Bidomains and Full Abstraction for Countable Nondeterminism

James Laird[*]

Dept. of Informatics, University of Sussex, UK
jiml@sussex.ac.uk

Abstract. We describe a denotational semantics for a sequential functional language with random number generation over a countably infinite set (the natural numbers), and prove that it is fully abstract with respect to may-and-must testing.

Our model is based on biordered sets similar to Berry's bidomains, and stable, monotone functions. However, (as in prior models of unbounded non-determinism) these functions may not be continuous. Working in a biordered setting allows us to exploit the different properties of both extensional and stable orders to construct a Cartesian closed category of sequential, discontinuous functions, with least and greatest fixpoints having strong enough properties to prove computational adequacy.

We establish full abstraction of the semantics by showing that it contains a simple, first-order "universal type-object" within which all types may be embedded using functions defined by (countable) ordinal induction.

1 Introduction

Non-determinism is an abstract property with which we may represent the inherent uncertainty of a computational system, whether ocurring by accident or by design. When describing a non-deterministic system, we are typically interested in the possibility of failure, whether by divergence or premature termination. However, it is well known that capturing these behaviours in systems exhibiting unbounded non-determinism — i.e. programs which may choose between an infinite set of possible steps without diverging — presents a challenge for denotational semantics, because semantic functions characterizing their divergent behaviours are not, in general, continuous (see e.g. [1]). The object of this paper is to describe a domain-theoretic setting in which we may successfully capture the observable properties of functional programs with countable non-determinism via a semantics which is fully abstract with respect to may and must testing.

The basis for our model is a category of biordered sets and order-preserving functions based on Berry's *bidomains* [2]. In previous work by the author [8, 7] these have been used to give fully abstract models of sequential languages such as unary PCF (which may itself be considered as a λ-calculus with a binary choice

[*] Supported by UK EPSRC grant GR/S72181.

operator). However, as well as capturing sequentiality, using two orders allows us to resolve some of the continuity problems associated with unbounded non-determinism. Essentially, they give separate extensional and intensional characterizations of programs, each having different completeness properties, which we exploit in proving (e.g.) computational adequacy.

Another feature of previous biorder-based semantics which is developed here is the focus on *observably sequentiality*, in which failure by divergence, and failure by premature termination (error) are distinguished. As observed by Cartwright and Felleisen [3], this simplifies the full abstraction problem for sequential functional languages, by making evaluation-order extensionally observable. In a non-deterministic setting, separating the two forms of failure makes a certain duality between "may" and "must" testing evident in both operational and denotational semantics: we give separate models of these in the same category of biorders, by interpreting error as a least element and recursion as a greatest fixed point with respect to may-testing, and interpreting error as a greatest element and recursion as a least fixed point with respect to must-testing.

We establish full abstraction for both may and must-testing semantics by developing a methodology used for proving definability results for observably sequential languages in particular (e.g. [11, 8]). We show that each type-object is a retract of a first-order "universal" type-object, and that these retractions are definable as terms in the language. This sheds some light on the process of computing interaction between functions with unbounded non-determinism, via countable sequences of unfoldings, in addition to sidestepping reliance of typical proofs of full abstraction on continuity and algebraicity.

1.1 Related Work

Apt and Plotkin [1] study a fully abstract denotational model of a simple imperative language with random assignment in a setting which brings together much of the preceding work on the semantics of countable non-determinism, and clarifies the role of non-continuity in particular. Lassen and Pitcher [9] study bisimulation equivalences based on may and must testing for a functional language similar to that modelled here. *Game semantics* has been used to describe denotational models of non-deterministic langages: Harmer and McCusker [6] have described a fully abstract may-and-must games model of Idealized Algol with bounded choice, whilst Levy [10] has described a game semantics of a language with unbounded non-determinism which captures an infinite trace equivalence. However, the biorder model described here appears to be the first fully abstract may-and-must testing semantics for a functional language with unbounded choice.

2 Syntax and Operational Semantics

We illustrate our approach by describing may and must semantics for a small functional language with countable non-determinism (which could be regarded as a target-language for CPS translation): a simply-typed λ-calculus with arithmetic, recursion and a random-number generator. Types are generated from two ground types: a data type of natural number *values* and a program (or "response") type $\mathtt{o}$ containing no values, but a single "error" term. Programs of function type may take either data or programs as arguments, but must return a program — i.e. $\mathtt{nat}$ may not occur on the right of an arrow. Thus the types of our language are:

$$T ::= \mathtt{nat} \mid \mathtt{o} \mid T \Rightarrow P$$

where $P \neq \mathtt{nat}$ (we refer to non-$\mathtt{nat}$ types as *pointed*).

Terms are obtained by extending the simply-typed λ-calculus with a set of basic arithmetic constants and (primitive recursive) operations on $\mathtt{nat}$, including:

- zero ($\mathtt{0}$), successor and predecessor ($\mathtt{succ(\_)}$, $\mathtt{pred(\_)}$)
- equality testing, $\_{=}\_$
- "injective pairing" $(\_ * \_)$ and projection $\mathtt{fst}\_$ and $\mathtt{snd}\_$.

and the following constants:

Error $\mathbf{e} : \mathtt{o}$,
Zero test $\mathtt{If0} : \mathtt{nat} \Rightarrow P \Rightarrow P \Rightarrow P$.
Fixpoints $\mathtt{Y} : (P \Rightarrow P) \Rightarrow P$.
Random number generation $\mathtt{rnd} : (\mathtt{nat} \Rightarrow \mathtt{o}) \Rightarrow \mathtt{o}$

We write $\mathtt{Eq}$ for $\lambda wxyz.((\mathtt{If0}\,(w{=}x))\,y)\,z : \mathtt{nat} \Rightarrow \mathtt{nat} \Rightarrow P \Rightarrow P \Rightarrow P$, and Ω for the divergent term $\mathtt{Y}\,\lambda x.x$ at each pointed type.

2.1 Operational Semantics

Note that any closed term $\mathtt{t} : \mathtt{nat}$ is an arithmetic expression derived from the total operations in the language. We assume an operation $|\_|$ evaluating such expressions to numerals, which thus has the properties:

- $|\mathtt{s} = \mathtt{t}| = \mathtt{0}$ if $|\mathtt{s}| = |\mathtt{t}|$ and $|\mathtt{s} = \mathtt{t}| = \mathtt{1}$, otherwise.
- $|\mathtt{fst(s * t)}| = |\mathtt{s}|$ and $|\mathtt{snd(s * t)}| = |\mathtt{t}|$.

We define two evaluation relations $\Downarrow^{may}$ and $\Downarrow^{must}$ between closed terms of pointed type and "canonical forms" (λ-abstractions, $\mathtt{If0}$, $\mathtt{rnd}$ and $\mathbf{e}$) by combining the following, standard, "deterministic" fragment:

$$\frac{}{\mathbf{e}\Downarrow\mathbf{e}} \qquad \frac{}{\mathtt{rnd}\Downarrow\mathtt{rnd}} \qquad \frac{}{\lambda x.M\Downarrow\lambda x.M}$$

$$\frac{}{\mathtt{If0}\Downarrow\mathtt{If0}} \qquad \frac{}{\mathtt{If0}\;\mathtt{t}\Downarrow\lambda xy.x}\;|\mathtt{t}| = \mathtt{0} \qquad \frac{}{\mathtt{If0}\;\mathtt{t}\Downarrow\lambda xy.y}\;|\mathtt{t}| \neq \mathtt{0}$$

$$\frac{}{\mathtt{Y}\Downarrow\lambda f.f\,(\mathtt{Y}\,f)} \qquad \frac{M\Downarrow\lambda x.M' \qquad M'[N/x]\Downarrow C}{M\,N\Downarrow C}$$

with one of the following rules for evaluating $\mathtt{rnd}$ by erratically generating a numeral and passing it to its argument:

$$\frac{\exists n{\in}\mathbb{N}.M\;\mathtt{n}\Downarrow^{may}\mathbf{e}}{\mathtt{rnd}\,M\Downarrow^{may}\mathbf{e}}\text{May} \qquad \frac{\forall n{\in}\mathbb{N}.M\;\mathtt{n}\Downarrow^{must}\mathbf{e}}{\mathtt{rnd}\,M\Downarrow^{must}\mathbf{e}}\text{Must}$$

We define notions of approximation and equivalence with respect to may and must testing. Given $M, N : T$:

- $M \lesssim^{may} N$ if for all compatible program contexts $C[\cdot] : \mathtt{o}$, $C[N]{\Downarrow}^{may}\mathbf{e}$ implies $C[M]{\Downarrow}^{may}\mathbf{e}$. $M \simeq^{may} N$ if $M \lesssim^{may} N$ and $N \lesssim^{may} M$.
- $M \lesssim^{must} N$ if for all compatible program contexts $C[\cdot] : \mathtt{o}$, $C[M]{\Downarrow}^{must}\mathbf{e}$ implies $C[N]{\Downarrow}^{must}\mathbf{e}$. $M \simeq^{must} N$ if $M \lesssim^{must} N$ and $N \lesssim^{must} M$.

Note the direction of the implication in the definition of may-approximation: $M \lesssim^{may} N$ if testing N leads to fewer errors than testing M.

As expected, there are functions which not continuous with respect to $\lesssim^{must}$ (considered as a partial order on $\simeq^{must}$ equivalence-classes of terms) — in particular, the operator $\mathtt{rnd}$ itself. For example, let $M_0 : \mathtt{nat} \Rightarrow \mathtt{o} = \lambda x.\Omega$ and $M_{i+1} = \lambda x.((\mathtt{If0}\, x)\, \mathbf{e})\, (M_i\, \mathtt{pred}(x))$ — i.e. M_i terminates if and only if its argument is less than i. So $\mathtt{rnd}\, M_i \not\Downarrow^{must}$ for all i, but the $\lesssim^{must}$ least upper bound of the chain $M_0 \lesssim^{must} M_1 \lesssim^{must} \ldots$ is $\lambda x.\mathbf{e}$, and $\mathtt{rnd}\, \lambda x.\mathbf{e}{\Downarrow}^{must}$. We study further examples of continuity and noncontinuity in Section 4.1.

The expressiveness of the language may be exploited by using it as the basis for CPS interpretation of more elaborate functional languages with unbounded nondeterminism. For example, we may translate PCF with random number generation simply by representing the type of natural number *computations*, $\mathtt{nat}_\bot^\top$, as $(\mathtt{nat} \Rightarrow \mathtt{o}) \Rightarrow \mathtt{o}$, giving $\mathtt{rnd} : \mathtt{nat}_\bot^\top$. (The corresponding bidomain model will contain additional elements corresponding to simple control operators and errors, yielding a fully abstract semantics of Cartwright and Felleisen's SPCF [3] with random number generation). Similarly, we may CPS translate Lassen and Pitcher's [9] version of Moggi's metalanguage extended with countable choice by representing the nondeterminism monad constructor $\mathsf{P}\_$ as the continuations monad $(\_ \Rightarrow \mathtt{o}) \Rightarrow \mathtt{o}$. (Again, the translation and associated model will be fully abstract if first-class continuations are included in the source language.)

By distinguishing the two notions of failure, and taking them as the basis for our notions of observation in our model, we can also reason about a variety of behaviours of programs in such languages. For instance we may capture the requirement that $M : \mathtt{nat}_\bot^\top$ *may* converge to some (non-error) value as the conjunction of $M\, \lambda x.\mathbf{e}{\Downarrow}^{may}$ and $M\, \lambda x.\Omega \not\Downarrow^{must}$, and the requirement that M *must* converge to a value as $M\, \lambda x.\mathbf{e}{\Downarrow}^{must}$ and $M\, \lambda x.\Omega \not\Downarrow^{may}$.

3 Complete Meet Biorders

Berry's biorders [2, 4] are based on a binary greatest lower bound operator (i.e. a meet semi-lattice) which may be used to interpret binary choice [8]. Thus to give a semantics of unbounded choice, we develop a notion of biorder based on complete lattices (i.e. having a greatest lower bound operator for arbitrary sets).

Definition 1. *A complete (meet) biorder is a triple* $\langle D, \sqsubseteq, \leq \rangle$ *consisting of a set* D *with two partial orders* $\sqsubseteq, \leq \subseteq D \times D$ *such that:*

- $(D, \sqsubseteq)$ *(the extensional order) is a complete lattice: every subset X has a greatest lower bound $\bigsqcap X$.*
- $(D, \leq)$ *(the stable order) is included in $\sqsubseteq$ ($\leq \subseteq \sqsubseteq$), has a least element, $\bot = \bigsqcap D$ and for any $X, Y \subseteq D$ such that $X \leq Y$ in the Egli-Milner order (i.e. $\forall x \in X \exists y \in Y.x \leq y \wedge \forall y \in Y.\exists x \in X.x \leq Y$) we have $\bigsqcap X \leq \bigsqcap Y$.*

We shall write $\uparrow X$ if $X \subseteq D$ is non-empty[1] and bounded above in $(D, \leq)$, observing that $(D, \leq)$ is bounded co-complete in the following sense:

Lemma 1. *If $\uparrow X$ then $\bigsqcap X$ is a greatest lower bound for X in $(D, \leq)$.*

Proof. Suppose X is bounded above by y in $\leq$. Then for any $x \in X$, $X \leq \{x, y\}$ and so $\bigsqcap X \leq x \sqcap y = x$, and if z is a $\leq$-lower bound for X, then $\{z\} \leq X$ and so $z \leq \bigsqcap X$.

Products of complete meet biorders are defined by taking the pointwise orderings on the product of the underlying sets. Particular examples include the one-element biorder 1 (the unit for the product), the "Sierpinski" biorder Σ containing two elements, ordered stably and extensionally.

Definition 2. *A function f from $(D, \sqsubseteq, \leq)$ to $(D', \sqsubseteq', \leq')$ is monotone if it preserves both orders, and (completely) stable if for every stably bounded set X, $f(\bigsqcap X) = \bigsqcap f(X)$.*

Proposition 1. *The category of complete meet biorders and completely stable and monotone functions is Cartesian closed.*

Proof. For complete biorders $(D, \sqsubseteq_D, \leq_D)$ and $(E, \sqsubseteq_E, \leq_E)$ the function-space $D \Rightarrow E$ is the biorder over the set of monotone and stable functions from D to E in which the extensional order is defined:

$$f \sqsubseteq_{D \Rightarrow E} g \text{ if } f(x) \sqsubseteq_E g(x) \text{ for all } x \in D.$$

and the stable order is defined:

$$f \leq_{D \Rightarrow E} g \text{ if for all } x \leq_D y,\ f(x) \leq_E g(y) \text{ and } f(x) = f(y) \sqcap g(x).$$

This satisfies the axioms for a complete biorder, with the greatest lower bound of a bounded set of functions F defined pointwise: $(\bigsqcap F)(x) = \bigsqcap\{f(x) \mid f \in F\}$.

Thus we have the basis for the semantics of functional languages with unbounded choice (a CCC with a greatest lower bound operator). To interpret the `Y` combinator we require least and greatest fixed points of each endomorphism $f : D \to D$. As in [1], we may compute these as the suprema/infima of chains of approximants obtained by iterating f countably many times.

Proposition 2. *Every endomorphism $f : A \to A$ has a $\sqsubseteq$-least fixed point $f^\dagger : \mathbf{1} \to A$ and a $\sqsubseteq$-greatest fixed point $f^\ddagger : \mathbf{1} \to A$.*

[1] In particular, $\top = \bigsqcup \varnothing$ is not in general a $\leq$-greatest element.

Proof. We obtain $f^\dagger$ as a stationary point of the $\sqsubseteq$-chain defined $f^\lambda = f(\bigsqcup_{\kappa<\lambda} f^\kappa)$ for each ordinal λ. Then $\lambda \leq \kappa$ implies $f^\lambda \sqsubseteq f^\kappa$, and if $f^\lambda < f^{\lambda+1}$ then $f^\kappa \sqsubset f^\mu$ for all $\kappa < \mu \leq \lambda$. So if κ has cardinality strictly greater than A, then we must have $f(f^\kappa) = f^\kappa$. Moreover f^κ is a least (pre)fixed point, since if $f(a) \sqsubseteq a$ then $f^\lambda \sqsubseteq a$ for all λ.

We construct the greatest fixed point $f^\ddagger$ similarly, as a stationary point in the descending $\sqsubseteq$-chain defined $f^\lambda = f(\bigsqcap_{\kappa<\lambda} f^\kappa)$.

However, since least upper bounds in complete meet biorders are defined indirectly, the mere existence of the least fixed point $f^\dagger$ is not sufficient to prove that it yields an interpretation of `Y` which is *computationally adequate.* It transpires that the continuity property required to prove adequacy is that for $f : (A \Rightarrow B) \rightarrow (A \Rightarrow B)$, $(\bigsqcup_{\kappa<\lambda} f^\kappa)(e) = \bigsqcup_{\kappa<\lambda} f(e)$. In general, the least upper bound of a $\sqsubseteq$-directed set of functions *cannot* be determined in this way (i.e. it is not the case that $(\bigsqcup F)(x) = \bigsqcup\{f(x) \mid f \in F\}$)— we give an example in the next section. However, we shall now show that we may define a full (Cartesian closed) subcategory of biorders in which the stable order is a cpo in which least upper bounds of directed sets of functions is determined pointwise.

Definition 3. *A complete meet biorder D is a complete meet bidomain*[2] *if it satisfies the following conditions:*

Stable Completeness. *Every set $X \subseteq D$ which is* stably directed *(i.e. upwards directed with respect to $\leq$) has a least upper bound $\bigvee X$ with respect to the stable order, such that $\bigvee X = \bigsqcup X$, and satisfying the following distributivity property:*
for any y with $y \uparrow \bigvee X$, $y \sqcap \bigvee X = \bigvee\{x \sqcap y \mid x \in X\}$.

Algebraicity. *An element $c \in D$ is* weakly compact *($c \in \mathcal{K}(D)$) if for every stably directed set X such that $c \sqsubseteq \bigvee X$ there exists $x \in X$ such that $c \sqsubseteq x$. D is (weakly) algebraic if every element in $d \in D$ is the ($\sqsubseteq$) least upper bound of its set of weakly compact approximants — $d = \bigsqcup\{c \in \mathcal{K}(D) \mid c \sqsubseteq d\}$.*

Lemma 2. *If D, E are stably complete and algebraic, then $D \Rightarrow E$ is stably complete.*

Proof. Given a stably directed set of functions F, the set $\{f(x) \mid f \in F\}$ is stably directed, so we may define the stable supremum of F pointwise: $(\bigvee F)(x) = \bigvee\{f(x) \mid f \in F\}$.

This is monotone — if $x \leq y$ then for all f, $f(x) \leq f(y) \leq (\bigvee F)(y)$ and so $(\bigvee F)(x) \leq (\bigvee F)(y)$ — and binary-stable: if $x \uparrow y$, then $(\bigvee F)(x) \sqcap (\bigvee F)(y) = \bigvee\{f(x) \sqcap \bigvee F(y) \mid f \in F\} = \bigvee\{f(x) \sqcap g(y) \mid f, g \in F\} \sqsubseteq \bigvee F(x \sqcap y)$.

To show that $\bigvee F$ is stable with respect to infima of stably bounded infinite sets, it is sufficient to show that it preserves infima of (downwards) stably directed sets. So suppose we have a downwards stably-directed set X. We need to show that $\bigsqcap(\bigvee F)(X) \sqsubseteq \bigvee F(\bigsqcap X)$.

[2] Note that a complete meet bidomain need not be a bidomain in the sense of Berry.

Suppose c is a compact element such that $c \sqsubseteq \bigsqcap(\bigvee F)(X)$. Choosing $x \in X$, we have $c \sqsubseteq (\bigvee F)(x)$ and so by compactness of c, there exists $f \in F$ such that $c \sqsubseteq f(x)$. Then for any $y \in X$, there exists $z \in X$ such that $z \leq x, y$, and so we may find $g \in F$ such that $c \sqsubseteq g(z)$, and $h \in F$ such that $f, g \leq h$. Thus $f(z) = f(x) \sqcap h(z)$, and so $c \sqsubseteq f(z) \sqsubseteq f(y)$. So $c \sqsubseteq \bigsqcap f(X)$, and $c \sqsubseteq \bigvee F(\bigsqcap X)$ as required.

Moreover $\bigvee F$ is a $\leq$-least upper bound for F (as well as being a $\sqsubseteq$ least upper bound): if $f \in F$ then for all x, $f(x) \leq (\bigvee F)(x)$ and if $x \leq y$ then $f(y) \sqcap (\bigvee F)(x) = \bigvee\{f(y) \sqcap g(x) \mid g \in F \sqcap f \leq g\} = f(x)$. If $f \leq g$ for all $f \in F$, then for all x, $(\bigvee F)(x) \leq g(x)$, and if $x \leq y$ then $(\bigvee F)(y) \sqcap g(x) = \bigvee\{f(y) \sqcap g(x) \mid f \in F\} = (\bigvee F)(x)$.

Since $\sqcap, \bigvee$ are both determined pointwise, the distributivity condition is straightforward to verify.

Lemma 3. *The complete meet bidomains and completely stable and monotone functions form a CCC.*

Proof. By Lemma 2, if D, E are complete meet bidomains then $D \Rightarrow E$ is stably complete, so it remains to show weak algebraicity. Given $f \in D \Rightarrow E$, $d \in D$, and weakly compact $c \in E$ such that $c \sqsubseteq f(d)$, we define $f^d_c \in D \Rightarrow E$ such that $f^d_c(x) = c$ if $d \sqsubseteq x$ and $f^d_c(x) = \bot$ otherwise.

Then f^c_d is monotone and completely stable: if $x \sqsubseteq y$, then if $d \sqsubseteq x$, $f^d_c(x) = f^d_c(y) = c$, otherwise $f^d_c(x) = \bot \leq f^d_c(y)$. Given a stably bounded set X, if $d \sqsubseteq \bigsqcap X$ then $d \sqsubseteq x$ for all $x \in X$, and so $f^d_c(\bigsqcap X) = c = \bigsqcap f^d_c(X)$. If $d \not sqleq \bigsqcap X$ then there exists $x \in X$ such that $d \not\sqsubseteq x$, and so $f^d_c(x) = \bot = f^d_c(\bigsqcap X)$.

It is straightforward to check that f^c_d is weakly compact (if $f^d_c \sqsubseteq \bigvee F$ then $f^d_c(d) = c \sqsubseteq (\bigvee F)(d)$ and so $c \sqsubseteq f(d)$ for some $f \in F$, and so $f^d_c \sqsubseteq f$) and $f = \bigsqcup\{f^d_c \mid d \in D \wedge c \in \mathcal{K}(E) \wedge c \sqsubseteq f(d)\}$.

To interpret unpointed types (in the current setting, just the type `nat` of natural number values), we define a notion of "pre-bidomain".

Definition 4. *A (complete meet) pre-bidomain $(D, \sqsubseteq, \leq)$ is a set D with partial orders $\leq \subseteq \sqsubseteq$ such that for each $x \in D$, $D_x = \{y \in D \mid \exists z \in D.z \sqsubseteq x, y\}$ is a co-complete bidomain.*

The co-product of pre-bidomains (formed pointwise) is a pre-bidomain, and gives the following characterization result.

Lemma 4. *For a pre-bidomain D, let $\lfloor D \rfloor$ be the set of $\sqsubseteq$-minimal elements of D. Then $D \cong \coprod_{x \in \lfloor D \rfloor} D_x$.*

Proof. Let $\bot(x) = \bigsqcap\{y \in D \mid y \sqsubseteq x\}$. Then for each x, $\bot(x)$ is a minimal element of D, and it is straightforward to show that the map sending x to $\mathsf{in}_{\bot(x)}(x)$ is an order-isomorphism.

Proposition 3. *The category of pre-bidomains and monotone and stable functions is bicartesian closed.*

Proof. We define the cartesian closed structure as for the category of complete bidomains and monotone and stable functions: thus the principal point to check is that the function-space yields a well-defined pre-bidomain, for which we use the decomposition into co-products (Lemma 4). We show that:

– for any complete meet bidomain A, and pre-bidomain D, $\coprod_{x \in \lfloor D \rfloor}(A \Rightarrow D_x) \cong A \Rightarrow \coprod_{x \in \lfloor D \rfloor} D_x \cong A \Rightarrow D$, and hence $A \Rightarrow D$ is a pre-bidomain.
– for any pre-bidomains D, E: $\Pi_{x \in \lfloor D \rfloor}(D_x \Rightarrow E) \cong (\coprod_{x \in \lfloor D \rfloor} D_x) \Rightarrow E \cong D \Rightarrow E$, and so $D \Rightarrow E$ is a pre-bidomain. (So if E is a complete bidomain then so is $D \Rightarrow E$.)

4 Denotational Semantics

We now give the may and must testing semantics of the functional language defined in Section 2. We interpret `nat` as the pre-bidomain $\mathbb{N}_* = \coprod_{i \in \mathbb{N}} 1$, and the remaining (pointed) types as the corresponding bidomains: i.e. $[\![o]\!] = \Sigma$ and $[\![S \Rightarrow T]\!] = [\![S]\!] \Rightarrow [\![T]\!]$.

We interpret terms-in-context $x_1 : S_1, \ldots, x_n : S_n \vdash M : T$ as monotone and completely stable functions from $[\![S_1]\!] \times \ldots [\![S_n]\!]$ to $[\![T]\!]$, giving two denotations $[\![M]\!]_{may}, [\![M]\!]_{must}$ for each term. We use the Cartesian closed structure to interpret λ-abstraction and application in standard fashion, and the associated operations on $\mathbb{N}$ to interpret the arithmetic constants and operations. Random assignment `rnd` is interpreted as the function which takes every argument except $\top$ to $\bot$:

$$[\![\texttt{rnd}]\!]_{may}(f) = [\![\texttt{rnd}]\!]_{must}(f) = \bigsqcap\{f(n) \mid n \in \mathbb{N}_*\}$$

Thus every program with neither recursion nor explicit errors has the same denotation in the may and must semantics.

In the may-testing semantics, we interpret the error as the *least* element $\bot$, and the fixpoint combinator $\mathtt{Y} : (P \Rightarrow P) \Rightarrow P$ as the *greatest* fixed point of the endomorphism $F : (P \Rightarrow P) \Rightarrow P \rightarrow (P \Rightarrow P) \Rightarrow P$ sending f to $\lambda g.g(f\ g)$. In the must-testing semantics we interpret the error as the *greatest* element $\top$, and $\mathtt{Y} : (P \Rightarrow P) \Rightarrow P$ as the *least* fixed point of F.

4.1 Examples

We give some examples of the continuity and noncontinuity properties of our model.

Noncontinuity. We have shown that the random number generator `rnd` is not continuous wiith respect to must-approximation. The same example suffices to show that its denotation (which we shall write as rnd) is not continuous with respect to extensional order nor the stable order. If we define $f_i : \mathbb{N} \Rightarrow \Sigma$ by $f_i(n) = \top$ if $n < i$ (so $[\![M_i]\!] = f_i$) then $f_i \leq f_{i+1}$ for all i. $\mathsf{rnd}(f_i) = \bot$ for all $i \in \omega$, but $\mathsf{rnd}(\bigvee\{f_i \mid i \in \omega\}) = \mathsf{rnd}(\top) = \top$.

Stable continuity of function application. We now give an example of a least upper bound of a stably-directed set of functions, defined pointwise. Let $g_i : ((\mathbb{N} \Rightarrow \Sigma) \Rightarrow \Sigma) \Rightarrow \Sigma$ be defined: $g_i(h) = h(f_i)$. Then $g_i \leq g_{i+1}$ for all $i \in \omega$ (since $f_i \leq f_{i+1}$), and so we may define the least upper bound $G = \bigvee\{g_i \mid i \in \omega\}$: $G(h) = \top$ if there exists i such that $h(f_i) = \top$. Note that G is distinct from the function $G'(h) = h(\bigsqcup\{f_i \mid i \in \omega\}) = h(\top)$, since $G(\mathsf{rnd}) = \bot$ and $G'(\mathsf{rnd}) = \top$.

Moreover, G is definable in our language — it is the denotation of $\lambda h.(\mathtt{Y}\,\lambda f.\lambda x.h\,(\lambda y.\mathtt{If0}\,(y < x)\ \text{ then } \top \text{ else } \ (f\,y)))\,0$.

Extensional noncontinuity of function application. By contrast, we may observe that the least upper bound of a $\sqsubseteq$-chain of functions may not be determined pointwise. Define $h_i : (\mathbb{N} \Rightarrow \Sigma) \Rightarrow \Sigma$ by $h_i(f) = \bigsqcap_{n \in \omega} f(n+i)$ (i.e. h_i is the denotation of the term $\lambda f.\mathtt{rnd}\,\lambda x.f\ (x + \mathtt{n})$. Then $h_i \sqsubseteq h_{i+1}$ for each i (but $h_i \not\leq h_{i+1}$). The least upper bound of $\{h_i \mid i \in \omega\}$ is $\top$. (To show this, define $k_i : \mathbb{N} \Rightarrow \Sigma$ by $k_i(n) = \bot$ if $i < n$ and $k_i(n) = \top$, otherwise. Then $h_i(k_i) = \top$, and so if H is an upper bound for $\{h_i \mid i \in \omega\}$, $H(k_i) = \top$ for all i. So by stability, $H(\bot) = H(\bigsqcap\{k_i \mid i \in \omega\}) = \bigsqcap\{H(k_i) \mid i \in \omega\} = \top$.) So $(\bigsqcup\{h_i \mid i \in \omega\})(\bot) = \top$, but $\bigsqcup\{h_i(\bot) \mid i \in \omega\} = \bot$.

4.2 Inequational Soundness

Proposition 4. *$M{\Downarrow}^{may}C$ implies $[\![M]\!]_{may} = [\![C]\!]$ and $M{\Downarrow}^{must}C$ implies $[\![M]\!]_{must} = [\![C]\!]$.*

Proof. Both cases are proved by induction on the derivation of $M \Downarrow C$: in the case of must-testing we decorate the judgement $\Downarrow$ with an ordinal (upper) bound on the depth of its derivation, following the schema:

$$\frac{}{M{\Downarrow}^{\lambda}C} \qquad \frac{M{\Downarrow}^{\lambda}\lambda x.M' \qquad M'[N/x]{\Downarrow}^{\kappa}C}{M\,N{\Downarrow}^{\kappa}C}\ \kappa < \lambda \qquad \frac{\forall n{\in}\mathbb{N}.M\,\mathtt{n}{\Downarrow}^{\kappa}\mathbf{e}}{\mathtt{rnd}\,M{\Downarrow}^{\lambda}\mathbf{e}}\ \kappa < \lambda$$

Then if $M \Downarrow C$, $M \Downarrow^{\lambda} C$ for some λ, and we may prove by ordinal induction that if $M \Downarrow^{\lambda} C$ then $[\![M]\!]_{must} = [\![C]\!]$.

Proposition 5 (Adequacy). *$[\![M]\!]_{may} = \bot$ implies $M{\Downarrow}^{may}\mathbf{e}$ and $[\![M]\!]_{must} = \top$ implies $M{\Downarrow}^{must}\mathbf{e}$.*

Proof. The proofs for both models are essentially the same: we sketch the case for must-testing. This uses "approximation relations" in the style of Plotkin [12]: first we define a relation $\lhd_T$ between elements of $[\![T]\!]$ and closed terms of type T for each T:

– $n \lhd_{\mathtt{nat}} M$ if $|M| = \mathtt{n}$.
– $e \lhd_{\mathsf{o}} M$ if $e = \top$ implies $M{\Downarrow}^{must}\mathbf{e}$.
– $f \lhd_{S \Rightarrow T} M$ if $e \lhd_S N$ implies $f(e) \lhd_T M\,N$.

We then define $f : [\![\Gamma]\!] \to [\![T]\!] \lhd_{\Gamma,T} \Gamma \vdash M : T$ if $\Gamma = x_1 : S_1, \ldots, x_n : S_n$ and for all $e_1 \lhd_{S_1} N_1, \ldots, e_n \lhd_{S_n} N_n$ implies $f(e_1, \ldots, e_n) \lhd_T M[N_1/x_1, \ldots, N_n/x_n]$.

We prove that if $\Gamma \vdash M : T$ then $[\![M]\!]_s \lhd_{\Gamma,T} M$ by a standard structural induction. The only potentially problematic case is the fixpoint combinator $\mathtt{Y}$, for which we use the following observations:

For any (closed) $M : T$, the set $\{e \in T \mid e \lhd_T M\}$ is (stably) chain complete, since the least upper bound of a stable chain of functions is determined pointwise. Note also that $e \lhd_P M\,(\mathtt{Y}\,M)$ implies $e \lhd_P \mathtt{Y}\,M$.

To prove $[\![\mathtt{Y}]\!] \lhd_{(P \Rightarrow P) \Rightarrow P} \mathtt{Y}$, we show that $F^\lambda \lhd_{(P \Rightarrow P) \Rightarrow P} \mathtt{Y}$ for all λ by induction on λ. For the induction case, assume $F^\kappa \lhd_{(P \Rightarrow P) \Rightarrow P} \mathtt{Y}$ for all $\kappa < \lambda$, and hence $\bigvee_{\kappa<\lambda} F^\kappa \lhd_{(P \Rightarrow P) \Rightarrow P} \mathtt{Y}$ by stable chain completeness. Suppose $f \lhd_{P \Rightarrow P} M$. Then $F^\lambda(f) = f((\bigvee_{\kappa<\lambda} F^\kappa)(f)) \lhd_P M\,(\mathtt{Y}\,M)$, and so $F^\lambda(f) \lhd_P \mathtt{Y}\,M$ as required.

Corollary 1 (Inequational Soundness). *$[\![M]\!]_{may} \sqsubseteq [\![N]\!]_{may}$ implies $M \lesssim^{may} N$. $[\![M]\!]_{must} \sqsubseteq [\![N]\!]_{must}$ implies $M \lesssim^{must} N$.*

Proof. Suppose e.g. $[\![M]\!]_{must} \sqsubseteq [\![N]\!]_{must}$. Then for any compatible context $C[\_]$, $C[M] \Downarrow$ implies $[\![C[M]]\!]_{must} = \top$ implies $[\![C[N]]\!]_{must} = \top$ implies $C[N] \Downarrow$ as required.

5 Full Abstraction

It remains to prove (inequational) completeness: we shall say that completeness holds at type T if for all closed $M, N : T$, if $M \lesssim^{may} N$ then $[\![M]\!]_{may} \sqsubseteq [\![N]\!]_{may}$ and if $M \lesssim^{must} N$ then $[\![M]\!]_{must} \sqsubseteq [\![N]\!]_{must}$.

So, for instance, completeness holds at $\mathtt{nat}$, since e.g. if $M \lesssim^{may} N$ then $(((\mathtt{Eq}\,M)\,\mathtt{n})\,\mathbf{e})\,\Omega {\Downarrow}^{may}$ implies $(((\mathtt{Eq}\,N)\,\mathtt{n})\,\mathbf{e})\,\Omega {\Downarrow}^{may}$, and hence $[\![M]\!]_{may} = [\![N]\!]_{may}$.

Lemma 5. *Completeness holds at the type* $\mathtt{nat} \Rightarrow \mathtt{o} \Rightarrow \mathtt{o}$.

Proof. Suppose e.g. $M \lesssim^{must} N$. Then by soundness and adequacy, for any $d \in \mathbb{N}$ and $e \in \{\top, \bot\}$ we have $([\![M]\!]_{must}\,d)\,e = \top$ implies $([\![N]\!]_{must}\,d)\,e = \top$, and so $[\![M]\!]_{must} \sqsubseteq [\![N]\!]_{must}$.

We reduce completeness at all pointed types to completeness at $\mathtt{nat} \Rightarrow \mathtt{o} \Rightarrow \mathtt{o}$ using the notion of *definable retraction.*

Definition 5. *Given types S, T, we write $[\![S]\!] \unlhd [\![T]\!]$ (with respect to an interpretation $\mathcal{M}$) if there is a retraction from $[\![S]\!]$ to $[\![T]\!]$ definable in $\mathcal{M}$: i.e. a pair of (closed) terms $(\mathtt{in} : S \Rightarrow T, \mathtt{out} : T \Rightarrow S)$ such that $[\![x : S \vdash \mathtt{out}\ (\mathtt{in}\ x) : S]\!]_{\mathcal{M}} = \mathtt{id}_{[\![S]\!]}$.*

Henceforth, unless noted otherwise, we will take $[\![S]\!] \unlhd [\![T]\!]$ to mean that there is a retraction definable in both may and must interpretations.

For example, we have $\mathbb{N}_* \Rightarrow \mathbb{N}_* \Rightarrow [\![P]\!] \unlhd \mathbb{N}_* \Rightarrow [\![P]\!]$ for any $[\![P]\!]$ via the definable retraction $(\lambda f.\lambda x.((f\ \mathtt{fst}(x))\ \mathtt{snd}(x)), \lambda g.\lambda y.\lambda z.g\ (y * z))$. Note that if $(\mathtt{in}_1, \mathtt{out}_1)$ and $(\mathtt{in}_2, \mathtt{out}_2)$ are definable retractions from $[\![S_1]\!]$ to $[\![S_2]\!]$ and from $[\![T_1]\!]$ to $[\![T_2]\!]$, then $(\lambda f.\lambda x.\mathtt{in}_2(f\ (\mathtt{out}_1\ x)), \lambda f.\lambda x.\mathtt{out}_2(f\ (\mathtt{in}_1\ x)))$ is a definable retraction from $[\![S_1 \Rightarrow T_1]\!]$ to $[\![S_2 \Rightarrow T_2]\!]$.

Let $U = \mathbb{N}_* \Rightarrow \Sigma \Rightarrow \Sigma$ — i.e. $U = [\![\mathtt{nat} \Rightarrow \mathtt{o} \Rightarrow \mathtt{o}]\!]$. We will show that U is *universal* amongst the pointed type-objects — i.e. $[\![P]\!] \trianglelefteq U$ for all pointed types P. This is sufficient to prove completeness at all types.

Lemma 6. *If $[\![S]\!] \trianglelefteq U$ in $\mathcal{M}$ then $\mathcal{M}$ is complete at type S.*

Proof. If $M \lesssim^{\mathcal{M}} N$ then $\mathtt{in}\, M \lesssim^{\mathcal{M}} \mathtt{in}\, N$ and so $[\![\mathtt{in}\, M]\!]_{\mathcal{M}} \sqsubseteq [\![\mathtt{in}\, N]\!]_{\mathcal{M}}$ and so $[\![M]\!]_{\mathcal{M}} = \mathsf{out}[\![\mathtt{in}\, M]\!]_{\mathcal{M}} \sqsubseteq \mathsf{out}[\![\mathtt{in}\, N]\!]_{\mathcal{M}} = [\![N]\!]_{\mathcal{M}}$ as required.

We will use the fact that we may regard elements of $\mathbb{N}_* \Rightarrow \Sigma$ as infinite lists of elements of Σ: for $M : \mathtt{o}, N : \mathtt{nat} \Rightarrow \mathtt{o}$, we define $M :: N : \mathtt{nat} \Rightarrow \mathtt{o} = \lambda x.((\mathtt{If0}\, x)\, M)\, (N\, \mathtt{pred}(x))$, $\mathtt{hd} : (\mathtt{nat} \Rightarrow \mathtt{o}) \Rightarrow \mathtt{o} = \lambda f.f\, \mathtt{0}$ and $\mathtt{tl} : (\mathtt{nat} \Rightarrow \mathtt{o}) \Rightarrow \mathtt{nat} \Rightarrow \mathtt{o} = \lambda f.\lambda x.f\, \mathtt{succ}(x)$. Then $[\![\mathtt{hd}\, (M :: N) = M]\!]$ and $[\![\mathtt{tl}\, (M :: N)]\!] = [\![N]\!]$.

Lemma 7. *$\Sigma \Rightarrow (\mathbb{N}_* \Rightarrow \Sigma) \Rightarrow \Sigma \trianglelefteq (\mathbb{N}_* \Rightarrow \Sigma) \Rightarrow \Sigma$.*

Proof. The retraction is definable via the terms $\mathtt{in} = \lambda f.\lambda g.(f\, (\mathtt{hd}\, g))\, (\mathtt{tl}\, g)$ and $\mathtt{out} = \lambda h.\lambda x.\lambda k.h\, (x :: k)$.

Definition 6. *Given $e \in \mathbb{N}_* \Rightarrow A$, $n \in \mathbb{N}_*$ and $d \in A$ let $e[d]_n \in \mathbb{N}_* \Rightarrow A$ (the "n-insertion" of d into e) be defined:*

- *$e[d]_n(m) = d$ if $n = m$,*
- *$e[d]_n(m) = e(m)$, otherwise.*

For terms $M : \mathtt{nat} \Rightarrow T$, $N : T$ and $t : \mathtt{nat}$, we define the coresponding term $M[N]_t : \mathtt{nat} \Rightarrow T = \lambda x.(((\mathtt{Eq}\, t)\, x)\, (N))\, (M\, x)$. We use insertion to define another key retraction.

Lemma 8. *$(\mathbb{N}_* \Rightarrow \Sigma) \Rightarrow \Sigma \trianglelefteq \mathbb{N}_* \Rightarrow \Sigma \Rightarrow \Sigma$.*

Proof. (Must testing case). Let $\mathsf{in} = \lambda f.\lambda x.\lambda y.f\, \lambda z.(((\mathtt{Eq}\, x)\, z)\, y)\, \mathtt{e}$ and $\mathtt{out} = \lambda f.\lambda g.\mathtt{rnd}\, \lambda x.(f\, x)\, (g\, x)$.

For any $g : \mathbb{N}_* \to \Sigma$, the set $\{\top[g(n)]_n \mid n \in \mathbb{N}_*\}$ is stably bounded above by the constantly $\top$ function, and $g = \bigsqcap\{\top[g(n)]_n \mid n \in \mathbb{N}_*\}$. Thus $(\mathtt{out}\, \mathsf{in}(f))(g) = \bigsqcap\{f(\top[g(n)]_n) \mid n \in \mathbb{N}_*\} = f(\bigsqcap\{\top[g(n)]_n \mid n \in \mathbb{N}_*\} = f(g)$ by stability.

We will now show that $U \Rightarrow \Sigma \trianglelefteq \mathbb{N}_* \Rightarrow \mathbb{N}_* \Rightarrow (\mathbb{N}_* \Rightarrow \Sigma) \Rightarrow \Sigma$, and hence by Lemma 8, $U \Rightarrow \Sigma \trianglelefteq U$. The key to defining this retraction is the *sequentiality* of the function-space $U \Rightarrow \Sigma$.

Definition 7. *Given $f \in (\mathbb{N}_* \Rightarrow A) \Rightarrow \Sigma$, where A is a complete bidomain, we say that f is i-strict if for all $g \in \mathbb{N}_* \Rightarrow A$, $g(i) = \bot$ implies $f(g) = \bot$. We write* $\mathsf{strict}(f)$ *for the set of $i \in \mathbb{N}$ such that f is i-strict.*

Lemma 9 (Sequentiality). *For any complete bidomain A, every $f \in (\mathbb{N}_* \Rightarrow A) \Rightarrow \Sigma$ is constant or i-strict for some i.*

Proof. Note that the set $\{\top[\bot]_i \mid i \in \mathbb{N}\} \subseteq \mathbb{N}_* \Rightarrow A$ is stably bounded above by $\top$. Suppose $f \neq \top$. Then $f(\bot) = f(\bigsqcap\{\top[\bot]_i \mid i \in \mathbb{N}_*\}) = \bot$. So by stability $\bigsqcap\{f(\top[\bot]_i) \mid i \in \mathbb{N}_*\} = \bot$. Hence $f(\top[\bot]_i) = \bot$ for some i, and $g(i) = \bot$ implies $g \sqsubseteq \top[\bot]_i$ and so $f(g) = \bot$ — i.e. f is i-strict as required.

Let $I \in \Sigma \Rightarrow \Sigma$ be the identity function (note that $I \sqsubseteq \top$, but $I \not\leq \top$).

Definition 8. *Given $f \in U \Rightarrow \Sigma$ and $n \in \mathbb{N}_*$, let $f_n = \lambda x.((x\,n)\,(f\,x[I]_n)) \sqcap (f\,x[\top]_n)$.*

Lemma 10. *If f is n-strict then $f = f_n$.*

Proof. Consider $e \in \mathbb{N}_* \Rightarrow \Sigma \Rightarrow \Sigma$.

If $e(n) = \bot$, then by n-strictness of f, $f(e) = \bot$, and $f_n(e) = (\bot\, f(e[I]_n) \sqcap f(e[\top]_n) = \bot$.

If $e(n) = I$, then $e = e[I]_n$, and $f_n(e) = (I\, f(e[I]_n) \sqcap f(e[\top]_n) = f(e) \sqcap f(e[\top]_n) = f(e)$ by monotonicity of f.

If $e(n) = \top$, then $e = e[\top]_n$, and $f_n(e) = (\top\, f(e[I]_n) \sqcap f(e[\top]_n) = \top \sqcap f(e) = f(e)$.

Thus we can represent any (non-constant) $f \in U \Rightarrow \Sigma$ as a strictness index n, together with the two functions $\lambda x.f\,x[I]_n$ and $\lambda x.f\,x[\top]_n$ which (as we shall show) may be computed using a strictly smaller part of their argument.

Lemma 11. *Suppose $e(n)(\top) = e(n)(\bot)$ for all $n \in \mathbb{N}_*$. Then for any $f \in U \Rightarrow \Sigma$, $f(e) = \bigsqcap\{e(n)(\bot) \mid n \in \mathsf{strict}(f)\}$.*

Proof. If $e(n)(\bot) = \bot$ for some $n \in \mathsf{strict}(f)$, then $f(e) = f_n(e) = (e(n)(f(e[I]_n))) \sqcap f(e[\top]_n) = \bot \sqcap f(e[\top]_n) = \bot$. So suppose $e(n)(\bot) = \top$ for all $n \in \mathsf{strict}(f)$. Then $e \sqsupseteq \bigsqcap\{\top[\bot]_n \mid n \notin \mathsf{strict}(f)\}$ and so $f(e) \sqsupseteq f(\bigsqcap\{\top[\bot]_n \mid n \notin \mathsf{strict}(f)\}) = \bigsqcap\{f(\top[\bot]_n) \mid n \notin \mathsf{strict}(f)\} = \top$ by stability.

We also require an "injective pairing" operation on $\mathbb{N}_* \Rightarrow A \Rightarrow \Sigma$, derived from the fact that that $\mathbb{N}_* \Rightarrow A \Rightarrow \Sigma \cong (A \Rightarrow \Sigma)^\omega \cong (A \Rightarrow \Sigma)^\omega \times (A \Rightarrow \Sigma)^\omega$.

Definition 9. *Given $M, N : \mathtt{nat} \Rightarrow T \Rightarrow \mathtt{o}$, let $\langle M, N\rangle : \mathtt{nat} \Rightarrow T \Rightarrow \mathtt{o} = \lambda x.\lambda y.((\mathtt{If0}\,\mathtt{fst}(x))\,((M\,\mathtt{snd}(x))\,y))\,((N\,\mathtt{snd}(x))\,y)$ and $\pi_i : (\mathtt{nat} \Rightarrow T \Rightarrow \mathtt{o}) \Rightarrow \mathtt{nat} \Rightarrow T \Rightarrow \mathtt{o} = \lambda f.\lambda x.\lambda y.(f\ (\mathtt{i} * x))\,y$. Then $\pi_i\,\langle M_0, M_1\rangle = M_i$ for $i \in \{0,1\}$.*

We finally note that $\sqcap$ is definable as erratic binary choice: given $M, N : \mathtt{o}$: $M\,\mathtt{or}\,N : \mathtt{o} = \mathtt{rnd}\,\lambda x.((\mathtt{If0}\,x)\,M)\,N$. So $[\![M\,\mathtt{or}\,N]\!] = [\![M]\!] \sqcap [\![N]\!]$.

We now define the retraction from $U \Rightarrow \Sigma$ to $\mathbb{N}_* \Rightarrow \mathbb{N}_* \Rightarrow (\mathbb{N}_* \Rightarrow \Sigma) \Rightarrow \Sigma$.

Definition 10. $\mathtt{in} : ((\mathtt{nat} \Rightarrow \mathtt{o} \Rightarrow \mathtt{o}) \Rightarrow \mathtt{o}) \Rightarrow \mathtt{nat} \Rightarrow \mathtt{nat} \Rightarrow (\mathtt{nat} \Rightarrow \mathtt{o}) \Rightarrow \mathtt{o} =$

$$\mathtt{Y}\lambda F.\lambda f.\lambda x.((\lambda g.f\ \lambda u.\lambda v.(g\,u)) :: \langle (F\ \lambda z.f\ z[\lambda w.w]_x), F\ \lambda z.f\ z[\lambda w.\mathbf{e}]_x\rangle)$$

and $\mathtt{out} : (\mathtt{nat} \Rightarrow \mathtt{nat} \Rightarrow (\mathtt{nat} \Rightarrow \mathtt{o}) \Rightarrow \mathtt{o}) \Rightarrow (\mathtt{nat} \Rightarrow \mathtt{o} \Rightarrow \mathtt{o}) \Rightarrow \mathtt{o} =$

$$\mathtt{Y}\lambda G.\lambda h.\lambda k.(\mathtt{hd}\,(h\,0))\ \lambda a.(((k\,a)\ (G\,(\pi_0\,(\mathtt{tl}\,(h\,a))))\ k)\mathtt{or}(G\,(\pi_1\,(\mathtt{tl}\,(h\,a)))))\ k)$$

We prove that these terms do indeed define a retraction by an ordinal induction on the unfolding of the fixpoints. For this we require a measure on $f \in U \Rightarrow \Sigma$ of the number of unfoldings required to compute $\mathsf{in}(f)$.

Definition 11. *For each ordinal λ we define the set of λ-*dependent *elements of $U \Rightarrow \Sigma$ inductively, as follows:*

f is λ-dependent if for all n such that f is n-strict there exists $\kappa < \lambda$ such that $\lambda x.f\,x[\lambda w.w]_n$ and $\lambda x.f\,x[\lambda w.\top]_n$ are κ-dependent.

Proposition 6. *If f is λ-dependent then $\mathsf{out}(\mathsf{in}(f)) = f$.*

Proof. By induction on λ. Unfolding the definition of out, we have $\mathsf{out}(\mathsf{in}(f))(d) = (\mathsf{hd}\,(\mathsf{in}(f)\,0))\,\lambda a.(((d\,a)\,(\mathsf{out}(\pi_0\,(\mathsf{tl}\,(\mathsf{in}(f)\,a)))(d))) \sqcap (\mathsf{out}(\pi_1\,(\mathsf{tl}\,(\mathsf{in}(f)\,a)))(d)))$.

Unfolding the recursive definition of in, we have: $\mathsf{in}(f) = \lambda x.((\lambda g.f\,\lambda u.\lambda v.(g\,u)) :: \langle \mathsf{in}(\lambda z.f\,z[\lambda w.w]_x), \mathsf{in}(\lambda z.f\,z[\lambda w.\top]_x\rangle))$.

So $\pi_0(\mathsf{tl}\,(\mathsf{in}(f)\,n)) = \mathsf{in}(\lambda z.f\,z[I]_n)$ and $\pi_1(\mathsf{tl}\,(\mathsf{in}(f)\,n)) = \mathsf{in}(\lambda z.f\,z[\top]_n)$, and $(\mathsf{hd}\,(\mathsf{in}(f)\,0))(e) = f\,\lambda u.\lambda v.(e\,u)$. Since $(\lambda u.\lambda v.(e\,u))(n)(\bot) = (\lambda u.\lambda v.(e\,u))(n)(\top)$ for all n, by Lemma 11 $(\mathsf{hd}\,(\mathsf{in}(f)\,0))(e) = f\,\lambda u.\lambda v.(e\,u) = \bigsqcap\{e(n) \mid n \in \mathsf{strict}(f)\}$.

Substituting these into the expansion of $\mathsf{out}(\mathsf{in}(f))(d)$, we have: $\mathsf{out}(\mathsf{in}(f))(d) = \bigsqcap\{(d(n)\,\mathsf{out}(\mathsf{in}(\lambda z.f\,z[I]_n))(d)) \sqcap \mathsf{out}(\mathsf{in}(\lambda f\,z[\top]_n))(d) \mid n \in \mathsf{strict}(f)\}$.

If $n \in \mathsf{strict}(f)$, then since f is λ-dependent, $\lambda z.f\,z[I]_n$ and $\lambda z.f\,z[\top]_n$ are κ-dependent for some $\kappa < \lambda$ and so by induction hypothesis, $\mathsf{out}(\mathsf{in}(\lambda z.f\,z[I]_n) = \lambda z.f\,z[I]_n$ and $\mathsf{out}(\mathsf{in}(\lambda z.f\,z[\top]_n)) = \lambda z.f\,z[\top]_n$.

So, as required, $\mathsf{out}(\mathsf{in}(f))(d) = \bigsqcap\{(d(n)\,f(d[I]_n) \sqcap f(d[\top]_n) \mid n \in \mathsf{strict}(f)\} = \bigsqcap\{f_n(d) \mid n \in \mathsf{strict}(f)\} = f(d)$ by Lemma 10.

Proposition 7. *Every function $f \in U \Rightarrow \Sigma$ is λ-dependent for some λ.*

Proof. Say that f is n-constant if $f(e[\bot]_n) = f(e[\top]_n)$ for all e. We show by induction on λ that for each $f \in U \Rightarrow \Sigma$ which is not λ-dependent, we can construct a sequence of distinct values $\langle n_\kappa(f) \mid \kappa \leq \lambda\rangle$ such that f is not $n_\kappa(f)$-constant for each $\kappa \leq \lambda$. Since the cardinality of such a sequence must be countable, f must be λ-dependent for some countable λ.

For the induction case, suppose f is not λ-dependent. Then for some $m \in \mathbb{N}_*$ such that f is m-strict, and for some $C \in \{I, \top\}$, $\lambda x.f\,x[C]_m$ is not κ-dependent for all $\kappa < \lambda$. Then $m \neq n_\kappa(\lambda x.f\,x[C]_m)$ for $\kappa < \lambda$, as $\lambda x.f\,x[C]_m$ is m-constant by definition. If f is n-constant, then so is $\lambda x.f\,x[C]_m$, and so f is not $n_\kappa(\lambda x.f\,x[C]_m)$-constant for any $\kappa < \lambda$. Hence we may define $n_\lambda(f) = m$ and $n_\kappa(f) = n_\kappa(\lambda x.f\,x[C]_m)$ for $\kappa < \lambda$.

Combining Propositions 6 and 7, we have shown:

Proposition 8. $U \Rightarrow \Sigma \trianglelefteq \mathbb{N}_* \Rightarrow \mathbb{N}_* \Rightarrow (\mathbb{N}_* \Rightarrow \Sigma) \Rightarrow \Sigma$

By composing definable retractions we may now show that U is a (definably) "reflexive object" (i.e. $U \Rightarrow U$ is a definable retract of U).

Proposition 9. $U \Rightarrow U \trianglelefteq U$.

Proof. We have $U \Rightarrow U \cong \mathbb{N}_* \Rightarrow \Sigma \Rightarrow (U \Rightarrow \Sigma)$
$\trianglelefteq \mathbb{N}_* \Rightarrow \Sigma \Rightarrow (\mathbb{N}_* \Rightarrow \mathbb{N}_* \Rightarrow (\mathbb{N}_* \Rightarrow \Sigma) \Rightarrow \Sigma)$ (by Proposition 8)
$\cong \mathbb{N}_* \Rightarrow \mathbb{N}_* \Rightarrow \mathbb{N}_* \Rightarrow (\Sigma \Rightarrow (\mathbb{N}_* \Rightarrow \Sigma) \Rightarrow \Sigma)$
$\trianglelefteq \mathbb{N}_* \Rightarrow \mathbb{N}_* \Rightarrow \mathbb{N}_* \Rightarrow ((\mathbb{N}_* \Rightarrow \Sigma) \Rightarrow \Sigma)$ (by Lemma 7)
$\trianglelefteq \mathbb{N}_* \Rightarrow \mathbb{N}_* \Rightarrow \mathbb{N}_* \Rightarrow U$ by (Lemma 8)
$\trianglelefteq U$.

Corollary 2. *For every pointed type-object P, $P \trianglelefteq U$.*

Proof. By structural induction on P. Clearly $\Sigma \trianglelefteq U$: if $P = \mathbb{N}_* \Rightarrow Q$ then $P \trianglelefteq \mathbb{N}_* \Rightarrow U \trianglelefteq U$, and if $P = Q_1 \Rightarrow Q_2$ then $P \trianglelefteq U \Rightarrow U \trianglelefteq U$.

Thus we have extended inequational completeness to all types and proved full abstraction.

Theorem 1. *For all terms M, N, $M \lesssim^{may} N$ if and only if $[\![M]\!]_{may} \sqsubseteq [\![N]\!]_{may}$ and $M \lesssim^{must} N$ if and only if $[\![M]\!]_{must} \sqsubseteq [\![N]\!]_{must}$.*

6 Conclusions and Further Directions

For the purposes of exposition we have restricted our attention to a very simple functional language, but bidomains have the potential to model a range of programming languages with non-deterministic features. As we have observed, one possible route to describing more expressive languages is via CPS interpretation. Alternatively, we may interpret lifted types such as call-by-value functions and lifted sums via the *powerdomain* monad: for a pre-bidomain $(D, \sqsubseteq, \leq)$, we define a complete bidomain $\mathcal{P}(D)$ as follows:

- elements are up-closed subsets of D, together with a least element $\bot$,
- $\sqsubseteq$ is the Smyth ordering — i.e. reverse inclusion, with $\bot \sqsubseteq X$ for all X,
- $\leq$ is the intersection of the Smyth and Egli-Milner orders: $X \leq Y$ if $X = \bot$ or $Y \subseteq X$ and for all $x \in X$ there exists $y \in Y$ such that $x \leq y$.

In general, using powerdomains to interpret lifting leads to models with "first-order" control operators (jumps) rather than all first-class continuations (we note that $\mathcal{P}(\mathbb{N}_*) \cong (\mathbb{N}_* \Rightarrow \Sigma) \Rightarrow \Sigma)$. It should also be possible to develop a semantics of recursive types in complete bidomains, based on limits of countable chains of approximants, as investigated in [5].

We have shown that the elements of our model are sequential functions: it would be interesting to relate them to *strategies* in game semantics, in which fully abstract models of functional-imperative languages with *bounded* non-determinism have been described [6]. Comparison with our extensional model may yield an approach to unbounded choice. (As we have suggested, we may interpret bounded choice using Berry's original notion of bidomain (which does require $\sqsubseteq$-continuity). As shown in [4], (Berry's) bidomains have a decomposition into a *bistructure* model of classical linear logic, yielding possible connections between models of concurrency and our semantics of bounded and unbounded non-determinism.

Acknowledgement

The author would like to thank Soren Lassen for comments on a previous version of this article.

References

1. K. R. Apt and G. D. Plotkin. Countable nondeterminism and random assignment. *Journal of the ACM*, 33(4):724–767, 1986.
2. G. Berry. Stable models of typed λ-calculi. In *Proceedings of the 5th International Colloquium on Automata, Languages and Programming*, number 62 in LNCS, pages 72–89. Springer, 1978.
3. R. Cartwright and M. Felleisen. Observable sequentiality and full abstraction. In *Proceedings of POPL '92*, 1992.
4. P.-L. Curien, G. Winskel, and G. Plotkin. Bistructures, bidomains and linear logic. In *Milner Festschrift*. MIT Press, 1997.
5. P. Di Gianantonio, F. Honsell, and G. Plotkin. Uncountable limits and the lambda calculus. *Nordic Journal of Computing*, 2(2):126 – 145, 1995.
6. R. Harmer and G. McCusker. A fully abstract games semantics for finite non-determinism. In *Proceedings of the Fourteenth Annual Symposium on Logic in Computer Science, LICS '99*. IEEE Computer Society Press, 1998.
7. J. Laird. Bistability: an extensional characterization of sequentiality. In *Proceedings of CSL '03*, number 2803 in LNCS. Springer, 2003.
8. J. Laird. Sequentiality in bounded bidomains. *Fundamenta Informaticae*, 65:173 – 191, 2005.
9. S. B. Lassen and C. Pitcher. Similarity and bisimilarity for countable non-determinism and higher-order functions. *Electronic Notes in Theoretical Computer Science*, 10, 1997.
10. P. B. Levy. Infinite trace equivalence. In *Games for Logic and Programming Languages*, pages 195 – 209, 2005.
11. J. Longley. Universal types and what they are good for. In *Domain Theory, Logic and Computation: Proceedings of the 2nd International Symposium on Domain Theory*. Kluwer, 2004.
12. G. Plotkin. Lectures on predomains and partial functions, 1985. Notes for a course given at the Center for the study of Language and Information, Stanford.