Recursively defined cpo algebras

Ohad Kammar and Paul Blain Levy

July 8, 2018

Outline

Bilimit compact categories

An axiomatization of a category of domains.

A Cpo-enriched category is a category $\mathcal{C}$ whose homsets are cpos, such that composition is continuous.

It is bilimit compact when

- each homset $\mathcal{C}(A, B)$ is pointed
- composition is bistrict
- $\mathcal{C}$ has a zero object (both initial and terminal)
- Every ω-chain $(A_n, e_n, p_n)_{n\in\mathbb{N}}$ in $\mathcal{C}^{\mathsf{ep}}$ has a bilimit, i.e. a cocone $V, (u_n, r_n)_{n\in\mathbb{N}}$ such that $\bigsqcup_{n\in\mathbb{N}} u_n \cdot r_n = \mathrm{id}_V$.

Examples

- The category $\mathbf{Cpo}^{\perp}$ of cppos and strict continuous maps.
- The opposite of a bilimit compact category.
- A product of bilimit compact categories.

Bifree algebras (Freyd)

Let $\mathcal{C}$ be a bilimit compact category.

Any locally continuous functor $H \colon \mathcal{C} \to \mathcal{C}$

has a bifree algebra (A, θ)

i.e. $\theta \colon HA \cong A$ is both an initial algebra and a final coalgebra.

In particular, the zero object is a bifree algebra of the identity functor.

The notion of bifree algebra extends to mixed variance functors.

Call-by-push-value

We want to apply this to the modelling of recursive types.

We'll use call-by-push-value, which subsumes call-by-value and call-by-name typed λ-calculus.

A value type A denotes a cpo. Call-by-value type.

A computation type $\underline{B}$ denotes a cppo. Call-by-name type.

Type syntax, including recursive types:

$$\begin{array}{lll} A, A' & ::= & U\underline{B} \mid 1 \mid A \times A' \mid 0 \mid A + A' \mid \sum_{i\in\mathbb{N}} A_i \mid \mathtt{X} \mid \mathtt{rec}\ \mathtt{X}.A \\ \underline{B}, \underline{B}' & ::= & FA \mid A \to \underline{B} \mid 1_\Pi \mid \underline{B}\,\Pi\,\underline{B}' \mid \prod_{i\in\mathbb{N}} \underline{B}_i \mid \underline{\mathtt{X}} \mid \mathtt{rec}\ \underline{\mathtt{X}}.\underline{B} \end{array}$$

Semantics of types:

$$[\![U\underline{B}]\!] = [\![\underline{B}]\!] \qquad\qquad [\![FA]\!] = [\![A]\!]_\bot$$

Recursive types

Recursive value type

$$\begin{array}{rcl} D & \stackrel{\text{def}}{=} & \mathtt{rec}\ \mathtt{X}.A \\ D & \cong & A[D/\mathtt{X}] \end{array}$$

Should denotes an isomorphism in $\mathbf{Cpo}$.

Recursive computation type $\underline{D} \stackrel{\text{def}}{=} \mathtt{rec}\ \underline{\mathtt{X}}.\underline{B}$

$$\begin{array}{rcl} \underline{D} & \stackrel{\text{def}}{=} & \mathtt{rec}\ \underline{\mathtt{X}}.\underline{B} \\ \underline{D} & \cong & \underline{B}[\underline{D}/\underline{\mathtt{X}}] \end{array}$$

Should denote an isomorphism in $\mathbf{Cpo}^{\perp}$.

Recursive types

Recursive value type

$$
\begin{array}{rcl}
D & \stackrel{\text{def}}{=} & \texttt{rec}\ \mathtt{X}.A \\
D & \cong & A[D/\mathtt{X}]
\end{array}
$$

Should denotes an isomorphism in $\mathbf{Cpo}$.

Recursive computation type $\underline{D} \stackrel{\text{def}}{=} \texttt{rec}\ \underline{\mathtt{X}}.\underline{B}$

$$
\begin{array}{rcl}
\underline{D} & \stackrel{\text{def}}{=} & \texttt{rec}\ \underline{\mathtt{X}}.\underline{B} \\
\underline{D} & \cong & \underline{B}[\underline{D}/\underline{\mathtt{X}}]
\end{array}
$$

Should denote an isomorphism in $\mathbf{Cpo}^{\perp}$.

But $\mathbf{Cpo}$ is not bilimit compact—it has no zero object.

Solution: expand the category

Let $\mathcal{B}$ be a Cpo-enriched category.
A bilimit compact expansion of $\mathcal{B}$ is a bilimit compact Cpo-enriched category $\mathcal{C}$ containing $\mathcal{B}$ as a subcategory such that

- $\mathcal{B}(A, B)$ is an admissible subset of $\mathcal{C}(A, B)$
- given
  - chains $(A_n, e_n, p_n)_{n \in \mathbb{N}}$ and (A'_n, e'_n, p'_n) in $\mathcal{C}^{\mathsf{ep}}$
  - bilimits $V, (u_n, r_n)_{n \in \mathbb{N}}$ and $V', (u'_n, r'_n)_{n \in \mathbb{N}}$
  - a map $\alpha_n \colon A_n \to A'_n$ commuting with e_n and with p_n

  the join of $e'_n \cdot \alpha_n \cdot p_n$ is in $\mathcal{B}(V, V')$.

Examples

- The category $\mathbf{pCpo}$ of cpos and partial continuous maps is a bilimit compact expansion of $\mathbf{Cpo}$.
- Preserved by $\mathcal{C} \mapsto \mathcal{C}^{\mathsf{op}}$.
- Preserved by product.

The recipe

We seek an fixpoint of a mixed variance functor H on $\mathcal{B}$.

Take a bilimit compact expansion $\mathcal{C}$ of $\mathcal{B}$.

Extend H to $\mathcal{C}$.

Obtain a fixpoint of H on $\mathcal{C}$.

Every isomorphism in $\mathcal{C}$ is an isomorphism in $\mathcal{B}$.

Kripke models

To model a language with dynamic generation (of names, references, etc.),

a value type denotes an object of $[\mathbb{I}, \mathbf{Cpo}]$

a computation type denotes an object of $[\mathbb{I}, \mathbf{Cpo}^{\perp}]$,

Theorem

- $[\mathbb{I}, \mathcal{C}]$ is bilimit compact if $\mathcal{C}$ is.
- Let $\mathcal{B}$ have bilimit compact expansion $\mathcal{C}$. Then $[\mathbb{I}, \mathcal{B}]$ has bilimit compact expansion, as follows: a map $A \to B$ is a map in $[\mathbb{I}, \mathcal{C}]$.

Semantics of a computation type $\underline{B}$

1. If programs either terminate or diverge, $[\![\underline{B}]\!]$ is a cppo.

Semantics of a computation type $\underline{B}$

1. If programs either terminate or diverge, $[\![\underline{B}]\!]$ is a cppo.
2. Suppose programs can crash. Then $[\![\underline{B}]\!]$ is a cppo A with a "crash" element.

Semantics of a computation type $\underline{B}$

1. If programs either terminate or diverge, $[\![\underline{B}]\!]$ is a cppo.
2. Suppose programs can crash. Then $[\![\underline{B}]\!]$ is a cppo A with a "crash" element.
3. Suppose programs can peform I/O, described by a functor $H\colon \mathbf{Cpo}^{\perp} \to \mathbf{Cpo}$. Then $[\![\underline{B}]\!]$ is a cppo A with a map $HA \to A$.

Semantics of a computation type $\underline{B}$

1. If programs either terminate or diverge, $[\![\underline{B}]\!]$ is a cppo.
2. Suppose programs can crash. Then $[\![\underline{B}]\!]$ is a cppo A with a "crash" element.
3. Suppose programs can peform I/O, described by a functor $H \colon \mathbf{Cpo}^{\perp} \to \mathbf{Cpo}$. Then $[\![\underline{B}]\!]$ is a cppo A with a map $HA \to A$.
4. Suppose programs can lookup and update memory, and S is the set of states. Then $[\![\underline{B}]\!]$ is a cppo A with maps

$$\begin{array}{lccc} \text{lookup} & : & A^S & \to & A \\ \text{update} & : & S \times A & \to & A \end{array}$$

satisfying some equations.

We need to form

- a recursive crash-cppo
- a recursive cppo-H-algebra
- a recursive cppo lookup/update algebra.

But the categories of crash-cppos, cppo-H-algebras, and cppo lookup/update algebras are not bilimit compact, as they have no zero object.

Recursive computation types with effects

We need to form

- a recursive crash-cppo
- a recursive cppo-H-algebra
- a recursive cppo lookup/update algebra.

But the categories of crash-cppos, cppo-H-algebras, and cppo lookup/update algebras are not bilimit compact, as they have no zero object.

Solution: expand the categories.

Crash cppos

The category of crash cppos and strict homomorphisms lacks a zero object.

Crash cppos

The category of crash cppos and strict homomorphisms lacks a zero object.

The expanded category: a morphism $f\colon (A, c) \to (B, d)$ is a lax homomorphism, a strict map such that $f(c) \leqslant d$.

H-algebras

The category of cppo-H-algebras and strict homomorphisms lacks a zero object.

H-algebras

The category of cppo-H-algebras and strict homomorphisms lacks a zero object.

The expanded category: a morphism $f\colon (A, c) \to (B, d)$ is a lax homomorphism, a strict map such that

$$\begin{tikzcd} HA \arrow[r, "Hf"] \arrow[d, "c"'] \arrow[dr, phantom, "\leqslant"] & HB \arrow[d, "d"] \\ A \arrow[r, "f"'] & B \end{tikzcd}$$

Bilimit compactness was proved by Fiore.

Summary

- Computation types denote cppo-algebras.
- The category of cppo-algebras is not bilimit compact.
- We interpret recursive computation types using the bilimit compact expansion in which a morphism is a lax homomorphism.
- The appropriate functors (e.g. $\rightarrow$) can be extended to this category.
- Caveat We conjecture this model is computationally adequate, but proving it is work in progress.