorner A^+ \urcorner \to F \ulcorner B^+ \urcorner))$
$M\ N : B^+$	$\ulcorner M \urcorner\ \mathsf{to}\ f. \ulcorner N \urcorner\ \mathsf{to}\ x.(\mathsf{force}\ f)\ x : F \ulcorner B^+ \urcorner$

Table 2: CBV types and expressions embedded into CBPV.

5 Encoding CBV/CBN into CBPV

Now, CBPV is a unified view of operational semantics through which we can capture both CBV and CBN. Let's rewrite our example program in strictly CBV semantics using our new syntax. (Author's note: please forgive me for not knowing how to indent).

print"hello" $\mathsf{to}\ \_.3\ \mathsf{to}\ \mathsf{return}\ (\mathsf{susp}\ (\lambda x.\mathsf{print}$"world" $\mathsf{to}\ \mathsf{return}\ x))\ \mathsf{to}\ f.(\mathsf{force}\ f)\ y\ \mathsf{to}\ r_1.(\mathsf{force}\ f)(y{+}1)\ \mathsf{to}\ r_2.\mathsf{return}\ (r_1, r_2)$

We can witness a generalization of this subsumption of both CBN and CBV into CBPV in Tables 2 and 3. To read those tables, take the function $\ulcorner \_ \urcorner$ to be overloaded as follows:

- $\ulcorner \_ \urcorner$: CBV Type $A^+ \to$ CBPV Value Type A^+
- $\ulcorner \_ \urcorner$: CBV Term Typed at $A^+ \to$ CBPV Computation Typed At FA^+
- $\ulcorner \_ \urcorner$: CBN Type $A^- \to$ CBPV Computation Type A^-
- $\ulcorner \_ \urcorner$: CBV Terms Typed at $A^- \to$ CBPV Computation Typed at A^-

Exercise 1. *Redo the example program in a CBN semantics using CBPV.*

References

[1] Levy, P. B. Call-by-push-value. *ACM SIGLOG News 9*, 2 (May 2022), 7–29.

CBV	CBPV
$\ulcorner A^- + B^- \urcorner$	$F(U\ulcorner A^- \urcorner + U\ulcorner B^- \urcorner)$
$\ulcorner A^+ \to B^- \urcorner$	$U\ulcorner A^- \urcorner \to \ulcorner B^- \urcorner$
$\ulcorner x : A^- \urcorner$	$\mathsf{force}\ x : UA^-$
$\ulcorner \mathsf{inl}\ M : A^- + B^- \urcorner$	$\mathsf{return}\ (\mathsf{inl}\ (\mathsf{susp}\ \ulcorner M \urcorner)) : F(U\ulcorner A^- \urcorner + U\ulcorner B^- \urcorner)$
$\ulcorner \mathsf{inr}\ M : A^- + B^- \urcorner$	$\mathsf{return}\ (\mathsf{inr}\ (\mathsf{susp}\ \ulcorner M \urcorner)) : F(U\ulcorner A^- \urcorner + U\ulcorner B^- \urcorner)$
$\ulcorner \lambda x.M : A^- \to B^- \urcorner$	$\lambda x.\ulcorner M \urcorner : U\ulcorner A^- \urcorner \to \ulcorner B^- \urcorner$
$M\ N : B^-$	$\ulcorner M \urcorner\ (\mathsf{susp}\ \ulcorner N \urcorner) : \ulcorner B^- \urcorner$

Table 3: CBN types and expressions embedded into CBPV.