Logical Methods in Computer Science
Vol. 3 (2:5) 2007, pp. 1–22
www.lmcs-online.org

Submitted Dec. 20, 2005
Published May 15, 2007

BISTABLE BIORDERS: A SEQUENTIAL DOMAIN THEORY

JAMES LAIRD

Dept. of Informatics, University of Sussex, UK
e-mail address: jiml@sussex.ac.uk

Abstract. We give a simple order-theoretic construction of a Cartesian closed category of sequential functions. It is based on bistable biorders, which are sets with a partial order — the extensional order — and a bistable coherence, which captures equivalence of program behaviour, up to permutation of top (error) and bottom (divergence). We show that monotone and bistable functions (which are required to preserve bistably bounded meets and joins) are strongly sequential, and use this fact to prove universality results for the bistable biorder semantics of the simply-typed lambda-calculus (with atomic constants), and an extension with arithmetic and recursion.

We also construct a bistable model of SPCF, a higher-order functional programming language with non-local control. We use our universality result for the lambda-calculus to show that the semantics of SPCF is fully abstract. We then establish a direct correspondence between bistable functions and sequential algorithms by showing that sequential data structures give rise to bistable biorders, and that each bistable function between such biorders is computed by a sequential algorithm.

1. Introduction

Since its inception, domain theory has been a dominant paradigm in denotational semantics; it is a natural and mathematically rich theory with broad applicability across a wide range of phenomena. However, a limitation of domain theory has been its failure to capture the intensional aspects of computation. The observation of Plotkin [28], that the continuous functional model of PCF is not fully abstract, because it contains functions which are not *sequential*, is symptomatic, but the problem cuts deeper; in the presence of computational effects such as state or concurrency, intensional properties such as the order of computation become critical, and must be captured by some means in any sound model.

Thus, a longstanding problem in domain theory, and the subject of a significant amount of research [2, 12, 4, 5], has been to find a simple characterization of higher-order sequential functions which is wholly extensional in character. Typically, what is sought is some form of mathematical structure, such that all set-theoretic functions which preserve this structure

2000 ACM Subject Classification: F.3.2.

Key words and phrases: Domain Theory, Sequentiality, Functional Programming, Universality, Full Abstraction, Sequential Algorithms.

Research Supported by EPSRC grant S72191.

DOI:10.2168/LMCS-3 (2:5) 2007

are sequential *and* can be used to construct a Cartesian closed category; the basis for a "sequential domain theory".

Clearly, any solution to this problem is dependent on what one means by sequential. It has been closely associated with the full abstraction problem for PCF, although it is now known that PCF sequentiality cannot be characterized effectively in this sense [12, 22].

Another notion of sequentiality — the observably sequential functionals — was discovered by Cartwright and Felleisen [6]. They observed that if one or more errors are added to a functional language, then the order of evaluation of programs becomes observable by varying their inputs. Thus each function corresponds to a unique evaluation tree or sequential algorithm [7], which can be reconstructed from its graph. The observably sequential functionals do form a cartesian closed category, which contains a fully abstract model of SPCF — PCF with errors and a simple control operator. However, the definitions of observably sequential functions and sequential algorithms are based implicitly or explicitly on intensional notions of sequentiality, and hence they cannot offer a characterization of it in the above sense. So we may refine our original problem to ask whether there is a simple, order-theoretic characterization of observable sequentiality.

This paper suggests such a characterization. We will construct a cartesian closed category of biordered sets and order-preserving "bistable" functions. We prove that bistable functions correspond to the observably sequential functions both indirectly — by showing that they may used to give models of observably sequential languages which are *universal* (every element is the denotation of a term) and fully abstract — and directly, by showing that each sequential data structure yields a bistable biorder, and that every bistable and continuous function between such orders is "realized" by a sequential algorithm.

Bistable biorders are analogous to Berry's bidomains [2, 3], which combine the extensional order with the stable order. Although the bidomain model of PCF is not sequential, even at first order types, the bidomain model of *unary* PCF (which contains a $\top$ element at each type) is sequential, and universal [15, 18]. The connection with observably sequential functions is made by viewing top as an error element. Under this interpretation, the monotone and stable functions on bidomains are not observably sequential, because they are not "error-propagating" (i.e. sequential with respect to $\top$ as well as $\bot$). However, the duality between $\bot$ and $\top$ suggests that we "symmetrize" the stable order, to obtain a notion of *bistable* order.

Bistable coherence may be thought of as "behavioural equivalence up to the point of failure — i.e. we may say that M and N are in the bistable order if they are in the extensional order, and M and N perform *the same* computation-steps. M and N are coherent if they behave in a way *except* that M may diverge where N raises an error, or vice-versa

Bistable functions are required to preserve the bistable order, and bistably bounded meets and joins. The proof that bistable functions are sequential is surprisingly simple. Informally, if we have a function which may evaluate two of its components in parallel, we may consider two arguments which are identical except that one diverges in the first component, and produces $\top$ in the second, and the other produces $\top$ in the first argument and diverges in the second. These arguments are bounded in the bistable order: their meet diverges in both components. Our function will produce an error when applied to either argument, but will diverge when applied to their meet, and hence it cannot be bistable.

1.1. **Related Work.** The notion of bistable biorder which is elaborated here was first presented (in a slightly different form) in [14], together with a (different) proof of full abstraction for a model of SPCF. Curien [9], Streicher [30] and Löw [32] have studied bistable functionals, and proved versions of some of the results described here (such as the correspondence between sequential algorithms and bistable functions in [9, 32]). The use of definable retractions to prove definability and full abstraction for observably sequential languages originates with Longley [23, 24]. The concluding section of this paper gives references to more recent work on bidomain models of sequential languages.

1.2. **Outline of the Paper.** In Section 2, we describe the notion of bistable biorder and bistable function, and prove that it yields a Cartesian closed category. We prove that this contains a universal model of the simply-typed λ-calculus $\Lambda^{\top}_{\perp}$ over a single atomic type containing two constants ($\top$ and $\perp$), equivalent to the "minimal model" of $\Lambda^{\top}_{\perp}$ [26]. In Section 3, we develop a notion of complete bistable biorder, or bistable bicpo, and show that we may define a CCC of bicpos and continuous and bistable functions. We give a semantics of SPCF in this category, and prove that it is fully abstract. In section 4 we describe a universal model of a λ-calculus extending $\Lambda^{\top}_{\perp}$ with arithmetic operations and recursion, which may be viewed as a target language for CPS interpretation of observably sequential languages such as SPCF. In Section 5, we investigate the correspondence between sequential algorithms on sequential data structures and bistable functions, showing that each of the latter gives rise to a bistable bicpo, and that each sequential algorithm on the "function-space" computes a bistable function. We then prove that every bistable function is computed in this way, and hence that there is a full embedding of the category of sequential data structures and sequential algorithms in the category of bistable bicpos and bistable and continuous functions.

2. Bistable Biorders

Definition 2.1. A bistable biorder is a tuple $(D, \leq^E, \updownarrow)$, where $(D, \leq^E)$ is a partial order (the *extensional* order), and $\updownarrow$ is an equivalence relation (*bistable coherence*) on D such that each $\updownarrow$-equivalence class is a *distributive* lattice with respect to $\leq^E$, and inclusion into D preserves meets and joins.

Bistable biorders were introduced in [14] as biordered sets (hence the name). In particular, we may define a bistable biorder to be a tuple $\langle D, \leq^E, \leq^B \rangle$, where $(D, \leq^E)$ and $(D, \leq^B)$ are partial orders such that:

- a and b are are bounded above in $\leq^B$ if and only if they are bounded below in $\leq^B$.
- If a and b are bounded above in $\leq^B$ then there are elements $a \wedge b, a \vee b \in D$ which are (respectively) the greatest lower bound and least upper bound of a and b with respect to both orders.
- If $\{a, b, c\}$ is bounded above in $\leq^B$, then $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ (and so $a \wedge (b \vee c) = (a \vee b) \wedge (a \vee c)$.

Proposition 2.2. *The definitions of bistable biorder are equivalent.*

Proof. From the bistable order, we may define the bistable coherence relation: $a \updownarrow b$ if a and b are bounded above in $(D, \leq^B)$. This is an equivalence relation, since if $f, g \leq^B p$ and $g, h \leq^B q$, then $g \leq^B p, q$ and hence p, q are bounded above and thus $f \updownarrow h$.

From the bistable coherence relation, we may define the bistable order $x \leq^B y$ if $x \updownarrow y$ and $x \leq^E y$. □

We shall now construct a Cartesian closed category of bistable biorders and monotone and bistable functions.

Definition 2.3. A function $f : D \to E$ is *monotone* if for all $x, y \in |D|$, $x \leq^E y$ implies $f(x) \leq^E f(y)$ and *bistable* if for each x, $f{\uparrow}[x]_{\updownarrow}$ is a lattice homomorphism into $[f(x)]_{\updownarrow}$ — i.e. for all $x, y \in |D|$ such that $x \updownarrow y$, $f(x) \updownarrow f(y)$, $f(x \wedge y) = f(x) \wedge f(y)$ and $f(x \vee y) = f(x) \vee f(y)$.

We define a category $\mathcal{BBO}$ in which objects are bistable biorders and morphisms are monotone and bistable functions.

Lemma 2.4. *$\mathcal{BBO}$ is bi-Cartesian.*

Proof. The product and co-product operations on bistable orders are defined directly (pointwise):

- $A \times B = (|A| \times |B|, \leq^E_A \times \leq^E_B, \updownarrow_A \times \updownarrow_B)$,
- $A + B = (|A| + |B|, \leq^E_A + \leq^E_B, \updownarrow_A + \updownarrow_B)$.

The unit for the product is the one-point biorder, **1** and the unit for the co-product is the empty biorder. □

We will now show that $\mathcal{BBO}$ is Cartesian closed by defining an exponential: a bistable biorder of functions, in which the extensional order is standard, and the bistable order is a symmetric version of the stable order.

Definition 2.5. Given bistable biorders D, E, we define the function-space $D \Rightarrow E$ to be the set of monotone and bistable functions from D to E, with

- $f \leq^E g$ if for all $x \in D$, $f(x) \leq^E g(x)$,
- $f \updownarrow g$ if for all $x \in D$ $f(x) \updownarrow g(x)$, and if $x \updownarrow y$ (and hence $f(y) \updownarrow g(x)$) then $f(x) \wedge g(y) = f(y) \wedge g(x)$ and $f(x) \vee g(y) = f(y) \vee g(x)$.

Lemma 2.6. *$D \Rightarrow E$ is a bistable biorder.*

Proof. If $f \updownarrow g$ then $f(a) \updownarrow g(a)$ for all a, and so we may define $\leq^E$ meets and joins $f \wedge g$ and $f \vee g$ pointwise:
$(f \wedge g)(a) = f(a) \wedge g(a)$ and $(f \vee g)(a) = f(a) \vee g(a)$.
We now show that $f \wedge g$ and $f \vee g$ are monotone and bistable functions — e.g. if $a \updownarrow b$ then $(f \wedge g)(a \vee b) = (f \wedge g)(a) \vee (f \wedge g)(b)$. Observe that $f(a) \wedge g(b) = g(a) \wedge f(b) \leq^E f(a), f(b), g(a), g(b)$ and so $f(a) \wedge g(b), f(b) \wedge g(a) \leq^E f(a) \wedge f(b), g(a) \wedge g(b)$. Hence:
$(f \wedge g)(a \vee b) = f(a \vee b) \wedge g(a \vee b) = (f(a) \vee f(b)) \wedge (g(a) \vee g(b)) = (f(a) \wedge g(a)) \vee (f(a) \vee g(b)) \vee (f(b) \wedge g(a)) \vee (f(b) \wedge g(b)) = (f(a) \wedge g(a)) \vee (f(b) \wedge g(b)) = (f \wedge g)(a) \vee (f \wedge g)(b)$.

Next, we show that $f \updownarrow f \wedge g$ and $f \updownarrow f \vee g$:
For all x, $f(x) \updownarrow (f \wedge g)(x) = f(x) \wedge g(x)$, and for all y such that $x \updownarrow y$, $f(x) \wedge (f \wedge g)(y) = f(x) \wedge f(y) \wedge g(y) = f(y) \wedge f(x) \wedge g(x) = f(y) \wedge (f \wedge g)(x)$ and
$f(x) \vee (f \wedge g)(y) = f(x) \vee (f(y) \wedge g(y)) = (f(x) \vee f(y)) \wedge (f(x) \vee g(y)) = (f(y) \vee f(x)) \wedge (f(y) \vee g(x)) = f(y) \vee (f \wedge g)(x)$.

Finally, we need to prove that $\updownarrow$ is transitive, for example, suppose $f \updownarrow g$ and $g \updownarrow h$. Suppose $x \updownarrow y$. Then:
$f(x) \wedge h(y) = f(x) \wedge h(y) \wedge (f(x) \vee g(y)) = f(x) \wedge h(y) \wedge (f(y) \vee g(x))$

$= (f(x) \wedge h(y) \wedge f(y)) \vee (f(x) \wedge h(y) \wedge g(x))$
$\leq^E f(y) \vee (f(x) \wedge g(y) \wedge h(x)) = f(y) \vee (f(y) \wedge g(x) \wedge h(x)) = f(y)$.
Similarly, $f(x) \wedge h(y) \sqsubseteq h(x)$, $f(y) \wedge h(x) \sqsubseteq f(y)$ and $f(y) \wedge h(x) \sqsubseteq h(x)$. So $f(x) \wedge h(y) = f(y) \wedge h(x)$. By duality, $f(x) \vee h(y) = f(y) \vee h(x)$ and so $f \updownarrow h$ as required. □

Proposition 2.7. $(\mathcal{BBO}, \mathbf{1}, \times)$ *is cartesian closed.*

Proof. We need to show that the natural bijection taking $f : A \times B \to C$ to $\Lambda(f) : A \to (B \Rightarrow C)$ such that $\Lambda(f)(a)(b) = f(\langle a, b\rangle)$, and its inverse, are well-defined on bistable biorders and bistable functions. This is similar to the proof for (stable) biorders and stable and monotone functions [3].

For example, to show that $\Lambda(f)$ preserves bistable coherence:
Suppose $a \updownarrow_A a'$. Then for all $b \updownarrow_B b'$, $\langle a, b\rangle \updownarrow \langle a', b'\rangle$, and e.g. $\Lambda(f)(a)(b) \wedge \Lambda(f)(a')(b') = f(\langle a, b\rangle \wedge \langle a', b'\rangle) = f(\langle a, b'\rangle) \wedge f(\langle a', b\rangle) = \Lambda(f)(a)(b') \wedge \Lambda(f)(a')(b)$. Similarly, $\Lambda(f)(a)(b) \vee \Lambda(f)(a')(b') = \Lambda(f)(a)(b') \vee \Lambda(f)(a')(b)$ and hence $\Lambda(f)(a) \updownarrow \Lambda(f)(b')$ as required.

Conversely, to show that if $g : A \to (B \Rightarrow C)$ is bistable, then $\Lambda^{-1}(g)$ is bistable, suppose $\langle a, b\rangle \updownarrow_{A\times B} \langle a', b'\rangle$. Then $a \updownarrow a'$ and $b \updownarrow b'$ and by bistability of g, $g(a)(b) \wedge g(a')(b') = g(a)(b') \wedge g(a')(b)$ and $g(a)(b) \vee g(a')(b') = g(a)(b') \vee g(a')(b)$. So e.g. $\Lambda^{-1}(g)(\langle a, b\rangle \wedge \langle a', b'\rangle) = g(a \wedge a')(b \wedge b') = g(a)(b) \wedge g(a)(b') \wedge g(a')(b) \wedge g(a')(b') = g(a)(b) \wedge g(a')(b') = \Lambda^{-1}(g)(\langle a, b\rangle) \wedge \Lambda^{-1}(g)(\langle a', b'\rangle)$. □

A bistable biorder D is *pointed* if $(D, \leq^E)$ has a least element $\bot$ and a greatest element $\top$, such that $\bot \updownarrow \top$. A monotone bistable function f of pointed biorders is *bistrict* if it preserves the meet and join of the empty set — i.e. $f(\top) = \top$ and $f(\bot) = \bot$. We define the category $\mathcal{BBO}_s$ of pointed bistable biorders and strict, monotone and bistable functions.

Proposition 2.8. *The inclusion of $\mathcal{BBO}_s$ into $\mathcal{BBO}$ has a left adjoint.*

Proof. The *bilifting* operation takes a bistable biorder A to a pointed bistable biorder by adding *two* new points, $\top$ and $\bot$: $A^\top_\bot = (A \times \{*\}) \cup \{\bot, \top\}$, where:
- $x \leq^E y$ if $x = \bot$ or $y = \top$, or $x = \langle x', *\rangle, y = \langle y', *\rangle$ and $x' \leq^E y'$,
- $x \updownarrow y$ if $x, y \in \{\bot, \top\}$ or $x = \langle x', *\rangle, y = \langle y', *\rangle$ and $x' \updownarrow y'$.

For any pointed B, $\mathcal{BBO}(A, B) \cong \mathcal{BBO}_s(A^\top_\bot, B)$. □

2.1. **First-Order Sequentiality and Universality.** A key step in proving universality for observably sequential languages is the observation that the monotone and bistable functions on pointed bistable biorders are *bisequential* (i.e. sequential with respect to both $\bot$ and $\top$ elements).

Definition 2.9. Given pointed bistable biorders $A_1, \ldots, A_n, B$, a function $f : A_1 \times \ldots \times A_n \to B$ is i-strict if $\pi_i(x) = \bot$ implies $f(x) = \bot$ and $\pi_i(x) = \top$ implies $f(x) = \top$.

Lemma 2.10. *Given pointed bistable biorders $A_1, \ldots, A_n$, every strict, monotone and bistable function $f : A_1 \times \ldots \times A_n \to \Sigma$ is i-strict for some $i \leq n$.*

Proof. Given $j \leq n$, let $\bot[\top]_j = \langle x_i \mid i \leq n\rangle$, where $x_i = \top$ if $i = j$, and $x_i = \bot$ otherwise. Similarly $\top[\bot]_j = \langle x_i \mid i \leq n\rangle$, where $x_i = \bot$ if $i = j$, and $x_i = \top$ otherwise.

If $\pi_i(x) = \bot$ then $x \leq^E \top[\bot]_i$, and if $\pi_i(x) = \top$, $\bot[\top]_i \leq^E x$. Thus f is i-strict if $f(\top[\bot]_i) = \bot$ and $f(\bot[\top]_i) = \top$. Since $\bot \leq^B \top$, we have $\top[\bot]_j \updownarrow \top[\bot]_k$ for all $j, k \leq n$, and $\bigwedge_{i\leq n} \top[\bot]_i = \bot$. Hence $\bigwedge_{i\leq n} f(\top[\bot]_j) = f(\bigwedge_{i\leq n} \top[\bot]_i) = f(\bot) = \bot$, and so for some

i, $f(\top[\bot]_i) = \bot$. Similarly $f(\bigvee_{i\in I}(\bot[\top]_i)) = \top$, and so $f(\bot[\top]_j) = \top$ for some j. Moreover, if $i \neq j$, then $\bot[\top]_j \leq^E \top[\bot]_i$, and so either $\bot[\top]_j = \top[\bot]_i$ — in which case each A_k is the one-point order — or else $i = j$ as required, and hence i is unique — i.e. bisequential functions are *strongly sequential*. □

2.2. **Universality for $\Lambda^\top_\bot$.** Let $\Lambda^\top_\bot$ be the simply-typed λ-calculus with products, over a single base type Σ containing the constants $\top$ and $\bot$. The "minimal" model of this language (that is, the model inducing the maximal consistent theory containing β and η) was shown to be effectively presentable by Padovani [26] using an analysis of the syntax. By Cartesian closure of $\mathcal{BBO}$, we obtain a model of $\Lambda^\top_\bot$ in which each type is interpreted as the corresponding bistable biorder. We will show that this is the minimal model.

For each type S of $\Lambda^\top_\bot$, an element of the corresponding biorder is *definable* if it is the denotation of a closed term of type S. Universality holds at S if every element of S is definable. Universality at *first-order* function types is a consequence of sequentiality.

Lemma 2.11. *The bistable model of $\Lambda^\top_\bot$ is universal at all types of the form $\Sigma^n \Rightarrow \Sigma^m$.*

Proof. Suppose $m = 1$. If f is constant ($\top$ or $\bot$), then f is definable. Otherwise, f is strict, and hence for some i, f is i-strict — i.e. $f = \pi_i$, and is therefore definable. If $m > 1$, we have $f = \langle f; \pi_i \mid i \leq m\rangle$, and $f;\pi_i$ is definable for each i and so f is definable.

□

We will now prove that universality at higher-order types *reduces* to universality at first-order, using the notion of *definable retraction*.

Definition 2.12. Given types S, T, a definable retraction from S to T (which we may write $\mathsf{inj} : S \trianglelefteq T : \mathsf{proj}$ or just $S \trianglelefteq T$) is a pair of terms: $\mathsf{inj} : S \Rightarrow T$ and $\mathsf{proj} : T \Rightarrow S$ which denote a retraction in $\mathcal{BBO}$ (i.e. $[\![\mathsf{inj}]\!];[\![\mathsf{proj}]\!] = \mathsf{id}_S$).

Lemma 2.13. *If universality holds at type T, and $\mathsf{inj} : S \trianglelefteq T : \mathsf{proj}$, then universality holds at type S.*

Proof. Given an element $e \in S$, we have a term $M : T$ such that $[\![M]\!] = e;\mathsf{inj}$ and thus $[\![\mathsf{proj}\, M]\!] = e;\mathsf{inj};\mathsf{proj} = e$. □

So we can prove universality for $\Lambda^\top_\bot$ by showing that every $\Lambda^\top_\bot$ type is a definable retract of a first order type. To do so, we require a few simple facts about definable retractions.

Lemma 2.14. *If $\mathsf{inj}_T : T_1 \trianglelefteq T_2 : \mathsf{proj}_T$, and $\mathsf{inj}_S : S_1 \trianglelefteq S_2 : \mathsf{proj}_S$, then $S_1 \Rightarrow T_1 \trianglelefteq S_2 \Rightarrow T_2$ and $S_1 \times T_1 \trianglelefteq S_2 \times T_2$.*

Proof. We have, for example, $\lambda f x.\mathsf{inj}_T\,(f\,(\mathsf{proj}_S\,x)) : S_1 \Rightarrow T_1 \trianglelefteq S_2 \Rightarrow T_2 : \lambda f x.\mathsf{proj}_T\,(f\,(\mathsf{inj}_S x))$. □

The key to reducing the order of the function-space is the fact that for any n, $(\Sigma^n \Rightarrow \Sigma) \Rightarrow \Sigma$ is a definable retract of $(\Sigma \Rightarrow \Sigma) \times \Sigma^n$.

Lemma 2.15. *If $f : (\Sigma^n \Rightarrow \Sigma) \to \Sigma$ is a strict bistable function, then for all $e \in \Sigma^n \Rightarrow \Sigma$, $f\,e = e\,\langle f\,\pi_i \mid 1 \leq i \leq n\rangle$.*

Proof. If $e = \top$ then $f\,e = \top$ by strictness of f, and $e\,\langle f(\pi_i) \mid 1 \leq i \leq n\rangle = \top$. Similarly, if $e = \bot$ then $f\,e = e\,\langle f\,\pi_i \mid 1 \leq i \leq n\rangle = \bot$. Otherwise, $e = \pi_i$ for some $1 \leq i \leq n$, and $e\,\langle f\,\pi_i \mid 1 \leq i \leq n\rangle = f\,\pi_i = f\,e$ as required. □

Lemma 2.16. *Let* $g = \lambda x.\lambda y.(x\,\lambda z.(y\,\langle x\,\pi_i \mid 1 \leq i \leq n\rangle))$[1]. *Then* $g = \mathsf{id}_{(\Sigma^n \Rightarrow \Sigma)\Rightarrow\Sigma}$

Proof. We show that for any element $f \in (\Sigma^n \Rightarrow \Sigma) \Rightarrow \Sigma$, $g(f) = f$. We first note that $g(\top) = \top$ and $g(\bot) = \bot$.

If $f \neq \top$ and $f \neq \bot$, then f is strict (since $f(\bot) \leq^E f(\top)$). Given $e \in \Sigma^n \Rightarrow \Sigma$, suppose $f(e) = \top$, then $(g\,f)(e) = f(\lambda z.(e\,\langle f\,\pi_i \mid 1 \leq i \leq n\rangle)) = f\,(\lambda z.f(e)) = f(\lambda z.\top) = \top = f(e)$, by Lemma 2.15 and strictness of f, and similarly if $f(e) = \bot$, then $(g\,f)(e) = \bot$. Hence $g(f) = f$ as required. □

Lemma 2.17. *For any* $n \geq 1$, $(\Sigma^n \Rightarrow \Sigma) \Rightarrow \Sigma$ *is a definable retract of* $(\Sigma \Rightarrow \Sigma) \times \Sigma^n$ *in* $\mathcal{BBO}$.

Proof. Consider the terms

$$\mathsf{inj} : ((\Sigma^n \Rightarrow \Sigma) \Rightarrow \Sigma) \Rightarrow ((\Sigma \Rightarrow \Sigma) \times \Sigma^i) = \lambda f.\langle \lambda x.(f\,\lambda y.x), \langle f\,\pi_i \mid 1 \leq i \leq n\rangle\rangle$$

$$\mathsf{proj} : ((\Sigma \Rightarrow \Sigma) \times \Sigma^n) \Rightarrow (\Sigma^n \Rightarrow \Sigma) \Rightarrow \Sigma = \lambda x.\lambda g.(\pi_1(x)\,(g\,\pi_2(x)))$$

We have $\lambda x.\mathsf{proj}\,(\mathsf{inj}\,x) =_{\beta\pi} \lambda x.\lambda y.x\,\lambda z.y\,\langle x\,\pi_i \mid 1 \leq i \leq n\rangle$, and hence by Lemma 2.16, $[\![\lambda x.\mathsf{proj}\,(\mathsf{inj}\,x)]\!] = \mathsf{id}_{(\Sigma^n\Rightarrow\Sigma)\Rightarrow\Sigma}$. □

Lemma 2.18. *For any* $n, m \geq 1$, $(\Sigma^n \Rightarrow \Sigma^m) \Rightarrow \Sigma$ *is a definable retract of* $\Sigma^{n+m} \Rightarrow \Sigma^{(2n)^m}$.

Proof. By induction on m. For the base case ($m = 1$), we have $(\Sigma^n \Rightarrow \Sigma) \Rightarrow \Sigma \trianglelefteq (\Sigma \Rightarrow \Sigma)\times\Sigma^n$ by Lemma 2.17, and since $\Sigma \Rightarrow \Sigma \trianglelefteq \Sigma^n + 1 \Rightarrow \Sigma$, and $\Sigma \trianglelefteq \Sigma^n \Rightarrow \Sigma$, so $\Sigma^n \trianglelefteq (\Sigma \Rightarrow \Sigma)^n$, we have $(\Sigma^n \Rightarrow \Sigma^m) \Rightarrow \Sigma \trianglelefteq ((\Sigma^{n+1} \Rightarrow \Sigma)^n)^2 \cong ((\Sigma^{n+1} \Rightarrow \Sigma)^{2n^m}$. For the induction case,
$(\Sigma^n \Rightarrow \Sigma^{m+1}) \Rightarrow \Sigma \cong (\Sigma^n \Rightarrow \Sigma^m) \Rightarrow ((\Sigma^n \Rightarrow \Sigma) \Rightarrow \Sigma)$
$\trianglelefteq (\Sigma^n \Rightarrow \Sigma^m) \Rightarrow ((\Sigma \Rightarrow \Sigma) \times \Sigma^n)$ by Lemma 2.17.
$\cong (\Sigma \Rightarrow (\Sigma^n \Rightarrow \Sigma^m) \Rightarrow \Sigma) \times ((\Sigma^n \Rightarrow \Sigma^m) \Rightarrow \Sigma)^n$
$\trianglelefteq (\Sigma \Rightarrow (\Sigma^{n+m} \Rightarrow \Sigma^{(2n)^m})) \times (\Sigma^{n+m} \Rightarrow \Sigma^{(2n)^m})^n$ by induction hypothesis
$\trianglelefteq (\Sigma^{n+m+1} \Rightarrow \Sigma^{(2n)^m})^n \times (\Sigma^{n+m+1} \Rightarrow \Sigma^{(2n)^m})^n$
$\trianglelefteq \Sigma^{n+m+1} \Rightarrow \Sigma^{(2n)^m \cdot 2n} \cong \Sigma^{n+m+1} \Rightarrow \Sigma^{(2n)^{m+1}}$ as required. □

Lemma 2.19. *For any type* T *there exists* $n(T), m(T) \in \mathbb{N}$ *such that* T *is a definable retract of* $\Sigma^{n(T)} \Rightarrow \Sigma^{m(T)}$.

Proof. is by induction on type structure. For the induction cases: $S \times T \trianglelefteq (\Sigma^{n(S)} \Rightarrow \Sigma^{m(S)}) \times (\Sigma^{n(T)} \Rightarrow \Sigma^{m(T)}) \trianglelefteq \Sigma^{\max\{n(S),n(T)\}} \Rightarrow \Sigma^{m(S)+m(T)}$.
$S \Rightarrow T \trianglelefteq (\Sigma^{n(S)} \Rightarrow \Sigma^{m(S)}) \Rightarrow (\Sigma^{n(T)} \Rightarrow \Sigma^{m(T)})$
$\cong (\Sigma^{n(T)} \Rightarrow (\Sigma^{n(S)} \Rightarrow \Sigma^{m(S)}) \Rightarrow \Sigma)^{m(T)}$
$\trianglelefteq (\Sigma^{n(T)} \Rightarrow (\Sigma^{n(S)+m(S)} \Rightarrow \Sigma^{(2n(S))^{m(S)}})^{m(T)}$
$\cong \Sigma^{n(T)+n(S)+m(S)} \Rightarrow \Sigma^{(2n(S))^{m(S)}\cdot m(T)}$. □

By applying Lemma 2.13 to Lemmas 2.11 and 2.19 we have established:

Theorem 2.20. *The bistable model of* $\Lambda^{\top}_{\bot}$ *is universal at all types.*

Corollary 2.21. *The bistable model is minimal.*

Proof. It is straightfoward to use universality to show that if $[\![M]\!] \neq [\![N]\!]$, then there is an (applicative) context such that $C[M] =_{\beta\eta\pi} \top$ and $C[N] =_{\beta\eta\pi} \bot$, or vice-versa. Hence any compatible theory containing $\beta\eta\pi$ as well as $M = N$ also contains $\bot = \top$. □

[1]Here we are using λ-calculus notation to describe an element of $\mathcal{BBO}$.

Our proof also yields a solution to a related problem: to give a simple axiomatization of the theory of the minimal model.

Definition 2.22. Let the theory $=^{\top}_{\bot}$ over the terms of $\Lambda^{\top}_{\bot}$ be the compatible, symmetric and transitive closure of $\beta\eta\pi$-equivalence extended with the axioms $f : (\Sigma^n \Rightarrow \Sigma) \Rightarrow \Sigma = \lambda h.f\ \lambda x.h\ \langle f\ \pi_i \mid 1 \leq i \leq n\rangle$ for each n.

For each type T, we have a definable retraction $\mathsf{inj}_T : T \trianglelefteq \Sigma^{n(T)} \Rightarrow \Sigma^{m(T)} : \mathsf{proj}_T$.

Lemma 2.23. $\mathsf{proj}_T\ (\mathsf{inj}_T\ x) =^{\top}_{\bot} \lambda x.x$.

Proof. This is by induction on T, following the definition of proj_T and inj_T, since to prove that they define a retraction in $\mathcal{BBO}$, we used only standard properties of all CCCs (i.e. $\beta\eta\pi$-equivalence) together with $(\Sigma^n \Rightarrow \Sigma) \Rightarrow \Sigma \trianglelefteq (\Sigma \Rightarrow \Sigma) \times \Sigma^n$. □

Proposition 2.24. *$M : T =^{\top}_{\bot} N : T$ if and only if $[\![M]\!] = [\![N]\!]$.*

Proof. From left-to-right, this follows from the soundness of the theory $=^{\top}_{\bot}$ in the bistable model of $\Lambda^{\top}_{\bot}$.

To prove the converse, suppose $[\![M]\!] = [\![N]\!]$. Then $[\![\mathsf{inj}\ M]\!] \in \Sigma^{n(T)} \Rightarrow \Sigma^{m(T)} = [\![\mathsf{inj}\ N]\!] \in \Sigma^{n(T)} \Rightarrow \Sigma^{m(T)}$. Hence for each $j \leq m(T)$, the terms $\lambda x.\pi_j((\mathsf{inj}\ M)\ x)$ and $\lambda x.\pi_j((\mathsf{inj}\ N)\ x)$ have the same *head-normal form* (i.e. $\lambda x.\top$, $\lambda x.\bot$ or $\lambda x.\pi_i\ x$ for some $1 \leq i \leq n(T)$). Thus $\mathsf{inj}\ M =^{\top}_{\bot} \mathsf{inj}\ N$ and so $M =^{\top}_{\bot} \mathsf{proj}\ (\mathsf{inj}\ M) =^{\top}_{\bot} \mathsf{proj}\ (\mathsf{inj}\ N) =^{\top}_{\bot} N$ as required. □

3. Bistable bicpos

We shall now extend our notion of bistable biorder with notions of completeness and continuity.

Definition 3.1. Given $\leq^E$-directed sets X, Y, we say that $X \updownarrow Y$ if for all $x \in X$ and $y \in Y$ there exists $x' \in X$ and $y' \in Y$ such that $x \leq^E x'$, $y \leq^E y'$ and $x' \updownarrow y'$. A *bistable bicpo* is a bistable biorder D such that $(|D|, \leq^E)$ is a cpo and if $X \updownarrow Y$ then $\bigsqcup X \updownarrow \bigsqcup Y$ and $\bigsqcup X \wedge \bigsqcup Y = \bigsqcup\{x \wedge y \mid x \in X \wedge y \in Y \wedge x \updownarrow y\}$

Let $\mathcal{BBC}$ be the category of bistable bicpos and continuous and bistable functions.

Proposition 3.2. *$(\mathcal{BBC}, \mathbf{1}, \times)$ is Cartesian closed.*

Proof. We show that for any directed set F of functions from A to B, a bistable and continuous least upper bound can be defined pointwise — $(\bigsqcup F)(a) = \bigsqcup F(a)$, where $F(a) = \{f(a) \mid f \in F\}$.

$\bigsqcup F$ is bistable: if $a \updownarrow b$, then we have $F(a) \updownarrow F(b)$ and hence $(\bigsqcup F)(a) \updownarrow (\bigsqcup F)(b)$, and $(\bigsqcup F)(a \vee b) = \bigsqcup\{f(a) \vee f(b) \mid f \in F\} \leq^E (\bigsqcup F)(a) \vee (\bigsqcup F)(b)$

To show preservation of glbs, we note that $\bigsqcup\{f(a) \wedge g(b) \mid f, g \in F \wedge f(a) \updownarrow g(b)\} = \bigsqcup\{f(a) \wedge f(b) \mid f \in F\}$ by directedness of F (for any f, g such that $f(a) \updownarrow g(b)$, we choose h such that $f, g \leq^E h$ and hence $f(a) \wedge g(b) \leq^E h(a) \wedge h(b)$. Thus $(\bigsqcup F)(a) \wedge (\bigsqcup F)(b) = \bigsqcup\{f(a) \wedge g(b) \mid f, g \in F \wedge f(a) \updownarrow g(b)\} = \bigsqcup\{f(a) \wedge f(b) \mid f \in F\} = (\bigsqcup F)(a \wedge b)$.

Now given directed sets of bistable functions F, G such that $F \updownarrow G$

$\bigsqcup F \updownarrow \bigsqcup G$**:** For all x, $F(x) \updownarrow G(x)$, and hence $(\bigsqcup F)(x) \updownarrow (\bigsqcup G)(x)$. Now suppose $x \updownarrow y$ — we need to show that $(\bigsqcup F)(x) \wedge (\bigsqcup G)(y) = (\bigsqcup F)(y) \wedge (\bigsqcup G)(x)$. By symmetry it suffices to show $(\bigsqcup F)(x) \wedge (\bigsqcup G)(y) = \bigsqcup\{f(x) \wedge g(y) \mid f \in F \wedge g \in$

$G \wedge f(x) \updownarrow g(y)\} \leq^E (\bigsqcup F)(y)$. Given $f \in F$ and $g \in G$ such that $f(x) \updownarrow g(y)$, there exists $f' \in F$ and $g' \in G$ such that $f \leq^E f', g \leq^E g'$ and $f' \updownarrow g'$. Hence $f(x) \wedge g(y) \leq^E f'(x) \wedge g'(y) \leq^E f(y)$ and so $f(x) \updownarrow g(y) \leq^E (\bigsqcup F)(y)$ as required.

$\bigsqcup F \wedge \bigsqcup G = \bigsqcup\{f \wedge g \mid f \updownarrow g\}$**:** For all x, $(\bigsqcup F \wedge \bigsqcup G)(x) = (\bigsqcup F)(x) \wedge (\bigsqcup G)(x) = \bigsqcup\{f(x) \wedge g(x) \mid f(x) \updownarrow g(x)\} = \bigsqcup\{f(x) \wedge g(x) \mid f \updownarrow g\}$, since for any f, g such that $f(x) \updownarrow g(x)$ there exists $f' \in F, g' \in G$ such that $f' \updownarrow g'$ and $f \leq^E f'$, $g \leq^E g'$, and so $f(x) \wedge g(x) \leq^E f'(x) \wedge g'(x)$.

□

The bistable bicpos are also closed under the lifting and coproduct operations.

3.1. **SPCF.** We have defined a cpo-enriched Cartesian closed category of sequential functionals, in which we may interpret PCF. We will now show that we have a fully abstract semantics of SPCF [6] — PCF with a non-local control operator — `catch` — and an "error", $\top$. Thus we may connect our bistable semantics of $\Lambda^\top_\perp$ to the "original" observably sequential language, SPCF. In doing so, we establish indirectly the correspondence between observably sequential functionals and bistable functionals, since both yield fully abstract imodels of SPCF. In the case of the bistable model, our proof of universality for $\Lambda^\top_\perp$ gives an easy proof of full abstraction, since every SPCF type-object is a limit for a chain of $\Lambda^\top_\perp$ types.

The types of SPCF are given by the following grammar:

$$S, T ::= \Sigma \mid \mathtt{nat} \mid S \Rightarrow T \mid S \times T$$

Terms are obtained by extending the simply-typed λ-calculus with pairing and projection and the following constants:

Divergence and Error: $\top, \perp : \Sigma$,
Numerals: $\mathtt{0} : \mathtt{nat}$, $\mathtt{succ}, \mathtt{pred} : \mathtt{nat} \Rightarrow \mathtt{nat}$,
Conditionals: $\mathtt{IF0} : \mathtt{nat} \Rightarrow (T \times T) \Rightarrow T$, where $T \in \{\Sigma, \mathtt{nat}\}$,
Fixpoints: $\mathbf{Y} : (T \Rightarrow T) \Rightarrow T$
Control: $\mathtt{catch}_n : (\Sigma^n \Rightarrow \Sigma) \Rightarrow \mathtt{nat}$

The control operator `catch` is a basic form of Cartwright and Felleisen's `catch` [6]; it sends i-strict functions (ith-projection) to `i`. Despite its simplicity, it can be used to derive (call-by-name versions of) control operators such as Felleisen's idealized call-with-current-continuation operator $\mathcal{C} : ((\mathtt{nat} \Rightarrow \Sigma) \Rightarrow \Sigma) \Rightarrow \mathtt{nat}$ [10]:

$$\mathcal{C} \equiv \mathbf{Y}\lambda f.\lambda g.((\mathtt{IF0}\,(\mathtt{catch}_2\,\lambda x.g\,\lambda y.(\mathtt{IF0}\,y)\,x))\,\langle 0, \mathtt{succ}\,(f\,\lambda h.g\,(\lambda z.h\,(\mathtt{pred}\,z)))\rangle)$$

(So $\mathtt{catch}_2$ is sufficient to express $\mathtt{catch}_n$ for any n.)

We may give a simple operational semantics for SPCF programs — *closed terms of type* Σ — using *evaluation contexts*,

Definition 3.3. Evaluation contexts of SPCF are given by the following grammar:

$$E[\cdot] ::= [\cdot] \mid E[\cdot]\,M \mid \mathtt{IF0}\,E[\cdot] \mid \pi_i\,E[\cdot] \mid \mathtt{succ}\,E[\cdot] \mid \mathtt{pred}\,E[\cdot]$$

The "small-step" operational semantics of SPCF programs is given in Table 1. The rule for `catch` makes its connection with control operators such as `callcc` apparent; the current continuation (represented as a tuple of evaluation contexts filled with the possible values for $\mathtt{catch}\,M$) is passed as an argument to M. For a program M we write $M \Downarrow$ if $M \twoheadrightarrow \top$. We adopt a standard definition of observational approximation and equivalence:

$$E[\top] \longrightarrow \top$$
$$E[(\lambda x.M)\, N] \longrightarrow E[M[N/x]]$$
$$E[\pi_i \langle M_1, M_2\rangle] \longrightarrow E[M_i]$$
$$E[\mathtt{pred}\,(\mathtt{succ}\,\mathtt{n})] \longrightarrow E[\mathtt{n}]$$
$$E[\mathtt{IF0}\,\mathtt{0}] \longrightarrow E[\pi_1]$$
$$E[\mathtt{IF0}\,(\mathtt{succ}\,\mathtt{n})] \longrightarrow E[\pi_2]$$
$$E[\mathtt{catch}_n\, M] \longrightarrow M\,\langle E[\mathtt{0}], \ldots E[\mathtt{n}-\mathtt{1}]\rangle$$
$$E[\mathbf{Y}\, M] \longrightarrow E[M\,(\mathbf{Y}M)]$$

Table 1: "Small-step" operational semantics for SPCF programs.

given terms $M, N : T$, $M \lesssim N$ if for all compatible program contexts $C[\cdot]$, $C[M] \Downarrow$ implies $C[N] \Downarrow$.

3.2. **The bistable model of SPCF.** The ground type $\mathtt{nat}$ is interpreted as $\mathbb{N}_\perp^\top$, where $\mathbb{N}$ is the set of natural numbers with the trivial extensional and bistable orderings. We interpret $\mathtt{catch}_n$ as the strict bistable function from $\Sigma^n \Rightarrow \Sigma$ to $\mathbb{N}_\perp^\top$ which sends the ith projection to the value i. The interpretation of the remainder of the language (i.e. PCF) is standard, since $\mathcal{BBC}$ is a cpo-enriched Cartesian closed category.

Proposition 3.4. *$M \Downarrow$ if and only if $[\![M]\!] = \top$.*

Proof. To show soundness, we need simply to verify that if $M \longrightarrow N$ then $[\![M]\!] = [\![N]\!]$. This is standard for all the rules except those for $\top$ and $\mathtt{catch}$. To establish these cases, we prove by induction that evaluation contexts are interpreted as strict maps — i.e. $[\![E[\top]]\!] = \top$ and $[\![E[\bot]]\!] = \bot$. Thus for any closed term $M : \Sigma^n \Rightarrow \Sigma$, if $[\![M]\!]$ is constant, then $[\![E[\mathtt{catch}\, M]]\!] = [\![M\, E[\mathtt{0}], \ldots, E[\mathtt{n}-\mathtt{1}]\rangle]\!]$, whilst if $[\![M]\!] = \pi_i$ then $[\![M\,\langle E[\mathtt{0}], \ldots, E[\mathtt{n}-\mathtt{1}]\rangle]\!] = [\![E[\mathtt{i}]]\!] = [\![E[\mathtt{catch}\, M]]\!]$

Adequacy is proved using a Tait-style computability predicate argument as for PCF [28]. □

We prove full abstraction by reduction to universality for $\Lambda_\perp^\top$. The key to doing this is the observation that for each i, the type $\Sigma^i \Rightarrow \Sigma$ is a definable retract of $\mathtt{nat}$.

For each $n \geq 1$ we have projection maps from $\mathbb{N}_\perp^\top$ to $\Sigma^n \Rightarrow \Sigma$ sending $i < n$ to the $i+1$th projection, and $i \geq n$ to $\bot$. These are definable as n-ary case statements $\mathtt{case}_n$, where $\mathtt{case}_1 = \lambda x.\lambda y.((\mathtt{IF0}\, x)\,\langle x, \bot\rangle)$, and

$$\mathtt{case}_{n+1} = \lambda x.\lambda y.(\mathtt{IF0}\, x)\langle \pi_1\, y, (\mathtt{case}_n\,(\mathtt{pred}\, x))\,(\pi_2\, y)\rangle$$

Lemma 3.5. *For each SPCF type S there is a sequence of $\Lambda_\perp^\top$ types $\{S_i \;:\; i \in \omega\}$ with SPCF-definable retractions: $\mathsf{inj}_i : S_i \trianglelefteq S : \mathsf{proj}_i$ such that $\bigsqcup_{i\in\omega}([\![\mathsf{proj}_i]\!]; [\![\mathsf{inj}_i]\!]) = \mathsf{id}_{[\![S]\!]}$.*

Proof. We define $\Sigma_i = \Sigma$, $\mathtt{nat}_i = \Sigma^i \Rightarrow \Sigma$ (and so $\mathsf{inj}_i = \mathtt{catch}_i$ and $\mathsf{proj}_i = \mathtt{case}_i$), $(S \times T)_i = S_i \times T_i$, and $(S \Rightarrow T)_i = S_i \Rightarrow T_i$. □

Theorem 3.6. *For all terms M, N, $M \lesssim N$ if and only if $[\![M]\!] \leq^E [\![N]\!]$.*

Proof. Inequational soundness follows from soundness and adequacy: if $[\![M]\!] \sqsubseteq [\![N]\!]$, then for every context $C[\cdot]$, if $C[M] \Downarrow$ then $[\![C[M]]\!] = \top$, by soundness, $[\![C[M]]\!] \sqsubseteq [\![C[N]]\!]$ by compositionality, $[\![C[N]]\!] = \top$, and so by adequacy $C[N] \Downarrow$ as required.

We prove inequational completeness by induction on the type of M, N (closed), for which the base case is Proposition 3.4. For example, if $M, N : S \Rightarrow T$, and $[\![M]\!] \neq [\![N]\!]$, then there exists $e \in [\![S]\!]$ such that $[M]\, e \not\sqsubseteq [\![N]\!]\, e$. Moreover, since $e = ([\![\bigsqcup_{i\in\omega}([\![\lambda x.\mathsf{inj}_i\,(\mathsf{proj}_i\, x)]\!]))(e)$, by continuity there exists i such that $[\![M]\!]([\![\lambda x.\mathsf{inj}_i\,(\mathsf{proj}_i\, x)]\!])(e) \not\sqsubseteq [\![N]\!]([\![\lambda x.\mathsf{inj}_i\,(\mathsf{proj}_i\, x)]\!])(e)$

By definability for $\Lambda^{\top}_{\perp}$, there is a term L such that $[\![L]\!] = [\![\mathsf{proj}_i]\!](e)$, and hence $[\![M\,(\mathsf{inj}_i\, L)]\!] \not\sqsubseteq [\![N\,(\mathsf{inj}_i\, L)]\!]$. By induction hypothesis, there exists a context $C[\_]$ such that $C[M\,(\mathsf{inj}_i\, L)]\!] \Downarrow$ and $C[N\,(\mathsf{inj}_i\, L)]\!] \not\Downarrow$ and so $M \not\lesssim N$ as required. □

4. Universality for a CPS Target Language

We have given a direct interpretation of SPCF in the category of bistable bicpos and bistable and continuous functions, but this is in fact equivalent to a CPS (continuation-passing-style) interpretation (in the style of Streicher and Reus [31]). This may be described as a translation into a target language, $\Lambda^{\top}_{\perp}(\omega)$, which is an extension of $\Lambda^{\top}_{\perp}$ with arithmetic and recursion (and which may also be used as a target calculus for CPS translation of call-by-value variants of SPCF). By proving universality for this calculus we show that it precisely captures the observably sequential functions over the given type-structure.

Types of $\Lambda^{\top}_{\perp}(\omega)$ are generated from two ground types: a data type of natural number *values* and the program (or "response") type Σ. Programs of function type may take either data or programs as arguments, but must return a program — i.e. `nat` may not occur on the right of an arrow. Thus the types of our language are:

$$T ::= \mathbb{N} \mid \Sigma \mid P \times P \mid T \Rightarrow P$$

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

Terms are obtained by extending the simply-typed λ-calculus (with products) with the following constants:

Divergence and Error: $\perp, \top : \Sigma$,

Zero test: $\mathsf{IF0} : \mathbb{N} \Rightarrow \Sigma \Rightarrow \Sigma \Rightarrow \Sigma$, interpreted as the function sending 0 to $\lambda xy.x$ and $n+1$ to $\lambda xy.y$.

Fixpoints: $\mathbf{Y} : (P \Rightarrow P) \Rightarrow P$, interpreted, in standard fashion, as $\bigsqcup_{i\in\omega} F^i(\perp)$, where $F = \lambda f.\lambda g.g\,(f\, g)$.

together with a set of basic arithmetic constants and unary and binary operations operations on $\mathbb{N}$, including:

- zero ($\mathtt{0}$),
- equality testing, $\_ = \_$,
- "injective pairing" $(\_ * \_)$ and projections $\mathsf{fst}\_$ and $\mathsf{snd}\_$, such that $n * m > 0$, $\mathsf{fst}(n * m) = n$ and $\mathsf{snd}(n * m) = t$.
- a unary operation ϕ_f for every total function $f : \mathbb{N} \to \mathbb{N}$, such that $\phi_f(n) = f(n)$.

4.1. **SPCF and $\Lambda^{\top}_{\perp}(\omega)$.** We may embed SPCF in $\Lambda^{\top}_{\perp}(\omega)$ via a fragment of the call-by-name CPS interpretation, by representing the type $\mathbb{N}$ as $(\mathbb{N} \Rightarrow \Sigma) \Rightarrow \Sigma$. The constants of SPCF may thus be expressed in $\Lambda^{\top}_{\perp}(\omega)$ as macros:

- $\mathtt{0} = \lambda x.x\,0$
- $\mathtt{succ} = \lambda f.\lambda x.f\ \lambda n.x\,\mathsf{succ}(n)$
- $\mathtt{IF0} = \lambda f.\lambda x.\lambda y.f\ \lambda n.((\mathsf{IF0}\, n)\, x)\, y$
- $\mathtt{catch}_n = \lambda f.\lambda x.f\ \langle x\,0, \ldots x\,(n-1)\rangle$

In fact, this yields an interpretation of SPCF in the category of bistable bicpos which is equivalent to the direct one, because the objects $\mathbb{N}_\perp^\top$ and $(\mathbb{N} \Rightarrow \Sigma) \Rightarrow \Sigma$ are isomorphic. To show this, we extend our sequentiality result for bistable functions to those which take an argument of the form $\mathbb{N} \Rightarrow D$. Noting that $\mathbb{N} \Rightarrow D \cong \Pi_{i \in \mathbb{N}} D$, if D and E are pointed, we say that a function $f : (\mathbb{N} \Rightarrow D) \Rightarrow E$ is i-strict if $g(i) = \perp$ implies $f(g) = \perp$ and $g(i) = \top$ implies $f(g) = \top$.

Lemma 4.1. *If A is pointed then every strict, continuous and bistable function $f : (\mathbb{N} \Rightarrow A) \Rightarrow \Sigma$ is i-strict for some i.*

Proof. Given $i \in \mathbb{N}$, $a \in A$, and $e \in \mathbb{N} \Rightarrow A$, let $e[a]_i \in \mathbb{N} \Rightarrow A$ denote the function defined:

- $e[a]_i(i) = j$,
- $e[a]_i(n) = e(n)$, if $n \neq i$.

Then $\perp[\top]_i \updownarrow \perp[\top]_j$ for all i, j, and so by continuity and bistability, $f(\top) = f(\bigvee\{\perp[\top]_i \mid i \in \mathbb{N}\}) = \bigvee\{f(\perp[\top]_i) \mid i \in \mathbb{N}\}$, and so $f(\perp[\top]_i) = \top$ for some i, and $e(i) = \top$ implies $\top = f(\perp[\top]_i) \leq^E f(e)$. $\top[\perp]_i \updownarrow \perp[\top]_i$, and so $f(\top[\perp]_i) \wedge f(\perp[\top]_i) = f(\top[\perp]_i \wedge \perp[\top]_i) = f(\perp) = \perp$, and so $f(\top[\perp]_i) = \perp$, and $e(i) = \perp$ implies $f(e) \leq^E f(\top[\perp]_i) = \perp$. □

Hence the strict function from $\mathbb{N}_\perp^\top$ to $(\mathbb{N} \Rightarrow \Sigma) \Rightarrow \Sigma$ sending $\mathtt{in}_1(n)$ to $\lambda f.f\,n$ is an isomorphism.

Corollary 4.2. $\mathbb{N}_\perp^\top \cong (\mathbb{N} \Rightarrow \Sigma) \Rightarrow \Sigma$.

So every SPCF type-object is isomorphic to the corresponding $\Lambda_\perp^\top(\omega)$ type-object. Moreoever, it is straightforward to show that the interpretation of SPCF constants factors through this isomorphism and hence:

Proposition 4.3. *The direct and indirect interpretations of SPCF are equivalent.*

4.2. **Universality for $\Lambda_\perp^\top(\omega)$.** We shall now prove that every element of every $\Lambda_\perp^\top(\omega)$ type-object is expressible as a term, using definable retractions.

Lemma 4.4. *There are definable retractions from $\mathbb{N} \Rightarrow \mathbb{N} \Rightarrow \Sigma$ to $\mathbb{N} \Rightarrow \Sigma$ and from $(\mathbb{N} \Rightarrow \Sigma) \Rightarrow (\mathbb{N} \Rightarrow \Sigma) \Rightarrow \Sigma$ to $(\mathbb{N} \Rightarrow \Sigma) \Rightarrow \Sigma$.*

Proof. Using the injective pairing operation, we have the embedding-projection pairs:
$(\lambda f.\lambda x.(f\ \mathsf{fst}(x)\ \mathsf{snd}(x), \lambda g.\lambda x.\lambda y.g\ (x * y))$ and
$(\lambda f.\lambda x.(f\ \lambda z.x\ (z * 0))\ \lambda z.x\ (z * 1), \lambda g.\lambda x.\lambda y.g\ \lambda z.((\mathsf{IF0}\ \mathsf{snd}(z))\ (x\ \mathsf{fst}(z)))\ (y\ \mathsf{fst}(z)))$. □

Now let U be the type $\mathbb{N} \Rightarrow (\mathbb{N} \Rightarrow \Sigma) \Rightarrow \Sigma$. We will show that U is universal amongst the (pointed) type-objects of $\Lambda_\perp^\top(\omega)$ — i.e. $T \trianglelefteq U$ for every pointed type – with a proof based on the the sequentiality of the model.

Lemma 4.5. *If $f : U \to \Sigma$ is i-strict then for any $h \in U$, $f(h) = (h\,i)\,\lambda v.f\,(h[\lambda y.y\,v]_i)$.*

Proof. If $h(i) = \perp$, then $f(h) = \perp = (h\,i)\,\lambda v.(f(h[\lambda y.y\,v]_i)$, and similarly if $h(i) = \top$. Otherwise $h(i) = \lambda p.p\,n$ for some $n \in \mathbb{N}$. Then $h[\lambda y.y\,n]_i = h$ and so $h(i)\,\lambda v.(f(h[\lambda y.y\,v]_i) = f(h[\lambda y.y\,n]_i) = f(h)$ as required. □

Hence if f is i-strict and $h(i)(k) = h(i)(k')$ for all $k, k' \in \mathbb{N} \Rightarrow \Sigma$, then $f(h) = h(i)(\bot)$. Note that we may express $h[a]_i$ in $\Lambda^{\top}_{\bot}(\omega)$ as $\lambda x.\mathsf{IF0}\, x = i \,\mathsf{then}\, a \,\mathsf{else}\, (h\, x)$.

Definition 4.6. Let $\mathsf{inj} : U \Rightarrow \Sigma =$
$\mathbf{Y}\lambda F.\lambda f.\lambda xy.f\ \lambda a.\lambda b.(\mathsf{IF0}\, x\, \mathsf{then}(y\, a)\, \mathsf{else}\, ((F\ (\lambda k.f\ k[\lambda p.p\, \mathsf{fst}(x)]_a))\, \mathsf{fst}(a))\, y$
and $\mathsf{proj} = \mathbf{Y}\lambda G.\lambda g.\lambda h.(g\, 0)\, \lambda u.((h\, u)\, \lambda v.(G\, \lambda w.g\, (v * w))\, h$.

To prove that this defines a retraction, we require a bound on the number of times the fixpoint must be unwound to compute $\mathsf{inj}(f)$ for finitary $f \in U \Rightarrow \Sigma$.

Definition 4.7. A function $f : U \to \Sigma$ is i-dependent if there exist $g, h \in U$ such that $g(j) = h(j)$ for all $j \neq i$ and $f(g) \neq f(k)$. We shall say that f has finite support if the set of $i \in \mathbb{N}$ such that f is i-dependent is finite.

Lemma 4.8. *Every $f : U \Rightarrow \Sigma$ is the least upper bound of a chain of functions with finite support.*

Proof. For each $g \in U$, define $g_i \in U$ by $g_i(x) = g(x)$ if $x \leq i$ and $g(x) = \bot$, otherwise. Then $f^i : U \to \Sigma$, defined $f^i(g) = f(g_i)$ is continuous and bistable, and k-dependent only for $k \leq i$. By continuity, $\bigsqcup\{f^i \mid i \in \mathbb{N}\} = f$. □

Lemma 4.9. *If f has finitary support then $\mathsf{inj}(\mathsf{proj}(f)) = f$.*

Proof. By induction on the size of the set of $n \in \mathbb{N}$ such that f is n-dependent. If f is not n-dependent for any n then it is constant, and so $\mathsf{inj}(\mathsf{proj}(f)) = f$ by strictness of $\mathsf{inj}, \mathsf{proj}$.

Suppose f is i-strict (hence i-dependent). Unfolding the fixpoint gives $\mathsf{inj}(f) =$
$\lambda x.\lambda y.f\ \lambda a.\lambda b.(\mathsf{IF0}\, x\, \mathsf{then}(y\, a)\, \mathsf{else}\, ((\mathsf{inj}\ (\lambda k.f\ k[\lambda p.p\, \mathsf{fst}(x)]_a))\, \mathsf{fst}(a))\, y$.
Lemma 4.5 (on $\lambda a.\lambda b.(\mathsf{IF0}\, x\, \mathsf{then}(y\, a)\, \mathsf{else}\, ((\mathsf{inj}\ (\lambda k.f\ k[\lambda p.p\, \mathsf{fst}(x)]_a))\, \mathsf{fst}(a))\, y)$, gives $\mathsf{inj}(f)(0)(e) = (e\, i)$ and $\mathsf{inj}(f)(m)(e) = ((\mathsf{inj}(\lambda k.f\ k[\lambda y.y\, \mathsf{fst}(m)]_i))\, \mathsf{fst}(a))\, e$ for $m > 0$.

Hence $\mathsf{proj}(\mathsf{inj}(f))(h)$
$= ((h\, i)\, (\lambda v.(G\, \lambda w.\mathsf{inj}(f)\, (v * w))\, h)$
$= ((h\, i)\, (\lambda v.(G\, \lambda w.\mathsf{inj}(\lambda k.f\ \ k[\mathsf{fst}(v * w)]_i)\, \mathsf{snd}(v * w))\, h)$
$= ((h\, i)\, (\lambda v.(\mathsf{proj}\, \lambda w.(\mathsf{inj}(\lambda k.f\ \ k[v]_i))\, w)\, h)$
$= (h\, i)\, (\lambda v.(\mathsf{proj}\, (\mathsf{inj}(\lambda k.f\ \ k[v]_i)))\, h)$.

Observe that $\lambda k.f\ \ k[v]_i$ is n-dependent on strictly fewer n than f, since it is not i-dependent but if it is n-dependent for some $n \neq i$ then so is f. Hence by hypothesis $\mathsf{proj}(\mathsf{inj}(\lambda k.f\, k[v]_i)) = \lambda k.f\, k[v]_i$. So $\mathsf{proj}(\mathsf{inj}(f))(h) = h(i)\, \lambda v.(f(h[v]_i) = f(h)$ by Lemma 4.5 as required. □

Proposition 4.10. $(\mathsf{inj}, \mathsf{proj})$ *form a definable retraction from $U \Rightarrow \Sigma$ to U.*

Proof. For each i, $\mathsf{proj}(\mathsf{inj}(f^i)) = f^i$ by Lemma 4.9, and so $\mathsf{proj}(\mathsf{inj}(f)) = \mathsf{proj}(\mathsf{inj}(\bigsqcup_{i\in\mathbb{N}} f^i)) = \bigsqcup_{i\in\mathbb{N}} \mathsf{proj}(\mathsf{inj}(f^i)) = \bigsqcup_{i\in\mathbb{N}} f^i = f$. □

It is now straightforward to prove universality of U.

Proposition 4.11. *For each pointed type T there is a definable retraction from T to U*
$U \Rightarrow U \trianglelefteq U$.

Proof. By induction on the structure of T. For the induction step, suppose $T = R \Rightarrow S$, then:
$T \trianglelefteq U \Rightarrow U \cong \mathbb{N} \Rightarrow (\mathbb{N} \Rightarrow \Sigma) \Rightarrow U \Rightarrow \Sigma \trianglelefteq \mathbb{N} \Rightarrow (\mathbb{N} \Rightarrow \Sigma) \Rightarrow U$
$\cong \mathbb{N} \Rightarrow \mathbb{N} \Rightarrow (\mathbb{N} \Rightarrow \Sigma) \Rightarrow (\mathbb{N} \Rightarrow \Sigma) \Rightarrow \Sigma \trianglelefteq \mathbb{N} \Rightarrow \mathbb{N} \Rightarrow (\mathbb{N} \Rightarrow \Sigma) \Rightarrow \Sigma$
$\cong (\mathbb{N} \Rightarrow \Sigma) \Rightarrow \mathbb{N} \Rightarrow \mathbb{N} \Rightarrow \Sigma \trianglelefteq (\mathbb{N} \Rightarrow \Sigma) \Rightarrow \mathbb{N} \Rightarrow \Sigma \cong U$. □

Lemma 4.12. *The bistable semantics of $\Lambda^{\top}_{\perp}(\omega)$ is universal at type U.*

Proof. Given $f \in \mathbb{N} \Rightarrow (\mathbb{N} \Rightarrow \Sigma) \Rightarrow \Sigma$, let $\widehat{f} : \mathbb{N} \to \mathbb{N}$ be defined:

- $\widehat{f}(n) = 0$, if $f(n) = \perp$,
- $\widehat{f}(n) = 1$, if $f(n) = \top$,
- $\widehat{f}(n) = m + 2$, if $f(n) = \mathsf{in}(m)$.

Then f is definable as the term:
$\lambda x.\lambda y.\mathsf{IF0}\, \phi_{\widehat{f}}(x)\, \mathsf{then}\, \perp\, \mathsf{else}\, (\mathsf{IF0}\, \mathsf{pred}(\phi_{\widehat{f}}(x))\, \mathsf{then}\, \top\, \mathsf{else}\, y\, (\mathsf{pred}(\mathsf{pred}(\phi_{\widehat{f}}(x)))))$. □

Hence by Lemma 2.13 we have shown the following.

Proposition 4.13. *The bistable semantics of $\Lambda^{\top}_{\perp}(\omega)$ is universal.*

4.3. **Extending the Bistable Semantics.** We may us bicpos to give (fully abstract) interpretations of functional programming languages with a variety of features of, including recursive types, call-by-value functions, and different control primitives. In general, these models follow the same lines as those based on cpos and continuous functions.

Sum Types: We may interpret sum types using either the coproduct, the "bilifted coproduct", $A \oplus B = (A + B)^{\top}_{\perp}$, or a "bi-coalesced" sum identifying the $\top$ and $\perp$ elements of its components. Using the bilifted co-product, for example, we may construct a fully abstract model of SPCF extended with sums [14]. It is straightforward to reduce full abstraction for this semantics to the case of the language without sums by using a definable retraction $A \oplus B \trianglelefteq \mathtt{nat} \times A \times B$. (The injection from $A \oplus B$ to $\mathtt{nat} \times A \times B$ sends $\mathtt{in_l}(e)$ to $\langle 0, e, \perp\rangle$ and $\mathtt{in_r}(e)$ to $\langle 1, \perp, e\rangle$, and the projection from $\mathtt{nat} \times A \times B$ to $A \oplus B$ sends $\langle 0, d, e\rangle$ to $\mathtt{in_l}(d)$, $\langle n+1, d, e\rangle$ to $\mathtt{in_r}(e)$.)

Recursive Types: We may interpret general recursive types using bistable variants of the standard techniques for determining colimits of ω-chains of cpos [29, 27]. For example, we may give an observably sequential version of Plotkin's FPC [11] by adding recursive types to SPCF. We may prove full abstraction for the resulting semantics by showing that every type is the limit of a chain of SPCF types, as shown for unary FPC in [15].

Call-by-value: Our constructions generalize naturally to a call-by-value setting using standard techniques; for example, the strong monad $(\_)^{\top}_{\perp}$ meets the requirements for a model of Moggi's computational metalanguage [25].

Hence we can interpret a call-by-value version of SPCF with `catch`. A proof of full abstraction for this model using definable retractions is given in [20]. Alternatively, we may interpret call-by-value SPCF with control (i.e. `callcc`) at all types by CPS interpretation.

Continuation-passing style interpretation: We have given a simple interpretation of SPCF inside $\Lambda^{\top}_{\perp}(\omega)$: this corresponds to a special case of the call-by-name CPS interpretation of Streicher and Reus [31], in which (closed) terms $M : T$ are interpreted as elements of $[\![T]\!]_c \Rightarrow \Sigma$, where $[\![T]\!]_c$ — the object of *continuations* of type T — is defined $[\![\mathtt{nat}]\!]_c = \mathbb{N} \Rightarrow \Sigma$ and $[\![S \Rightarrow T]\!]_c = ([\![S]\!]_c \Rightarrow \Sigma) \times [\![T]\!]_c$.

In general (in call-by-value, or call-by-name with sum types), continuation-passing style interpretations will not be equivalent to those based on the lifting monad (the latter is equivalent to a *linear* CPS monad [17]). CPS interpretation yields models

with "higher-order" control (`callcc` at *all* types), whilst lifting yields models with "first-order control" (`catch` or `callcc` at ground type only).

5. Bistable functions and Sequential Algorithms

As we have already observed, sequential algorithms also provide a description of the fully abstract model of SPCF [7], and thus correspond to bistable functions. We shall now make this correspondence explicit, by showing that the set of sequential data algorithms on a sequential data structure forms a bistable bicpo, and that all bistable functions between such spaces of strategies are observably sequential, in that they are computed by a sequential algorithm on the corresponding function-space SDS. (Similar results have been described by Curien [9] and Streicher [30].)

As observed in [21, 8], sequential algorithms on sequential data structures may be represented as strategies on a game (of the basic form described in [1]). We adopt this presentation, capturing interactions which result in an error ($\top$) as odd-length traces.

A sequential data structure game A is specified by a triple (M_A, λ_A, P_A), where M_A is a set of moves with a labelling function $\lambda_A : M_A \to \{P, O\}$ which partitions M_A into sets of Player and Opponent moves. $P_A \subseteq M_A^{\circledast}$ is the set of plays of A, where $M_A^{\circledast}$ is the set of sequences over A which are finite, alternating (i.e. P-moves are immediately preceded by O-moves and vice-versa), and contain at most one occurrence of each move and at least as many Opponent as Player moves. Key examples are the "empty game" $\langle \varnothing, \varnothing, \{\varepsilon\}\rangle$, and the game with one (Opponent) move $o = \langle \{o\}, \langle o, O\rangle, \{\varepsilon, o\}\rangle$.

Sequential algorithms, or Player strategies on A, are represented as sets of plays, using odd-length sequences to represent divergences. We write $s \sqsubseteq^E t$ for the partial order on sequences defined "s is an even-length prefix of t *or* $s = t$".

Definition 5.1. A sequential algorithm over a game A is a non-empty subset of P_A, subject to the conditions:

- Even-prefix closure — if $s \sqsubseteq^E t \in \sigma$, then $s \in \sigma$.
- Even-branching — if $s, t \in \sigma$ then $s \sqcap t \sqsubseteq^E s, t$. (So the only odd-length sequences in σ are of maximal length.)

We shall write $\mathbf{strat}(A)$ for the set of strategies over A. Given a strategy σ, we shall write $E(\sigma)$ for its set of even-length sequences (which is a strategy).

So, for instance, there are two strategies over o, $\{\varepsilon\}$ and $\{\varepsilon, o\}$. We shall now define an extensional order and bistable coherence making $\mathbf{strat}(A)$ a bistable bicpo.

Definition 5.2. We first define the extensional order on plays:
$s \leq^E t$ if s is even-length and $s \sqsubseteq t$, or t is odd-length and $t \sqsubseteq s$.

This is a partial order — to show antisymmetry, note that if $s \leq^E t$ and $t \leq^E s$ then s and t are either both even or both odd, and hence $s = t$. Thus we may define a partial order on strategies: $\sigma \leq^E \tau$ if $\forall s \in \sigma. \exists t \in \tau. s \leq^E t$. We establish that this is a partial order by proving antisymmetry.

Lemma 5.3. *If $\sigma \leq^E \tau$ and $\tau \leq^E \sigma$, then $\sigma = \tau$.*

Proof. We prove that $s \in \sigma$ if and only if $s \in \tau$ by induction on length. For the induction case suppose $sab \in E(\sigma)$. Then $s \in \sigma$ and so $s \in \tau$, and there exists $t \in \tau$ such that $sab \leq^E t$. If $sab \sqsubseteq t$ then $sab \in \tau$. Otherwise t is odd-length and $t \sqsubseteq sab$. Then there exists $t' \in \sigma$ such that $t \leq^E t'$ and so $t' \sqsubseteq t \sqsubseteq sab$. But this contradicts determinacy of σ. □

So in $\mathbf{strat}(o)$, for instance, we have $\{\varepsilon\} \leq^E \{\varepsilon, o\}$. More generally, for each game there is a $\leq^E$-least element $\bot$ — the empty strategy — and a $\leq^E$-greatest element $\top$, which contains every play consisting of at most one move.

Definition 5.4. Two strategies are bistably coherent if they have the same non-divergent traces — i.e. $\sigma \updownarrow \tau$ if $E(\sigma) = E(\tau)$.

Lemma 5.5. *For any game A,* $\mathrm{strat}(A) = (\mathbf{strat}(A), \leq^E, \updownarrow)$ *is a pointed bistable bicpo.*

Proof. If $E(\sigma) = E(\tau)$ then we may define $\sigma \wedge \tau = \sigma \cap \tau$ and $\sigma \vee \tau = \sigma \cup \tau$. These clearly satisfy the even-prefix-closure and even-branching conditions.

It is straightforward to see that $\sigma \cup \tau$ is a least upper bound, since $\sigma, \tau \subseteq \sigma \cup \tau$, and if $\sigma, \tau \leq^E \rho$, then for all $s \in \sigma \cup \tau$, either $s \in \sigma$ or $s \in \tau$ and we have $r \in \rho$ such that $s \leq^E r$.

Similarly, $\sigma \cap \tau$ is a lower bound — $\sigma \cap \tau \subseteq \sigma, \tau$. To show that it is a greatest lower bound, suppose $\rho \leq^E \sigma, \tau$ and $r \in \rho$. Then there exist $s \in \sigma, t \in \tau$ such that $r \leq^E s, t$. If s is even-length, then $s \in E(\tau) \subseteq \sigma \cap \tau$. If s is odd-length, then $s \sqsubseteq r$ and $s, t \in \sigma \cup \tau$ and so $s = t \in \sigma \cap \tau$ as required.

We now prove completeness. For a directed set of strategies $S \subseteq \mathrm{strat}(A)$, we define $\bigsqcup S = \{s \in P_A \mid \exists \sigma(s) \in S. \forall \tau \in S. \sigma(s) \leq^E \tau \Longrightarrow s \in \tau\}$.

This is a well-defined strategy: if $s, t \in \bigsqcup S$ then there exists $\tau \in S$ such that $\sigma(s), \sigma(t) \leq^E \tau$ and so $s, t \in \tau$ and are therefore even-branching.

$\bigsqcup S$ is an upper bound for S: We prove by induction on sequence length that if $s \in \sigma \in S$ then there exists $t \in \bigsqcup S$ such that $s \leq^E t$. Suppose there exists $\tau \in S$ with $\sigma \leq^E \tau$ such that $s \notin \tau$. Then there exists $s' \in \tau$ with $s \leq^E s'$, and s' must be a (proper) prefix of s so by hypothesis, there exists $t \in \bigsqcup S$ with $s \leq^E s' \leq^E t$.

$\bigsqcup S$ is a *least* upper bound: If $\sigma \leq^E \tau$ for all $\sigma \in S$, then if $s \in \bigsqcup S$ then $s \in \sigma(s) \leq^E \tau$, and so there exists $t \in \tau$ with $s \leq^E t$.

$\bigsqcup$ preserves coherence: Suppose $X \updownarrow Y$, and $s \in E(\bigsqcup X)$. Then $s \in \sigma_S(s)$, and there exists $\sigma' \in X, \tau \in Y$ such that $\sigma \leq^E \sigma'$ and $\sigma' \updownarrow \tau$. so $s \in \sigma'$ and $s \in \tau$. If $\tau' \in Y$ and $\tau \leq^E \tau'$ then either $s \in \tau'$ or else there exists odd-length $t \in \tau'$ with $t \sqsubseteq s$. But in the latter case, we may find σ'', τ'' with $\sigma' \leq^E \sigma''$ and $\tau' \leq^E \tau''$ and $\sigma'' \updownarrow \tau''$ and so $s \in \tau''$. Since there exists $t' \in \tau''$ with $t \leq^E t'$, this is a contradiction.

The proof that $\bigsqcup$ preserves bistable glbs is similar. □

We shall say that a biorder arising as $\mathrm{strat}(A)$ for some sequential data structure is an SDS-biorder.

5.1. **Bistable Functions and observably sequential functions.** We shall now show that bistable functions between spaces of sequential algorithms correspond to sequential algorithms on the corresponding "function-space" sequential data structure. We follow Lamarche [21] and Curien [8] in decomposing this into an affine function space $A \multimap B$, and a ! operator.

Definition 5.6. The affine function-space $A \multimap B$ is formed as follows.

- $M_{A \multimap B} = \overline{M_A} + M_B$,
- $\lambda_{A \multimap B} = [\overline{\lambda_A}, \lambda_B]$,
- $P_{A \multimap B} = \{t \in M^{\circledast}_{A \multimap B} \mid t{\restriction}A \in A \wedge t{\restriction}B \in B\}$.

We define the affine application of $\sigma : A \to B$ to $\tau : A$:

$$\tau; \sigma = \{t\restriction B \mid t \in \sigma \wedge t\restriction A \in \tau\}$$

Lemma 5.7. *For any sequential algorithm $\sigma : A \multimap B$, the function from* $\mathrm{strat}(A)$ *to* $\mathrm{strat}(B)$ *sending τ to $\tau;\sigma$ is continuous and bistable.*

Proof. For monotonicity, suppose $\rho : A \leq^E \tau : A$. Then given $r \in \rho;\sigma$, we have $s \in \sigma$ such that $s\restriction B = r$ and $s\restriction A \in \rho$. Hence there exists $t \in \tau$ such that $s\restriction A \leq^E t$. If $s\restriction A \sqsubseteq t$, then $s\restriction A$ is even-length and so $s\restriction A \in \tau$ and so $r = s\restriction B \in \tau$ as required. If $s\restriction A \not\sqsubseteq t$, t is odd-length and $t \sqsubseteq s\restriction A$. Hence there exists $s' \sqsubseteq^E s$ such that $s'\restriction A = t$, and $s'\restriction B \sqsubseteq s\restriction B = r$, and $s'\restriction B$ is odd-length, so $s\restriction B \leq^E s'\restriction B$ as required. For continuity, suppose $s \in (\bigsqcup S);\sigma$. Then there exists $t \in A \multimap B$ such that $t\restriction B = s$, and there exists $\tau \in S$ such that $\tau \leq^E \tau'$ implies $t\restriction A \in \tau'$. So $\tau \leq^E \tau'$ implies $s \in \tau'$ and hence $s \in \bigsqcup\{\sigma;\tau \mid \sigma \in S\}$.

For bistability, we show that for all $\tau : A$, $E(\tau;\sigma) = E(E(\tau);\sigma)$. Given $t \in E(\tau;\sigma)$ there exists $s \in \sigma$ such that $s\restriction A \in \tau$ and $s\restriction B = t$. But since $s\restriction B$ is even-length, so are s and $s\restriction A$, and therefore $s\restriction B \in E(E(\tau);\sigma)$. Preservation of bistable lubs and glbs is straightforward. For example, if $\tau \updownarrow \rho$ then $s \in (\tau;\sigma) \cup (\rho;\sigma)$ if and only if there exists $t \in \sigma$ such that $t\restriction B = s$ and $t\restriction A \in \rho$ or $t\restriction A \in \tau$ if and only if $s \in (\rho \cup \tau);\sigma$. □

We form the game $!A$ as in [21] by using plays of A as moves of $!A$. For a sequence s of such moves, let $|s|^E = \{p \in P_A \mid \exists t.tp \sqsubseteq^E s\}$.

Definition 5.8. From a game A, we define a game $!A$ as follows:

- $M_{!A} = P_A - \{\varepsilon\}$,
- $\lambda_{!A}(sa) = \lambda_A(a)$,
- $P_{!A} = \{s \in M_{!A}^{\circledast} \mid \forall t \sqsubseteq s.|t|^E \in \mathrm{strat}(A)\}$.

We define the *promotion* of a strategy $\sigma : A$ to a strategy $\sigma^\dagger : !A$:
$\sigma^\dagger = \{s \in P_{!A} \mid |s|^E \subseteq \sigma\}$.

Lemma 5.9. *The function sending σ to $\sigma^\dagger$ is continuous and bistable.*

Proof.

Monotonicity: We prove by induction on the length of s that if $s \in \sigma^\dagger$ then there exists $t \in \tau^\dagger$ such that $s \leq^E t$. For the induction case, suppose $s = s'(pa)$ or $s = s'(pa)(pab)$, where s' is even-length. Then by hypothesis there exists $t' \in \tau^\dagger$ such that $s' \leq^E t'$. If t' is odd-length then $t' \sqsubseteq s' \sqsubseteq s$ and we are done. If t' is even-length, then $s' \sqsubseteq t'$ and so $s' \in \tau^\dagger$. If $s = s'(pa)$ then since $pa \in \sigma$, there must exist $q \in \tau$ with $q \leq^E pa$ — i.e. q is odd-length and $q \sqsubseteq pa$. Since $p \in \tau$ by even-prefix closure, q cannot be a proper prefix of pa and so $q = pa$, and so $s = s'(pa) \in \tau^\dagger$. Similarly, if $s = s'(pa)(pab)$, then either $pab \in \tau$ — and so $s \in \tau^\dagger$ — or else $pa \in \tau$ and so $s \leq^E s'(pa) \in \tau^\dagger$.

Continuity: Given a directed set of strategies S, suppose $s \in (\bigsqcup S)^\dagger$. We prove by induction on the length of s that $s \in \bigsqcup\{\sigma^\dagger \mid \sigma \in S\}$. Suppose $s = s'(pa)$. Then by hypothesis there exists $\sigma \in S$ such that $s' \in \sigma^\dagger$ and $\sigma \leq^E \tau$ implies $s' \in tau^\dagger$. Since $|s|^E \subseteq \bigsqcup S$, there exists $\rho \in S$ such that $\rho \leq^E \tau$ implies $pa \in \tau$. So there exists θ such that $\sigma, \rho \leq^E \theta$ and so $\theta \leq^E \tau$ implies $s' \in \tau^\dagger$ and $pa \in \tau$ and so $s'(pa) \in \tau^\dagger$.

Bistability: Note that if $s \in P_{!A}$ is even-length then $|s|^E = E(|s|^E)$. Hence $E(\sigma)^\dagger = E(\sigma^\dagger)$, and so if $E(\sigma) = E(\tau)$ then $E(\sigma^\dagger) = E(\tau^\dagger)$. Moreover $\sigma^\dagger \cap \tau^\dagger = \{s \in$

$P_{!A} \mid |s|^E \subseteq \sigma \wedge |s|^E \subseteq \tau\} = \sigma^\dagger \cap \tau^\dagger$ and $\sigma^\dagger \cup \tau^\dagger = \{s \in P_{!A} \mid |s|^E \subseteq \vee \wedge |s|^E \subseteq \tau\} = \sigma^\dagger \cup \tau^\dagger$.

□

We define the application of a strategy $\sigma : A \Rightarrow B$ to a strategy $\sigma : B$ by combining the promotion and affine application operations: $\sigma \cdot \tau = \tau^\dagger; \sigma$ (or directly, $\sigma \cdot \tau = \{s \upharpoonright B \mid s \in \sigma \wedge |s \upharpoonright !A|^E \subseteq \tau\}$). We define an observably sequential function between sequential data structures A and B to be a function $f : \mathrm{strat}(A) \to \mathrm{strat}(B)$ which is "realized" by a sequential algorithm $\sigma_f : A \Rightarrow B$ — i.e. $f(\tau) = \sigma_f \cdot \tau$. By Lemmas 5.7 and 5.9, we have shown the following.

Proposition 5.10. *Every observably sequential function is continuous and bistable.*

We shall now show that every strategy on $A \Rightarrow B$ corresponds to a continuous and bistable function from $\mathrm{strat}(A)$ to $\mathrm{strat}(B)$. To do so, we observe that bistable functions are *stable* with respect to the inclusion order — i.e. continuous with respect to $\subseteq$, and conditionally multiplicative (if $\sigma, \sigma' \subseteq \tau$ then $f(\sigma \cap \sigma') = f(\sigma) \cap f(\sigma')$).

Proposition 5.11. *Every bistable and continuous function of SDS-biorders is stable.*

Proof. Suppose $\sigma, \tau \subseteq \rho$. Let $\sigma' = (\sigma \cap \tau) \cup \{pa \in P_A \mid p \in E(\sigma \cap \tau) \wedge \exists q \in \sigma. pa \sqsubseteq q \wedge q \notin \tau\}$, and $\tau' = (\sigma \cap \tau) \cup \{pa \in P_A \mid \exists q \in \tau. pa \sqsubseteq q \wedge q \notin \sigma\}$.

Then $\sigma \leq^E \sigma'$ (if $s \in \sigma$ then either $s \in \sigma \cap \tau \subseteq \sigma'$, or else $s'a \in \sigma'$, where $s'a$ is the maximal prefix of s such that $s' \in \sigma \cap \tau$) and similarly $\tau \leq^E \tau'$. Moreover $\sigma' \updownarrow \tau'$ and $\sigma' \cap \tau' = \sigma \cap \tau$, and so $f(\sigma \cap \tau) = f(\sigma) \cap f(\tau) \leq^E f(\sigma') \cap f(\tau') = f(\sigma' \cap \tau') = f(\sigma \cap \tau) \leq^E f(\sigma) \cap f(\tau)$ as required.

Hence f is also monotone with respect to $\subseteq$, and moreover continuous because every $\subseteq$-directed set is $\leq^E$-directed.

□

Thus each continuous and bistable function $f : \mathrm{strat}(A) \to \mathrm{strat}(B)$ has a *trace*: $\mathrm{tr}(f) \subseteq \mathrm{strat}(A) \times P_B = \{(\sigma, t) \mid t \in f(\sigma) \wedge \forall \tau. (\sigma \uparrow \tau \wedge t \in f(\tau) \implies \sigma \subseteq \tau)\}$. We define a sequential algorithm $\sigma_f : A \Rightarrow B$ for computing f by "sequentializing" this trace: $\sigma_f = \{s \in P_{A \Rightarrow B} \mid \forall t \sqsubseteq^E s. (|t \upharpoonright !A|^E, t \upharpoonright B) \in \mathrm{tr}(f)\}$.

Lemma 5.12. *σ_f is a well-defined strategy on $A \Rightarrow B$.*

Proof. σ_f is even-prefix-closed by definition. To prove that it is even-branching, suppose $sab, sac \in \sigma_f$. We show that $b = c$.

- If b and c are both moves in B then $sab \upharpoonright B, sac \upharpoonright C \in f(|sa \upharpoonright !A|^E)$ and so $b = c$.
- If b is a move in $!A$ and c is a move in B (or vice-versa), then b is an odd-length sequence on A and so $|sab \upharpoonright !A|^E = |sa|^E \cup \{b\} \updownarrow |sa \upharpoonright !A|^E = |sac \upharpoonright !A|^E$. Hence $f(|sab \upharpoonright !A|^E) \updownarrow f(|sac \upharpoonright !A|^E)$ and so $sac \upharpoonright B = (sa \upharpoonright B)c \in f(|sab \upharpoonright !A|^E)$ since it is even-length. But this contradicts the assumption that the (odd-length) $sab \upharpoonright B = sa \upharpoonright B \in f(|sab \upharpoonright !A|^E)$.
- If b and c are both (Opponent) moves in A, then if $b \neq c$ then $|sab \upharpoonright !A|^E \updownarrow |sac \upharpoonright !A|^E$ and $|sab \upharpoonright !A|^E \wedge |sac \upharpoonright !A|^E = |sa \upharpoonright !A|^E$. Thus $sa \upharpoonright B \in f(|sab \upharpoonright !A|^E) \wedge f(|sac \upharpoonright !A|^E) = f(|sab \upharpoonright !A|^E) \wedge |sac \upharpoonright !A|^E) = f(|sa \upharpoonright !A|^E)$. But by definition of σ_f, $(|sab \upharpoonright !A|^E, sab \upharpoonright B) \in \mathrm{tr}(f)$, which is a contradiction.

We show that if $s \in \sigma_f$ is odd-length, then there is no extension of s in σ_f by the same argument. □

We now show that the sequential algorithm σ_f does indeed compute f, based on the following lemmas.

Lemma 5.13. *Suppose $(\tau, tab) \in \mathrm{tr}(f)$ or $(\tau, ta) \in \mathrm{tr}(f)$, where t is even-length, and $\sigma \subset E(\tau)$ is such that $t \in f(\sigma)$. Then there exists a unique* sequentiality index $\mathrm{seq}(\sigma)$ *for f at (σ, t) — an even-length sequence $pc \in \tau - \sigma$ such that $ta \in f(\sigma \cup \{p\})$.*

Proof. Suppose $(\tau, tab) \in \mathrm{tr}(f)$ (the case $(\tau, ta) \in \mathrm{tr}(f)$ is similar). In this case $\tau = E(\tau)$ by bistability. Let $\sigma' = \sigma \cup \{pc \in P_A \mid p \in \sigma \wedge \exists d.pcd \in \tau\}$. We have $f(\sigma) \updownarrow f(\sigma')$ and so $t \in f(\sigma')$. Moreover, $\tau \leq^E \sigma'$, and so $f(\tau) \leq^E f(\sigma')$. Hence there exists $r \in f(\sigma')$ such that $tab \leq^E r$. If $tab \sqsubseteq r$ then $tab \in f(\sigma')$. But then $tab \in f(\sigma)$, since $\sigma \updownarrow \sigma'$, which contradicts $\subseteq$-minimality of τ. So $r \sqsubseteq tab$ is odd-length, and since $t \in f(\sigma')$, this entails $r = ta$. Since $\sigma' = \bigvee\{\sigma \cup \{pc\} \mid p \in \sigma \wedge \exists d.pcd \in \tau\}$, by bistability there exists a unique $pcd \in \tau$ such that $tc \in f(\sigma \cup \{pc\})$. □

Given $(\tau, tab) \in \mathrm{tr}(f)$ or $(\tau, ta) \in \mathrm{tr}(f)$, where t is even-length, we define a (finite) chain of strategies $\sigma_0 \subset \ldots \sigma_n \subseteq \tau$:

- $\sigma_0 = \bigcap\{\rho \subseteq \tau \mid t \in f(\rho)\}$.
- If $\sigma_i \neq E(\tau)$, then we define $\sigma_{i+1} = \sigma_i \cup \mathrm{seq}(\sigma_i)$.

Lemma 5.14. *If $\rho \subseteq E(\tau)$, $t \in f(\rho)$ and $\mathrm{seq}(\rho) = pc$, then $(\rho \cup \{p\}, ta) \in \mathrm{tr}(f)$ if and only if $\rho = \sigma_i$ for some i.*

Proof. Suppose $(\rho \cup \{p\}, ta) \in \mathrm{tr}(f)$. Since there exists n such that $\sigma_n = E(\tau)$, there must be some i such that $\mathrm{seq}(\sigma_i) = pc$. Then $\mathrm{seq}_i \cup \{p\}$ and $\rho \cup \{p\}$ are stably coherent, and so $tc \in f(\rho \cap \sigma_i \cup \{p\})$. But since $(\rho \cup \{p\}, ta) \in \mathrm{tr}(f)$, we have $\rho = \sigma_i$ as required.

We prove the converse by induction on the size of ρ. Suppose $\rho = \sigma_i$, but there exists $\theta \subset \rho \cup \{p\}$ with $(\theta, ta) \in \mathrm{tr}(f)$. Then $t \in f(\theta - \{p\})$ (by bistability), and so by induction hypothesis $\theta - \{p\} = \sigma_j$ for some $j < i$. But then pc is the sequentiality index for σ_j, and so $pc \in \sigma_i$, which is a contradiction. □

Proposition 5.15. *If $(\tau, tab) \in \mathrm{tr}(f)$ or $(\tau, ta) \in \mathrm{tr}(f)$, where t is even-length, then either $(E(\tau), t) \in \mathrm{tr}(f)$ or else there exists (a unique) $pcd \in E(\tau)$ such that $((E(\tau) - \{pcd\} \cup \{pc\}, ta) \in \mathrm{tr}(f)$.*

Proof. If $(\tau, t) \notin \mathrm{tr}(f)$ then since there exists n such that $\sigma_n = \tau$ $\sigma_{n+1} = \sigma_n \cup \{\mathrm{seq}(\sigma_i)\} = E(\tau)$, we may take $pcd = \mathrm{seq}(\sigma_i)$ as required. □

We may now show how to sequentialize each element of $\mathrm{tr}(f)$.

Lemma 5.16. *Let $f : \mathrm{strat}(A) \to \mathrm{strat}(B)$ be a continuous bistable function. Then for any $(\tau, t) \in \mathrm{tr}(f)$, there exists a sequence $\tau \natural t \in \sigma_f$ such that $|\tau \natural t| !A|^E = \tau$ and $\tau \natural t | B = t$.*

Proof. By induction on the total lengths of the sequences in $\tau \cup \{t\}$.

If t is even-length and non-empty — i.e. $t = t'ab$ — then $\tau = E(\tau)$ by bistability and by Proposition 5.15, either $(\tau, t') \in \mathrm{tr}(f)$ — and so we may define $\tau \natural t = (\tau \natural t')ab$ — or there exists $pcd \in \tau$ such that $((\tau - \{pcd\} \cup \{pc\}, ta) \in \mathrm{tr}(f)$ — and so we may define $\tau \natural t = (((\tau - \{pcd\} \cup \{pc\}) \natural ta)(pcd)b$.

Similarly, if t is odd-length i.e. $t = t'a$ — then if $\tau = E(\tau)$, by Proposition 5.15, either $(\tau, t') \in \mathrm{tr}(f)$ — and so we may define $\tau \natural t = (\tau \natural t')ab$ — or there exists $pcd \in \tau$ such that $((\tau - \{pcd\} \cup \{pc\}, ta) \in \mathrm{tr}(f)$ — and so we may define $\tau \natural t = (((\tau - \{pcd\} \cup \{pc\}) \natural ta)(pcd)$. Otherwise τ contains an odd-length sequence q. By minimality of τ with respect to $\subseteq$, and

bistability of f, q is unique. By Proposition 5.15, either $(E(\tau), t') \in \mathrm{tr}(f)$ — so we may define $\tau \not\downarrow t = (E(\tau) \not\downarrow t')qa$ — or there exists $pcd \in E(\tau)$ such that $((E(\tau) - \{pcd\} \cup \{pc\}, ta) \in \mathrm{tr}(f)$ and so we may define $\tau \not\downarrow t = (((E(\tau) - \{pcd\} \cup \{pc\}) \not\downarrow ta)(pcd)q$. □

Thus we have shown that every bistable and continuous function $f : \mathrm{strat}(A) \to \mathrm{strat}(B)$ is observably sequential (and hence given an alternaative proof that observably sequential functions may be composed).

Proposition 5.17. *The SDS-biorders and observably sequential functions form a full subcategory of $\mathcal{BBC}$.*

6. Further Directions

Research into bidomain models of sequential programming languages is ongoing, and includes the following themes:

Elimination of nesting in SPCF: In [16, 20] we use the full abstract bicpo model of SPCF to show that nested and recursive function calls in SPCF may be *eliminated*: every SPCF term is observationally equivalent to one typable in an affine typing system which does not permit nesting. The proof is based on the universality of the type of first-oder functions: we show that all retractions into this type may be defined in our affine system. Since every first-order function is definable without nesting, we show that every SPCF-definable element of the model is definable in affine SPCF.

Locally Boolean Domains: We have shown that the category of sequential algorithms and sequential data structures can be fully embedded in the category of bistable bicpos and bistable and continuous functions. This leaves open the question of how the correspondence works in the opposite direction; what is the image of the embedding, and given an object in that image, can we construct the corresponding sequential data structures? Furthermore, is there a "linear decomposition" of bistable bidomains into a model of linear logic, which corresponds to that for sequential algorithms [21, 8]? In [17] we answer these qestions by describing a notion of "locally boolean" domain — a partial order (the extensional order) with an involutive negation, which can be used to give simple definitions of the stable and bistable orders. Our fundamental representation result for these domains is that they can all be generated (up to isomorphism) by taking products and co-products, lifting, and limits of ω-chains. Hence, in particular, locally boolean domains may be viewed as games in which one player chooses indices in the product, and the other in the lifted sum.

Semantics of imperative effects: Locally boolean domains form a model of linear type theory equivalent to the simple games and strategies (or affine sequential algorithms) model described by Lamarche [21, 8]. A more general "linear decomposion" of bistable functions is still under investigation. A next step is to extend our semantics beyond functional languages with control to include imperative features, nondeterminism and concurrency, inspired by games models of functional-imperative languages such as Idealized Algol. The key to constructing such models is the identification of categorical structures shared by games and bistable models, and used to capture subtle intensional properties of such languages [13]. This in turn may lead to higher-order principles for reasoning about them.

In another direction, we may obtain a semantics of fresh *name generation* in a category of "FM-biorders" — bistable biorders acted upon by the topological group of natural number automorphisms. This fits with a a natural CPS interpretation of fresh name generation given by Shinwell and Pitts to give a sequential model of a "CPS-nu-calculus".

Other Bidomain Models: Bistable bidomains share many properties with Bérry's original (stable) bidomains [2]. This captures a different but related notion of *non-deterministic* observable sequentiality, as shown by may-nand-must full abstraction results for a version of $\Lambda^{\top}_{\perp}(\omega)$ with countable non-determinism [19] (as well as fully abstract models of languages such as the lazy λ-calculus [18]). This poses the question of whether there is a general notion of bidomain embracing both stable and bistable instances, and other phenomena such as probabilistic non-determinism.

Acknowledgements

Thanks to Pierre-Louis Curien and Thomas Streicher for discussions and encouragement, the referees for their comments, and Paul B. Levy for a new proof of the transitivity of $\updownarrow$.

References

[1] S. Abramsky, R. Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59:543–574, 1994.

[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] G. Berry. *Modèles complètement adéquats et stables des lambda-calculs typés*. PhD thesis, Université Paris 7, 1979.

[4] G. Berry and P.-L. Curien. Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20:265–321, 1982.

[5] A. Bucciarelli and T. Ehrhard. A theory of sequentiality. *Theoretical Computer Science*, 113:273–292, 1993.

[6] R. Cartwright and M. Felleisen. Observable sequentiality and full abstraction. In *Proceedings of POPL '92*, 1992.

[7] R. Cartwright, P.-L. Curien and M. Felleisen. Fully abstract semantics for observably sequential languages. *Information and Computation*, 1994.

[8] P.-L. Curien. On the symmetry of sequentiality. In *Mathematical Foundations of Computer Science*, number 802 in LNCS. Springer, 1993.

[9] P.-L. Curien. Sequential algorithms as bistable maps. Unpublished note, 2002.

[10] Matthias Felleisen, Daniel P. Friedman, Eugene E. Kohlbecker, and Bruce Duba. A syntactic theory of sequential control. *Theoretical Computer Science*, 52:205 – 207, 1987.

[11] M. Fiore and G. Plotkin. An axiomatisation of compuationally adequate domain thoeretic models of FPC. In *Proceedings of LICS '94*, pages 92–102. IEEE Computer Society Press, 1994.

[12] G. Kahn, and G. Plotkin. Concrete domains. *Theoretical Computer Science*, Böhm Festschrift special issue, 1993. First appeared as technical report 338 of INRIA-LABORIA, 1978.

[13] J. Laird. A categorical semantics of higher-order store. In *Proceedings of CTCS '02*, number 69 in ENTCS. Elsevier, 2002.

[14] J. Laird. Bistability: an extensional characterization of sequentiality. In *Proceedings of CSL '03*, number 2803 in LNCS. Springer, 2003.

[15] J. Laird. A fully abstract bidomain model of unary FPC. In *Proceedings of TLCA '03*, number 2701 in LNCS, 2003.

[16] J. Laird. The elimination of nesting in SPCF. In *Proceedings of TLCA '05*, number 3461 in LNCS, pages 234–245. Springer, 2005.

[17] J. Laird. Locally boolean domains. *Theoretical Computer Science*, 342:132–148, 2005.
[18] J. Laird. Sequentiality in bounded bidomains. *Fundamenta Informaticae*, 65:173–191, 2005.
[19] J. Laird. Bidomains and full abstraction for countable non-determinism. In *Proceedings of FoSSaCS'06*, number 3921 in LNCS, pages 352–366. Springer, 2006.
[20] J. Laird. On the expressiveness of affine progams with non-local control: The elimination of nesting in SPCF. *Fundamenta Informaticae*, 2007. To appear.
[21] F. Lamarche. Sequentiality, games and linear logic. In *Proceedings, CLICS workshop, Aarhus University*. DAIMI-397–II, 1992.
[22] R. Loader. Finitary PCF is not decidable. *Theoretical Computer Science*, 266(1 -2):341 –364, 2000.
[23] J. Longley. The sequentially realizable functionals. *Annals of Pure and Applied Logic*, 1998.
[24] J. Longley. Universal types and what they are good for. In *Domain Theory, Logic and Computation: Proceedings of the 2^{nd} International Symposium on Domain Theory*. Kluwer, 2004.
[25] E. Moggi. Computational lambda-calculus and monads. Technical Report ECS-LFCS-88-66, University of Edinburgh Department of Computer Science, 1988.
[26] V. Padovani. Decidability of all minimal models. In M. Coppo and S. Berardi, editor, *Types for proofs and programs*, volume 1158 of *LNCS*. Springer, 1996.
[27] A. M. Pitts. Relational properties of domains. *Information and Computation*, 127:66–90, 1996.
[28] G. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223 – 255, 1977.
[29] G. Plotkin. Postgraduate lecture notes in advanced domain theory (incorporating the 'Pisa notes'). Available from http://www.dcs.ed.ac.uk/home/gdp/publications/, 1981.
[30] T. Streicher. Laird domains. Unpublished note, 2002.
[31] T. Streicher and B. Reus. Classical logic: Continuation semantics and abstract machines. *Journal of Functional Programming*, 8(6):543 –572, 1998.
[32] T. Löw. Locally Boolean domains and universal models for infinitary sequential languages. Doctoral Thesis, Technical University of Darmstadt, 2006.