Universality Results for Models in Locally Boolean Domains

Tobias Löw and Thomas Streicher

TU Darmstadt, Schloßgartenstraße 7, D-64289 Darmstadt

Abstract. In [6] J. Laird has shown that an infinitary sequential extension of PCF has a fully abstract model in his category of locally boolean domains (introduced in [8]). In this paper we introduce an extension SPCF_∞ of his language by recursive types and show that it is universal for its model in locally boolean domains.
Finally we consider an infinitary target language CPS_∞ for (the) CPS translation (of [16]) and show that it is universal for a model in locally boolean domains which is constructed like Dana Scott's D_∞ where $D = \{\bot, \top\}$.

1 Introduction

In [4] Cartwright, Curien and Felleisen have shown that for SPCF, an extension of PCF with error elements and a **catch** construct, one can construct extensional fully abstract models whose induced theory in the finitary case (i.e. over base type boolean) is still decidable and thus much simpler than the fully abstract models for PCF (see [1, 5, 13]) as demonstrated by Loader's result [9]. The model of [4] consists of error-propagating sequential algorithms between concrete data structures (with errors). About a decade later in [8] J. Laird has arrived at a reformulation of this model in terms of a category **LBD** of *locally boolean domains* (lbds) and *sequential maps* between them.

In the current paper we show that in **LBD** one can interpret an infinitary variant SPCF_∞ of the language SPCF of [4]. Roughly speaking, the language SPCF_∞ is an extension of simply typed λ-calculus by countable sums and products, error elements $\top$ for each type, a control construct **catch** and recursive types. For SPCF_∞ without recursive types it has been shown in [6] that the **LBD** model is fully abstract, i.e. that all finite elements arise as denotations of programs. We show that actually *all* elements of (possibly recursive) SPCF_∞ types can be denotes by SPCF_∞ terms, i.e. that SPCF_∞ is universal for its **LBD** model. In the proof we first show that every SPCF_∞ type can be obtained as an SPCF_∞ definable retract of the first order type $\mathbf{U} = \mathbf{N} {\to} \mathbf{N}$ (adapting an analogous result in [10] for ordinary sequential algorithms without error elements) and then conclude by observing that every element of $\mathbf{U}$ is (trivially) SPCF_∞ definable.

In [16] it has been observed that $\mathsf{0}_\infty$, i.e. Scott's D_∞ with $D = \mathsf{0} = \{\bot, \top\}$, can be obtained as bifree solution of $D = [D^\omega {\to} \mathsf{0}]$. Since solutions of recursive type equations are available in **LBD** (see section 2) we may consider also the bifree solution of the equation for D in **LBD**. Canonically associated with this type equation is the language CPS_∞ whose terms are given by the grammar

$$M ::= x \mid \lambda\vec{x}.M\langle\vec{M}\rangle \mid \lambda\vec{x}.\top$$

where $\vec{x}$ ranges over infinite lists of pairwise disjoint variables and $\vec{M}$ over infinite lists of terms. Notice that CPS_∞ is more expressive than (untyped) λ-calculus with an error element $\top$ in the respect that one may apply a term to an infinite list of arguments. Consider e.g. the term $\lambda\vec{x}.x_0\langle\vec{\bot}\rangle$ whose interpretation retracts D to $\mathsf{0}$ (i.e. sends $\top$ to $\top$ and everything else to $\bot$) whereas this retraction is not expressible in λ-calculus with a constant $\top$. We show that CPS_∞ is universal for its model in D. For this purpose we proceed as follows.

We first observe that the finite elements of D all arise from simply typed λ-calculus over $\mathsf{0}$. Since the latter is universal for its **LBD** model (as shown in [6]) and all retractions of D to finite types are CPS_∞ definable it follows that all finite elements of D are definable in CPS_∞. Then borrowing an idea from [7] we show that the supremum of stably bounded elements of D is CPS_∞ definable. Using this we show that the supremum of every chain of finite elements increasing w.r.t. $\leq_s$ is CPS_∞ and thus every element of D is CPS_∞ definable as well.

Although interpretation of CPS_∞ in D is surjective it happens that interpretation in D may identify terms with different infinite normal form, i.e. the interpretation is not faithful. Finally, we discuss a way how this shortcoming can be avoided, namely to extend CPS_∞ with a parallel construct $\parallel$ and refining the observation type $\mathsf{0}$ to $\widetilde{\mathsf{0}} \cong \mathsf{List}(\widetilde{\mathsf{0}})$. This amounts to a "qualitative" reformulation of a "quantitative" method introduced by F. Maurel in his Thesis [12].

2 Locally Boolean Domains

This section contains a short introduction to the theory of lbds and sequential maps (cf. [8]).

Definition 1. *A* locally boolean order (lbo) *is a triple $A = (|A|, \sqsubseteq, \neg)$ where $(A, \sqsubseteq)$ is a partial order and $\neg : |A| \to |A|$ is antitonic and an involution (i.e. $x \sqsubseteq y \Rightarrow \neg y \sqsubseteq \neg x$ and $\neg\neg x = x$ for all $x, y \in |A|$) such that*

(1) *for every $x \in A$ the set $\{x, \neg x\}$ has a least upper bound $x^\top = x \sqcup \neg x$ (and, therefore, also a greatest lower bound $x_\bot = \neg(x^\top) = x \sqcap \neg x$)*
(2) *whenever $x \sqsubseteq y^\top$ and $y \sqsubseteq x^\top$ (notation $x \uparrow y$) then $\{x, y\}$ has a supremum $x \sqcup y$ and an infimum $x \sqcap y$.*

A is complete *if $(|A|, \sqsubseteq)$ is a cpo, i.e. every directed subset X has a supremum $\bigsqcup X$. A is* pointed *if it has a least element $\bot$ (and thus also a greatest element $\top = \neg\bot$).* $\diamond$

We write $x \downarrow y$ as an abbreviation for $\neg x \uparrow \neg y$, and $x \updownarrow y$ for $x \uparrow y$ and $x \downarrow y$. Notice that $x \updownarrow y$ iff $x_\bot = y_\bot$ iff $x^\top = y^\top$. A subset $X \subseteq A$ is called *stably coherent* (notation $\uparrow X$) iff $x \uparrow y$ for all $x, y \in X$. Analogously, X is called *costably coherent* (notation $\downarrow X$) iff $x \downarrow y$ for all $x, y \in X$. Finally, X is called *bistably coherent* (notation $\updownarrow X$) iff $\uparrow X$ and $\downarrow X$.

Definition 2. *For a lbo A and $x, y \in A$ we define*

stable order: $x \leq_s y$ *iff* $x \sqsubseteq y$ *and* $x \uparrow y$
costable order: $x \leq_c y$ *iff* $x \sqsubseteq y$ *and* $x \downarrow y$ (*iff* $\neg y \leq_s \neg x$)
bistable order: $x \leq_b y$ *iff* $x \leq_s y$ *and* $x \leq_c y$ $\diamond$

For the definition of locally boolean domains we introduce the notion of *finite* and *prime* elements.

Definition 3. *Let A be a lbo.*
An element $p \in |A|$ is called prime *iff*

$$\forall x, y{\in}A.\, ((x \uparrow y \vee x \downarrow y) \wedge p \sqsubseteq x \sqcup y) \rightarrow (p \sqsubseteq x \vee p \sqsubseteq y)$$

We write $P(A)$ for the set $\{p \in |A| \mid p \text{ is prime}\}$ and $P(x)$ for the set $\{p \in P(A) \mid p \leq_s x\}$.
An element $e \in |A|$ is called finite *iff the set $\{x \in A \mid x \leq_s e\}$ is finite. We put*

$$F(A) := \{e \in |A| \mid e \text{ is finite}\} \quad \text{and} \quad F(x) := \{e \in F(A) \mid e \leq_s x\}\,.$$

For handling finite primes, *i.e. elements that are finite and prime, we define $FP(A) := P(A) \cap F(A)$ and $FP(x) := P(x) \cap F(A)$.* $\diamond$

Definition 4. *A* locally boolean domain (lbd) *is a pointed, complete lbo A such that for all $x \in A$*

(1) $x = \bigsqcup FP(x)$ *and*
(2) *all finite primes in A are compact w.r.t. $\sqsubseteq$, i.e. for all $p \in FP(A)$ and directed sets X with $p \sqsubseteq \bigsqcup X$ there is an $x \in X$ with $p \sqsubseteq x$.* $\diamond$

One can show that stably coherent subsets X of a lbd A have a supremum $\bigsqcup X$ which is a supremum also w.r.t. $\leq_s$. Moreover, if X is also nonempty then X has an infimum $\bigsqcap X$ which is an infimum also w.r.t. $\leq_s$. For costably coherent subsets the dual claims hold. Further, we have the following property of maximal bistably coherent subsets.

Lemma 5. *Let A be a lbd and $x \in A$. Then $[x]_\updownarrow := \{y \in A \,|\, y \updownarrow x\}$ with $\sqcap$, $\sqcup$ and $\neg$ restricted to $[x]_\updownarrow$ forms a complete atomic boolean algebra.*

The following lemma is needed for showing that our definition of locally boolean domain is equivalent with the original one given by J. Laird in [8].

Lemma 6. *Let x and y be elements of a lbd A then the following are equivalent*

(1) $x \sqsubseteq y$
(2) $\forall p{\in}FP(x).\exists q{\in}FP(y).\, p \sqsubseteq q$
(3) $\forall c{\in}F(x).\exists d{\in}F(y).\, c \sqsubseteq d$

Thus A is a coherently complete dI*-domain* (*cf.* [2]) *w.r.t. the stable order $\leq_s$.*

Next we define an appropriate notion of sequential map between lbds.

Definition 7. *Let A and B be lbds. A* sequential *map from A to B is a Scott continuous function $f : (|A|, \sqsubseteq) \to (|B|, \sqsubseteq)$ such that for all $x \updownarrow y$ it holds that $f(x) \updownarrow f(y)$, $f(x \sqcap y) = f(x) \sqcap f(y)$ and $f(x \sqcup y) = f(x) \sqcup f(y)$.* $\diamond$

We denote the ensuing category of lbds and sequential maps by **LBD**. The category **LBD** is cpo-enriched w.r.t. $\sqsubseteq$ and $\leq_s$ and order extensional w.r.t. $\sqsubseteq$, i.e. in particular well-pointed. In [8] J. Laird has shown that the category **LBD** is equivalent to the category **OSA** of *observably sequential algorithms* which has been introduced in [4] where it was shown that it gives rise to a fully abstract model for the language SPCF, an extension of PCF by error elements and a control operator **catch**.

The category **LBD** enjoys all the properties required for interpreting the language SPCF_∞ introduced subsequently in Section 3, namely that **LBD** is cartesian closed, has countable products and bilifted sums and inverse limits of ω-chains of projections. We just give the construction of these lbds, for a detailed verification of their characterising properties see [11].

Cartesian Products. For each family of lbds $(A_i)_{i\in I}$ the cartesian product $\prod_{i\in I} A_i$ is constructed as follows: $(\prod_{i\in I} |A_i|, \sqsubseteq, \neg)$ with $\sqsubseteq$ and $\neg$ defined pointwise.

Exponentials. For lbds A, B the function space $[A{\to}B]$ is constructed as follows: $|[A{\to}B]| = \mathbf{LBD}(A, B)$, the extensional order is defined pointwise and negation is given by $(\neg f)(x) := \bigsqcup\{\neg f(\neg c) \mid c \in F(x)\}$.

Terminal Object. The object 1 is given by $(\{*\}, \sqsubseteq, \neg)$.

Bilifted Sum. For each family of lbds $(A_i)_{i\in I}$ the bilifted sum $\sum_{i\in I} A_i$ is constructed as follows: $(\bigcup_{i\in I}\{i\} \times |A_i| \cup \{\bot, \top\}, \sqsubseteq, \neg)$ with

$$x \sqsubseteq y \Leftrightarrow x{=}\bot \vee y{=}\top \vee (\exists i{\in}I.\exists x', y'{\in}A_i.\, x{=}(i, x') \wedge y{=}(i, y') \wedge x'{\sqsubseteq_i}y')$$

and negation given by $\neg\bot = \top$ and $\neg(i, x) = (i, \neg_i x)$.

Natural Numbers. The data type $\mathsf{N} = \sum_{i\in\omega} 1$ will serve as the type of bilifted natural numbers. More explicitly N can be described as the lbd $(\mathbb{N}{\cup}\{\bot, \top\}, \sqsubseteq, \neg)$ with $x \sqsubseteq y$ iff $x = \bot$ or $y = \top$ or $x = y$, and negation is given by $\neg\bot = \top$ and $\neg n = n$ for all $n \in \mathbb{N}$.

Type of Observations. The type of observations $\mathsf{0} = \sum_{i\in\emptyset}$. More explicitly $\mathsf{0}$ can be described as the lbd $(\{\bot, \top\}, \sqsubseteq, \neg)$ with $\bot \sqsubseteq \top$ and $\neg\bot = \top$. Notice that $[A{\to}\mathsf{0}]$ separates points in A for any lbd A.

The exponential transpose of functions is defined as usual and since evaluation is sequential it follows that the category **LBD** is cartesian closed.

Notice that for exponentials we cannot simply define negation of a sequential map by $(\neg f)(x) = \neg f(\neg x)$ as the following example shows that sequentiality does not imply cocontinuity w.r.t. $\leq_c$.

Example 8. Let $F : [\mathsf{N} \to \mathsf{0}^{\mathsf{0}}] \to \mathsf{0}$ be defined recursively as

$$F(f) = f(0)(F(\lambda n.f(n{+}1)))\,.$$

Let $f = \lambda n.\mathrm{id}_{\mathsf{0}}$ and $f_n(k) = \mathrm{id}_{\mathsf{0}}$ for $k < n$ and $f(k) = \lambda n.\top$ for $k \geq n$. Obviously, the set $X := \{f_n \mid n \in \mathbb{N}\}$ is costably coherent, codirected w.r.t. $\leq_c$ and $f = \bigsqcap X$. As f is a minimal solution of its defining equation we have $F(f) = \bot$ and $F(f_n) = \top$ for all n. Thus, we have $f(\bigsqcap X) = \bot$ whereas $\bigsqcap f[X] = \top$, i.e. F fails to be cocontinuous w.r.t. $\leq_c$.

Nevertheless we always have

Lemma 9. *Let $f : A \to B$ be a* **LBD** *morphism and $x \in A$. Then $(\neg f)(x) \sqsubseteq \neg f(\neg x)$*

Proof. For all $c \in F(x)$ we have $\neg x \sqsubseteq \neg c$, thus, $f(\neg x) \sqsubseteq f(\neg c)$ and $\neg f(\neg c) \sqsubseteq \neg f(\neg x)$. Hence, it follows that $(\neg f)(x) = \bigsqcup\{\neg f(\neg c) \mid c \in F(x)\} \sqsubseteq \neg f(\neg x)$.

For the construction of recursive types in **LBD** we have to introduce an appropriate notion of embedding/projection pairs for lbds.

Definition 10. *An* embedding/projection pair (ep-pair) *from X to Y in* **LBD** *(notation $(\iota, \pi) : X \to Y$) is a pair of* **LBD** *morphisms $\iota : X \to Y$ and $\pi : Y \to X$ with $\pi \circ \iota = \mathrm{id}_X$ and $\iota \circ \pi \leq_s \mathrm{id}_Y$.*

If $(\iota, \pi) : X \to Y$ and $(\iota', \pi') : Y \to Z$ then their composition is defined as $(\iota', \pi') \circ (\iota, \pi) = (\iota' \circ \iota, \pi \circ \pi')$. We write $\mathbf{LBD}^E$ *for the ensuing category of embedding/projection pairs in* **LBD**. ◇

Notice that this is the usual definition of ep-pair when viewing **LBD** as order enriched by the stable and not by the extensional order.

Next we describe the construction of inverse limits of ω-chains of ep-pairs in **LBD**. The underlying cpos are constructed as usual. However, it needs some care to define negation appropriately (since in general projections do not preserve negation).

Theorem 11. *Given a functor $A : \omega \to \mathbf{LBD}^E$ its inverse limit of the projections is given by $(A_\infty, \sqsubseteq, \neg)$ where*

$$A_\infty = \{x \in \prod_{n \in \omega} A_n \mid x_n = \pi_{n,n+1}(x_{n+1}) \text{ for all } n \in \omega\}$$

the extensional order $\sqsubseteq$ is defined pointwise and

$$(\neg x)_n = \bigsqcap\nolimits_{k \geq n} \pi_{n,k}(\neg x_k)$$

for all $n \in \omega$.

Notice that the full subcategory of *countably based* lbds, i.e. lbds A where $FP(A)$ is countable, is closed under the above constructions as long as products and bilifted sums are assumed as countable.

3 The Language SPCF_∞

The language SPCF_∞ is an infinitary version of SPCF as introduced in [4]. More explicitly, it is obtained from simply typed λ-calculus by adding (countably) infinite sums and products, error elements, a control operator **catch** and recursive types. For a detailed presentation of SPCF_∞ see Table 1.

The operational semantics of SPCF_∞ is given in Table 2. Notice that each SPCF_∞ term t which is not already a value has a unique decomposition into an evaluation context E and a redex t' with $E[t'] \equiv t$.

The interpretation of SPCF_∞ in locally boolean domains can be found in Table 3. The interpretation of recursive types is done as usual via inverse limits whose existence is guaranteed by Theorem 11. One can prove adequacy of the model like in [14, 15].

Since by definition sequential maps preserve infima and suprema of bistably coherent arguments a sequential map from $\mathsf{0}^\omega$ to $\mathsf{0}$ is either constant (with value $\perp$ or $\top$) or is a projection $\pi_i : \mathsf{0}^\omega \to \mathsf{0}$. For this reason there exists an isomorphism $\mathsf{catch} : [\mathsf{0}^\omega {\to} \mathsf{0}] \overset{\cong}{\to} \mathsf{N}$ with

$$\mathsf{catch}(f) = i \quad \text{iff} \quad f \text{ is } i\text{-strict, i.e. } f(x) = \perp \Leftrightarrow \pi_i(x) = \perp$$

which will serve as interpretation of the control operator **catch** of SPCF_∞.

4 Universality for SPCF_∞

In this section we show that the first order type $\mathbf{U} = \mathbf{N} {\to} \mathbf{N}$ is universal for the language SPCF_∞ by proving that every type is a SPCF_∞ definable retract of $\mathbf{U}$. Since all elements of the lbd $[\![\mathbf{U}]\!]$ can be defined syntactically we get universality of SPCF_∞ for its model in **LBD**.

Definition 12. *A closed* SPCF_∞ *type* σ *is called a* SPCF_∞ definable retract *of a* SPCF_∞ *type* τ *(denoted* $\sigma \lhd \tau$*) iff there exist closed terms* $e : \sigma {\to} \tau$ *and* $p : \tau {\to} \sigma$ *with* $[\![p]\!] \circ [\![e]\!] = \mathrm{id}_{[\![\sigma]\!]}$. ◇

Theorem 13. *Every* SPCF_∞ *type appears as* SPCF_∞ *definable retract of the type* $\mathbf{U} := \mathbf{N} {\to} \mathbf{N}$.

Proof. It suffices to show that for all $n \in \omega + 1$ the types

$$\mathbf{U} {\to} \mathbf{U} \qquad \Pi_{i \in n} \mathbf{U} \qquad \Sigma_{i \in n} \mathbf{U}$$

are SPCF_∞ definable retracts of $\mathbf{U}$.

The SPCF_∞ programs exhibiting $\Pi_{i \in n} \mathbf{U}$ as definable retract of $\mathbf{U}$ are given in Table 5. Using this we get $\Sigma_{i \in n} \mathbf{U} \lhd \mathbf{U}$ since obviously $\Sigma_{i \in n} \mathbf{U} \lhd \mathbf{U} {\times} \Sigma_{i \in n} \mathbb{1}$.

By currying we have $\mathbf{U} {\to} \mathbf{U} \cong (\mathbf{U} {\times} \mathbf{N}) {\to} \mathbf{N}$. As $\mathbf{U} {\times} \mathbf{N} \lhd \mathbf{U} {\times} \mathbf{U} \lhd \mathbf{U}$ it suffices to construct a retraction $\mathbf{U} {\to} \mathbf{N} \lhd \mathbf{U}$ for showing that $\mathbf{U} {\to} \mathbf{U} \lhd \mathbf{U}$ holds. For this purpose we adapt an analogous result given by J. Longley in [10] for ordinary sequential algorithms without error elements. The programs establishing the retraction are given in Table 6. The function p interprets elements of $\mathbf{U}$ as sequential algorithms for functionals of type $\mathbf{U}{\to}\mathbf{N}$ as described in [10]. For a given $F : \mathsf{U}{\to}\mathsf{N}$ the element $[\![e]\!](F) : \mathsf{N}{\to}\mathsf{N}$ is a strategy / sequential algorithm for computing F. This is achieved by computing sequentiality indices iteratively using **catch**. □

Since all sequential function from N to N can be programmed using a (countably infinite) case analysis (available by the **case**-construct for index set ω) it follows that

Theorem 14. *The language* SPCF_∞ *is universal for its model in* **LBD**.

5 Universality for an infinitary untyped CPS target language CPS_∞

The interpretation of the SPCF_∞ type $\delta := \mu\alpha.(\alpha^\omega{\to}\mathbf{0})$ (where for arbitrary types σ we henceforth write σ^ω as an abbreviation for $\Pi_{i\in\omega}\sigma$) is the minimal solution of the domain equation $D \cong [D^\omega{\to}\mathsf{0}]$. Obviously, we have $D \cong [D{\to}D]$. Moreover, it has been shown in [16] that D is isomorphic to $\mathsf{0}_\infty$, i.e. what one obtains by performing D. Scott's D_∞ construction in **LBD** when instantiating D by $\mathsf{0}$.

We now describe an untyped infinitary language CPS_∞ canonically associated with the domain equation $D \cong [D^\omega{\to}\mathsf{0}]$. The precise syntax of CPS_∞ is given in Table 4. We interpret CPS_∞ terms in context Γ, i.e. a set of variables, as sequential maps from D^Γ to D in the obvious way.

The language CPS_∞ is an extension of pure untyped λ-calculus since applications MN can be expressed by $\lambda\vec{x}.M\langle N,\vec{x}\rangle$ with fresh variables $\vec{x}$ and abstraction $\lambda x.M$ by $\lambda x\vec{y}.M\langle\vec{y}\rangle$ with fresh variables $\vec{y}$. Thus, CPS_∞ allows for recursion and we can define recursion combinators in the usual way.

Notice that CPS_∞ is more expressive than pure untyped λ-calculus since the latter does not contain a term semantically equivalent to

$$\lambda\vec{x}.x_0\langle\vec{\bot}\rangle$$

which sends $\top_D$ to $\top_D$ and all other elements of D to $\bot_D$. Since the retraction of D to $\mathsf{0}$ is CPS_∞ definable all other retractions to the finite approximations of D (which are isomorphic to simple types over $\mathsf{0}$) are definable as well.

Lemma 15. *The lbds* N *and* U *are both* CPS_∞ *definable retract of the lbd* D.

Proof. Since we can retract the lbd D to the lbd $\mathsf{0}$ and $[\mathsf{0}^\omega{\to}\mathsf{0}] \cong \mathsf{N}$ it follows that N is a CPS_∞ definable retract of D. As $[D{\to}D]$ is a CPS_∞ definable retract of D it follows that $\mathsf{U} = [\mathsf{N}{\to}\mathsf{N}]$ is a CPS_∞ definable retract of D. □

Thus, we can do arithmetic within CPS_∞. Natural numbers are encoded by $\underline{n} \equiv \lambda\vec{x}.x_n\langle\vec{\bot}\rangle$ and a function $f{:}\mathbb{N}{\to}\mathbb{N}$ by its graph, i.e. $\underline{f} \equiv \lambda x\vec{y}.x\langle\lambda\vec{z}.\underline{f(i)}\langle\vec{y}\rangle\rangle_{i\in\omega}$. Notice that CPS_∞ allows for the implementation of an infinite case construct.

Universality for CPS_∞ will be shown in two steps. First we argue why all finite elements of D are CPS_∞ definable. Then adapting a trick from [7] we show that suprema of chains increasing w.r.t. $\leq_s$ are CPS_∞definable, too.

Lemma 16. *All finite elements of the lbd D are CPS_∞ definable.*

Proof. In [6] Jim Laird has shown that the language $\Lambda_\bot^\top$, i.e. simply typed λ-calculus over the base type $\{\bot,\top\}$ is universal for its model in **LBD**. Thus, since all retractions of D to its finitary approximations are CPS_∞ definable it follows that all finite elements of D are CPS_∞ definable. $\square$

Next we show that for all $f : A \to \mathsf{0}$ in **LBD** the map $\widetilde{f} : A \to [\mathsf{0}{\to}\mathsf{0}]$ with

$$\widetilde{f}(a)(u) := \begin{cases} u & \text{if } f(a^\top) = \bot_\mathsf{0} \text{ and} \\ f(a) & \text{otherwise} \end{cases} \tag{1}$$

is an **LBD** morphism as well.

Lemma 17. *If $f : A \to \mathsf{0}$ is a sequential map between lbds then the function $\widetilde{f} : A \to [\mathsf{0}{\to}\mathsf{0}]$ given by* (1) *is sequential.*

Proof. For showing monotonicity suppose $a_1, a_2 \in A$ with $a_1 \sqsubseteq a_2$ and $u \in \mathsf{0}$. We proceed by case analysis on $f(a_1^\top)$.
Suppose $f(a_1^\top) = \bot_\mathsf{0}$. Thus, $\widetilde{f}(a_1)(u) = u$. If $f(a_2^\top) = \bot_\mathsf{0}$ then $\widetilde{f}(a_2)(u) = u$, and we get $\widetilde{f}(a_1)(u) = u = \widetilde{f}(a_2)(u)$. If $f(a_2^\top) = \top_\mathsf{0}$ then $\widetilde{f}(a_2)(u) = f(a_2)$. As $f(a_1^\top) = \bot_\mathsf{0}$ it follows that $f(\neg a_1) = \bot_\mathsf{0}$ and $f(\neg a_2) = \bot_\mathsf{0}$ (because $\neg a_2 \sqsubseteq \neg a_1$). As $\top_\mathsf{0} = f(a_2^\top) = f(a_2) \sqcup f(\neg a_2)$ it follows that $f(a_2) = \top_\mathsf{0}$ as desired.
If $f(a_1^\top) = \top_\mathsf{0}$ then $\widetilde{f}(a_1)(u) = f(a_1)$. W.l.o.g. assume $f(a_1) = \top_\mathsf{0}$. Then $\top_\mathsf{0} = f(a_1) \sqsubseteq f(a_2) \sqsubseteq f(a_2^\top)$. Hence, $f(a_2) = \top_\mathsf{0} = f(a_2^\top)$ and we get $\widetilde{f}(a_2)(u) = f(a_2) = \top_\mathsf{0}$.
Next we show that $\widetilde{f}$ is bistable. Let $a_1 \updownarrow a_2$, thus (†) $a_1^\top = a_2^\top = (a_1 \sqcap a_2)^\top$. If $f(a_1^\top) = f(a_2^\top) = \bot_\mathsf{0}$ then $\widetilde{f}(a_1) = \mathrm{id}_\mathsf{0} = \widetilde{f}(a_2)$. If $f(a_1^\top) = f(a_2^\top) = \top_\mathsf{0}$ then $\widetilde{f}(a_i) = \lambda x{:}\mathsf{0}.\, f(a_i)$ for $i \in \{1,2\}$. Since $\lambda x{:}\mathsf{0}.\, \bot_\mathsf{0} \updownarrow \lambda x{:}\mathsf{0}.\, \top_\mathsf{0}$ it follows that $\widetilde{f}$ preserves bistable coherence.
Finally we show that $\widetilde{f}$ preserves bistably coherent suprema. If $f((a_1 \sqcap a_2)^\top) = \bot_\mathsf{0}$ then $\widetilde{f}(a_1 \sqcap a_2)(u) = u = \widetilde{f}(a_1)(u) \sqcap \widetilde{f}(a_2)(u)$ (since $f(a_1^\top) = f(a_2^\top) = \bot_\mathsf{0}$ by (†)). Otherwise, if $f((a_1 \sqcap a_2)^\top) = \top_\mathsf{0}$ then $\widetilde{f}(a_1 \sqcap a_2)(u) = f(a_1 \sqcap a_2) = f(a_1) \sqcap f(a_2) = \widetilde{f}(a_1)(u) \sqcap \widetilde{f}(a_2)(u)$ (since f is bistable and $f(a_1^\top) = f(a_2^\top) = \top_\mathsf{0}$ by (†)).
Analogously, it follows that $\widetilde{f}$ preserves bistably coherent suprema. $\square$

The following observation is useful when computing with functions of the form $\widetilde{f}$.

Lemma 18. *If $f : A \to \mathsf{0}$ is a* **LBD** *morphism then $\widetilde{f}(a)(\bot_\mathsf{0}) = f(a)$.*

Proof. If $f(a) = \bot_0$ then $\widetilde{f}(a)(\bot_0) = \bot_0 = f(a)$ since $\bot$ and $f(a)$ are the only possible values of $\widetilde{f}(a)(\bot_0)$. If $f(a) = \top_0$ then $f(a^\top) = \top_0$ and thus $\widetilde{f}(a)(\bot_0) = f(a)$ as desired.

If $f \in D \cong [D^\omega \to 0]$ the we write $\widehat{f}$ for that element of D with

$$\widehat{f}(d, \vec{d}) = \begin{cases} \widetilde{f}(\vec{d})(\top_0) & \text{if } d \neq \bot \\ \widetilde{f}(\vec{d})(\bot_0) & \text{if } d = \bot \end{cases}$$

Lemma 19. *For every finite f in D the element $\widehat{f}$ is also finite and thus CPS_∞ definable.*

Proof. If A is a finite lbd then for every $f : A \to 0$ the **LBD** map $\widetilde{f} : A \to [0{\to}0]$ is also finite. This holds in particular for f in the finite type hierarchy over 0.

Since embeddings of lbds preserves finiteness of elements we conclude that for every finite f in D the element $\widehat{f}$ is finite as well. Thus, by Lemma 16 the element $\widehat{f}$ is CPS_∞ definable. ☐

Lemma 20. *For $f, g : A \to 0$ with $f \leq_s g$ it holds that $\widetilde{g} = \lambda a{:}A.\ \widetilde{f}(a) \circ \widetilde{g}(a)$.*

Proof. Suppose $f \leq_s g$. Let $a \in A$ and $u \in 0$. We have to show that $\widetilde{g}(a)(u) = \widetilde{f}(a)(\widetilde{g}(a)(u))$.
If $g(a^\top) = \bot_0$ then $f(a^\top) = \bot_0$ (since $f \leq_s g$) and thus $\widetilde{g}(a)(u) = \widetilde{f}(a)(\widetilde{g}(a)(u))$.
Thus, w.l.o.g. suppose $g(a^\top) = \top_0$. Then $\widetilde{g}(a)(u) = g(a)$.
If $f(a) = \top_0$ then $f(a^\top) = \top_0 = g(a)$ and, therefore, we have $\widetilde{f}(a)(\widetilde{g}(a)(u)) = f(a) = \top_0 = g(a) = \widetilde{g}(a)(u)$.
Now suppose $f(a) = \bot_0$.
If $g(a) = \bot_0$ then we have $\widetilde{f}(a)(\widetilde{g}(a)(u)) = \widetilde{f}(a)(g(a)) = \widetilde{f}(a)(\bot_0) = \bot_0$ where the last equality holds by Lemma 18.
Now suppose $g(a) = \top_0$. We proceed by case analysis on the value of $f(a^\top)$. If $f(a^\top) = \bot_0$ then $\widetilde{f}(a)(\widetilde{g}(a)(u)) = \widetilde{g}(a)(u)$. We show that $f(a^\top) = \top_0$ cannot happen.
Suppose $f(a^\top) = \top_0$ holds. Then by bistability we have $\top_0 = f(a^\top) = f(a) \sqcup f(\neg a) = \bot_0 \sqcup f(\neg a) = f(\neg a)$ and thus also $\neg f(\neg a) = \bot_0$. Since $f \leq_s g$ we have $g \sqsubseteq f^\top$. Moreover, by Lemma 9 we have $(\neg f)(a) \sqsubseteq \neg f(\neg a)$. Thus, we have $\top_0 = g(a) \sqsubseteq f^\top(a) = f(a) \sqcup (\neg f)(a) = (\neg f)(a) \sqsubseteq \neg f(\neg a) = \bot_0$ which clearly is impossible. ☐

Now we are ready to prove our universality result for CPS_∞.

Theorem 21. *All elements of the lbd D are CPS_∞ definable.*

Proof. Suppose $f \in D$. Then $f = \bigsqcup f_n$ for some increasing (w.r.t. $\leq_s$) chain $(f_n)_{n \in \omega}$ of finite elements. Since by Lemma 19 all $\widehat{f_n}$ are CPS_∞ definable there exists a CPS_∞ term F with $[\![F\underline{n}]\!] = \widehat{f_n}$ for all $n \in \omega$.

Since recursion is available in CPS_∞ one can exhibit a CPS_∞ term Ψ such that

$$\Psi g = \lambda x.\, g(\underline{0})(\Psi(\lambda n.\, g(n{+}1))x) = \bigsqcup_{n\in\omega} (g(\underline{0}) \circ \cdots \circ g(\underline{n}))(\bot)$$

Thus, the term $M_f \equiv \lambda\vec{x}.\,\Psi(\lambda y.\lambda z.F\langle y, z, \vec{x}\rangle)$ denotes f since

$$\begin{aligned}
M_f(\vec{d}) &= \Psi(\lambda y.\lambda z.F(y, z, \vec{d})) \\
&= \bigsqcup_{n\in\omega} ((\lambda z.F\underline{0}(z, \vec{d})) \circ \cdots \circ (\lambda z.F\underline{n}(z, \vec{d})))(\bot) \\
&= \bigsqcup_{n\in\omega} ((\lambda z.\widehat{f_0}(z, \vec{d})) \circ \cdots \circ (\lambda z.\widehat{f_n}(z, \vec{d})))(\bot) \\
&= \bigsqcup_{n\in\omega} ((\lambda z.\widetilde{f_0}(\vec{d})(z)) \circ \cdots \circ (\lambda z.\widetilde{f_n}(\vec{d})(z)))(\bot) \\
&= \bigsqcup_{n\in\omega} (\widetilde{f_0}(\vec{d}) \circ \cdots \circ \widetilde{f_n}(\vec{d}))(\bot) \\
&= \bigsqcup_{n\in\omega} (\widetilde{f_n}(\vec{d}))(\bot) && \text{(by Lemma 20)} \\
&= \bigsqcup_{n\in\omega} f_n(\vec{d}) && \text{(by Lemma 18)} \\
&= f(\vec{d})
\end{aligned}$$

for all $\vec{d} \in D^\omega$. □

6 Faithfulness of the interpretation

In the previous section we have shown that the interpretation of closed CPS_∞ terms in the lbd D is surjective. There arises the question whether the interpretation is also faithful. Recall that infinite normal forms for CPS_∞ are given by the grammar

$$N ::= x \mid \lambda\vec{x}.\top \mid \lambda\vec{x}.x\langle\vec{N}\rangle$$

understood in a coinductive sense.

Definition 22. *We call a model* faithful *iff for all normal forms N_1, N_2 if $[\![N_1]\!] = [\![N_2]\!]$ then $N_1 = N_2$.* ◇

We will show that the **LBD** model of CPS_∞ is not faithful. For a closed CPS_∞ term M consider

$$M^* \equiv \lambda\vec{x}.x_0\langle\bot, \lambda\vec{y}.x_0\langle M, \vec{\bot}\rangle, \vec{\bot}\rangle$$

Lemma 23. *For closed CPS_∞ terms M_1, M_2 it follows that $[\![M_1^*]\!] = [\![M_2^*]\!]$.*

Proof. We will show that for all terms M the term M^* is semantically equivalent to $\lambda\vec{x}.x_0\langle\vec{\bot}\rangle$, i.e. for all $\vec{d} \in D^\omega$ we have $[\![M^*]\!](\vec{d}) = \top$ iff $d_0 = \top$. Suppose $d_0 \neq \top$. Then $d_0 = \bot$ or there is an n such that d_0 evaluates the n-th argument first. If $n = 1$ then $d_0\langle M, \vec{\bot}\rangle = \bot$, thus

$$d_0\langle \bot, \lambda\vec{y}.d_0\langle M, \vec{\bot}\rangle, \vec{\bot}\rangle = \bot$$

which is also the case if $n \neq 1$. $\square$

Suppose N_1 and N_2 are different infinite normal forms. Then N_1^* and N_2^* have different infinite normal forms and we get $[\![N_1^*]\!] = [\![N_2^*]\!]$ by the above consideration. Thus, the **LBD** model of CPS_∞ is not faithful.

Lemma 24. *There exist infinite normal forms N_1, N_2 in CPS_∞ that can not be separated.*

Notice that in pure untyped λ-calculus different normal forms can always be separated. (cf. [3])

We think that the lack of faithfulness of CPS_∞ can be overcome by extending the language by a parallel construct and refining the observation type $\mathsf{0}$ to $\mathsf{0}' \cong \mathsf{List}(\mathsf{0}')$. The language $\mathsf{CPS}_\infty^{\|}$ associated with the domain equation $D \simeq D^{\mathsf{N}} \to \mathsf{0}'$ is given by

$$\begin{aligned} M &::= x \mid \lambda\vec{x}.t \\ t &::= \top \mid M\langle\vec{M}\rangle \mid (\!|\, t \parallel \ldots \parallel t \,|\!) \end{aligned}$$

the syntactic values are given by the grammar $V ::= \top \mid (\!| V \| \ldots \| V |\!)$ operational semantics of $\mathsf{CPS}_\infty^{\|}$ is the operational semantics of CPS_∞ extended by the rule

$$\frac{(\lambda\vec{x}.t_i)\langle\vec{M}\rangle \Downarrow V_i \quad \text{for all } i \in \{1,\ldots,n\}}{(\lambda\vec{x}.(\!| t_1 \| \ldots \| t_n |\!))\langle\vec{M}\rangle \Downarrow (\!| V_1 \| \ldots \| V_n |\!)}$$

and the normal forms of $\mathsf{CPS}_\infty^{\|}$ are given by the grammar

$$\begin{aligned} N &::= x \mid \lambda\vec{x}.t \\ t &::= \top \mid x\langle\vec{N}\rangle \mid (\!| t \| \ldots \| t |\!) \end{aligned}$$

understood in a coinductive sense. Notice that there is no possibility to combine the results of a parallel computation of $(\!| t_1 \| \ldots \| t_n |\!)$. Hence $\mathsf{CPS}_\infty^{\|}$ does not allow for the definition of a parallel or operator.

Obviously, separation of normal forms can be shown for an affine version of CPS_∞ by substituting the respective projections for head variables. Using the parallel construct $(\!| \ldots \| \ldots |\!)$ of $\mathsf{CPS}_\infty^{\|}$ we can substitute for a head variable quasi simultaneously *both* the respective projection *and* the head variable itself. Since the interpretation of $\mathsf{CPS}_\infty^{\|}$ is faithful w.r.t. the parallel construct $(\!| \ldots \| \ldots |\!)$ we get separation for $\mathsf{CPS}_\infty^{\|}$ normal forms as in the affine case. This kind of argument can be seen as as a "qualitative" reformulation of a related "quantitative" method introduced by F. Maurel in his Thesis [12] albeit in the somewhat more complex context of J.-Y. Girard's *Ludics*.

References

1. Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Inf. Comput.*, 163(2):409–470, 2000.
2. Roberto M. Amadio and Pierre-Louis Curien. *Domains and lambda-calculi*. Cambridge University Press, New York, NY, USA, 1998.
3. H. P. Barendregt. *The Lambda Calculus - its syntax and semantics*. North Holland, 1981, 1984.
4. R. Cartwright, P.-L. Curien, and M. Felleisen. Fully abstract models of observably sequential languages. *Information and Computation*, 111(2):297–401, 1994.
5. J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF: I. models, observables and the full abstraction problem, ii. dialogue games and innocent strategies, iii. a fully abstract and universal game model. *Information and Computation*, 163:285–408, 2000.
6. J. Laird. Bistable biorders: a sequential domain theory. Submitted, 2005.
7. J. Laird. *A semantic analysis of control*. PhD thesis, University of Edinburgh, 1998.
8. J. Laird. Locally boolean domains. *Theoretical Computer Science*, 342:132 – 148, 2005.
9. Ralph Loader. Finitary PCF is not decidable. *Theor. Comput. Sci.*, 266(1-2):341–364, 2001.
10. John Longley. The sequentially realizable functionals. *Ann. Pure Appl. Logic*, 117(1-3):1–93, 2002.
11. T. Löw. *Locally Boolean Domains and Universal Models for Infinitary Sequential Languages*. PhD thesis, Technical University of Darmstadt, 2006. available from `http://www.mathematik.tu-darmstadt.de/~streicher/THESES/loew.pdf.gz` or `http://elib.tu-darmstadt.de/diss/000790/`.
12. F. Maurel. *Un cadre quantitatif pour la Ludique*. PhD thesis, Université Paris 7, Paris, 2004.
13. Peter W. O'Hearn and Jon G. Riecke. Kripke logical relations and PCF. *Information and Computation*, 120(1):107–116, 1995.
14. Andrew M. Pitts. Relational properties of domains. *Information and Computation*, 127(2):66–90, 1996.
15. G. D. Plotkin. Lectures on predomains and partial functions. Course notes, Center for the Study of Language and Information, Stanford, 1985.
16. B. Reus and T. Streicher. Classical logic, continuation semantics and abstract machines. *J. Funct. Prog.*, 8(6):543–572, 1998.

Table 1. The language SPCF_∞

Types: $\sigma ::= \alpha \mid \sigma{\to}\sigma \mid \mu\alpha.\sigma \mid \Sigma_{i\in I}\sigma \mid \Pi_{i\in I}\sigma$ with I countable,
$\mathbb{1} := \Pi_\emptyset, \quad \mathbf{0} := \Sigma_\emptyset, \quad \mathbf{N} := \Sigma_{i\in\omega}\mathbb{1}, \quad \sigma^\omega := \Pi_{i\in\omega}\sigma,$
$\mathbf{n} := \Sigma_{i\in n}\mathbb{1}$ (for all $n \in \omega{+}1$)

Environments: $\Gamma \equiv x_1 : \sigma_1, \ldots, x_n : \sigma_n$ for closed types σ_i

Terms: $t ::= \; x \mid (\lambda x : \sigma.t) \mid (tt) \mid \langle t_i\rangle_{i\in I} \mid \mathbf{pr}_i(t) \mid \mathbf{in}_i(t) \mid \mathbf{case}\, t\, \mathbf{of}\, (\mathbf{in}_i\, x \Rightarrow t_i)_{i\in I} \mid$
$\mathbf{fold}(t) \mid \mathbf{unfold}(t) \mid \top \mid \mathbf{catch}(t)$

Values $v ::= \; (\lambda x : \sigma.t) \mid \langle t_i\rangle_{i\in I} \mid \mathbf{in}_i(t) \mid \mathbf{fold}(t) \mid \top$

Abbreviations / Combinators: $\underline{n} := \mathbf{in}_n\langle\rangle$ for all $n \in \omega$

$\mathbf{catch}^{\sigma_1\to\ldots\to\sigma_n\to\mathbf{N}} := \lambda f.\, \mathbf{catch}(\lambda x : \mathbf{0}^\omega.\, \mathbf{case}$
$f(e_1(\mathbf{pr}_1\, x), \ldots, e_n(\mathbf{pr}_n\, x))\, \mathbf{of}\, (\mathbf{in}_i\, y \Rightarrow \mathbf{pr}_{i+n}\, x)_{i\in\omega})$
with $e_i := \lambda x : \mathbf{0}.\, \mathbf{case}^{\mathbf{0},\sigma_i}\, x\, \mathbf{of}\, ()$ for all $i \in \{1, \ldots, n\}$

$\mathbf{Y}_\sigma := k(\mathbf{fold}^\tau(k))$ with $\tau := \mu\alpha.(\alpha{\to}(\sigma{\to}\sigma){\to}\sigma)$
and $k := \lambda x : \tau.\lambda f : \sigma{\to}\sigma.f(\mathbf{unfold}^\tau(x)xf)$

Typing rules:

$$\frac{}{\Gamma, x : \sigma, \Delta \vdash x : \sigma} \qquad \frac{}{\Gamma \vdash \top^{\Sigma_{i\in I}\sigma_i} : \Sigma_{i\in I}\sigma_i} \qquad \frac{\Gamma \vdash t : \mathbf{0}^\omega{\to}\mathbf{0}}{\Gamma \vdash \mathbf{catch}(t) : \mathbf{N}}$$

$$\frac{\Gamma, x : \sigma \vdash t : \tau}{\Gamma \vdash (\lambda x : \sigma.t) : \sigma{\to}\tau} \qquad \frac{\Gamma \vdash t : \sigma{\to}\tau \quad \Gamma \vdash s : \sigma}{\Gamma \vdash (ts) : \tau} \qquad \frac{\Gamma \vdash t_i : \sigma_i \quad \text{for all } i \in I}{\Gamma \vdash \langle t_i\rangle_{i\in I} : \Pi_{i\in I}\sigma_i}$$

$$\frac{\Gamma \vdash t : \Pi_{i\in I}\sigma_i}{\Gamma \vdash \mathbf{pr}_i^{\Pi_{i\in I}\sigma_i}(t) : \sigma_i} \qquad \frac{\Gamma \vdash t : \sigma[\mu\alpha.\sigma/\alpha]}{\Gamma \vdash \mathbf{fold}^{\mu\alpha.\sigma}(t) : \mu\alpha.\sigma} \qquad \frac{\Gamma \vdash t : \mu\alpha.\sigma}{\Gamma \vdash \mathbf{unfold}^{\mu\alpha.\sigma}(t) : \sigma[\mu\alpha.\sigma/\alpha]}$$

$$\frac{\Gamma \vdash t : \Sigma_{i\in I}\sigma_i \quad \Gamma, x : \sigma_i \vdash s_i : \tau \quad \text{for all } i \in I}{\Gamma \vdash \mathbf{case}^{\Sigma_{i\in I}\sigma_i,\tau}\, t\, \mathbf{of}\, (\mathbf{in}_i\, x \Rightarrow s_i)_{i\in I} : \tau} \qquad \frac{\Gamma \vdash t : \sigma_i}{\Gamma \vdash \mathbf{in}_i^{\Sigma_{i\in I}\sigma_i}(t) : \Sigma_{i\in I}\sigma_i}$$

Table 2. Operational semantics of SPCF_∞

Evaluation contexts:
$E ::= [\,] \mid Et \mid \mathbf{pr}_i(E) \mid \mathbf{unfold}(E) \mid \mathbf{case}\, E\, \mathbf{of}\, (\mathbf{in}_i \Rightarrow t)_{i\in I} \mid \mathbf{catch}(\lambda x : \mathbf{0}^\omega.E)$

Redex reduction:

$$(\lambda x : \sigma.t)s \to_{\mathsf{red}} t[s/x] \qquad \mathbf{case}\, \mathbf{in}_i\, s\, \mathbf{of}\, (\mathbf{in}_i\, x \Rightarrow t_i) \to_{\mathsf{red}} t_i[s/x]$$
$$\mathbf{pr}_i(\langle t_i\rangle_{i\in I}) \to_{\mathsf{red}} t_i \qquad \mathbf{unfold}(\mathbf{fold}(t)) \to_{\mathsf{red}} t$$

Evaluation context reduction:

$$E[t] \to_{\mathsf{op}} E[t'] \qquad \text{if } t \to_{\mathsf{red}} t'$$
$$E[\top] \to_{\mathsf{op}} \top \qquad \text{if } E \neq [\,]$$
$$E[\mathbf{catch}\, t] \to_{\mathsf{op}} t\langle E[\underline{n}]\rangle_{n\in\omega}$$

Table 3. Interpretation of SPCF_∞

$$[\![x_1:\sigma_1,\ldots,x_n:\sigma_n\vdash x_i:\sigma_i]\!] := \pi_i$$
$$[\![\Gamma\vdash\top:\Sigma_{i\in I}\sigma_i]\!] := x^{[\![\Gamma]\!]}\mapsto\top_{[\![\Sigma_{i\in I}\sigma_i]\!]}$$
$$[\![\Gamma\vdash(\lambda x:\sigma.t):\sigma\to\tau]\!] := \mathsf{curry}_{[\![\Gamma]\!],[\![\sigma]\!]}([\![\Gamma,x:\sigma\vdash t:\tau]\!])$$
$$[\![\Gamma\vdash ts:\tau]\!] := \mathsf{eval}\circ\langle[\![\Gamma\vdash t:\sigma\to\tau]\!],[\![\Gamma\vdash s:\sigma]\!]\rangle$$
$$[\![\Gamma\vdash\langle t_i\rangle_{i\in I}^{\Pi_{i\in I}\sigma_i}:\Pi_{i\in I}\sigma_i]\!] := \langle[\![\Gamma\vdash t_i:\sigma_i]\!]\rangle_{i\in I}$$
$$[\![\Gamma\vdash\mathbf{pr}_i(t):\sigma]\!] := \pi_i\circ[\![\Gamma\vdash t:\sigma]\!]$$
$$[\![\Gamma\vdash\mathbf{case}^{\Sigma_{i\in I}\tau_i,\sigma}\,t\,\mathbf{of}\,(\mathbf{in}_i\,x\Rightarrow t_i):\sigma]\!] := \mathsf{case}\circ\langle[\![\Gamma\vdash t]\!],\langle[\![\Gamma\vdash(\lambda x:\tau_i.t_i):\tau_i\to\sigma]\!]\rangle_{i\in I}\rangle$$
$$[\![\Gamma\vdash\mathbf{in}_i(t):\Sigma_{i\in I}\sigma_i]\!] := \iota_i\circ[\![\Gamma\vdash t:\sigma_i]\!]$$
$$[\![\Gamma\vdash\mathbf{catch}(t):\mathbf{N}]\!] := \mathsf{catch}\circ[\![\Gamma\vdash t:\mathbf{0}^\omega\to\mathbf{0}]\!]$$
$$[\![\Gamma\vdash\mathbf{fold}^{\mu\alpha.\sigma}(t):\mu\alpha.\sigma]\!] := \mathsf{fold}\circ[\![\Gamma\vdash t:\sigma[\mu\alpha.\sigma/\alpha]]\!]$$
$$[\![\Gamma\vdash\mathbf{unfold}^{\mu\alpha.\sigma}(t):\sigma[\mu\alpha.\sigma/\alpha]]\!] := \mathsf{unfold}\circ[\![\Gamma\vdash t:\mu\alpha.\sigma]\!]$$

Table 4. The language CPS_∞

Contexts: $\Gamma\equiv[x_i\mid i\in I]$ with $I\in\omega+1$

Terms: $M ::= x \mid \lambda\vec{x}.t \qquad \vec{x}\equiv(x_i)_{i\in\omega}$

$\qquad t ::= \top \mid M\langle\vec{M}\rangle \qquad \vec{M}\equiv(M_i)_{i\in\omega}$

Terms-in-context:

$$\frac{}{[x_i\mid i\in I]\vdash x_i}\qquad\frac{}{\Gamma\vdash\lambda\vec{x}.\top}\qquad\frac{\Gamma\cup[x_i\mid i\in I]\vdash M\quad\Gamma\cup[x_i\mid i\in I]\vdash N_i}{\Gamma\vdash\lambda\vec{x}.M\langle\vec{N}\rangle}$$

Operational semantics:

$$\frac{}{\top\Downarrow\top}\qquad\frac{t[M_i/x_i]_{i\in\omega}\Downarrow\top}{(\lambda\vec{x}.t)\langle\vec{M}\rangle\Downarrow\top}$$

Table 5. Retraction $\Pi_{i\in n}\mathbf{U}\lhd\mathbf{U}$

$$e := \lambda f:\Pi_{i\in n}\mathbf{U}.\lambda n:\mathbf{N}.\,\mathbf{case}\,\mathbf{pr}_0(\pi_{\mathbf{2}}(n))\,\mathbf{of}\left(\begin{array}{l}\mathbf{in}_0\,x\Rightarrow\mathbf{catch}^{\mathbf{U}}(\mathbf{pr}_i(f))\\ \mathbf{in}_1\,x\Rightarrow\mathbf{case}\,\mathbf{pr}_0(\pi_{\mathbf{n}}(\mathbf{pr}_1(\pi_{\mathbf{2}}(n))))\,\mathbf{of}\\ \qquad(\underline{j}\Rightarrow\mathbf{pr}_j(f)(\mathbf{pr}_1(\pi_{\mathbf{n}}(\mathbf{pr}_1(\pi_{\mathbf{2}}(n))))))_{j\in n}\end{array}\right)$$

$$p := \lambda f:\mathbf{U}.\langle\mathbf{case}\,f(\iota_{\mathbf{2}}\langle\underline{0},\underline{i}\rangle)\,\mathbf{of}\left(\begin{array}{l}\mathbf{in}_0\,x\Rightarrow\lambda n:\mathbf{N}.f(\iota_{\mathbf{2}}\langle\underline{1},\iota_{\mathbf{n}}\langle\underline{i},n\rangle\rangle)\\ \mathbf{in}_{j+1}\,x\Rightarrow\lambda n:\mathbf{N}.\underline{j}\end{array}\right)_{j\in\omega}\rangle_{i\in n}$$

with $\iota_{\mathbf{n}}:(\mathbf{n}\times\mathbf{N})\to\mathbf{N}$ and $\pi_{\mathbf{n}}:\mathbf{N}\to(\mathbf{n}\times\mathbf{N})$ satisfying $[\![\pi_{\mathbf{n}}(\iota_{\mathbf{n}}\langle\underline{i},\underline{m}\rangle)]\!]=[\![\langle\underline{i},\underline{m}\rangle]\!]$ for all $i\in n$ and $m\in\omega$

Table 6. Retraction $\mathbf{U}{\to}\mathbf{N} \lhd \mathbf{U}$

$$e := \lambda F : \mathbf{U}{\to}\mathbf{N}.\lambda n : \mathbf{N}.\, \mathbf{case}\, \alpha^*(n)\, \mathbf{of} \begin{pmatrix} \mathbf{in}_0\, t \Rightarrow \mathbf{case}\, \mathbf{catch}^{\mathbf{U}\to\mathbf{N}}(F)\, \mathbf{of}\ R \\ \mathbf{in}_1\, t \Rightarrow \alpha(\mathbf{in}_1(F(\lambda x : \mathbf{N}.t))) \\ \mathbf{in}_2\, t \Rightarrow \mathbf{case}\, S\, \mathbf{of} \begin{pmatrix} \mathbf{in}_{2i}\, x \Rightarrow \alpha(\mathbf{in}_1\, \underline{i}) \\ \mathbf{in}_{2i+1}\, x \Rightarrow \alpha(\mathbf{in}_2\, \underline{i}) \end{pmatrix}_{i\in\omega} \end{pmatrix}$$

$$R := \begin{pmatrix} \mathbf{in}_0\, x \Rightarrow \alpha(\mathbf{in}_0\langle\rangle) \\ \mathbf{in}_{i+1}\, x \Rightarrow \alpha(\mathbf{in}_1\, \underline{i}) \end{pmatrix}_{i\in\omega}$$

$$S := \mathbf{catch}(\lambda x : \mathbf{0}^\omega.\, \mathbf{case}\, F(\lambda n : \mathbf{N}. \\ \mathbf{case}\, \mathsf{find}(t, n)\, \mathbf{of} \begin{pmatrix} \mathbf{in}_0\, s \Rightarrow s \\ \mathbf{in}_1\, s \Rightarrow \mathbf{case}^{\mathbf{0},\mathbf{N}}(\mathbf{case}\, n\, \mathbf{of}\, (\mathbf{in}_j\, u \Rightarrow \mathbf{pr}_{2j+1}\, x)_{j\in\omega})\, \mathbf{of}\, () \end{pmatrix}) \\ \mathbf{of}\, (\mathbf{in}_i\, s \Rightarrow \mathbf{pr}_{2i}\, x)_{i\in\omega})$$

$$p := \lambda r : \mathbf{N}{\to}\mathbf{N}.\lambda f : \mathbf{N}{\to}\mathbf{N}.\, \mathbf{case}\, \alpha^*(r(\alpha(\mathbf{in}_0\langle\rangle)))\, \mathbf{of} \begin{pmatrix} \mathbf{in}_0\, t \Rightarrow T \\ \mathbf{in}_1\, t \Rightarrow t \\ \mathbf{in}_2\, t \Rightarrow \bot \end{pmatrix}$$

$$T := \mathbf{case}\, \mathbf{catch}^{\mathbf{N}\to\mathbf{N}}(f)\, \mathbf{of} \begin{pmatrix} \mathbf{in}_0\, t \Rightarrow U(\mathsf{nil}) \\ \mathbf{in}_{i+1}\, t \Rightarrow \mathbf{case}\, \alpha^*(r(\alpha(\mathbf{in}_1\, \underline{i})))\, \mathbf{of} \begin{pmatrix} \mathbf{in}_0\, t \Rightarrow \bot \\ \mathbf{in}_1\, t \Rightarrow t \\ \mathbf{in}_2\, t \Rightarrow \bot \end{pmatrix} \end{pmatrix}_{i\in\omega}$$

$$U := \mathbf{Y}_{\mathbf{N}\to\mathbf{N}}(\lambda h : \mathbf{N}{\to}\mathbf{N}.\lambda g : \mathbf{N}.\, \mathbf{case}\, \alpha^*(r(\alpha(\mathbf{in}_2\, g))) \\ \mathbf{of} \begin{pmatrix} \mathbf{in}_0\, t \Rightarrow \bot \\ \mathbf{in}_1\, t \Rightarrow t \\ \mathbf{in}_2\, t \Rightarrow h(\mathsf{cons}(g, (t, f(t)))) \end{pmatrix})$$

with $\alpha : (\mathbb{1}{+}\mathbf{N}{+}\mathbf{N}){\to}\mathbf{N}$ and $\alpha^* : \mathbf{N}{\to}(\mathbb{1}{+}\mathbf{N}{+}\mathbf{N})$ satisfying $[\![\alpha^*(\alpha(\mathbf{in}_0\langle\rangle))]\!] = [\![\mathbf{in}_0\langle\rangle]\!]$ and $[\![\alpha^*(\alpha(\mathbf{in}_i\, n))]\!] = [\![\mathbf{in}_i\, n]\!]$ for $i = 1, 2$ and $n \in \omega$, and the following auxiliary list-handling functions in Haskell-style where γ encodes lists (of pairs) of natural numbers as natural numbers

$$\mathsf{nil} := \gamma(\texttt{[]}) \qquad \mathsf{cons}(g, (x, y)) := \gamma((x, y) : \gamma^{-1}(g))$$

$$\begin{aligned} \mathsf{find}(g, x) := \texttt{case}\ \gamma^{-1}(g)\ \texttt{of} & \\ \texttt{[]} \quad & \texttt{->}\ \mathbf{in}_1\langle\rangle \\ ((x, y) : r) \quad & \texttt{->}\ \mathbf{in}_0\, y \\ (\_ : r) \quad & \texttt{->}\ \mathsf{find}(\gamma(r), x) \end{aligned}$$