Recursively defined cpo algebras

Ohad Kammar and Paul Blain Levy

July 13, 2018
Outline

1. Bilimit compact categories
2. Modelling recursive types
3. Recursively defined cppo algebras
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 $\omega$-chain $(A_n, e_n, p_n)_{n \in \mathbb{N}}$ in $\mathcal{C}^{\text{ep}}$ has a bilimit, i.e. a cocone $V, (u_n, r_n)_{n \in \mathbb{N}}$ such that $\bigsqcup_{n \in \mathbb{N}} u_n \cdot r_n = \text{id}_V$. 

Ohad Kammar and Paul Blain Levy

Recursively defined cpo algebras

July 13, 2018
Examples

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

Any locally continuous functor $H : C \to C$

has a bifree algebra $(A, \theta)$

i.e. $\theta : HA \cong A$ 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.
We want to apply this to the modeling 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** $B$ denotes a cppo. **Call-by-name type.**

**Type syntax, including recursive types:**

$$A, A' ::= UB \mid 1 \mid A \times A' \mid 0 \mid A + A' \mid \sum_{i \in \mathbb{N}} A_i \mid X \mid \text{rec } X.A$$

$$B, B' ::= FA \mid A \to B \mid 1\Pi \mid B \Pi B' \mid \Pi_{i \in \mathbb{N}} B_i \mid X \mid \text{rec } X.B$$

**Semantics of types:**

$$[UB] = [B] \quad [FA] = [A] \perp$$
Recursive types

Recursive value type

\[ D \overset{\text{def}}{=} \text{rec } X.A \]
\[ D \cong A[D/X] \]

Should denotes an isomorphism in \( \mathbf{Cpo} \).

Recursive computation type \[ D \overset{\text{def}}{=} \text{rec } X.B \]

\[ D \overset{\text{def}}{=} \text{rec } X.B \]
\[ D \cong B[D/X] \]

Should denote an isomorphism in \( \mathbf{Cpo}^{\perp} \).
Recursive types

Recursive value type

\[ D \overset{\text{def}}{=} \text{rec } X.A \]
\[ D \simeq A[D/X] \]

Should denotes an isomorphism in \( \text{Cpo} \).

Recursive computation type \( D \overset{\text{def}}{=} \text{rec } X.B \)

\[ D \overset{\text{def}}{=} \text{rec } X.B \]
\[ D \simeq B[D/X] \]

Should denote an isomorphism in \( \text{Cpo}^\perp \).

But \( \text{Cpo} \) is not bilimit compact—it has no zero object.
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 $\mathcal{C}^{ep}$
  - bilimits $V, (u_n, r_n)_{n \in \mathbb{N}}$ and $V', (u'_n, r'_n)_{n \in \mathbb{N}}$
  - a map $\alpha_n : A_n \to A'_n$ commuting with $e_n$ and with $p_n$

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

Ohad Kammar and Paul Blain Levy
The category $\mathbf{pCpo}$ of cpos and partial continuous maps is a bilimit compact expansion of $\mathbf{Cpo}$.

Preserved by $C \mapsto C^{op}$.

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

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

Extend $H$ to $C$.

Obtain a fixpoint of $H$ on $C$.

Every isomorphism in $C$ 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 $[\mathbb{I}, \text{Cpo}]$ and a computation type denotes an object of $[\mathbb{I}, \text{Cpo}^\perp]$.

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 $[\mathbb{I}, \mathcal{C}]$. 
If programs either terminate or diverge, $[B]$ is a cppo.
1. If programs either terminate or diverge, $[B]$ is a cppo.
2. Suppose programs can crash. Then $[B]$ is a cppo $A$ with a “crash” element.
Semantics of a computation type $\mathcal{B}$

1. If programs either terminate or diverge, $\llbracket \mathcal{B} \rrbracket$ is a cppo.

2. Suppose programs can crash. Then $\llbracket \mathcal{B} \rrbracket$ is a cppo $A$ with a “crash” element.

3. Suppose programs can perform I/O, described by a functor $H : \mathbf{Cpo}^{\perp} \rightarrow \mathbf{Cpo}$. Then $\llbracket \mathcal{B} \rrbracket$ is a cppo $A$ with a map $HA \rightarrow A$. 
Semantics of a computation type \( \mathcal{B} \)

1. If programs either terminate or diverge, \( \llbracket \mathcal{B} \rrbracket \) is a cppo.

2. Suppose programs can crash. Then \( \llbracket \mathcal{B} \rrbracket \) is a cppo \( A \) with a “crash” element.

3. Suppose programs can perform I/O, described by a functor \( H : \text{Cpo}^\perp \to \text{Cpo} \). Then \( \llbracket \mathcal{B} \rrbracket \) 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 \( \llbracket \mathcal{B} \rrbracket \) is a cppo \( A \) with maps:

\[
\begin{align*}
\text{lookup} & : \quad A^S \to A \\
\text{update} & : \quad S \times A \to A
\end{align*}
\]

satisfying some equations.
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.
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.
The category of crash cpos and strict homomorphisms lacks a zero object.
The category of crash cpos and strict homomorphisms lacks a zero object.

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

\[
\begin{array}{c}
HA \xrightarrow{Hf} HB \\
\downarrow c & \leq & \downarrow d \\
A \xrightarrow{f} B
\end{array}
\]

Bilimit compactness (for the Eilenberg-Moore version) was proved by Fiore.
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.