Electronic Notes in Theoretical Computer Science 69 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume69.html 18 pages

A Categorical Semantics of Higher Order Store

J. Laird

COGS, University of Sussex
Brighton BN1 9QH, UK
E-mail: jiml@cogs.susx.ac.uk

Abstract

We give a categorical description of a class of sound and adequate models of a functional language with assignable variables. This is based on a notion of "sequoidal category", a symmetric monoidal category with an additional non-commutative and non-associative tensor product. We describe a category $\mathcal{G}$ of games and strategies, and show that it satisfies our axioms. We give an axiomatic characterization of those categories (including $\mathcal{G}$) which give rise to fully abstract models.

1 Introduction

This paper is an attempt to give a categorical account of a semantics of higher-order functional-imperative computation (specifically, a typed λ-calculus with locally declared reference variables). The desirability of such an account can perhaps be most effectively argued by a comparison with the situation of purely functional computation — cartesian closed categories are an essential part of a broad theoretical understanding which links logics, type-theories, λ-calculi, programming languages and denotational models.

The task of giving denotational semantics to functional languages with imperative behaviour is complicated by their fundamentally "dynamic" nature, manifested in extreme sensitivity to the order in which programs are evaluated. (To capture such behaviour, we need finer grained structure than is available in CCCs; our notion of model will be based on *linear logic*, with *non-commutative* and even *non-associative* refinements.) Associated with this is the particular subtlety of functional computation with locally declared references, leading to phenomena such as interference and cyclic sharing. These issues account for some of the limitations on previous attempts to model or reason about the store explicitly, and demonstrate the possible dividends of a comprehensive characterization of state based on "logical" principles.

A singular success has been achieved in this area using *game semantics* to model first Idealized Algol [2] and then higher-order (local) references [3] without representing state explicitly at all; programs are modelled as processes

(strategies) which can store and retrieve data by interacting with individual reference cells, each of which is a strategy itself. Having such a denotational semantics naturally prompts the question: what is its underlying categorical structure? As we have suggested, the space of "possible behaviours" for programs with higher-order store is very complex. The games model accurately describes the boundaries of this space, but not its internal structure, rendering it difficult to reason about even some elementary properties. Our aim is to refine the games model in order to bring out its underlying "logical" character and to abstract from it a general, categorical description of the semantics of higher-order local store [1]. Moreover, we shall extend our characterization of sound and adequate models of references with a form of completeness result by capturing axiomatically the most significant property of the games semantics — *full abstraction*.

1.1 Related Work

Naturally, our work is based on the games models of Idealized Algol [2], and higher-order references [3]. The former, in particular, contains some analysis of what a categorical model of state might be like, developing proposals of Reynolds [20] to interpret the type of integer-valued references as the product $(\mathtt{nat} \Rightarrow \mathtt{com} \times \mathtt{nat})$ of its "methods", assignment and dereferencing, which are simply projections from this type. Thus according to [2], a model of Idealized Algol is (in essence) a model of intuitionistic (affine) linear type theory together with a morphism $\mathsf{cell} : I \to !(\mathtt{nat} \Rightarrow \mathtt{com} \times \mathtt{nat})$ representing a reference cell; new variable binding is interpreted as linear composition with cell. The limitations of this notion of model as a general theory stem from the fact that cell is treated as a "black box" — its requisite properties are not characterised in categorical terms, it is simply described as a strategy in a particular games model. Composition with cell allows history-sensitive (non-functional) behaviour to "percolate" through the model, in a way which can be expressed elegantly using *factorization theorems*, but which makes it difficult to give an equational characterisation of the model. The games semantics of higher-order references [3] is similar, except that it is not described via a games model of linear type theory — thus one of the novel features of the semantics given here is that it is a linear category [7].

Another area of research which uses game semantics to reason about functional/imperative languages is based on the observation that the strategies in a restricted version of Idealized Algol are generated by *regular languages*, which can be obtained from the corresponding programs in an elegantly simple way [8]. This "model-checking" approach seems to be quite complementary to the "equational" one described here.

[1] Although we give only one example of such a model, it can be extended by adding various combinations of the computational effects which have been studied in games semantics, such as continuations [13], non-determinism [9], exceptions [14] and concurrency [15].

The treatment of full abstraction developed here is clearly inspired by Abramsky's "Axioms for Definability and Full Completeness" [1]. This gives categorical axioms which capture abstractly the properties of both the AJM and Hyland-Ong games models of PCF [4,11] (and the simply-typed λ-calculus) which allow full abstraction to be proved via inductive *decomposition* of morphisms. Our axiomatization of fully abstract models of references is similar, and it also leads to such a decomposition (and hence to decomposition trees, which represent a new notion of "normal form" for programs with references). However, we are able to characterize full abstraction directly, by showing that tests can be defined to distinguish operationally between any terms which are distinct in our model.

The closest precursor to this work outside game semantics is Reddy's "Linear logic model of state" [19], which anticipates it in several features, including the use of a linear type theory with a non-commutative connective (albeit a different one). However, there are important points of difference between the two models, both in specific terms — the model described in [19] is for a language with only ground-type, *non-interfering* variables — and more general ones; the representation of state is more explicit in Reddy's model.

1.2 Organization of this paper

In section 2, a simply-typed, call-by-name λ-calculus with higher-typed references is defined. The remainder of the paper is devoted to describing a sound and fully abstract categorical semantics of this language, and an example of such a model in a category of games. Section 3 describes the categorical notions upon which this account will be grounded. These form two groups; a "structural" basis — essentially an extension of the notion of symmetric monoidal category to include a non-commutative and non-associative tensor — together with additional conditions which allow a model of references to be constructed. Section 4 describes an example of such structure in a category of games, $\mathcal{G}$. In Section 5 we give the interpretation of $\lambda_{\mathtt{ref}}$, and prove that it is sound and adequate. Section 6 is concerned with full abstraction; we give an axiomatisation of "sequential" models of references, and show that these are fully abstract.

2 A language with references

We shall now describe the syntax and operational semantics of a simply-typed call-by-name λ-calculus with lazy assignment. It can be thought of as a kind of "idealized scheme". To simplify the semantic analysis, we shall consider only types generated from a single ground type, $\mathbf{0}$, containing only a constant for divergence, Ω. (Our inequational soundness result, at least, can be easily generalised to include call-by-name and call-by-value languages with more interesting datatypes.)

Formation and typing rules for terms in contexts (sets of typed variables) are as follows:

$$\frac{}{\Gamma \vdash \Omega : \mathbf{0}} \qquad \frac{}{\Gamma, x:T \vdash x:T}$$

$$\frac{\Gamma, x:S \vdash M:T}{\Gamma \vdash \lambda x.M : S \Rightarrow T} \qquad \frac{\Gamma \vdash M : S \Rightarrow T \quad \Gamma \vdash N : S}{\Gamma \vdash M\, N : T} \qquad \frac{\Gamma, x:T \vdash M:T \quad \Gamma, x:T \vdash N:S}{\Gamma, x:T \vdash x := M.N : S}$$

The operational semantics of $\lambda_{\mathtt{ref}}$ is given by a "small-step" reduction relation for terms of ground type (possibly containing free identifiers), in an environment $\mathcal{E}$ consisting of a set of pairs $(a : T, M : T)$, where a is the name of a location and M is the program stored there.

Definition 2.1 The evaluation rules for $\lambda_{\mathtt{ref}}$ programs are as follows:

$$\lambda y.M\, N, \mathcal{E} \longrightarrow M[a'/a], \mathcal{E} \cup \{(a', N)\} \quad a' \notin \pi_1(\mathcal{E})$$
$$a := N.M, \mathcal{E} \longrightarrow M, \mathcal{E}[(a, N)]$$
$$(a\, N_1) \ldots N_k, \mathcal{E} \longrightarrow (M\, N_1) \ldots N_k, \mathcal{E} \quad (a, M) \in \mathcal{E}$$
$$\Omega, \mathcal{E} \longrightarrow \Omega, \mathcal{E}$$

Note that despite the simple typing, we can express recursive behaviour in $\lambda_{\mathtt{ref}}$ using self-referencing variables — for example, we can express the fixed-point combinator $\mathbf{Y} : (\mathbf{0} \Rightarrow \mathbf{0}) \Rightarrow \mathbf{0}$ as $\lambda f.((\lambda y.y := (f\, y).y)\, \Omega)$. We shall write $M \Downarrow$ if evaluation of M terminates — in which case M evaluates to a head-normal form. We use standard notions of observational approximation and full abstraction.

Definition 2.2 Let $M, N : T$ be terms of $\lambda_{\mathtt{ref}}$. We write $M \lesssim N$ if for all compatible contexts $C[\cdot]$, $C[M] \Downarrow$ implies $C[N] \Downarrow$.
A (order-enriched) model of $\lambda_{\mathtt{ref}}$ is fully abstract (for closed terms) if for all closed $M, N : T$, $[\![M]\!] \sqsubseteq [\![N]\!]$ if and only if $M \lesssim N$.

3 Categorical structure

We shall now describe some categorical structure which will allow a model of $\lambda_{\mathtt{ref}}$ to be constructed. Our analysis is based on a model of intuitionistic linear logic — a symmetric monoidal closed category — with extra structure in the form of an additional non-commutative and non-associative connective (a kind of right-strict product) — $\oslash$. This is reminiscent of the use of premonoidal categories [18] to account for differences in the order of evaluation in the semantics of languages with effects — but the key difference (which is significant) is that we will use non-commutativity to *express* some facts about sequential behaviour. (In fact, unlike a premonoid, $\oslash$ will be a bifunctor). Informally, we can read $A \otimes B$ as "the events A and B occur, starting in indeterminate order", and $A \oslash B$ as "A starts after B, if at all". On this reading, $\oslash$ clearly should not be commutative. Moreover, it should not be associative either — "(A after B) after C" means that C, B and A start in succession,

but "A after (B after C)" simply means that A and B both start after C, but doesn't tell us anything at all about the relative ordering of A and B. Accordingly, we have the "pseudo-associativity" coherence $A \oslash (B \oslash C) \cong (A \otimes B) \oslash C$. The unit for $\otimes$ — the "empty event" should be a unit for $\oslash$ on one side — nothing starting after A is equivalent to A — but not on the other; instead we have $A \oslash I \cong I$ — if A can only start after the empty event then A cannot start at all.

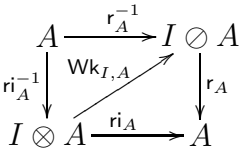
In fact, the equations satisfied by $\oslash$ are those satisfied by the exponential $\multimap$ in a symmetric monoidal closed category — so $\oslash$ can be thought of as a covariant version of $\multimap$, or as a generalisation of the algebraic structure of the natural numbers with the operations of multiplication and exponentiation, which satisfy $(a^b)^c = a^{bc} = a^{cb}$, $1^a = 1$ and $a^1 = a$. (As we shall see, certain sequoidal categories will also satisfy an *additive* rule with respect to a cartesian product, corresponding to $a^{bc} = a^b \times a^c$.)

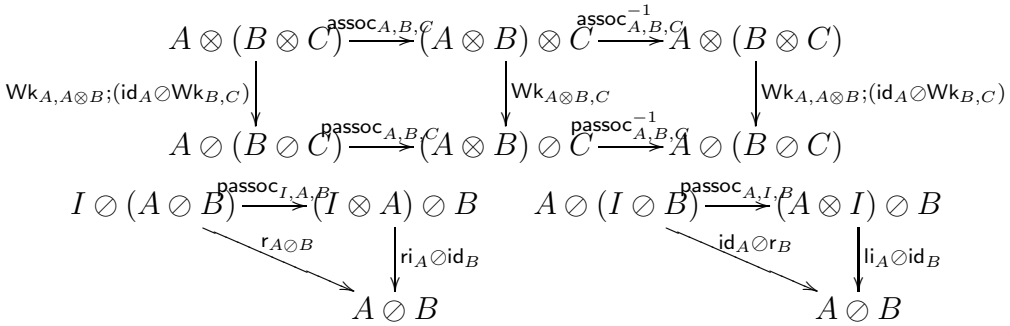
Our starting point will be pointed symmetric monoidal categories — that is, SMCs with a distinguished map $\bot_{A,B} : A \to B$ for each A, B, such that $f; \bot = \bot$ for all f, and $\bot_{A,B} \otimes \bot_{C,D} = \bot_{A \otimes C, B \otimes D}$.

Definition 3.1 Given a pointed SMC, $(\mathcal{C}, I, \otimes)$, let $\mathcal{C}_S$ be the subcategory of $\mathcal{C}$ consisting of the objects of $\mathcal{C}$ and the strict maps between them, where $f : A \to B$ is *strict* if $\bot_{C,A}; f = \bot_{C,B}$ for all C.

It's standard that if $\mathcal{C}$ is a pointed SMC then the symmetric monoidal structure of $\mathcal{C}$ restricts to $\mathcal{C}_S$.

Definition 3.2 A sequoidal category is a pointed SMC $(\mathcal{C}, I, \otimes)$ with a functor $\_ \oslash \_ : \mathcal{C} \otimes \mathcal{C}_s \to \mathcal{C}_s$ (such that $\bot_{A,B} \oslash \bot_{C,D} = \bot_{A \oslash C, B \oslash D}$) and a natural transformation $\mathsf{Wk} : \_ \otimes \_ \to \_ \oslash \_$ and isomorphisms $\mathsf{r}_A : I \oslash A \to A$ and $\mathsf{l}_A : A \oslash I \to I$ and $\mathsf{passoc}_{A,B,C} : A \oslash (B \oslash C) \to (A \otimes B) \oslash C$ satisfying the following coherence diagrams:





(Where $\mathsf{ri}_A : I \otimes A \to A$, $\mathsf{li}_A : A \otimes I \to A$, $\mathsf{assoc}_{A,B,C} : (A \otimes B) \otimes C \to A \otimes (B \otimes C)$ are the monoidal isomorphisms for $(\mathcal{C}, I, \otimes)$.)

The coherence conditions are already sufficient to entail that a sequoidal category is *affine* — for any A there is a map $t_A : A \to I = \mathsf{r}_A^{-1}; \mathsf{Wk}_{A,I}; \mathsf{l}_A$ such that for all $f : A \to B$, $f; t_B = t_A$, and hence there are natural transformations $\mathsf{proj}_{A,B} : A \oslash B \to B = (t_A \oslash \mathsf{id}_B); \mathsf{r}_B$. We shall also make use of the commutativity iomorphism $\mathsf{twist}_{A,B} : A \otimes B \to B \otimes A)$ to define the "partial-commutativity" natural isomorphisms $\mathsf{pcomm}_{A,B,C} : A \oslash (B \oslash C) \to B \oslash (A \oslash C) = \mathsf{passoc}_{A,B,C}; (\mathsf{twist}_{A,B} \oslash \mathsf{id}_C); \mathsf{passoc}^{-1}_{B,A,C}$. (where $\mathsf{twist}_{A,B} : A \otimes B \to B \otimes A$).

Definition 3.3 A sequoidal *closed* category is a sequoidal category $(\mathcal{C}, I, \otimes, \oslash)$ together with a functor $\_ \multimap \_ : \mathcal{C}^{OP} \times \mathcal{C} \to \mathcal{C}$ which restricts to a functor from $\mathcal{C}^{OP} \times \mathcal{C}_S$ to $\mathcal{C}_S$ such that for each object A in $\mathcal{C}$, $A \otimes \_$ is left adjoint to $A \multimap \_$ (so that $(\mathcal{C}, I, \otimes, \multimap)$ is a SMCC), and $A \oslash \_ : \mathcal{C}_s \to \mathcal{C}_s$ is left adjoint to $A \multimap \_ : \mathcal{C}_s \to \mathcal{C}_s$. We also require that the co-units $(\mathsf{app}, \mathsf{ap})$ and units $(\mathsf{co}-\mathsf{app}, \mathsf{co}-\mathsf{ap})$ of the adjunctions commute with Wk:

$$\begin{array}{ccc} A \otimes (A \multimap B) & \xrightarrow{\mathsf{Wk}_{A,A\multimap B}} & A \oslash (A \multimap B) \\ & \searrow^{\mathsf{app}_{A\multimap B,A}} & \downarrow{\scriptstyle \mathsf{ap}_{A\multimap B,A}} \\ & & B \end{array} \qquad \begin{array}{ccc} B & \xrightarrow{\mathsf{co}-\mathsf{app}_{A,A\multimap B}} & A \multimap (A \otimes B) \\ & \searrow^{\mathsf{co}-\mathsf{ap}_{A,B}} & \downarrow{\scriptstyle \mathsf{id}_A \multimap \mathsf{Wk}_{A,B}} \\ & & A \multimap (A \oslash B) \end{array}$$

Definition 3.4 A sequoidal closed category $(\mathcal{C}, I, \otimes, \oslash, \multimap)$ is cpo-enriched if each hom-set of $\mathcal{C}$ is a cpo (with least element $\bot$) and the functors $\otimes, \oslash$ and $\multimap$ are all continuous.

Note that a simple concrete example of a sequoidal closed category is the category **pcpo** of pointed cpos and continuous functions, in which the $\oslash$ is the right-strict product: $A \oslash B = \{\langle a, b\rangle \in A \times B \mid b = \bot \Rightarrow a = \bot\}$. This lacks, however, the following key property for interpreting references.

Definition 3.5 A *commutative* sequoidal closed category is a sequoidal closed category $(\mathcal{C}, I, \otimes, \oslash, \multimap)$ such that for each pair of objects A, B the functors $A \oslash \_ : \mathcal{C}_S \to \mathcal{C}_S$ and $B \multimap \_ : \mathcal{C}_S \to \mathcal{C}_S$ commute — i.e. there is a natural isomorphism $\mathsf{comm}_{A,B} : A \multimap (B \oslash \_) \to B \oslash (A \multimap \_)$ — *and* for all A the maps $\mathsf{co}-\mathsf{pa}_{A,B} : B \to A \oslash (A \multimap B) = \mathsf{co}-\mathsf{ap}_{A,B}; \mathsf{comm}^{-1}_{A,A,B}$ and $\mathsf{pa} : A \multimap (A \oslash B) \to B = \mathsf{comm}^{-1}_{A,A,B}; \mathsf{ap}_{A,B}$ are the units and co-units of an adjunction $A \oslash \_ \vdash A \multimap \_$.

So in a commutative sequoidal closed category, $A \oslash \_$ and $A \multimap \_$ are both left and right adjoint. We shall make use of the following connection with *traced monoidal categories* [12].

Proposition 3.6 *A commutative sequoidal closed category is a traced monoidal category, with the trace of $f : A \otimes X \to B \otimes X$ given by:*

$$\mathrm{trace}^X_{A,B}(f) : A \to B = \Lambda(f; \mathsf{twist}_{B,X}; \mathsf{Wk}_{X,B}); \mathsf{pa}_{X,B}.$$

We also note that because right-adjoints preserve finite limits, in a commutative sequoidal closed category with (cartesian) products we will have an isomorphism: $\mathrm{dist}_{A,B,C} : (A \oslash B) \times (A \oslash C) \to A \oslash (B \times C)$.

Example 3.7 Let $\mathbf{Rel}_\perp$ be the category of pointed sets and "right-strict" relations: a relation $R \subseteq S \times T$ is right-strict if $xR\perp$ implies $x = \perp$. Then $\mathbf{Rel}_\perp$ is a commutative sequoidal closed category — the tensor product is given by cartesian product of pointed sets, and both $\oslash$ and $\multimap$ are given by the right-strict-product of pointed sets. ($\mathbf{Rel}_\perp$ also has finite products given by the coalesced sum of pointed sets.)

3.1 The exponential

We now need to characterise the structure on a sequoidal closed category $\mathcal{C}$ which is required to define a cartesian closed category (and hence recursive behaviour, since in a traced monoidal category with contraction this can be implemented by cyclic sharing [10]). Essentially, we require a co-monad $! : \mathcal{C} \to \mathcal{C}$ such that $!A \cong !A \oslash A$. Thus $!A$ is an infinite *sequence* or stream of copies of A or "threads". More specifically, we require a linear category [7] — a SMCC with a monoidal comonad $(!, (\_)^\dagger, \mathtt{der})$ with a natural transformation $\mathtt{con} : !\_ \to !\_ \otimes !\_$ such that for all $f : !A \to B$. $f^\dagger; \mathtt{con}_B = \mathtt{con}_A; (f^\dagger \otimes f^\dagger)$.

Definition 3.8 A linear store (LS) category is a tuple $(\mathcal{C}, I, \otimes, \oslash, \times, !)$ such that $(\mathcal{C}, I, \otimes, \oslash, \times)$ is a commutative, cpo-enriched sequoidal closed category with cartesian products, and $(\mathcal{C}, I, \otimes, \multimap, !)$ is a linear category such that $\mathtt{out}_A : !A \cong !A \oslash A : \mathtt{in}_A$ where $\mathtt{out}_A = \mathtt{con}_A; (\mathsf{id} \oslash \mathtt{der}_A)$.

This suggests that $!A$ should be constructed as the least fixpoint of the functor $F(X) = X \oslash A$, and we shall now give sufficient conditions for this to be possible.

Definition 3.9 Let $(\mathcal{C}, I, \otimes, \oslash)$ be a sequoidal category. The tensor product on $\mathcal{C}$ is *sequentially decomposable* if for every pair of maps $f : A \to B \oslash C$ and $g : A \to C \oslash B$ there exists a unique map $\mathsf{sym}(f, g) : A \to B \otimes C$ such that $\mathsf{sym}(f, g); \mathsf{Wk} = f$ and $\mathsf{sym}(f, g); \mathsf{twist}_{B,C}; \mathsf{Wk}_{B,C} = g$.

Thus in a sequoidal closed category with finite products, the monoid is sequentially decomposable if and only if the map $\langle \mathsf{Wk}_{A,B}, \mathsf{twist}_{A,B}; \mathsf{Wk}_{B,A} \rangle : A \otimes B \to (A \oslash B) \times (B \oslash A)$ is an isomorphism. Returning to our analogy between sequoidal categories as models of a "logic of events", we can interpret the cartesian product as external choice — $A \times B$ means A occurs or B occurs. So sequential decomposability means "A and B" is equivalent to "A after B or B after A".

Recall that a *minimal invariant* [17] for a functor $F : \mathcal{C} \to \mathcal{C}$ on cpo-enriched categories is an object $\Delta(F)$ such that there is an isomorphism $\mathtt{out} : \Delta(F) \cong F(\Delta(F)) : \mathtt{in}$, and $\mathsf{id}_{\Delta(F)}$ is the least fixpoint of the operation which takes $f : \Delta(F) \to \Delta(F)$ to $\mathtt{out}; F(f); \mathtt{in}_A : \Delta(F) \to \Delta(F)$.

Proposition 3.10 *Let $(\mathcal{C}, I, \otimes, \oslash, \multimap, \times)$ be a commutative sequoidal closed category with cartesian products, such that $\otimes$ is sequentially decomposable. Suppose for each object A, there is a minimal invariant $!A$ for the functor $\_ \oslash A$. Then $(\mathcal{C}, I, \otimes, \oslash, \multimap, \times, !)$ is a LS-category.*

Proof. We define a monoidal co-monad with contraction maps as follows:

- $\mathtt{der}_A : !A \to A = \mathtt{out}_A; \mathsf{proj}_{!A,A}$,
- $\mathtt{con}_A : !A \to !A \otimes !A$ is the least fixed point of an operation $\Phi_A : \mathcal{C}(!A, !A \otimes !A) \to \mathcal{C}(!A, !A \otimes !A)$, — i.e. $\mathtt{con}_A = \bigsqcup_{i \in \omega} \Phi_A^i(\bot_{!A,!A\otimes!A})$, where

  $$\Phi_A(f) = \mathtt{out}_A; (f \oslash \mathsf{id}_A); \mathsf{passoc}^{-1}_{!A,!A,A}; \mathsf{sym}(\mathsf{id}_{!A} \oslash \mathtt{in}_A, \mathsf{pcomm}_{!A,!A,A}; (\mathsf{id}_{!A} \oslash \mathtt{in}_A)).$$

- For each $f : !A \to B$, we define $f^\dagger : !A \to !B$ by taking the least fixpoint of $\Psi_f : \mathcal{C}(!A, !B) \to \mathcal{C}(!A, !B)$, where $\Psi_f(g) = \mathtt{con}_A; g \otimes f; \mathsf{Wk}_{!A,A}; \mathtt{in}_A$ — i.e. $f^\dagger = \bigsqcup_{i \in \omega} \Psi_f^i(\bot)$.

The monoidal natural transformations $m_I : I \to !I$ and $m_{A,B} : !(A \otimes B) \to !A \otimes !B$ are similarly defined by taking fixpoints of continuous maps between hom-sets. □

4 A Sequoidal category of Games

We shall now describe a LS-category of games and strategies; in Sections 5 and 6 we shall show that this category contains a fully abstract model of $\lambda_{\mathtt{ref}}$. Unlike Hyland-Ong games [11] (and related models, including [3]), this is not based on "justified sequences". Instead we have introduced notions of *triggering* and *blocking* to determine which combinations of moves are legal. (The analogy between triggering and *enabling* will be apparent to those familiar with Hyland-Ong games.)

Definition 4.1 A game A is a tuple $\langle M_A, \lambda_A, S_A, \triangleright_A, \triangleleft_A \rangle$ where:

- M_A is a set of moves,
- $\lambda_A : M_A \to \{O, P\}$ is Player/Opponent labelling (by convention, $\overline{O} = P, \overline{P} = O$),
- $S_A \subseteq M_A$ is a set of Opponent moves — the *starting* moves,
- $\triangleright \subseteq M_A \times M_A$ is a binary relation called *triggering* ($m \triangleright n$ means m is a trigger for n),
- $\triangleleft_A \subseteq M_A \times M_A$ is a binary relation on moves called *blocking*.

Definition 4.2 For any game A, the set of legal sequences L_A is the prefix-closed set of finite sequences of moves in M_A satisfying the following conditions.

Alternation Player and Opponent moves alternate:
$sab \in L_A \Longrightarrow \lambda_A(a) = \overline{\lambda_A(b)}$.

Triggering Every non-starting move is preceded by at least one trigger:
$tb \in L_A \wedge b \notin S_A \Longrightarrow \exists sa \sqsubseteq tb.a \triangleright_A b$.

Blocking No move is preceded by a move which can block it::
$tb \in L_A \Longrightarrow \forall sa \sqsubseteq tb.\neg a \triangleleft_A b$.

In all of the games which will concern us, every move will be self-blocking; the blocking condition entails that all legal sequences over such games will contain no repeated moves.

We define operations $\otimes, \multimap, \&$ along the lines of previous games semantics. $A \otimes B$ inherits its labelling, triggering and blocking from A and B, so a legal sequence in $A \otimes B$ consists of interleaved sequences in A and B (which are not necessarily legal as they may fail the alternation condition). $A \multimap B$ is similar, except that polarities in A are interchanged, and the only starting moves are those from B, which become triggers for the starting moves of A. In the additive product $A\&B$, starting moves in A are blocked by starting moves from B and vice-versa — thus a legal sequence in $A\&B$ is a sequence from A, or from B.

Definition 4.3 Given games A, B, form:

- $A \otimes B = (M_{A\otimes B}, \lambda_{A\otimes B}, S_A \otimes B, \triangleright_{A\otimes B}, \triangleleft_{A\otimes B})$, where $M_{A\otimes B} = M_A + M_B$, $\lambda_{A\otimes B} = [\lambda_A, \lambda_B]$, $S_{A\otimes B} = S_A + S_B$, $\triangleright_{A\otimes B} = \mathsf{in_l}(\triangleright_A) \cup \mathsf{in_r}(\triangleright_B)$ and $\triangleleft_{A\otimes B} = \mathsf{in_l}(\triangleleft_A) \cup \mathsf{in_r}(\triangleleft_B)$.
- $A \multimap B = (M_{A\multimap B}, \lambda_{A\multimap B}, S_{A\multimap B}, \triangleright_{A\multimap B}, \triangleleft_{A\multimap B})$, where $M_{A\multimap B} = M_A + M_B$, $S_{A\multimap B} = S_B$, $\lambda_{A\multimap b} = [\overline{\lambda_A}, \lambda_B]$, $\triangleright_{A\multimap B} = \mathsf{in_l}(\triangleright_A) \cup \mathsf{in_r}(\triangleright_B) \cup (\mathsf{in_l}(S_B) \times \mathsf{in_r}(S_A))$ and $\triangleleft_{A\multimap B} = \mathsf{in_l}(\triangleleft_A) \cup \mathsf{in_r}(\triangleleft_B)$.
- $A\&B = (M_{A\&B}, \lambda_{A\&B}, S_{A\&B}, \triangleright_{A\&B}, \triangleleft_{A\&B})$, where $M_{A\&B} = M_A + M_B$, $\lambda_{A\&B} = [\lambda_A, \lambda_B]$, $\triangleright_{A\&B} = \mathsf{in_l}(\triangleright_A) \cup \mathsf{in_r}(\triangleright_B)$, $S_{A\&B} = S_A + S_B$, $\triangleleft_{A\&B} = \mathsf{in_l}(\triangleleft_A) \cup \mathsf{in_r}(\triangleleft_B) \cup (\mathsf{in_l}(M_A) \times \mathsf{in_r}(M_B)) \cup (\mathsf{in_r}(M_B) \times \mathsf{in_l}(M_A))$.

(If $R \subseteq A_1 \times B_1$ is a relation, then $\mathsf{in_l}(R) \subseteq (A_1 + A_2) \times (B_1 + B_2)$ is the relation $\{\langle \mathsf{in_l}(a), \mathsf{in_l}(b)\rangle \mid \langle a, b\rangle \in R\}$.)

The *strategies* on a game A are the even-prefix-closed and evenly-branching sets of even-length legal sequences over A. Composition of strategies by "parallel composition plus hiding" follows the standard definition.

Definition 4.4 From strategies σ on $A \multimap B$ and τ on $B \multimap C$ form $\sigma; \tau$ on $A \multimap C$:
$\{s \in L_{A\multimap C} \mid \exists t \in (M_A + M_B + M_C)^*.s = t\restriction A, C \wedge t\restriction A, B \in \sigma \wedge t\restriction B, C \in \tau\}$.

Thus we can form a category $\mathcal{G}$ in which the objects are games and the morphisms from A to B are strategies on $A \multimap B$.

Proposition 4.5 *$(\mathcal{G}, I, \otimes, \multimap)$ is a symmetric monoidal closed category.*

Proof. The requisite isomorphisms are all *copycat strategies*. □

$\mathcal{G}$ is cpo-enriched by the inclusion order on strategies, so the $\perp$ maps are

the empty strategies. Thus the strict strategies from A to B are those which respond to the opening move in B with a move in A.

The sequoidal product $A \oslash B$ differs only from $A \otimes B$ in that the starting moves are the starting moves of B, which act as triggers for the starting moves in A. (So $A \oslash B$ differs from $A \multimap B$ only in that the polarity of moves in A is not inverted.)

Definition 4.6 From games A, B, define $A \oslash B = (M_{A \oslash B}, \lambda_{A \oslash B}, S_{A \oslash B}, \triangleright_{A \oslash B}, \triangleleft_{A \oslash B})$, where $M_{A \oslash B} = M_A + M_B$, $\lambda_{A \oslash B} = [\lambda_A, \lambda_B]$, $S_{A \oslash B} = S_B$, $\triangleright_{A \oslash B} = \mathsf{in_l}(\triangleright_A) \cup \mathsf{in_r}(\triangleright_B) \cup (\mathsf{in_r}(S_B) \times \mathsf{in_l}(S_A))$ and $\triangleleft_{A \oslash B} = \mathsf{in_l}(\triangleleft_A) \cup \mathsf{in_r}(\triangleleft_B)$.

In $\mathcal{G}$, the tensor $\otimes$ is sequentially decomposable: the isomorphism $(A \oslash B) \& (B \oslash A) \to B \otimes A$ is the strategy which chooses the left-hand component of $(A \oslash B) \& (B \oslash A)$ if Opponent starts in the left-hand part of $B \otimes A$, and vice-versa. Since we can also solve domain equations as described in [16] — in particular, finding a minimal invariant $!A = !A \oslash A$ — by Proposition 3.10 we have a LS-category. However, we can give a simple description of $!A$ more directly, as an ω-indexed product of copies of A, in which the starting moves for the i-indexed copy act as triggers for the starting moves in the $i+1$-indexed copy.

Definition 4.7 From a game A, define $!A = \langle M_{!}A, \lambda_{!A}, S_{!A} \triangleright_{!A}, \triangleleft_{!A} \rangle$, where $M_{!A} = \coprod_{i \in \omega} M_A = \omega \times M_A$, $\lambda_{!A} = [\lambda_A \,|\, i \in \omega]$, $S_{!A} = S_A \times \{1\}$, $\triangleright_{!A} = (\bigcup_{i \in \omega} \mathsf{in}_i(\triangleright_A)) \cup (\bigcup_{i \in \omega} (\mathsf{in}_i(S_A) \times \mathsf{in}_{i+1}(S_A)))$, $\triangleleft_{!A} = \bigcup_{i \in \omega} \mathsf{in}_i(\triangleleft_A)$.

Proposition 4.8 *$(\mathcal{G}, I, \otimes, \oslash, \multimap, \times, !)$ is a LS-category.*

5 Semantics of $\lambda_{\mathtt{ref}}$ in a LS-category

Our final requirement for modelling $\lambda_{\mathtt{ref}}$ is a non-terminal object ι to be the denotation of the base type — in $\mathcal{G}$ we shall take the minimal such object (denoted o), the game with a single, starting, self-blocking Opponent move. We shall write Σ for the object $\iota \multimap \iota$; by non-terminality of ι, $\mathcal{C}(I, \Sigma)$ contains at least two elements — $\bot$, and $\top = \Lambda(\mathsf{li}_\iota)$.

Types of $\lambda_{\mathtt{ref}}$ are interpreted in standard fashion: $[\![\mathbf{0}]\!] = \iota, [\![S \Rightarrow T]\!] = ![\![S]\!] \multimap [\![T]\!]$. For each type-object $[\![T]\!]$ we can derive a "sequential composition" morphism $\mathsf{seq}_T : \Sigma \otimes [\![T]\!] \to [\![T]\!]$ — $\mathsf{seq_0} = \mathsf{app}_{\iota,\iota}$, and $\mathsf{seq}_{S \Rightarrow T} = \Lambda((\mathsf{id}_\Sigma \otimes \mathsf{app}_{![\![S]\!],[\![T]\!]}); \mathsf{seq}_T)$.

To interpret references, we define a store modality (reminiscent of [19]); a functor $\S : \mathcal{C} \to \mathcal{C}$, with natural transformations $\mathsf{assign}_A : \S A \otimes !A \to \Sigma$ and $\mathsf{deref}_A : \S A \to A$ and interpret terms-in-context $x_1 : S_1, \ldots, x_m : S_m \vdash M : T$ as morphisms from $!(\S[\![S_1]\!] \times \ldots \times \S[\![S_m]\!])$ to $[\![T]\!]$.
So we can define $[\![x : T, \Gamma \vdash x := M]\!] : [\![T, \Gamma]\!] \to \Sigma =$

$$(\mathtt{con}_{[\![T,\Gamma]\!]}; (\mathtt{der}_{[\![T,\Gamma]\!]}; \pi_1) \otimes [\![x : T, \Gamma \vdash M : T]\!]); \mathsf{assign}_{[\![T]\!]}$$

and thus

$$[\![x : T, \Gamma \vdash x := M.N]\!] = \mathsf{con}_{[\![T,\Gamma]\!]}; ([\![x : T, \Gamma \vdash x := M]\!] \otimes [\![x : T, \Gamma \vdash N : S]\!]); \mathsf{seq}_S)$$

$$[\![x : T, \Gamma \vdash x : T]\!] = \mathsf{der}_{[\![T,\Gamma]\!]}; \pi_1; \mathsf{deref}_{[\![T]\!]}$$

In fact (as in [2,3]), $\S A$ is simply a product of the assign and dereference "methods" on A — i.e. $\S A = A \times (!A \multimap \Sigma)$, and $\mathsf{assign}_A = \pi_r \otimes \mathsf{id}_A; \mathsf{ap}_{!A,\Sigma}$ and $\mathsf{deref}_A = \pi_l$.

To interpret λ-abstraction — which involves declaring a new storage cell and initializing it — we define morphisms $\mathsf{cell}_A : !A \to !\S A$ for each A — so that for any $f : !A \to B$, $f^\dagger; \mathsf{cell}_B : !A \to !\S B$ represents a reference cell with f assigned to it. We define cell_A by taking the least fixed point of an operation $\Theta_A : \mathcal{C}(!A, !\S A) \to \mathcal{C}(!A, !\S A)$, i.e. $\mathsf{cell}_A = \bigsqcup_{i \in \omega} \Theta^i(\bot_{!A,\S A})$, where:

$$\Theta_A(f) = \langle t_{!A}; \top; \mathsf{pa}; (f \oslash \mathsf{id}_{!A \multimap \Sigma}), \mathtt{out}_A; f \oslash \mathsf{id}_A \rangle; \mathrm{dist}_{\S A, A, !A \Rightarrow \Sigma}; \mathtt{in}_{\S A}$$

Note that the key behaviour of cell_A is captured by the unit of the adjunction $A \multimap \vdash A \oslash \_ \dashv \top; \mathsf{pa}_{A,\Sigma} : I \to A \oslash (A \multimap \Sigma)$ — this represents a kind of write once, read once instance of cell. (In $\mathcal{G}$, this is the strategy which responds to Opponent's first move (in Σ) with the corresponding P-move in Σ, and thereafter plays copycat between the two copies of A.)

The semantics of λ-abstraction can now be given by composition with cell:

$$[\![\Gamma \vdash \lambda x : T.M]\!] = \Lambda((\mathsf{cell}_{[\![T]\!]} \otimes \mathsf{id}_{[\![\Gamma]\!]}); [\![x : T, \Gamma \vdash M]\!])$$

To establish soundness of the semantics, we first define the interpretation of a term M in an environment $\mathcal{E}$.

Definition 5.1 Given terms $\Gamma, \Delta \vdash M : \mathbf{0}$, and $\Gamma, \Delta \vdash N_1 : T_1, \Gamma, \Delta \vdash T_n$, where $\Delta = a_1 : T_1, \ldots a_n : T_n$ let $\mathcal{E} = (a_1, N_1), \ldots, (a_n, N_n)$, and define $[\![M, \mathcal{E}]\!] : [\![\Gamma]\!] \to \iota =_{df} \mathrm{trace}^{[\![\Delta]\!]}_{[\![\Gamma]\!],\iota}(\mathtt{con}^{i+1}_{[\![\Gamma,\Delta]\!]}; ([\![M]\!] \otimes ([\![N_1]\!]^\dagger; \mathsf{cell}_{[\![T_1]\!]} \otimes \ldots \otimes [\![N_n]\!]^\dagger; \mathsf{cell}_{[\![T_n]\!]}))$.

Proposition 5.2 *If $M, \mathcal{E} \longrightarrow M', \mathcal{E}'$ then $[\![M, \mathcal{E}]\!] = [\![M', \mathcal{E}']\!]$.*

The proof is based on the following lemma:

Lemma 5.3 *Assignment: for any $f, g : !A \to B$:*

$$\mathtt{con}_A; (f \oslash (g; \mathsf{cell}_B; \mathtt{out}_{\S B})); \mathsf{pcomm}_{A,!\S A,\S A}; (\mathsf{id}_{!\S A} \oslash \mathsf{assign}_A) = \mathsf{l}_A; (f; \mathsf{cell}_B \otimes \top)$$

Dereferencing: for any $f : !A \to B$:

$$f^\dagger; \mathsf{cell}_B; \mathtt{out}_{\S B}; (\mathsf{id} \oslash \mathsf{deref}_B) : !A \to !\S B \oslash B = \mathtt{out}_A; ((f^\dagger; \mathsf{cell}_B) \oslash f)$$

We can also show that the model of $\lambda_{\mathtt{ref}}$ in a LS-category is *computationally adequate*. First note that non-terminality of ι implies that $\mathsf{id}_\iota \neq \bot_{\iota,\iota}$.

Proposition 5.4 *Let $x : \mathbf{0} \vdash M : \mathbf{0}$ be a $\lambda_{\mathtt{ref}}$ term such that $[\![M]\!] \neq \bot$, then $M \Downarrow$.*

The proof proceeds by showing that operationally, $\lambda_{\mathtt{ref}}$ can be viewed as the "limit" of $\lambda^n_{\mathtt{ref}} : n \in \omega$, where $\lambda^n_{\mathtt{ref}}$ is a version of $\lambda_{\mathtt{ref}}$ in which each location can be dereferenced at most n times.

Definition 5.5 A $\lambda^n_{\mathtt{ref}}$ environment $\mathcal{E}$ is a set of triples, (l, M, m) consisting of a location, its contents, and an integer bound on the number of future occasions on which it can be dereferenced. The latter is initialized to n, and decremented by 1 each time the contents of the location are required. Thus the key reduction rules of $\lambda^n_{\mathtt{ref}}$ are:

$$\lambda x.M\ N, \mathcal{E} \longrightarrow E[M[a'/a]], \mathcal{E}' \cup \{(a, N, n)\} \quad : a' \notin \pi_1(\mathcal{E}))$$
$$(a\, N_1) \ldots N_k, \mathcal{E} \longrightarrow (M\, N_1) \ldots N_k, \mathcal{E}[(a, M, i)/(a, M, i+1)] \quad : (a, M, i+1) \in \mathcal{E}$$

We shall write $M \Downarrow^n$ if evaluation of M in $\lambda^n_{\mathtt{ref}}$ terminates at a head normal form. Clearly, if $M \Downarrow^n$, then $M \Downarrow$.

Proposition 5.6 *For any n, every reduction of $\lambda^n_{\mathtt{ref}}$ terminates.*

Proof. We give a sound translation $(\_)^n$ of $\lambda^n_{\mathtt{ref}}$ terms-in-environments into $\lambda^n_{\mathtt{ref}}$ terms, and show that $\lambda_{\mathtt{ref}}$ reduction strictly reduces the length of the translated terms. Let $(x := M)^n$ be the sequential composition of n copies of $x := n$ — i.e. $(x := M)^1 =_{df} x := M$, $(x := M)^{n+1} = x := M.(x := M)^n$. Then for an environment $\mathcal{E} = (a_1, N_1, m_1), \ldots, (a_k, N_k, m_k)$ and program M, we define $(M, \mathcal{E})^n = (a_1 := N_1)^{m_1}. \ldots .(a_k := N_k)^{m_k}.M^n$, where M^n is the program obtained by replacing each assignment $x := L$ in M with $(x := L)^n$, and each application $M\, N$ by $\lambda y.(y := N)^{2n}.(M\, N)\, N$. It is straightforward to show that $M, \mathcal{E} \Downarrow^n$ if and only if $(M, \mathcal{E})^n \Downarrow^n$. Finally, we observe that if $M, \mathcal{E} \longrightarrow M', \mathcal{E}'$, then $(M', \mathcal{E}')^n$ is strictly shorter than $(M, \mathcal{E})^n$ and hence reduction of $\lambda^n_{\mathtt{ref}}$ must terminate. □

For each term M, we define an approximant $[\![M]\!]_n$ for each n by replacing the $\mathsf{cell}_{[\![T]\!]}$ strategy used to interpret λ-abstraction in M by the approximant $\mathsf{cell}^n_{[\![T]\!]} = \Theta^n(\bot)$. The proof of the following lemma follows that of Proposition 5.2.

Lemma 5.7 *If $M, \mathcal{E} \longrightarrow_i M', \mathcal{E}'$, then $[\![M, \mathcal{E}]\!]_i \subseteq [\![M', \mathcal{E}']\!]_i$.*

Corollary 5.8 *If $[\![M]\!]^i \neq \bot$, then $M \Downarrow^i$.*

Thus to prove Proposition 5.4 suppose $[\![M]\!] \neq \bot$. Since $[\![M]\!] = \bigsqcup_{i \in \omega} [\![M]\!]_i$, $[\![M]\!]_i \neq \bot$ for some i. By Corollary 5.8, $M \Downarrow^i$ and hence $M \Downarrow$. Inequational soundness of our model of $\lambda_{\mathtt{ref}}$ now follows by a standard argument.

Corollary 5.9 (Inequational Soundness) *Let $\mathcal{C}$ be a LS-category, then for any terms $M, N : T$, if $[\![M]\!]_{\mathcal{C}} \sqsubseteq [\![N]\!]_{\mathcal{C}}$ then $M \lesssim N$.*

6 Full Abstraction

It is relatively straightforward to establish full abstraction for the model of $\lambda_{\mathtt{ref}}$ in $\mathcal{G}$, since we can show that every legal sequence over a type-object

corresponds to a unique HO-style "justified sequence" (a form of "pointifixion" [5]). We can use this fact to define notions of *view* and *innocence*, and show that every finite strategy is definable using factorization theorems as in [3]. However, to give a more general characterization of the LS-categories which give rise to fully abstract models of $\lambda_{\mathtt{ref}}$ requires a different approach. We will give a simple axiomatic characterization of a notion of *sequential* LS-category (of which $\mathcal{G}$ is an example), and show that we can give a *decomposition theorem* for morphisms in sequential categories, analogous to the the decompositions of compact strategies used to prove definability in PCF [4,11] and axiomatically in [1] (except with the essential difference that our decomposition gives rise to a full abstraction result, without the need for any extensional quotient).

Our axioms are also similar to those in [1], (but somewhat simpler, since much of the decomposition can be accomplished using properties of the ! which hold in all LS-categories). They take the general form of an assertion that a natural transformation between functors from $(\mathcal{C}^{OP})^m \times \mathcal{C}^n$ into **pcpo** (the category of pointed cpos and strict continuous functions) which exists in all LS-categories is an isomorphism in sequential LS-categories.

Definition 6.1 An object ι in a LS-category $\mathcal{C}$ is *atomic* if the following conditions hold:

- (i) $\mathcal{C}(A, B \oslash \iota) = \mathcal{C}_S(A, B \oslash \iota)$ — i.e. for any A, B, every $f : A \to B \oslash \iota$ is strict.
- (ii) ι is "π-atomic" in the sense of [1]: for every A, B: $\mathcal{C}(A \times B, \iota) \cong \mathcal{C}(A, \iota) \oplus \mathcal{C}(B, \iota)$ — i.e. the map $\mathcal{C}(\_, \iota) : \mathcal{C}^{op} \to \mathbf{pcpo}$ preserves co-products (coalesced sums in **pcpo**).
- (iii) $\mathcal{C}_S(A \multimap \iota, B \multimap \iota) \cong (\mathcal{C}(B, A))_\perp$ — i.e. the natural transformation from $\mathcal{C}(\__2, \__1)_\perp$ to $\mathcal{C}_S(\__1 \multimap \iota, \__2 \multimap \iota)$ which sends $\perp$ to $\perp$ and $f_\perp$ to $f \multimap \mathsf{id}_\iota : B \multimap \iota \to A \multimap \iota$ is an isomorphism.

A LS-category is *sequential* if it contains an atomic object.

Proposition 6.2 *$\mathcal{G}$ is a sequential LS-category, with atom o.*

Proof. Straightforward — (i) holds by the alternation condition, since there are no P-moves in $A \oslash o$ triggered by the opening move. (ii) holds by determinacy of strategies and the blocking condition, since the first P move in $A \& B \multimap o$ must be in A (and so block the starting moves in B) or in B (and so block the starting moves in A). (iii) holds because any strict, non-$\perp$ strategy from $A \multimap o$ to $B \multimap o$ can be converted to a strategy from A to B by removing the first two moves. □

We shall also use a lemma which applies in all LS-categories.

Lemma 6.3 *In any LS-category, $!A \multimap !B \trianglelefteq !(!A \multimap B)$.*

Proof. In any linear category there is a morphism $\Lambda((\mathtt{der}_{!A \multimap B}; \mathtt{app}_{!A \multimap B, !A})^\dagger) : !(!A \multimap B) \to !A \multimap !B$. In an LS-category, we also have a right-inverse:

$(\Lambda(\mathsf{app}_{!A\multimap !B,!A};\mathtt{out}_B));\mathsf{comm}_{!A,!B};(K^{\dagger}_{B,}\oslash\mathsf{id}_{!A\multimap B}) : !A \multimap !B \to !(!A \multimap B)$ (where $K : !B \to !A \multimap B = \Lambda(\mathsf{proj}_{A,B})$). Hence in an LS-category, for any A, B: $\mathcal{C}(!A, !B) \trianglelefteq \mathcal{C}(I, !(!A \multimap B))$. □

The key to our proof of full abstraction for sequential categories is the following decomposition lemma, which also asserts that a natural transformation between functors into **pcpo** is a retraction. Note that in the latter category we have the identity $\Sigma_{i\in I} A_i = (\coprod_{i\in I} A_i)_\perp$, where Σ is the lifted sum. We shall write $\vec{A}$ for $\Pi_{i\leq n} A_i$, $\vec{B_i}$ for $\Pi_{j\leq m_i} B_{ij}$, etc.

Lemma 6.4 *Let $\mathcal{C}$ be a sequential LS-category, and let $A_1, \ldots, A_n$ be objects of $\mathcal{C}$ such that $A_i = !B_{i1} \multimap \ldots \multimap !B_{im_i} \multimap \iota \cong !\vec{B_i} \multimap \iota$, for each $i \leq n$, and $B_i j = !C_{ij1} \multimap \ldots \multimap !C_{ijl_{ij}} \multimap \iota$ for each $j \leq m_i$. Then:*

$$\mathcal{C}(I, !(\Pi_{i\leq n} A_i)) \trianglelefteq \prod_{i\leq n} \sum_{j\leq m_i} \mathcal{C}(I, !(\Pi_{i\leq n}(!\vec{B_{ij}} \multimap A_i) \times \Pi_{k\leq l_{ij}}(!\vec{B_{ij}} \multimap C_{ijk})))$$

More precisely, in all LS-categories there is a continuous map

$$\phi_{\vec{A}} : \prod_{i\leq n} \sum_{j\leq m_i} \mathcal{C}(I, !(\Pi_{i\leq n}(!\vec{B_{ij}} \multimap A_i) \times \Pi_{k\leq l_{ij}}(!\vec{B_{ij}} \multimap C_{ijk}))) \to \mathcal{C}(I, !(\Pi_{i\leq n} A_i))$$

and in sequential categories this has a left inverse $\phi^{-1}_{\vec{A}}$ such that $\phi^{-1}_{\vec{A}}; \phi_{\vec{A}} = \mathsf{id}$.

Proof. $\mathcal{C}(I, !\vec{A}) \cong \mathcal{C}(I, !\vec{A} \oslash \vec{A})$, since $!\vec{A} \cong !\vec{A} \oslash \vec{A}$,
$\mathcal{C}(I, !\vec{A} \oslash (\Pi_{i\leq n} A_i)) \cong \mathcal{C}(I, \Pi_{i\leq n}(!\vec{A} \oslash A_i))$ ($A \oslash \_$ is a right adjoint),
$\mathcal{C}(I, \Pi_{i\leq n}(!\vec{A} \oslash A_i)) \cong \prod_{i\leq n} \mathcal{C}(I, !\vec{A} \oslash A_i)$
$\mathcal{C}(I, !\vec{A} \oslash !\vec{B_i} \multimap \iota) \cong \mathcal{C}(I, !\vec{B_i} \multimap (!\vec{A} \oslash \iota))$ by commutativity of $\multimap$ and $\oslash$,
$\mathcal{C}(!\vec{B_i}, (!\vec{A} \oslash \iota)) \cong \mathcal{C}_S(!\vec{B_i}, (!\vec{A} \oslash \iota))$ by ax. (i),
$\mathcal{C}_S(!\vec{B_i}, (!\vec{A} \oslash \iota)) \cong \mathcal{C}_S(!\vec{A} \multimap !\vec{B_i}, \iota)$ by the adjunction $!\vec{A} \multimap \_ \vdash !\vec{A} \oslash \_$,
$\mathcal{C}_S(!\vec{A} \multimap !\vec{B_i}, \iota) \cong \mathcal{C}_S(!\vec{A} \multimap (!\vec{B_i} \oslash \vec{B_i}), \iota)$ since $!\vec{B_i} \cong !\vec{B_i} \oslash \vec{B_i}$,
$\mathcal{C}_S(!\vec{A} \multimap (!\vec{B_i} \oslash \Pi_{j\leq m_i}(!\vec{C_i j} \multimap \iota)), \iota) \cong \mathcal{C}_S(\Pi_{j\leq m_i}(!\vec{A} \multimap (!\vec{B_i} \oslash (!\vec{C_i j} \multimap \iota))), \iota)$,
$\mathcal{C}_S(\Pi_{j\leq m_i}(!\vec{A} \multimap (!\vec{B_i} \oslash (!\vec{C_i j} \multimap \iota))), \iota) \cong \coprod_{j\leq m_i} \mathcal{C}_S(!\vec{A} \multimap (!\vec{B_i} \oslash (!\vec{C_i j} \multimap \iota)), \iota)$,
by ax. (ii),
$\mathcal{C}_S(!\vec{A} \multimap (!\vec{B_i} \oslash (!\vec{C_i j} \multimap \iota)), \iota) \cong \mathcal{C}_S(!\vec{B_i} \oslash (!\vec{A} \multimap (!\vec{C_i j} \multimap \iota)), \iota)$ since $\multimap$ and $\oslash$ commute,
$\mathcal{C}_S(!\vec{B_i} \oslash (!\vec{A} \multimap (!\vec{C_i j} \multimap \iota)), \iota) \cong \mathcal{C}_S(!\vec{B_i} \oslash ((!(\vec{A} \times \vec{C_i j}) \multimap \iota), \iota)$
$\mathcal{C}_S(!\vec{B_i} \oslash ((!(\vec{A} \times \vec{C_i j}) \multimap \iota), \iota) \cong \mathcal{C}_S((!(\vec{A} \times \vec{C_i j}) \multimap \iota, !\vec{B_i} \multimap \iota)$,
$\mathcal{C}_S(!(\vec{A} \times \vec{C_i j}) \multimap \iota, !\vec{B_i} \multimap \iota) \cong (\mathcal{C}(!\vec{B_i}, !(\vec{A} \times \vec{C_i j})))_\perp$ by ax. (iii),
$\mathcal{C}(!\vec{B_i}, !(\vec{A} \times \vec{C_i j})) \trianglelefteq \mathcal{C}(I, !(!\vec{B_i} \multimap (\vec{A} \times \vec{C_i j})))$ by Lemma 6.3.
Hence $\mathcal{C}(I, !\vec{A}) \trianglelefteq \prod_{i\leq n}(\coprod_{j\leq m_i}(\mathcal{C}(I, !(!\vec{B_i} \multimap (\vec{A} \times \vec{C_i j})))))_\perp$
$\cong \prod_{i\leq n} \sum_{j\leq m_i} \mathcal{C}(I, !(\Pi_{i\leq n}(!\vec{B_{ij}} \multimap A_i) \times \Pi_{k\leq l_{ij}}(!\vec{B_{ij}} \multimap C_{ijk})))$, as required. □

We shall use the decomposition lemma to define a notion of *decomposition tree* for each $f : I \to !(\llbracket T_1 \rrbracket \times \ldots \times \llbracket T_n \rrbracket)$, where $T_1, \ldots, T_n$ are $\lambda_{\mathtt{ref}}$ types.

This can be seen as a finite-branching (but possibly infinite depth) tree with branches and nodes each labelled with a natural number. (In the decomposition tree of a strategy in $\mathcal{G}$, branches correspond to O-moves and nodes to P-moves.)

Definition 6.5 Given a finite list of $\lambda_{\mathtt{ref}}$-types $T_1, \ldots, T_n$, where $T_i = S_{i1} \Rightarrow \ldots \Rightarrow S_{im_i} \Rightarrow \mathbf{0}$ for each $i \leq n$, and $S_{ij} = R_{ij1} \Rightarrow \ldots \Rightarrow R_{ijl_{ij}} \Rightarrow \mathbf{0}$ for each $j \leq m(i)$, let $\mathsf{D}_{ij}(T_1, \ldots, T_n)$ be the finite list of types: $\vec{S_i} \Rightarrow T_1, \ldots \vec{S_i} \Rightarrow T_n, \vec{S_i} \Rightarrow R_{ij1}, \ldots, \vec{S_i} \Rightarrow R_{ijl_{ij}}$.

Definition 6.6 For each $i \leq \omega$, and each finite sequence of types $T_1, \ldots T_n$, we shall define a pointed cpo $\mathsf{DT}^i(\vec{T})$, such that $i \leq j$ implies $\mathsf{DT}^i(\vec{T}) \trianglelefteq \mathsf{DT}^j(T)$.
$\mathsf{DT}^0(\vec{T}) = \{\bot\}$ for all $\vec{T}$,
$\mathsf{DT}^{i+1}(\vec{T}) = \prod_{i \leq n} \Sigma_{j \leq m_)} \mathsf{DT}^i(\mathsf{D}_{ij}(\vec{T}))$.
$\mathsf{DT}^\omega(\vec{T}) = \bigsqcup_{i \in \omega} \mathsf{DT}^i(\vec{T})$.

Definition 6.7 Let $\mathcal{C}$ be a sequential LS-category. For each sequence of $\lambda_{\mathtt{ref}}$ types $T_1, \ldots, T_n$, and each $k \in \omega$ we define maps $\mathsf{DI}^k_{\vec{T}} : \mathcal{C}(I, !\llbracket T_1 \rrbracket_{\mathcal{C}} \times \ldots \times \llbracket T_n \rrbracket_{\mathcal{C}}) \to \mathsf{DT}^i(T_1, \ldots, T_n)$, and $\mathsf{DP}^k_{\vec{T}} : \mathsf{DT}^k(T_1, \ldots, T_n) \to \mathcal{C}(I, !\llbracket T_1 \rrbracket_{\mathcal{C}} \times \ldots \times \llbracket T_n \rrbracket_{\mathcal{C}})$
$\mathsf{DI}^0_{\vec{T}} = \bot, \quad \mathsf{DP}^0_{\vec{T}} = \bot$
$\mathsf{DI}^{i+1}_{\vec{T}} = \phi^{-1}_{\llbracket \vec{T} \rrbracket}; (\Pi_{i \leq n} \Sigma_{i \leq n} \mathsf{DI}^k_{\mathsf{D}_{ij}(\vec{T})}), \quad \mathsf{DP}^{i+1}_{\vec{T}} = (\Pi_{i \leq n} \Sigma_{i \leq n} \mathsf{DP}^k_{\mathsf{D}_{ij}(\vec{T})}); \phi_{\llbracket \vec{T} \rrbracket}$,
$\mathsf{DI}^\omega_{\vec{T}} = \bigsqcup_{i \in \omega} \mathsf{DI}^i_{\vec{T}}, \quad \mathsf{DP}^\omega_{\vec{T}} = \bigsqcup_{i \in \omega} \mathsf{DP}^i_{\vec{T}}$.

Lemma 6.8 *For any $f : I \to !(\llbracket T_1 \rrbracket \times \ldots \llbracket T_n \rrbracket)$, for all i, $\mathsf{DP}^i_{\vec{T}}(\mathsf{DI}^i_{\vec{T}}(f)) \sqsubseteq f$, and if $\mathsf{DI}^i_{\vec{T}}(f) = \mathsf{DI}^{i+1}_{\vec{T}}$ then $\mathsf{DP}^i_{\vec{T}}(\mathsf{DI}^i_{\vec{T}}(f)) = f$.*

Proof. is by induction on i. □

Lemma 6.9 *If $\mathcal{C}$ is a sequential and ω-algebraic LS-category, then for any $f, g : I \to !(\llbracket T_1 \rrbracket \times \ldots \llbracket T_n \rrbracket)$, $\mathsf{DI}^\omega_{\vec{T}}(f) \leq \mathsf{DI}^\omega_{\vec{T}}(g)$ implies $f \sqsubseteq g$.*

Proof. By Lemma 6.8, and ω-algebraicity. $f = \bigsqcup_{i \in \omega} \mathsf{DP}^i_{\vec{T}}(\mathsf{DI}^i_{\vec{T}}(f))$, and $g = \bigsqcup_{i \in \omega} \mathsf{DP}^i_{\vec{T}}(\mathsf{DI}^i_{\vec{T}}(g))$. So if $f \not\sqsubseteq g$, then there exists i such that for all j, $\mathsf{DI}^i_{\vec{T}}(g) \not\leq \mathsf{DI}^j_{\vec{T}}(g)$ and hence $\mathsf{DI}^\omega_{\vec{T}}(f) \not\leq \mathsf{DI}^\omega_{\vec{T}}(g)$ as required. □

For a map $f : I \to !A_1 \multimap \ldots \multimap !A_n \multimap B$, we shall write $\mathsf{unc}(f) : !(A_1 \times \ldots \times A_n) \to B$ for the uncurried version.

Proposition 6.10 *Let $\mathcal{C}$ be a sequential LS-category. Suppose for some $f, g : I \to !(\llbracket T_1 \rrbracket \times \ldots \times \llbracket T_n \rrbracket)$ that $\mathsf{DI}^\omega_{\vec{T}}(f) \not\leq \mathsf{DI}^\omega_{\vec{T}}(g)$. Then there exists a (closed) $\lambda_{\mathtt{ref}}$ term $M(f,g) : T_1 \Rightarrow \ldots \Rightarrow T_n \Rightarrow (\mathbf{0} \Rightarrow \mathbf{0})$ such that $f; \mathsf{unc}(\llbracket M \rrbracket) = \top$ and $g; \mathsf{unc}(\llbracket M \rrbracket) = \bot$.*

Proof. If $\mathsf{DI}^\omega_{T_1, \ldots T_n}(f) \not\leq \mathsf{DI}^\omega_{T_1, \ldots T_n}(g)$ then there exists a minimal $0 < q < \omega$ such that $\mathsf{DI}^q_{\vec{T}}(f) \not\leq \mathsf{DI}^q_{\vec{T}}(g)$ We define $M(f,g)$ by induction on q.
Suppose $q = 1$ — i.e. there exists $1 \leq i \leq n$ and $1 \leq j \leq m_i$ such that

$\pi_i(\mathsf{DI}^1_{\vec{T}}(f)) = \mathsf{in}_j(\bot)$ and $\pi_(\mathsf{DI}^1_{\vec{T}}(g)) \not\leq \mathsf{in}_j(\bot)$. Then

$$M(f,g) = \lambda\overline{x}.\lambda y : \mathbf{0}.x_j\,\Omega_1 \ldots \Omega_{k-1}\,(\lambda\vec{z}.y)\,\Omega_{j+1}\ldots\Omega_n.$$

Suppose $q > 1$. Then there exists $1 \leq i \leq n$ and $1 \leq j \leq m_i$, and $f_{ij}, g_{ij} : I \to !([\![\mathsf{D}_{ij}(\vec{T})]\!]$ such that $\pi_i(\mathsf{DI}^q(f)) = \mathsf{in}_j(\mathsf{DI}^{q-1}(f_{ij}))$ and $\pi_i(\mathsf{DI}^q(g)) = \mathsf{in}_j(\mathsf{DI}^{q-1}(g_{ij}))$ and $\mathsf{DI}^{q-1}(f_{ij}) \not\leq \mathsf{DI}^{q-1}(g_{ij})$.
Then by induction hypothesis, there is a term $M(f_i, g_j) : (\vec{S_i} \Rightarrow T_1) \Rightarrow \ldots(\vec{S_i} \Rightarrow T_n) \Rightarrow (\vec{S_i} \Rightarrow R_{ij1}) \Rightarrow \ldots \Rightarrow (\vec{S_i} \Rightarrow R_{ijl_{ij}}) \Rightarrow (\mathbf{0} \Rightarrow \mathbf{0})$ such that $f_{i,j}; \mathsf{unc}([\![M(f_{ij}, g_{ij})]\!]) = \top$ and $g_{i,j}; \mathsf{unc}([\![M(f_{ij}, g_{ij})]\!]) = \bot$.

$$M(f,g) = \lambda\overline{x}.\lambda u : \mathbf{0}.((\lambda a_1 : S_{i1} \ldots \lambda a_{m_i} : S_{im_i}.x_i\,a_1 \ldots a_{j_1}\,N\,a_{j+1}\ldots a_{m_i})\,\Omega\ldots\Omega)$$

where $N = \lambda w_1 : R_{ij1} \ldots \lambda w_{l_{ij}} : R_{ijl_{ij}}.(M(f_{ij}, g_{ij})\,P_1 \ldots P_n\,Q_1 \ldots Q_{l_{ij}}\,u)$ and,
$P_k : \vec{S_i} \Rightarrow T_k = \lambda v_1 : S_{i1} \ldots \lambda v_{m_i} : S_{im_i}.a_1 := v_1 \ldots a_{m_i} := v_{m_i}.x_k$,
and for $1 \leq k \leq l_{ij}$, $Q_k : \vec{S_i} \Rightarrow R_{ijk} = \lambda\vec{v}.a_1 := v_1 \ldots a_{m_i} := v_{m_i}.w_k$. □

Theorem 6.11 *Any sequential, ω-algebraic LS-category $\mathcal{C}$ contains a model of $\lambda_{\mathtt{ref}}$ which is fully abstract for closed terms.*

Proof. Suppose $f = [\![P : T]\!]_{\mathcal{C}} \not\sqsubseteq [\![Q : T]\!]_{\mathcal{C}} = g$. Then $\mathsf{DI}^\omega_T(f; \mathtt{der}_{[\![T]\!]}) \not\leq \mathsf{DI}^\omega_T(g; \mathtt{der}_T)$ by Lemma 6.9. Hence by Proposition 6.10, there exists a term $M(f,g) : T \Rightarrow \mathbf{0} \Rightarrow \mathbf{0}$ of $\lambda_{\mathtt{ref}}$ such that $f; \mathtt{der}_{[\![T]\!]}; \mathsf{unc}([\![M(f,g)]\!]) = \top$ and $g; \mathtt{der}_{[\![T]\!]}; \mathsf{unc}([\![M(f,g)]\!]) = \bot$, and so by adequacy, $M(f,g))\,P\,x \Downarrow$ and $M(f,g))\,Q\,x \not\Downarrow$ — i.e. $P \not\lesssim Q$ as required. □

7 Conclusions

We have suggested that one motivation for the research reported here is the parallel with functional programming, cartesian closed categories and the λ-calculus; this raises the question of whether an informative notion of deduction system and term calculus can be associated with our class of categorical models. One difficulty in describing such a calculus is that it is liable to become very complicated, due to the amount of information required to determine the meaning of a term of $\lambda_{\mathtt{ref}}$ within a context. This could be said to reflect the complexity of observational equivalence in the presence of references, thus an important goal is to determine sensible restrictions on programs which would allow a simpler categorical and type-theoretical interpretation — for instance, constraints on *interference* between variables.

A somewhat simpler characterization of models of (Idealized Algol style) first-order store can be given based on sequoidal categories. It is possible to give a linear λ-calculus based term language for such categories. We can also give an axiomatic characterization of the fully abstract models together with a decomposition theorem.

Games models of several non-functional features — including exceptions [14] and concurrency [15] — have been defined along similar lines to the mod-

els of store; a natural question to ask is whether we can give categorical characterizations of these models. The difficulty here is that these features also manipulate the flow of control, so further reasoning principles are required to analyze these "hybrid effects". One possibility would be to combine the "linear-continuation-passing" [6] interpretation of control effects with the notion of linear model described here.

Acknowledgements

This research was funded by a UK EPSRC grant "A Unified Semantics of Hybrid Effects". I would like to thank Guy McCusker, Russ Harmer and Matthew Wall for some helpful discussions.

References

[1] S. Abramsky. Axioms for full abstraction and full completeness. In *Essays in Honour of Robin Milner*. MIT Press, 1997.

[2] S. Abramsky and G. McCusker. Linearity, Sharing and State: a fully abstract game semantics for Idealized Algol with active expressions. In P.W. O'Hearn and R. Tennent, editors, *Algol-like languages*. Birkhauser, 1997.

[3] S. Abramsky, K. Honda, G. McCusker. A fully abstract games semantics for general references. In *Proceedings of the 13th Annual Symposium on Logic In Computer Science, LICS '98*, 1998.

[4] S. Abramsky, R. Jagadeesan and P. Malacaria. Full abstraction for PCF. *Information and Computation*, 163:409–470, 2000.

[5] P. Baillot. *Approches Dynamiques en Semántique de la Logique Linéaire: Jeux et Géométrie de l'Interaction*. PhD thesis, Université d'Aix-Marseille II, 1999.

[6] Josh Berdine, Peter W. O'Hearn, Uday S. Reddy, and Hayo Thielecke. Linearly used continuations. In *Informal Proceedings of The Third ACM SIGPLAN Workshop on Continuations (CW'01)*, January 2001. To appear.

[7] G. Bierman. What is a categorical model of intuitionistic linear logic? In *Proceedings of the Second International Conference on Typed Lambda Calculus*, number 902 in LNCS, 1995.

[8] D. Ghica and G. McCusker. Reasoning about Idealised Algol using regular languages. In *Proceedings of the Twenty-Seventh International Colloquium on Automata, Languages and Programming (ICALP 2000)*, 2000.

[9] R. Harmer and G. McCusker. A fully abstract games semantics for finite nondeterminism. In *Proceedings of the Fourteenth Annual Symposium on Logic in Computer Science, LICS '99*. IEEE Computer Society Press, 1998.

[10] M. Hasegawa. Recursion from cyclic sharing: traced monoidal categories and models of cyclic lambda calculi. In *Proceedings of TLCA '97*, number 1210 in LNCS, pages 196–213. Springer, 1997.

[11] J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF: I, II and III. *Information and Computation*, 163:285–408, 2000.

[12] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. *Math. Proc. Camb. Phil. Soc.*, 119:447 – 468, 1996.

[13] J. Laird. *A Semantic Analysis of Control*. PhD thesis, Department of Computer Science, University of Edinburgh, 1998.

[14] J. Laird. A fully abstract game semantics of local exceptions. In *Proceedings of the Sixteenth International Symposium on Logic In Computer Science, LICS '01*. IEEE Computer Society Press, 2001.

[15] J. Laird. A game semantics of ICSP. In *Proceedings of MFPS XVII*, number 45 in Electronic notes in Theoretical Computer Science. Elsevier, 2001.

[16] G. McCusker. *Games and full abstraction for a functional metalanguage with recursive types*. PhD thesis, Imperial College London, 1996.

[17] A. M. Pitts. Relational properties of domains. *Information and Computation*, 127:66–90, 1996.

[18] J. Power and E. Robinson. Premonoidal categories and notions of computation. *Mathematical Structures in Computer Science*, 1997.

[19] U. Reddy. A linear logic model of state. Typescript, 1993.

[20] J. Reynolds. Syntactic control of interference. In *Conf. Record 5^{th} ACM Symposium on Principles of Programming Languages*, pages 39–46, 1978.