Possible World Semantics for General Storage in Call-By-Value

Paul Blain Levy

PPS, Université Denis Diderot, Case 7014, 2 Place Jussieu
75251 Paris Cedex 05, FrancePaul.Levy@pps.jussieu.fr

Abstract. We describe a simple denotational semantics, using possible worlds, for a call-by-value language with ML-like storage facilities, allowing the storage of values of any type, and the generation of new storage cells. We first present a criticism of traditional Strachey semantics for such a language: that it requires us to specify what happens when we read non-existent cells. We then obtain our model by modifying the Strachey semantics to avoid this problem.
We describe our model in 3 stages: first no storage of functions or recursion (but allowing storage of cells), then we add recursion, and finally we allow storage of functions. We discuss similarities and differences between our model and Moggi's model of ground store. A significant difference is that our model does not use monadic decomposition of the function type.

1 Storage and its Denotational Models

1.1 Overview

Many call-by-value (CBV) programming languages such as ML and Scheme provide a facility to store values in *cells*, i.e. memory locations. In ML, these cells are typed using `ref`: a cell storing values of type A is itself a value of type `ref` A. To date, besides recent work [1] blending operational and denotational semantics, there have been 3 ways of modelling such a CBV language denotationally:

- traditional Strachey-style semantics, used e.g. in [2]
- possible world semantics, used in [3–5] to model storage of ground values only
- game semantics [6].

In this paper, we argue that Strachey-style semantics, whilst very natural for a language with a fixed set of cells, is unnatural for a language in which new cells can be generated, because in the latter case it requires us to specify what happens when we read a non-existent cell, something that can never occur in reality. We modify Strachey semantics to avoid this problem, and obtain thereby a surprisingly simple possible world model for general store (not just ground store). The model is different from, and in some ways simpler than, the ground store model of [3, 4]. One notable difference is that our model does not use

Moggi's *monadic decomposition* of $A \to_{\mathbf{CBV}} B$ as $A \to TB$ [7], whereas the ground store model does.

For the purposes of exposition, we consider 3 levels of liberality in languages with storage.

1. Only ground values such as booleans and numbers can be stored.
2. As well as ground values, cells themselves can be stored.
3. Any value at all—including a function—can be stored. This is the case in ML and Scheme.

Languages of level 1 and 2 can also be classified according to whether they provide recursion. This division does not apply to languages of level 3, because recursion can be encoded using function storage, as noted by Landin (folklore).

The paper is organized as follows. We first present our criticism of Strachey semantics and give the basic ideas of the possible world semantics. After giving the syntax and big-semantics for the language, we present our model incrementally.

– We first model level 2 storage without recursion—here we can use sets instead of cpos.
– Then we model level 2 storage with recursion.
– Finally we model level 3 storage, i.e. the full language.

We compare with the ground store model and discuss some further directions.

1.2 From Strachey-Style to Possible World Semantics

For convenience of exposition we will consider a language with the following properties:

– it has level 2 storage and no recursion, so that we can work with sets rather than cpos.
– it distinguishes between a *value* $\Gamma \vdash^{\mathsf{v}} V : A$ and a *producer* $\Gamma \vdash^{\mathsf{p}} M : A$. The latter is an ordinary CBV term that can perform effects before producing an answer. (Moggi's monadic metalanguage would represent it as a term of type TA.) This explicit distinction at the level of judgements—which we call *fine-grain CBV*—makes it easier to describe the semantics.

We give a summary of the traditional Strachey semantics for such a language, where we write S for the set of states.

– A type A (and hence a context Γ) denotes a set, which we think of as the set of denotations of closed values of type A.
– A value $\Gamma \vdash^{\mathsf{v}} V : A$ denotes a function from $[\![\Gamma]\!]$ to $[\![A]\!]$.
– A producer $\Gamma \vdash^{\mathsf{p}} M : A$ denotes a function from $S \times [\![\Gamma]\!]$ to $S \times [\![A]\!]$.

A key question is how we are to interpret `ref` A. This is easy if the number of cells is fixed. If, for example, the language provides 3 boolean-storing cells, then `ref bool` will denote $\$3 = \{0,1,2\}$. Here, we use the notation $\$n$ for the set $\{0,\ldots,n-1\}$, the canonical set of size n.

But in languages such as ML and Scheme, new cells can be generated in the course of execution, and the state of the memory is given by two pieces of information:

- the *world*, which tells us how many cells there are of each type—we write $\mathcal{W}$ for the poset of worlds
- the *store*, which tells us what the cells contain—we write Sw for the set of stores in a given world w.

Thus the set S of states is given as $\sum_{w\in\mathcal{W}} Sw$. The Strachey-style semantics [2] for such a language interprets `ref` A by $\mathbb{N}$.

We claim, however, that this approach is problematic. For suppose w is a world in which there are 3 boolean-storing cells and s is a store in this world and M is the term

$$\mathtt{x} : \mathtt{ref\ bool} \ \vdash^{\mathsf{p}} \ \mathtt{read\ x\ as\ y.\ produce\ y} \ : \mathtt{bool}$$

What is $[\![M]\!](w,s)(\mathtt{x} \mapsto 7)$ going to be in our semantics? It is quite arbitrary, because what $[\![M]\!](w,s)(\mathtt{x} \mapsto 7)$ describes is absurd: the term M, executed in state (w,s), reads cell 7—which does not exist in world w—and returns the boolean that it finds there. This is operationally impossible precisely because the world can only grow bigger—if there were a "destroy cell" instruction, this situation could actually happen.

An obvious way to avoid this problem of non-existent cells is for $[\![M]\!]$ to take as arguments a state (w,s) and an environment that *makes sense in world* w. To set up such a semantics, the denotation of a type must depend on the world. For example, if w is a world where there are 3 boolean-storing cells, then $[\![\mathtt{ref\ bool}]\!]w$ is $\$3$. So the above problem does not arise.

1.3 Denotation of Function Type

Recall that in the Strachey-style semantics, using $S = \sum_{w\in\mathcal{W}} Sw$, the semantics of the function type is given by

$$\begin{aligned}[\![A \to B]\!] &= S \to [\![A]\!] \to (S \times [\![B]\!]) \\ &\cong \prod_{w'\in\mathcal{W}} (Sw' \to [\![A]\!] \to \sum_{w''\in\mathcal{W}} (Sw'' \times [\![B]\!]))\end{aligned}$$

This means that a value V of type $A \to B$ will be applied to a state (w',s') and operand U of type A, and then terminate in some state (w'',s'') with a result W of type B. But we know that if V is a w-value, then $w' \geqslant w$ and U is a w'-value, and that $w'' \geqslant w'$ and W is a w''-value. We therefore modify the above equation as follows:

$$[\![A \to B]\!]w = \prod_{w'\geqslant w} (Sw' \to [\![A]\!]w' \to \sum_{w''\geqslant w'} (Sw'' \times [\![B]\!]w'')) \tag{1}$$

In summary, this equation says that a w-value of type $A \to B$, when applied in a future state (w', s') to an operand (a w'-value of type A), will terminate in a state (w'', s'') even further in the future, returning a w''-value of type B.

1.4 Relating the Different Worlds

As we move from world w to the bigger world w', each w-value of type A in the environment becomes a w'-value of type A. In the syntax, the conversion from w-terms to w'-terms is just a trivial inclusion, but in the denotational semantics, we must explicitly provide a function from $[\![A]\!]w$ to $[\![A]\!]w'$, which we call $[\![A]\!]^{w}_{w'}$. We require

$$[\![A]\!]^{w}_{w} a = a \tag{2}$$

$$[\![A]\!]^{w}_{w''} a = [\![A]\!]^{w'}_{w''}([\![A]\!]^{w}_{w'} a) \quad \text{for } w \leqslant w' \leqslant w'' \tag{3}$$

In the terminology of category theory, A denotes a functor from the poset $\mathcal{W}$ (regarded as a category) to **Set**.

2 The Language

A *world* w is a finite multiset on types; i.e. a function from the set types of types to $\mathbb{N}$ such that the set $\mathsf{cells}\ w = \sum_{A \in \mathsf{types}} \$ w_A$ is finite. We use worlds to formulate the syntax in Fig. 1. Notice that if $w \leqslant w'$ then every w-term is also a w'-term—this fact will be used implicitly in the big-step semantics.

A *syntactic w-store* π is a function associating to each cell $(A, l) \in \mathsf{cells}\ w$ a closed w-value of type A. By contrast we will use s to represent a *denotational-semantic* store—this distinction is important when we have function storage. A *syntactic state* is a pair w, π where π is a syntactic w-store. We use syntactic states to present big-step semantics in Fig. 2.

Definition 1 (observational equivalence). *Given two producers $\Gamma \vdash^{\mathsf{p}} M, N : A$, we say that $M \simeq N$ when for every ground context $\mathcal{C}[\cdot]$, i.e. context which is a producer of ground type* `bool`*, and for every syntactic state w, π and every n we have*

$$\exists w', \pi'(w, \pi, \mathcal{C}[M] \Downarrow w', \pi', n) \text{ iff } \exists w', \pi'(w, \pi, \mathcal{C}[N] \Downarrow w', \pi', n)$$

We similarly define $\simeq$ for values.

We list in Fig. 3 some basic equivalences that all the CBV models, including the Strachey semantics, validate.

3 Denotational Semantics Without Divergence

In this section we exclude `diverge` and μ and storage of functions, so that we can model using sets rather than cpos. We say that a type D is a *data type* if

Types $\quad A ::= \ \mathtt{bool} \mid 1 \mid A \times A \mid A \to A \mid \mathtt{ref}\ A$

Rules for 1 are omitted, as it is analogous to $\times$.

Judgements $\quad w|\Gamma \vdash^{\mathsf{v}} V : A \qquad w|\Gamma \vdash^{\mathsf{p}} M : A \qquad$ where w is a world.

In the special case $w = 0$, we write $\Gamma \vdash^{\mathsf{v}} V : A$ and $\Gamma \vdash^{\mathsf{p}} M : A$.

Terms

$$\frac{}{w|\Gamma, \mathtt{x} : A, \Gamma' \vdash^{\mathsf{v}} \mathtt{x} : A} \qquad \frac{w|\Gamma \vdash^{\mathsf{v}} V : A \quad w|\Gamma, \mathtt{x} : A \vdash^{\mathsf{p}} M : B}{w|\Gamma \vdash^{\mathsf{p}} \mathtt{let}\ V\ \mathtt{be}\ \mathtt{x}.\ M : B}$$

$$\frac{w|\Gamma \vdash^{\mathsf{v}} V : A}{w|\Gamma \vdash^{\mathsf{p}} \mathtt{produce}\ V : A} \qquad \frac{w|\Gamma \vdash^{\mathsf{p}} M : A \quad w|\Gamma, \mathtt{x} : A \vdash^{\mathsf{p}} N : B}{w|\Gamma \vdash^{\mathsf{p}} M\ \mathtt{to}\ \mathtt{x}.\ N : B}$$

$$\frac{}{w|\Gamma \vdash^{\mathsf{v}} \mathtt{true} : \mathtt{bool}} \qquad \frac{}{w|\Gamma \vdash^{\mathsf{v}} \mathtt{false} : \mathtt{bool}}$$

$$\frac{w|\Gamma \vdash^{\mathsf{v}} V : \mathtt{bool} \quad w|\Gamma \vdash^{\mathsf{p}} M : B \quad w|\Gamma \vdash^{\mathsf{p}} M' : B}{w|\Gamma \vdash^{\mathsf{p}} \mathtt{if}\ V\ \mathtt{then}\ M\ \mathtt{else}\ M' : B}$$

$$\frac{w|\Gamma \vdash^{\mathsf{v}} V : A \quad w|\Gamma \vdash^{\mathsf{v}} V' : A'}{w|\Gamma \vdash^{\mathsf{v}} (V, V') : A \times A'} \qquad \frac{w|\Gamma \vdash^{\mathsf{v}} V : A \times A' \quad w|\Gamma, \mathtt{x} : A, \mathtt{y} : A' \vdash^{\mathsf{p}} M : B}{w|\Gamma \vdash^{\mathsf{p}} \mathtt{pm}\ V\ \mathtt{as}\ (\mathtt{x}, \mathtt{y}).M : B}$$

$$\frac{w|\Gamma, \mathtt{x} : A \vdash^{\mathsf{p}} M : B}{w|\Gamma \vdash^{\mathsf{v}} \lambda \mathtt{x}.M : A \to B} \qquad \frac{w|\Gamma \vdash^{\mathsf{v}} V : A \quad w|\Gamma \vdash^{\mathsf{v}} W : A \to B}{w|\Gamma \vdash^{\mathsf{p}} V`W : B}$$

Terms For Divergence/Recursion

$$\frac{}{w|\Gamma \vdash^{\mathsf{p}} \mathtt{diverge} : B} \qquad \frac{w|\Gamma, \mathtt{f} : A \to B, \mathtt{x} : A \vdash^{\mathsf{p}} M : B}{w|\Gamma \vdash^{\mathsf{v}} \mu \mathtt{f} \lambda \mathtt{x}.M : A \to B}$$

Terms For Storage

$$\frac{}{w|\Gamma \vdash^{\mathsf{v}} \mathtt{cell}_A l}\ (A, l) \in \mathsf{cells}\ w \qquad \frac{w|\Gamma \vdash^{\mathsf{v}} V : \mathtt{ref}\ A \quad w|\Gamma \vdash^{\mathsf{v}} W : A \quad w|\Gamma \vdash^{\mathsf{p}} M : B}{w|\Gamma \vdash^{\mathsf{p}} V := W.\ M : B}$$

$$\frac{w|\Gamma \vdash^{\mathsf{v}} V : \mathtt{ref}\ A \quad w|\Gamma, \mathtt{x} : A \vdash^{\mathsf{p}} M : B}{w|\Gamma \vdash^{\mathsf{p}} \mathtt{read}\ V\ \mathtt{as}\ \mathtt{x}.\ M : B} \qquad \frac{w|\Gamma \vdash^{\mathsf{v}} V : A \quad w|\Gamma, \mathtt{x} : \mathtt{ref}\ A \vdash^{\mathsf{p}} M : B}{w|\Gamma \vdash^{\mathsf{p}} \mathtt{new}\ \mathtt{x} := V.\ M : B}$$

$$\frac{w|\Gamma \vdash^{\mathsf{v}} V : \mathtt{ref}\ A \quad w|\Gamma \vdash^{\mathsf{v}} V' : \mathtt{ref}\ A \quad w|\Gamma \vdash^{\mathsf{p}} M : B \quad w|\Gamma \vdash^{\mathsf{p}} M' : B}{w|\Gamma \vdash^{\mathsf{p}} \mathtt{if}\ V = V'\ \mathtt{then}\ M\ \mathtt{else}\ M' : B}$$

Here, we do not allow $V = V'$ to be a boolean value, because the operational semantics exploits the fact that values do not need to be evaluated.

Fig. 1. Terms of Fine-Grain CBV

The form of the big-step semantics is $w, \pi, M \Downarrow w', \pi', W$ where

- w, π is a syntactic state
- M is a closed w-producer
- w', π' is a syntactic state such that $w' \geqslant w$
- W is a closed w'-value of the same type as M.

$$\frac{w, \pi, M[V/\mathtt{x}] \Downarrow w', \pi', W}{w, \pi, \mathtt{let}\ V\ \mathtt{be}\ \mathtt{x}.\ M \Downarrow w', \pi', W}$$

$$\frac{}{w, \pi, \mathtt{produce}\ V \Downarrow w, \pi, V} \qquad \frac{w, \pi, M \Downarrow w', \pi', V \quad w', \pi', N[V/\mathtt{x}] \Downarrow w'', \pi'', W}{w, \pi, M\ \mathtt{to}\ \mathtt{x}.\ N \Downarrow w'', \pi'', W}$$

$$\frac{w, \pi, M[V/\mathtt{x}, V'/\mathtt{y}] \Downarrow w', \pi', W}{w, \pi, \mathtt{pm}\ (V, V')\ \mathtt{as}\ (\mathtt{x}, \mathtt{y}).\ M \Downarrow w', \pi', W}$$

$$\frac{w, \pi, M[V/\mathtt{x}] \Downarrow w', \pi', W}{w, \pi, V`\lambda\mathtt{x}.M \Downarrow w', \pi', W}$$

$$\frac{w, \pi, \mathtt{diverge} \Downarrow w', \pi', W}{w, \pi, \mathtt{diverge} \Downarrow w', \pi', W} \qquad \frac{w, \pi, M[V/\mathtt{x}, \mu\mathtt{f}\lambda\mathtt{x}.M/\mathtt{f}] \Downarrow w', \pi', W}{w, \pi, V`\mu\mathtt{f}\lambda\mathtt{x}.M \Downarrow w', \pi', W}$$

$$\frac{w, \pi, M[V/\mathtt{x}] \Downarrow w', \pi', W}{w, \pi, \mathtt{read}\ \mathtt{cell}_A l\ \mathtt{as}\ \mathtt{x}.\ M \Downarrow w', \pi', W} \quad V \text{ is the contents of } A\text{-storing cell } l \text{ in } \pi$$

$$\frac{w, \pi', M \Downarrow w'', \pi'', W}{w, \pi, \mathtt{cell}_A l := V;\ M \Downarrow w'', \pi'', W} \quad \pi' \text{ is } \pi \text{ with } A\text{-storing cell } l \text{ assigned } V$$

$$\frac{w', \pi', M[\mathtt{cell}_A l/\mathtt{x}] \Downarrow w'', \pi'', W}{w, \pi, \mathtt{new}\ \mathtt{x} := V;\ M \Downarrow w'', \pi'', W} \quad (w', \pi') \text{ is } (w, \pi) \text{ extended with a cell } l \text{ storing } V$$

$$\frac{w, \pi, M \Downarrow w', \pi', W}{w, \pi, \mathtt{if}\ \mathtt{cell}_A l = \mathtt{cell}_A l\ \mathtt{then}\ M\ \mathtt{else}\ M' \Downarrow w', \pi', W}$$

$$\frac{w, \pi, M' \Downarrow w', \pi', W}{w, \pi, \mathtt{if}\ \mathtt{cell}_A l = \mathtt{cell}_A l'\ \mathtt{then}\ M\ \mathtt{else}\ M' \Downarrow w', \pi', W}\ (l \neq l')$$

Exploiting determinism, we say that $w\pi, M$ *diverges* when there is no w', π', V such that $w, \pi, M \Downarrow w', \pi', V$.

Fig. 2. Big-Step Semantics for Fine-Grain CBV with Storage

We employ the *bound/unbound* convention: when, in an equation—such as the η-law $M = \lambda\mathtt{x}.(\mathtt{x}`M)$—the term $\Gamma \vdash^{\mathsf{c}} M : \underline{B}$ occurs both in the scope of an $\mathtt{x}$-binder and not in the scope of an $\mathtt{x}$-binder, we assume $\mathtt{x} \notin \Gamma$. We do not write the weakening explicitly.

$$
\begin{array}{llcl}
(\beta) & \mathtt{let}\ \mathtt{x}\ \mathtt{be}\ V.\ M & = & M[V/\mathtt{x}] \\
(\beta) & \mathtt{if}\ \mathtt{true}\ \mathtt{then}\ M\ \mathtt{else}\ M' & = & M \\
(\beta) & \mathtt{if}\ \mathtt{false}\ \mathtt{then}\ M\ \mathtt{else}\ M' & = & M' \\
(\beta) & \mathtt{pm}\ (V, V')\ \mathtt{as}\ (\mathtt{x}, \mathtt{y}).M & = & M[V/\mathtt{x}, V'/\mathtt{y}] \\
(\beta) & (\lambda\mathtt{x}.M)V & = & M[V/\mathtt{x}] \\
(\beta) & \mathtt{produce}\ V\ \mathtt{to}\ \mathtt{x}.\ M & = & M[V/\mathtt{x}] \\
(\eta) & M[V/\mathtt{z}] & = & \mathtt{if}\ V\ \mathtt{then}\ M[\mathtt{true}/\mathtt{z}]\ \mathtt{else}\ M[\mathtt{false}/\mathtt{z}] \\
(\eta) & M[V/\mathtt{z}] & = & \mathtt{pm}\ V\ \mathtt{as}\ (\mathtt{x}, \mathtt{y}).M[(\mathtt{x}, \mathtt{y})/\mathtt{z}] \\
(\eta) & V & = & \lambda\mathtt{x}.(V\mathtt{x}) \\
(\eta) & M & = & M\ \mathtt{to}\ \mathtt{x}.\ \mathtt{produce}\ \mathtt{x} \\
 & (P\ \mathtt{to}\ \mathtt{x}.\ M)\ \mathtt{to}\ \mathtt{y}.\ N & = & P\ \mathtt{to}\ \mathtt{x}.\ (M\ \mathtt{to}\ \mathtt{y}.\ N) \\
 & (V := W; M)\ \mathtt{to}\ \mathtt{y}.\ N & = & V := W; (M\ \mathtt{to}\ \mathtt{y}.\ N) \\
 & (\mathtt{read}\ V\ \mathtt{as}\ \mathtt{x}.\ M)\ \mathtt{to}\ \mathtt{y}.\ N & = & \mathtt{read}\ V\ \mathtt{as}\ \mathtt{x}.\ (M\ \mathtt{to}\ \mathtt{y}.\ N) \\
 & (\mathtt{new}\ \mathtt{x} := V;\ M)\ \mathtt{to}\ \mathtt{y}.\ N & = & \mathtt{new}\ \mathtt{x} := V;\ (M\ \mathtt{to}\ \mathtt{y}.\ N)
\end{array}
$$

Fig. 3. Basic CBV Equivalences, Using Bound/Unbound Convention

values of type D can be stored. The types of the restricted language are given by

$$
\begin{array}{rl}
D ::= & \mathtt{bool} \mid D \times D \mid \mathtt{ref}\ D \\
A ::= & D \mid \mathtt{bool} \mid A \times A \mid A \to A
\end{array}
$$

Proposition 1. *Let M be a w-producer and s a w-store in this restricted language. Then $w, \pi, M \Downarrow w', \pi', W$ for (clearly unique) w', π', W.*

This is proved by a standard Tait-style argument. We now present the denotational semantics for this restricted language. As we stated in the introduction, each type A in each world w denotes a set $[\![A]\!]w$. These sets are given by

$$
\begin{aligned}
Sw &= \prod_{(D,l) \in \mathsf{cells}\ w} [\![D]\!]w \\
[\![\mathtt{bool}]\!]w &= \{\mathsf{true}, \mathsf{false}\} \\
[\![A \times A']\!]w &= [\![A]\!]w \times [\![A']\!]w \\
[\![\mathtt{ref}\ A]\!]w &= \$w_A \\
[\![A \to B]\!]w &= \prod_{w' \geqslant w} (Sw' \to [\![A]\!]w' \to \sum_{w'' \geqslant w'} (Sw'' \times [\![B]\!]w''))
\end{aligned}
$$

The functions $[\![A]\!]^w_{w'}$ are given simply:

- $[\![\mathtt{bool}]\!]^w_{w'}$ is the identity on $\{\mathsf{true}, \mathsf{false}\}$.
- $[\![A \times A']\!]^w_{w'}$ takes (a, a') to $([\![A]\!]^w_{w'} a, [\![A]\!]^w_{w'} a')$.
- $[\![\mathtt{ref}\ D]\!]^w_{w'}$ is the inclusion from $\$w_D$ to $\$w'_D$.

– $[\![A \to B]\!]^w_{w'}$ takes a family $\{f_{w''}\}_{w'' \geqslant w}$ to the restricted family $\{f_{w''}\}_{w'' \geqslant w'}$.

It is easily verified that they satisfy (2)–(3). A context Γ is interpreted similarly.

A value $w_0|\Gamma \vdash^{\mathsf{v}} V : A$ will denote, for each world $w \geqslant w_0$, a function $[\![V]\!]w$ from $[\![\Gamma]\!]w$ to $[\![A]\!]w$. These functions are related: if $w_0 \leqslant w \leqslant w'$ then

$$\begin{tikzcd} {[\![\Gamma]\!]w} \arrow[r, "{[\![V]\!]w}"] \arrow[d, "{[\![\Gamma]\!]^w_{w'}}"'] & {[\![A]\!]w} \arrow[d, "{[\![A]\!]^w_{w'}}"] \\ {[\![\Gamma]\!]w'} \arrow[r, "{[\![V]\!]w'}"'] & {[\![A]\!]w'} \end{tikzcd} \qquad \text{must commute.} \tag{4}$$

Informally, (4) says that if we have an environment ρ of closed w-values, substitute into V and then regard the result as a closed w'-value, we obtain the same as if we regard ρ as an environment of closed w'-values and substitute it into V.

The special case that $w_0 = 0$, in which we have a value $\Gamma \vdash^{\mathsf{v}} V : A$, is interesting. In categorical terminology, V denotes a natural transformation from $[\![\Gamma]\!]$ to $[\![A]\!]$.

A producer $w_0|\Gamma \vdash^{\mathsf{p}} M : A$ denotes, for each $w \geqslant w_0$, a function $[\![M]\!]w$ from $Sw \times [\![\Gamma]\!]w$ to $\sum_{w' \geqslant w}(Sw' \times [\![A]\!]w')$. This is because in a given state (w, π) where $w \geqslant w_0$ and environment of w-values, it terminates in a state (w', π'), where $w' \geqslant w$, producing a w'-value. There is no required relationship between the functions $[\![M]\!]w$ for different w. The semantics of terms is straightforward.

Remark 1. According to the prescription above, the denotation of a closed value $w| \vdash^{\mathsf{v}} V : A$ is an element of

$$\{a \in \prod_{w' \geqslant w} [\![A]\!]w : [\![A]\!]^{w'}_{w''} a(w') = a(w'') \text{ when } w \leqslant w' \leqslant w''\}$$

There is an obvious bijection between this set and $[\![A]\!]w$. This shows that our thinking of $[\![A]\!]w$ as the set of denotations of closed w-values of type A, which pervades the informal parts of this paper, is in agreement with the technical development.

Remark 2. For each datatype D, the function $[\![-]\!]$ from the set of closed w-values of type D to the set $[\![D]\!]w$ is a bijection, by induction on D. Because of this, until Sect. 6, we neglect the distinction between syntactic and denotational-semantic store, and we write both as s.

Proposition 2 (soundness). *If $w, s, M \Downarrow w', s', W$ then $[\![M]\!]ws = (w', s', [\![W]\!]w')$.*

This is proved by straightforward induction.

Corollary 1. *(by Prop. 1) If M is a closed ground w-producer (i.e. producer of type* `bool`*) then $w, \pi, M \Downarrow w', \pi', n$ iff $[\![M]\!]ws = (w', \pi', n)$. Hence terms with the same denotation are observationally equivalent.*

4 Adding Recursion

In this section, we allow the `diverge` and recursion constructs, but we continue to prohibit function storage. We thus avoid Sw and $[\![A \to B]\!]w$ being mutually recursive.

In the denotational model, $[\![A]\!]w$ is a cpo rather than a set, although $[\![Dw]\!]$ (for a datatype D) and Sw will continue to be sets (or flat cpos). The functions $[\![A]\!]^{w}_{w'}$ and $[\![V]\!]w$ and $[\![M]\!]w$ are required to be continuous. In the language of category theory, a type denotes a functor from $\mathcal{W}$ to **Cpo** (the category of cpos and continuous functions), and a value $\Gamma \vdash^{\mathsf{v}} V : A$ again denotes a natural transformation.

The key semantic equation (1) must be modified for the possibility of divergence

$$[\![A \to_{\mathbf{CBV}} B]\!]w = \prod_{w' \geqslant w} (Sw' \to [\![A]\!]w' \to (\sum_{w'' \geqslant w'} (Sw'' \times [\![B]\!]w''))_{\perp}) \qquad (5)$$

This equation says that a w-value of type $A \to B$, when applied in a future-world store (w', π') to an operand (a w'-value of type A), will *either* diverge *or* terminate in state (w'', s'') returning a w''-value of type B.

Similarly a producer $w_0|\Gamma \vdash^{\mathsf{p}} M : A$ will now denote, in each world $w \geqslant w_0$, a continuous function $[\![M]\!]w$ from $Sw \times [\![\Gamma]\!]w$ to $(\sum_{w' \geqslant w}(Sw' \times [\![A]\!]w'))_{\perp}$. The lifting allows for the possibility of divergence. The interpretation of terms is straightforward.

Proposition 3 (soundness/adequacy).

1. *If $w, s, M \Downarrow w', s', W$ then $[\![M]\!]ws = (w', s', [\![W]\!]w')$.*
2. *If w, s, M diverges, then $[\![M]\!]ws = \perp$.*

Proof. (1) is straightforward. For (2), we define admissible relations $\leqslant^{\mathsf{v}}_{A\,w}$ between $[\![A]\!]w$ and closed w-values of type A, for which $a \leqslant^{\mathsf{v}}_{Aw} V$ and $w \leqslant x$ implies $([\![A]\!]^{w}_{x})a \leqslant^{\mathsf{v}}_{Ax} V$, and $\perp$-containing admissible relations $\leqslant^{\mathsf{p}}_{A\,w}$ between $(\sum_{w' \geqslant w}(Sw' \times [\![A]\!]w'))_{\perp}$ and triples x, s, M (where $x \geqslant w$ and s is a x-store and M is a closed x-producer of type A). These are defined by mutual induction on types in the evident way. For data types D, we will have $d \leqslant^{\mathsf{v}}_{D,w} V$ iff $d = [\![V]\!]w$. We prove that for any producer $w|A_0, \ldots, A_{n-1} \vdash^{\mathsf{p}} M : A$, if $w \leqslant x$ and $s \in Sx$ and $a_i \leqslant^{\mathsf{v}}_{A_i x} W_i$ for $i = 0, \ldots, n-1$ then $[\![M]\!]xs\overrightarrow{a_i} \leqslant^{\mathsf{p}}_{Ax} x, s, M[\overrightarrow{W_i/\mathtt{x}_i}]$; and similarly for values. The required result is immediate.

Corollary 2. *If M is a closed ground w-producer then $w, \pi, M \Downarrow w', \pi', n$ iff $[\![M]\!]ws = \mathsf{lift}(w', \pi', n)$ and w, π, M diverges iff $[\![M]\!]ws = \perp$. Hence terms with the same denotation are observationally equivalent.*

5 Theory of Enriched-Compact Categories

We review some key results about solution of domain/predomain equations from [8, 9]. Whilst those papers work with the category $\mathbf{Cpo}^{\perp}$ of pointed cpos and strict continuous functions, everything generalizes[1] to enriched-compact categories, as we now describe. All of this material is somewhat implicit in [11].

Definition 2. *An* enriched-compact category $\mathcal{C}$ *is a* **Cpo**-*enriched category with the following properties.*

- *Each hom-cpo $\mathcal{C}(A, B)$ has a least element $\perp$.*
- *Composition is bi-strict i.e. $\perp; g = \perp = f; \perp$.*
- *$\mathcal{C}$ has a zero object i.e. an object which is both initial and terminal. (Because of bi-strictness, just one of these properties is sufficient.)*
- *Writing $\mathcal{C}^{\mathsf{ep}}$ for the category of embedding-projection pairs in $\mathcal{C}$, we have that for every countable directed diagram $D : \mathbb{D} \longrightarrow \mathcal{C}^{\mathsf{ep}}$ has an* O-colimit *(necessarily unique up to unique isomorphism). We recall from [9] that an O-colimit for D is defined to be a cocone $(V, \{(e_d, p_d)\}_{d \in \mathbb{D}})$ from D in $\mathcal{C}^{\mathsf{ep}}$ satisfying*
$$\bigvee_{d \in \mathbb{D}} (p_d; e_d) = \mathsf{id}_V \tag{6}$$

Definition 3. *Let F be a locally continuous functor from $\mathcal{C}^{\mathsf{op}} \times \mathcal{C}$ to $\mathcal{C}$. Then an* invariant *for F is an object D together with an isomorphism $i : F(D, D) \cong D$. It is a* minimal invariant *when the least fixed point of the continuous endofunction on $\mathcal{C}(D, D)$ taking e to $i^{-1}; F(e, e); i$ is the identity.*

Proposition 4. *Let F be a locally continuous functor from $\mathcal{C}^{\mathsf{op}} \times \mathcal{C}$ to $\mathcal{C}$. Then F has a minimal invariant, and it is unique up to unique isomorphism.*

Proof. This is proved as in [8].

Definition 4. *1. A subcategory $\mathcal{C}$ of a category $\mathcal{D}$ is* lluf *[12] when $\mathsf{ob}\ \mathcal{C} = \mathsf{ob}\ \mathcal{D}$.*

2. If $\mathcal{B}$ is a subcategory of $\mathcal{D}$ we write $\mathcal{B} \bullet\!\!\rightarrow \mathcal{D}$ for the category with the objects of $\mathcal{B}$ and the morphisms of $\mathcal{D}$, i.e. the unique category $\mathcal{C}$ such that
$$\mathcal{B} \subset_{\mathsf{lluf}} \mathcal{C} \subset_{\mathsf{full}} \mathcal{D}$$

3. A lluf admissible subcategory $\mathcal{B}$ of an enriched-compact category $\mathcal{C}$ is embedding-complete *when it contains all the embeddings (and in particular the isomorphisms) in $\mathcal{D}$.*

Def. 4(3) is important because frequently we seek an isomorphism in a **Cpo**-enriched category $\mathcal{B}$ which is not enriched-compact (such as **Cpo**). So we look for an enriched-compact category $\mathcal{D}$ that contains $\mathcal{C}$ as an embedding-complete subcategory.

Proposition 5. *1. The category $\mathbf{Cpo}^{\perp}$ is enriched-compact.*

*2. The category **Cpo** is an embedding-complete subcategory of the enriched-compact category **pCpo** of cpos and partial continuous functions.*

[1] Another generalization is to the "rational categories" of [10], but they are for call-by-name.

3. *Any small product* $\prod_{i\in I}\mathcal{C}_i$ *of enriched-compact categories is enriched-compact. If* $\mathcal{B}_i \subset \mathcal{C}_i$ *is embedding-complete for all* $i\in I$*, then so is* $\prod_{i\in I}\mathcal{B}_i \subset \prod_{i\in I}\mathcal{C}_i$*.*
4. *Let* $\mathfrak{I}$ *be a small category and* $\mathcal{C}$ *be enriched-compact. Then the functor category* $[\mathfrak{I},\mathcal{C}]$ *is enriched-compact.*
5. *Let* $\mathfrak{I}$ *be a small category. Then* $[\mathfrak{I},\mathbf{Cpo}]$ *is an embedding-complete subcategory of the enriched-compact category* $[\mathfrak{I},\mathbf{Cpo}] \bullet\!\!\to [\mathfrak{I},\mathbf{pCpo}]$*.*

Proof. (1)–(3) are standard.

(4) Given a countable directed diagram D in $[\mathfrak{I},\mathcal{C}]$, set $(Vi,\{(e_{di},p_{di})\}_{d\in\mathbb{D}})$ to be the O-colimit in $\mathcal{C}$ of Di and set $Vi \xrightarrow{Vf} Vj$ to be $\bigvee_{d\in\mathbb{D}}(p_{di};D_{df};e_{dj})$ for $i \xrightarrow{f} j$. The required properties are trivial.

(5) We construct the O-colimit of a countable directed diagram D in $[\mathfrak{I},\mathbf{Cpo}] \bullet\!\!\to [\mathfrak{I},\mathbf{pCpo}]$ as in the previous case. We need to show that $Vi \xrightarrow{Vf} Vj$ is total for any $f : i \longrightarrow j$. Given $x \in Vw$, we know that

$$\bigvee_{d\in\mathbb{D}}(p_{dw};e_{dw})x = x$$

Therefore, for sufficiently large d, x is in the domain of p_{dw}. Hence, for such d, x is in the domain of $p_{dw};D_{df};e_{dw'}$, because D_{df} and $e_{dw'}$ are total. So x is in the domain of $Vf = \bigvee_{d\in\mathbb{D}}(p_{di};D_{df};e_{dj})$ as required.

6 Storing Functions

We now want to model the full language. We want to provide a cpo Sw for each world w and a functor $\mathcal{W} \xrightarrow{[\![A]\!]w} \mathbf{Cpo}$ for each type A. Thus we seek an object (and isomorphism) in the category

$$\mathcal{C}_0 = \prod_{w\in\mathcal{W}}\mathbf{Cpo} \times \prod_{A\in\mathsf{types}}[\mathcal{W},\mathbf{Cpo}]$$

By Prop. 5, this is an embedding-complete subcategory of the enriched-compact category

$$\mathcal{C} = \prod_{w\in\mathcal{W}}\mathbf{pCpo} \times \prod_{A\in\mathsf{types}}([\mathcal{W},\mathbf{Cpo}] \bullet\!\!\to [\mathcal{W},\mathbf{pCpo}])$$

We define a locally continuous functor F from $\mathcal{C}^{\mathrm{op}} \times \mathcal{C}$ to $\mathcal{C}$ in Fig. 4; its minimal invariant is an object and isomorphism in $\mathcal{C}_0$—this is our semantics of types. Semantics of terms proceeds as in Sect. 4, with isomorphisms inserted where required.

Proposition 6 (soundness/adequacy).

1. *If* $w,\pi,M \Downarrow w',\pi',W$ *then* $[\![M]\!]ws = (w',\pi',[\![W]\!]w')$*.*
2. *If* w,π,M *diverges, then* $[\![M]\!]ws = \bot$*.*

Construction of $F : \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \longrightarrow \mathcal{C}$
For objects D, E

$$\begin{aligned}
F(D,E)_{Sw} &= \prod_{(A,l)\in \mathsf{cells}\ w} E_A \\
F(D,E)_{\mathtt{bool}w} &= \{\mathsf{true}, \mathsf{false}\} \\
F(D,E)_{\mathtt{bool}{}^{w}_{x}} b &= b \\
F(D,E)_{(A\times A')w} &= E_{Aw} \times E_{A'w} \\
F(D,E)_{A\times A'{}^{w}_{x}} &= (E_{A{}^{w}_{x}} c, E_{A'{}^{w}_{x}} c') \\
F(D,E)_{(\mathtt{ref}\ A)w} &= \$w_A \\
F(D,E)_{\mathtt{ref}\ A{}^{w}_{x}} i &= i \\
F(D,E)_{(A\to B)w} &= \prod_{w'\geqslant w} (D_{Sw'} \to D_{Aw'} \to (\sum_{w''\geqslant w'} (E_{Sw''} \times E_{Bw''}))_\bot) \\
F(D,E)_{A\to B{}^{w}_{x}} &= \lambda x's'.fx's'
\end{aligned}$$

For morphisms $D' \xrightarrow{h} D$ and $E \xrightarrow{k} E'$

$$\begin{aligned}
F(h,k)_{Sw} s &= \begin{cases} ((A,l) \mapsto k_{Sw} s(A,l)) \\ \quad \text{if } k_{Sw} s(A,l) \text{ is defined for all } (A,l) \in \mathsf{cells}\ w \\ \text{undefined otherwise} \end{cases} \\
F(h,k)_{\mathtt{bool}w} b &= b \\
F(h,k)_{(\mathtt{ref}\ A)w} i &= i \\
F(h,k)_{(A\times A')w}(c,c') &= \begin{cases} (k_{Aw}c, k_{A'w}c') \\ \quad \text{if } k_{Aw}c \text{ and } k_{A'w}c' \text{ are defined} \\ \text{undefined otherwise} \end{cases} \\
F(h,k)_{(A\to B)w} f &= \lambda w'.\lambda s'.\lambda a'. \begin{cases} \mathsf{lift}(w'', k_{Sw''}s'', k_{Bw''}b'') \\ \quad \text{if } h_{Sw'}s' \text{ and } h_{Aw'}a' \text{ are defined} \\ \quad \text{and } fw'(h_{Sw'}s')(h_{Aw'}a') = \mathsf{lift}(w'', s'', b'') \\ \quad \text{and } k_{Sw''}s'' \text{ and } k_{Bw''}a'' \text{ are defined} \\ \bot \text{ otherwise} \end{cases}
\end{aligned}$$

Fig. 4. Construction of F

Proof. (1) is straightforward induction. The proof of (2) is obtained from that of Prop. 3(2), using Pitts' techniques [8], which generalize to an arbitrary enriched-compact category.

Corollary 3. *If M is a closed ground w-producer then $w, \pi, M \Downarrow w', \pi', n$ iff $[\![M]\!]ws = \mathsf{lift}(w', \pi', n)$ and w, π, M diverges iff $[\![M]\!]ws = \bot$. Hence terms with the same denotation are observationally equivalent.*

7 Monadic Decomposition and the Ground-Store Model

The set model of Sect. 3 gives us the following structure on the cartesian category $[\mathcal{W}, \mathbf{Set}]$:

$$
\begin{aligned}
(A \rightarrow_{\mathbf{CBV}} B)w &= \prod_{w' \geqslant w} (Sw' \to Aw' \to \sum_{w'' \geqslant w'} (Sw'' \times Bw'')) \\
(A \rightarrow_{\mathbf{CBV}} B)^w_x &= \lambda \mathsf{x}' s'. f x' s' \\
TBw &= \prod_{w' \geqslant w} (Sw' \to \sum_{w'' \geqslant w'} (Sw'' \times Bw'')) \\
TB^w_x &= \lambda \mathsf{x}' s'. f x' s'
\end{aligned}
$$

We know from Moggi's theory that, for any model of fine-grain CBV, when TB is set to be $1 \rightarrow_{\mathbf{CBV}} B$ as it is here, we can extend T to a strong monad, and then $A \rightarrow_{\mathbf{CBV}} B$ must be an exponential from A to TB. But the decomposition of $A \rightarrow_{\mathbf{CBV}} B$ as $A \to TB$ is hardly obvious here. It seems that a more natural categorical organization for our model is the "closed Freyd category" [13].

We recall the ground store model of [3], as generalized in [4], and see how it differs from ours. Let $\mathfrak{I}$ be the category of worlds and *injections*. Because we are dealing with ground store only, S is a functor from $\mathfrak{I}^{\mathrm{op}}$ to $\mathbf{Set}$: a store in a bigger world can always be restricted to a store in a smaller world.

The ground store model interprets values in the cartesian category $[\mathfrak{I}, \mathbf{Set}]$. This category has exponentials described as an end

$$(A \to B)w = \int_{w' \in (w/\mathfrak{I})} (Aw' \to Bw') \tag{7}$$

and a strong monad described using a coend

$$(TB)w = Sw \to \int^{w' \in (w/\mathfrak{I})} (Sw' \times Bw') \tag{8}$$

By monadic decomposition we obtain

$$(A \rightarrow_{\mathbf{CBV}} B)w = \int_{w' \in (w/\mathfrak{I})} (Aw' \to Sw' \to \int^{w'' \in (w'/\mathfrak{I})} (Sw'' \times Bw'')) \tag{9}$$

whose similarity to (1) is evident. Notice the importance of the contravariance of S for (8) to be covariant in w, and indeed for the coend to be meaningful.

Once we can store cells, S is no longer contravariant: if $w \leqslant w'$, a w'-store s' cannot necessarily be restricted to a w-store, because some w-cell in s' might be storing a non-w-cell. Another difficulty is moving from sets to cpos, because although colimits of cpos exist [14], they are unwieldy.

An advantage of the ground store model over ours is that it validates the equivalences (employing the bound/unbound convention)

$$\texttt{new x} := V; M \simeq M$$
$$\texttt{new x} := V; \texttt{new y} := W; M \simeq \texttt{new y} := W; \texttt{new x} := V; M$$

We hope that our work will provide a starting-point for work on parametric models validating these and other equivalences.

8 Relationship with Call-By-Push-Value

Finally, we mention two links between our model and the *call-by-push-value* language of [15].

object language The model reflects the decomposition of $\rightarrow_{\mathbf{CBV}}$ into call-by-push-value given in [15].

metalanguage We want to use call-by-push-value as a metalanguage for the cpo equations of Sect. 6, in order to avoid having to construct the functor F in detail, and also to model storage combined with other effects [16].

We hope to treat these links in detail in future work. We also hope that working with call-by-push-value will help to establish connections with possible world models for call-by-name [17–20], especially Ghica's model for pointers [21].

Acknowledgements

Thanks to Peter O'Hearn for discussion and advice.

References

1. Ahmed, A., Appel, A., Virga, R.: A stratified semantics of general references embeddable in higher-order logic. In: Proceedings of IEEE Symposium on Logic in Computer Science, Copehagen, 2002. (2002) to appear.
2. Kelsey, R., Clinger, W., (Editors), J.R.: Revised^5 report on the algorithmic language Scheme. ACM SIGPLAN Notices **33** (1998) 26–76
3. Moggi, E.: An abstract view of programming languages. Technical Report ECS-LFCS-90-113, Dept. of Computer Science, Edinburgh Univ. (90)
4. Plotkin, G.D., Power, A.J.: Notions of computation determine monads. In: Proceedings of Foundations of Software Science and Computation Structures, Grenoble, France (FoSSaCS '02). LNCS (2002) to appear.
5. Stark, I.D.B.: Names and Higher-Order Functions. PhD thesis, University of Cambridge (1994)

6. Abramsky, S., Honda, K., McCusker, G.: A fully abstract game semantics for general references. Proceedings, Thirteenth Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press (1998)
7. Moggi, E.: Notions of computation and monads. Information and Computation **93** (1991) 55–92
8. Pitts, A.M.: Relational properties of domains. Information and Computation **127** (1996) 66–90 (A preliminary version of this work appeared as Cambridge Univ. Computer Laboratory Tech. Rept. No. 321, December 1993.).
9. Smyth, M., Plotkin, G.D.: The category-theoretic solution of recursive domain equations. SIAM J. Computing **11** (1982)
10. Abramsky, S., Jagadeesan, R., Malacaria, P.: Full abstraction for PCF (extended abstract). In Hagiya, M., Mitchell, J.C., eds.: Theoretical Aspects of Computer Software. International Symposium TACS'94. Volume 789 of LNCS., Sendai, Japan, Springer-Verlag (1994) 1–15
11. Stark, I.: A fully abstract domain model for the π-calculus. In: Proceedings of the Eleventh Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press (1996) 36–42
12. Freyd, P.J.: Algebraically complete categories. In Carboni, A., et al., eds.: Proc. 1990 Como Category Theory Conference, Berlin, Springer-Verlag (1991) 95–104 Lecture Notes in Mathematics Vol. 1488.
13. Power, A.J., Thielecke, H.: Closed Freyd- and kappa-categories. In: Proc. ICALP '99. Volume 1644 of LNCS., Springer-Verlag, Berlin (1999) 625–634
14. Jung, A.: Colimits in DCPO. 3-page manuscript, available by fax (1990)
15. Levy, P.B.: Call-by-push-value: a subsuming paradigm (extended abstract). In Girard, J.Y., ed.: Typed Lambda-Calculi and Applications. Volume 1581 of LNCS., Springer (1999) 228–242
16. Levy, P.B.: Call-by-push-value. PhD thesis, Queen Mary, University of London (2001)
17. Odersky, M.: A functional theory of local names. In ACM, ed.: Proceedings of 21st Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL), New York, NY, USA, ACM Press (1994) 48–59
18. O'Hearn, P.W., Tennent, R.D.: Semantics of local variables. In Fourman, M.P., Johnstone, P.T., Pitts, A.M., eds.: Applications of Categories in Computer Science. Proceedings of the LMS Symposium, Durham July 1991, Cambridge University Press (1992) 217–238
19. Oles, F.J.: A Category-Theoretic Approach to the Semantics of Programming Languages. Ph. D. dissertation, Syracuse University (1982)
20. Reynolds, J.C.: The essence of Algol. In de Bakker, J.W., van Vliet, J.C., eds.: Algorithmic Languages, Amsterdam, North-Holland (1981) 345–372
21. Ghica, D.R.: Semantics of dynamic variables in algol-like languages. Master's thesis, Queens' University, Kingston,Ontario (1997)