Replace this file with prentcsmacro.sty for your meeting, or with entcsmacro.sty for your meeting.
Both can be found through links on the ENTCS Web Page.

From Global to Local State, Coalgebraically and Compositionally

Jim Laird

University of Bath, UK

Abstract

We describe a type theory or metalanguage for constructing and reasoning about higher-order programs with global and local state, and its categorical model. This provides an encapsulation primitive for abstracting global state and making it local to an object, so that it is passed only between its invocations. Our calculus and its semantics extend the interpretation of lambda-terms in a Cartesian closed category with a monoidal action on a category of evaluation contexts — *the sequoid* — which is dual to the action of the function type. This gives an interpretation of a new type constructor which allows the representation of both global state — via "state-passing-style" interpretation which uses it to represent output states — and local state, via encapsulation, which corresponds to the unique map into a *final coalgebra* for the sequoid. This provides the equational theory of our calculus with a coinduction rule for proving equivalence between objects with local state. We show that this theory is sound and complete with respect to the categorical semantics by constructing a term model and we show that it is consistent by giving a concrete example based on a category of games and strategies previously used to interpret general references.

1 Introduction

This paper describes a methodology for constructing higher-order, stateful programs — in particular, by making state *local* to a function or object — and for establishing equivalence between them. By restricting where and how changes to the state can occur, locality simplifies the modular construction and verification of stateful programs — e.g. it is no longer necessary to keep track of an ever-increasing collection of global variables to avoid undesirable state changes. However, reasoning about local state can be subtle, as changes to the state are implicit in interaction with the objects which interface with it. Because it has a formal, categorical model, our operation for making state local comes with a sound and complete theory of program equivalence.

It is based on a powerful and well-established principle for abstracting the state from a computational system which can be represented as a coalgebra for a functor, by taking the unique map into its *final coalgebra* [20]. For example, if we have a (deterministic) system which at each step takes in an input state s from a set S to an output state $s' \in S$ and produces an observable value $v \in V$, this may be represented as a function $f : S \to V \times S$ — i.e. a coalgebra for the functor $V \times \_$ on the category of sets. The final coalgebra for this functor is given by the set of infinite sequences (streams) of values in V and the function taking such a stream $v_1 v_2 v_3 \ldots$ to $(v_1, v_2, v_3 \ldots)$. The unique coalgebra morphism from f is the function which takes an initial state s and returns the corresponding stream of observable values $\pi_1 f(s) \pi_1 f(\pi_2 f(s)) \pi_1 f(\pi_2 f(\pi_2 f(s))) \ldots$. This characterizes exactly the observable behaviour of the system, while hiding the internal passing of state from one step to the next. It also comes with coinductive principles for reasoning about infinite streams based on the coalgebras which generate them [21].

Coalgebraic principles have been used to describe the semantics of stateful programming languages such as Java [6] in this way. The novelty and technical challenge of our approach is to describe such a construction inside a metalanguage itself, in a setting which soundly captures higher-order functions with shared access to our stateful objects, allowing them to be combined *compositionally*. To capture this shared access requires a new coalgebraic representation of global state which is not available in the category of sets: our model is motivated and underpinned by game semantics. It has developed from investigation of the game semantics of local state, in particular higher-order references [2]. This seminal model elegantly distils stateful behaviour into the interpretation of local variable declaration as a strategy behaving as a reference cell, so that composition with this cell yields a class of strategies for which the whole history of play can act as an internal state. The

©2019 Published by Elsevier Science B. V.

combinatorial definition of the cell — and thus the denotations of programs it generates — lacks a direct relationship to the abstract notion of state; in [10,5] it was shown that it may be interpreted coalgebraically, using the final coalgebra for a new kind of functor, the *sequoid*.

The work reported here abstracts these techniques from the technical setting of game semantics into a general metalanguage for reasoning about state. Specifically, we describe a conservative extension of the simply-typed λ-calculus to a typing system with a new constructor for output states — for which an object with global state may be represented as a coalgebra — and an *encapsulation operator* corresponding to a unique map into its final coalgebra. The uniqueness property of the latter provides us with a new coinductive principle for proving equivalence between programs with local state, based on their local reductions. This is part of a sound and complete equational theory for our metalanguage, which could be used as the basis for new programming languages, or for analyzing existing ones via translations.

1.1 Overview and Related Work

We first present the part of our calculus in which we may represent objects with global state. This is based on adding explicit definitions for variables to the λ-calculus, as in the $\lambda\sigma$-calculus of explicit substitutions [1]. A key feature is that variables may be invoked outside the lexical scope of their definition. Accordingly, we introduce two further binding operations: variable hiding with scope extrusion in the style of the π-calculus [14], and an operation abstracting the capability to define a variable in our model (similar to to the notion of σ-abstraction of the capacity to write to a global reference described in [4]) which is typed with a new constructor, $\oslash$.

In Section 3 we describe additional structure on a model of the λ-calculus (Cartesian closed category) which may be used to interpret these constructs: a sequoidal CCC. This is based on interpreting the basic operation of adding a variable definition to a term via its monoidal *action* on a category of linear morphisms corresponding to evaluation contexts. The use of a monoidal action to describe state-passing style semantically has been explored by Møgelberg and Staton [15], who have also demonstrated a precise correspondence with continuation-passing-style which relates to a further property of our model: a duality between the sequoid and the internal hom from the CCC. This induces a form of *trace operator* [9] with which variable hiding is interpreted (cf. models of the π-calculus such as [7]). Our soundness and completeness result with respect to the equational theory of our calculus is proved by constructing a term model, showing that it is an internal language for sequoidal CCCs, extending (for example) the original result for the simply-typed λ-calculus [11].

Examples of sequoidal CCCs arise from any (pre)monoidal adjunction between a CCC and a compact closed category — e.g. the categories of sets and functions, and sets and relations. However, there are instances which do not arise in this way. We give an example based on the Hyland-Ong style games used to give a denotational semantics of higher-order references [2]. Another example of a sequoidal CCC was introduced in [10]: this is built from a model of intuitionistic linear logic in which the exponential $!A$ is the *final coalgebra* for the sequoidal functor $A \oslash \_$. As discussed in [5], this gives a recipe for defining strategies by taking the anamorphism for a coalgebra $\sigma : S \to A \oslash S$ (which, as we have noted, corresponds to an object with global state in S), leading to coalgebra morphism from S to $!A$ in which the state is hidden. A similar operation was proposed by Longley [13]. A contribution of the current work is to reframe this recipe for models which do not have a linear decomposition.

In Section 4, we extend our calculus with a corresponding syntactic *encapsulation* operation which binds a variable to a term in which the globally visible state (e.g. derived by state-passing style interpretation) has been made local by using hidden variables to share it between between successive invocations of the variable. This is a characterization of the *abstraction* operation of [17], in which *anti-frame* rules are introduced to reason about it in an axiomatic semantics setting. Here, we describe a coinductive rule for proving equivalence between definitions of objects with encapsulated state and apply it to some simple examples including a reference cell. In Section 5, we extend our categorical model and its soundness and completeness results for the equational theory to the encapsulation operator, by developing a new category of coalgebras and "sequoidal natural transformations" on sequoidal categories.

2 A Calculus for Global State

In this section we describe the basis for our representation of higher-order state, in the form of a novel type theory in which terms may contain explicit definitions of variables (as terms) *and* the capability to make such definitions may be abstracted. It is a direct extension of the (call-by-name) simply-typed λ-calculus with finite products — although this does not directly reflect the semantics of a typical "real-life" stateful higher-order programming language particularly closely, its models (Cartesian closed categories) are the simplest setting in which we can introduce our key ideas, and can be readily extended to effect calculi, used for CPS translation, or otherwise extended with more expressive types and features.

Raw terms are generated by the following grammar:

$$t := x \mid \lambda x.t \mid t\,t \mid t\{x := t\} \mid \nu x.t \mid \overline{\lambda x}.t \mid t\,\overline{x} \mid t.i \mid \langle t_1, \ldots, t_{n-1}$$

— i.e. by extending the λ-calculus with the following operations:

- *Definition* — $s\{x := t\}$ adds a definition of the variable x as t to s.
- *Declaration* (hiding) — $\nu x.t$ hides the variable x in t.
- *Co-abstraction* — $\overline{\lambda x}.t$ abstracts the ability to define the variable x — and
- *Co-application* — $t\,\overline{y}$ assigns to y the definition abstracted in t.

Types are given by the grammar: $S, T := B \mid \Pi_{i<n} S_i \mid S \to T \mid S \oslash T$ — i.e. from a set of ground types B by the finite product and function type constructors of the simply-typed λ-calculus extended with a notion of "output type" $S \oslash T$, which is introduced by co-abstraction and eliminated by co-application: when supplied with a variable of type T, a term of type $S \oslash T$ defines it and returns a result of type s. We will assume that subterms may be annotated with their types where necessary but in general omit these annotations.

The rules for typing judgments are given in Table 1. They generate terms-in-context of the form $\Gamma \vdash t : T; \Delta$, where Γ and Δ are sets of typed variables (non-repeating sequences, considered up to permutation). Γ (the input context) contains free variables which t may invoke, and Δ (the output context) free variables which it has defined. If a name occurs in Γ it may or may not appear in Δ, and vice-versa.

- Definitions may be recursive: e.g. we may define the fixed-point combinator $\mathbf{Y}_T : (T \to T) \to T =_{df} \lambda f.\nu x.x\{x := f\,x\}$.
- Inputs are treated non-linearly whereas outputs are treated linearly (a variable is defined exactly once but may be invoked as many times as required). These constraints entail that terms represent sequential and deterministic programs, although they may be relaxed.
- Although application and co-application are not symmetric, since a term may be co-applied only to a variable but applied to any term, we may regain symmetry between these operations by replacing the application $t\,s$ with $\nu x.t\,x\{x := s\}$.

$$\frac{}{\Gamma,x{:}T \vdash x{:}T;\_} \qquad \frac{\Gamma,x{:}S \vdash t{:}T;\Delta,x{:}S}{\Gamma \vdash \nu x.t{:}T;\Delta} \qquad \frac{\Gamma \vdash s{:}S;\Delta \qquad \Gamma \vdash t{:}T;\_}{\Gamma \vdash s\{x{:=}t\}{:}S\Delta,x{:}T}\, x \notin \Delta$$

$$\frac{\Gamma,x{:}S \vdash t{:}T;\Delta}{\Gamma \vdash \lambda x.t{:}S \to T;\Delta}\, x \notin \Delta \qquad \frac{\Gamma \vdash t{:}T;\Delta,x{:}S}{\Gamma \vdash \overline{\lambda x}.t{:}T \oslash S;\Delta}\, x \notin \Gamma \qquad \frac{\Gamma \vdash t_i{:}T_i;\Delta \qquad i<n}{\Gamma \vdash \langle t_1,\ldots,t_{n-1}\rangle{:}\Pi_{i<n}T_i;\Delta}$$

$$\frac{\Gamma \vdash t{:}S \to T;\Delta \qquad \Gamma \vdash s{:}S;\_}{\Gamma \vdash t\;s{:}T;\Delta} \qquad \frac{\Gamma \vdash s{:}S \oslash T;\Delta}{\Gamma \vdash s\,\overline{x}{:}S;\Delta,x{:}T} \qquad \frac{\Gamma \vdash t{:}\Pi_{i<n}T_i;\Delta}{\Gamma \vdash t.i{:}T_i;\Delta}\; i < n$$

Table 1
Typing Rules

To give an equational theory concisely, we introduce *head contexts*, which have holes exactly at their head positions (which are encountered exactly once during evaluation).

Definition 2.1 Head contexts are given by the grammar:

$$L ::= \quad [\_] \mid \lambda x.L \mid \langle L_1, \ldots, L_n\rangle \mid L\{x := t\} \mid \nu x.L \mid L\,t \mid L.i$$

We assume that head contexts only bind variables if the binders are made explicit — e.g. $L[t]$ is assumed to not bind variables in t, but $\nu x.L[t]$ may bind occurences of x in t. The equational theory $=_{\mathcal{T}}$ is the least congruence containing the axioms in Table 2, which extend the $\beta\eta$-equality of the λ-calculus (with finite products) with axioms asserting that:

- Definition and hiding commute with head contexts, providing no variables are captured (so, in particular, they commute with each other).
- Bound input variables may be replaced with their definitions; if a variable is never invoked then its definition may be garbage-collected; a bound output variable aliased to another variable may be replaced by it.

$$\begin{array}{cc}
(\lambda x.t)\, s =_{\mathcal{T}} t[s/x] & \lambda x.(t\, x) =_{\mathcal{T}} t\ (x \notin FV(t)) \\
(\overline{\lambda x}.t)\, \overline{y} =_{\mathcal{T}} t[y/x] & \overline{\lambda x}.(t\, \overline{x}) =_{\mathcal{T}} t\ (x \notin FV(t)) \\
\langle t_1, \ldots, t_n \rangle.i =_{\mathcal{T}} t_i & \langle t.1, \ldots, t.n \rangle =_{\mathcal{T}} t \\
\nu x.L[t] =_{\mathcal{T}} L[\nu x.t]\ (x \notin FV(L[\_]) & L[t\{x := s\}] =_{\mathcal{T}} L[t]\{x := s\} \\
\nu x.t\{x := s\} =_{\mathcal{T}} \nu x.t[s/x]\{x := s\} & \nu x.t\{y := x\} =_{\mathcal{T}} t[\overline{y}/\overline{x}]\ (x \notin FV(t)) \\
\nu x.t\{x := s\} =_{\mathcal{T}} t\ (x \notin FV(t)) &
\end{array}$$

Table 2
Equational Theory

3 Sequoidal Categories

We now define a notion of categorical model for our type theory which conservatively extends the interpretation of the λ-calculus in a cartesian closed category [11] with structure for interpreting variable definitions and output types, based on the notion of sequoidal category introduced in [10].

An *action* of a symmetric monoidal category $\mathcal{C}$ is given by a category $\mathcal{L}$ and a strong monoidal functor from $\mathcal{C}$ into $\mathcal{L}^{\mathcal{L}}$ (the monoidal category of endofunctors on $\mathcal{L}$ with composition). In other words, a functor $\_ \oslash \_ : \mathcal{L} \times \mathcal{C} \to \mathcal{L}$ with isomorphisms $\lambda_{X,B,C} : X \oslash (B \otimes C) \cong (X \oslash B) \oslash C$ and $\kappa_X : X \oslash I \cong X$ satisfying the coherence equations for a strong monoidal functor [8]. We write $(\mathcal{C}, \otimes)$ for the canonical action of $\mathcal{C}$ on itself. By a weak morphism between the $\mathcal{C}$-actions $(\mathcal{L}, \oslash)$ and $(\mathcal{L}', \oslash')$ we mean a functor $J : \mathcal{L} \to \mathcal{L}'$ with a natural transformation $\omega_{X,A} : JX \oslash' B \to J(X \oslash A)$ such that:

$$\begin{tikzcd}
JX \oslash' (A \otimes B) \arrow[r, "\omega_{X,A\otimes B}"] \arrow[d, "\lambda'_{JX,A,B}"] & J(X \oslash (A \otimes B)) \arrow[dr, "J(\lambda_{X,A,B})"] & & JX \oslash' I \arrow[r, "\omega_{X,I}"] \arrow[dr, "\kappa'_{JX}"] & J(X \oslash I) \arrow[d, "J(\kappa_X)"] \\
(JX \oslash A) \oslash' B \arrow[r, "(\omega_{X,A} \oslash' B)"] & J(X \oslash A) \oslash' B \arrow[r, "\omega_{X \oslash A,B}"] & J((X \oslash A) \oslash B) & & JX
\end{tikzcd}$$

It is *strong* if ω is an isomorphism, and *strict* if it is the identity.

Definition 3.1 A *sequoidal category* is a tuple $(\mathcal{C}, \mathcal{L}, \oslash, J)$ given by a symmetric monoidal category $(\mathcal{C}, I, \otimes)$, a $\mathcal{C}$-action $(\mathcal{L}, \oslash)$ and a weak morphism of $\mathcal{C}$-actions (J, ω) from $(\mathcal{L}, \oslash)$ to $(\mathcal{C}, \otimes)$.
It is *cartesian* if the symmetric monoidal structure on $\mathcal{C}$ is cartesian, and the image under J of a specified finite product structure on $\mathcal{L}$ "on the nose" (i.e. $J(X \times Y) = JX \times JY$).

We will use the sequoid to interpret the explicit definition operation: given terms s and t denoting morphisms $f : A \to JX$ and $g : A \to JY$ in a cartesian sequoidal category, $s\{x := t\}$ denotes the morphism $\langle f, g \rangle; \omega_{X,JY} : A \to J(X \oslash JY)$.

Examples of sequoidal categories may be derived from *premonoidal adjunctions* [18]. For any symmetric monoidal category $(\mathcal{C}, \otimes, I)$ and premonoidal category $(\mathcal{L}, \odot, I)$, a premonoidal functor $J : \mathcal{L} \to \mathcal{C}$ with a premonoidal left adjoint $G : \mathcal{C} \to \mathcal{L}$ yields a sequoidal category $(\mathcal{C}, \mathcal{L}, \_ \odot G\_, J)$: G is necessarily strong monoidal into the *centre* of $\mathcal{L}$ (morphisms for which $\odot$ is always bifunctorial) and hence $X \oslash A = X \otimes_{\mathcal{L}} GA$ is a $\mathcal{C}$-action on $\mathcal{L}$, with a natural transformation $\omega_{X,B} : JX \otimes_{\mathcal{C}} B \to J(X \otimes_{\mathcal{L}} GB)$ making (J, ω) a weak morphism of $\mathcal{C}$-actions from $\oslash$ to $\otimes$. However, some of the sequoidal categories that we are particularly interested in do not arise from premonoidal adjunctions in this way: in particular the sequoidal categories of games described below (and in [10,3,5]).

3.1 Sequoidal Closed Categories

λ-abstraction and function application in our calculus are interpreted in standard fashion by requiring that the symmetric monoidal category is *closed* — i.e. has an internal hom functor: we can then interpret all of the other rules by requiring that the sequoid is dual to this, in a sense in which we now explain.

Definition 3.2 A *dual* for a $\mathcal{C}$-action $(\mathcal{L}, \oslash)$ is a $\mathcal{C}^{op}$-action $\multimap$ on $\mathcal{L}$ such that for each A in $\mathcal{C}$, $\_ \oslash A$ is dual to $A \multimap \_$ in the monoidal category $\mathcal{L}^{\mathcal{L}}$ — i.e. both right and left adjoint to the endofunctor $\_ \oslash A$.

Observe that dual actions commute — i.e. there is a natural isomorphism $\theta_{A,B,X} : A \multimap (X \oslash B) \cong (A \multimap X) \oslash B$, since for any functions $F \dashv\vdash G$ and $F' \dashv\vdash G'$ such that F commutes with F' and G with G', we may construct a natural isomorphism:

$$FG' \xrightarrow{FG'\eta} FG'FG \cong FGG'F \xrightarrow{\epsilon_{G'F}} G'F$$

Definition 3.3 A sequoidal closed category is given by a tuple $(\mathcal{C}, \mathcal{L}, \oslash, J)$ where

(i) $\mathcal{C}$ is a symmetric monoidal closed category.
(ii) $(\mathcal{L}, \oslash)$ is a $\mathcal{C}$-action with a dual, $(\mathcal{L}, \multimap)$.
(iii) J is a strong morphism of $\mathcal{C}^{op}$ actions from $(\mathcal{L}, \multimap)$ to $(\mathcal{C}, \multimap)$.

A sequoidal CCC is a sequoidal category which is cartesian, and closed.

To justify the terminology, observe that by uncurrying $J(\eta_{X,B}) : JX \to B \multimap J(X \oslash B)$ (where η is the unit of the adjunction $B \multimap \_ \dashv \_ \oslash B$) the above data determine a natural transformation $\omega_{X,B} : (JX \otimes B) \to J(X \oslash B)$ making (J, ω) a lax morphism of $\mathcal{C}$-actions from $(\mathcal{L}, \oslash)$ to $(\mathcal{C}, \otimes)$. So we can equivalently say that a sequoidal category is closed if it arises in this way. For adjunction models, sequoidal closure corresponds to *compact closure* of the image of the functor $G : \mathcal{C} \to \mathcal{L}$.

Proposition 3.4 *An adjunction model $(\mathcal{C}, \mathcal{L}, J, G)$ is sequoidal closed if and only if $\mathcal{C}$ is symmetric monoidal closed and every object GA has a dual in the premonoidal category $\mathcal{L}$.*

Proof. If $\mathcal{L}$ has duals of every object GA then the $\mathcal{C}^{op}$-action $\_ \otimes (G\_)^*$ is dual to $\_ \otimes G\_$ by definition, and for any objects A in $\mathcal{C}$ and B in $\mathcal{L}$, $J(B \otimes (!A)^*)$ is an exponential in $\mathcal{C}$, and so J is a strong functor of $\mathcal{C}^{op}$ actions. Conversely, if $(\mathcal{C}, \mathcal{L}, J, G)$ is sequoidal closed, then for any A, $GA \multimap I$ is a dual for GA in $\mathcal{L}$. □

A simple example of a sequoidal CCC is thus given by the monoidal adjunction between the Cartesian closed category $\mathcal{C}$ of sets and functions and the compact closed category $\mathcal{L}$ of sets and relations — i.e. $J : \mathcal{L} \to \mathcal{C}$ is the powerset functor and $\oslash$ is the action sending sets A, X to $A \times X$. This is self-dual and satisfies $J(A \multimap X) = \mathcal{P}(A \times X) \cong \mathcal{P}(X)^A = A \multimap JX$.

Every compact closed category $(\mathcal{C}, \otimes, I)$ admits a canonical *trace operator* [9] on $\mathcal{C}$: sequoidal closed structure on $(\mathcal{C}, \mathcal{L}, J, \oslash)$ induces similar operators on both $\mathcal{C}$ and $\mathcal{L}$, which we use to interpret variable hiding.

Definition 3.5 Given $f : A \otimes B \to J(X \oslash B)$ in $\mathcal{C}$, define $\mathrm{tr}^B_{A,X}(f) = \Lambda(f); J(\epsilon_{B,X})$, where $\epsilon_{B,X} : B \multimap (X \oslash B) \to X$ is the co-unit of the adjunction $B \Rightarrow \_ \dashv \_ \oslash B$. The corresponding operation on $\mathcal{L}$ sends $f : X \oslash B \to Y \oslash B$ to $\Lambda_{\oslash}(f); \epsilon_{B,Y} : X \to Y$. [1]

These operators satisfy axioms analogous to the corresponding properties of a traced monoidal category (we refer to [9] for diagrammatic representation) i.e.

- Naturality in A and X: $\mathrm{tr}^B_{A,X}(f); J(g) = \mathrm{tr}^B_{A,Y}(f; J(g \oslash B))$ for $g : X \to Y$ and $h; \mathrm{tr}^B_{A,X}(f) = \mathrm{tr}^B_{C,X}((h \otimes B); f$ for $h : C \to A$.
- Dinaturality in B: $\mathrm{tr}^C_{A,X}(f; J(X \oslash g)) = \mathrm{tr}^B_{A,X}((A \otimes g); f)$ for $f : A \otimes C \to J(X \oslash B)$ and $g : B \to C$.
- Vanishing: $\mathrm{tr}^I_{A,X}(f) = f$ and $\mathrm{tr}^B_{A,X}(\mathrm{tr}^C_{A \times B, J(X \oslash B)}(f)) = \mathrm{tr}^{B \otimes C}_{A,X}(f; J(\lambda_{X,B,C}))$.
- Superposition: $tr^B_{A,X}(f)\{g\} = \mathrm{tr}^B_{A \otimes C, X \oslash C}((A \otimes \gamma_{B,C}); f\{g\}; J(X \oslash \gamma_{C,B}))$, where γ is the symmetry isomorphism of $(\mathcal{C}, \otimes, I)$.
- Yanking: $\mathrm{tr}(\gamma_{JX,JX}; \omega_{JX,JX}) = \mathsf{id}_{JX}$.

3.2 A Sequoidal CCC of Games

We now briefly describe a sequoidal closed category, and related sequoidal CCC of "Hyland-Ong-style" two-player games and strategies, by adding sequoidal structure to the categories of games used by Abramsky, Honda and McCusker to give a semantics of higher-order references [2], to which we refer for further details. It will serve as a more concrete example of the definitions already introduced (establishing their consistency) and provide motivations and intuitions for extending our type theory and categorical model to capture its coalgebraic properties.

Definition 3.6 An arena $\langle M_A, M^I_A, \vdash_A, \lambda_A \rangle$ is a directed, acyclic graph —- a set of nodes (moves) M_A with an edge-relation $\vdash_A \subseteq M_A \times M_A$ (enabling) and a set of specified source nodes $M^I_A \subseteq M_A$ (initial moves) —

[1] In an adjunction model, this corresponds directly to the trace operator induced by the compact closed structure.

with a labelling $\lambda_A : M_A \to \{O, P\}$ of each move as a Player or Opponent move such that every initial move is an Opponent move.

Apart from dropping the labelling of moves as questions and answers (as we shall not make use of it), the only change from [2] is that we do not require that non-initial O moves are enabled by P-moves and vice-versa.

Definition 3.7 A legal sequence on the arena A is a finite O/P alternating sequence of moves equipped with an explicit *justification pointer* from each non-initial move to a preceding, enabling move. (Note that O-moves may point to O-moves, and P-moves to P-moves.)

A (deterministic) strategy σ on A is a non-empty, even-prefix-closed set of even-length legal sequences on A such that $sa, sb \in \sigma$ implies $a = b$.

The category of games $\mathcal{G}$ has games as objects: morphisms from A to B are strategies on the game $A \Rightarrow B \triangleq (M_A + M_B, \mathtt{inr}(M^I_B), \vdash_A + \vdash_B \cup (\mathtt{inr}(M^I_B) \times \mathtt{inl}(M^I_A), [\overline{\lambda_A}, \lambda_B])$. The identity on A is the set of "copycat sequences" on $A \Rightarrow A$ (every even prefix has equal projections to both copies of A) and the composition of morphisms is their "parallel composition with hiding" — i.e. given strategies $\sigma : A \to B$ and $\tau : B \to C$, $\sigma;\tau$ consists of the legal sequences which are restrictions to $A \Rightarrow C$ of sequences t in $M_A + M_B + M_C$ such that $t\lceil A, B$ is in σ and $t\lceil B, C$ is in τ. We may define symmetric monoidal (closed) structure on $\mathcal{G}$, with $A \otimes B \triangleq (M_A + M_B, M^I_A + M^I_B, \vdash_A + \vdash_B, [\lambda_A, \lambda_B])$. We now define sequoidal closed structure on $\mathcal{G}$:

Definition 3.8 A strategy on $A \Rightarrow B$ is *linear* if:

- every sequence $s \in \sigma$ contains at most one initial move from A and B, respectively.
- for every initial move $b \in M^I_B$ there exists $a \in M^I_A$ such that $ba \in \sigma$.

Games and linear strategies form a category $\mathcal{G}_l$, with an identity-on-objects functor $J : \mathcal{G}_l \to \mathcal{G}$ which sends the linear morphism $\sigma : A \Rightarrow B$ to the strategy $\sigma^\dagger$ consisting of legal sequences on $A \Rightarrow B$ which are the interleaving of some $s_1, \ldots, s_n \in \sigma$.

The internal hom functor $\_ \Rightarrow \_ : \mathcal{G}^{op} \times \mathcal{G} \to \mathcal{G}$ restricts to linear morphisms, giving an action of $\mathcal{G}^{op}$ on $\mathcal{G}_l$ such that J is a strict map of $\mathcal{G}^{op}$ actions. Moreover $\Rightarrow$ has a dual:

Definition 3.9 Given arenas A and B, the sequoid $A \oslash B$ is defined:

$$(M_A + M_B, \mathtt{inl}(M^I_A), \vdash_A + \vdash_B \cup (\mathtt{inl}(M^I_A) \times \mathtt{inl}(M^I_B), [\lambda_A, \lambda_B])$$

i.e. $A \oslash B$ is defined as for $B \Rightarrow A$ except that Player/Opponent labelling is not swapped in A. Thus there are evident natural isomorphisms:

$$\frac{\mathcal{G}_l(A \oslash B, C)}{\mathcal{G}_l(A, B \Rightarrow C)} \qquad\qquad \frac{\mathcal{G}_l(B \Rightarrow A, C)}{\mathcal{G}_l(A, C \oslash B)}$$

making $(\mathcal{G}, \mathcal{G}_l, J, \oslash)$ a sequoidal closed category. This is not a sequoidal CCC because $\otimes$ is not a cartesian product on $\mathcal{G}$. However, we may obtain a sequoidal CCC by restricting to *well-opened* strategies — essentially the same step as in [2].

Definition 3.10 A legal sequence is *well-opened* if it contains at most one initial move. A strategy is well-opened if it consists of well-opened legal sequences, and the category $\mathcal{G}_w$ of arenas and well-opened strategies has as morphisms from A to B, the well-opened strategies on $A \Rightarrow B$, with the composition of $\sigma : A \to B$ with $\tau : B \to C$ being the parallel composition with hiding of $\sigma^\dagger$ with τ.

$\mathcal{G}_w$ is cartesian closed (with product $\otimes$) with the inclusion functor $J : \mathcal{G}_l \to \mathcal{G}_w$ being a strict morphism of actions which preserves products $\mathcal{G}_l$. Thus $(\mathcal{G}_w, \mathcal{G}_l, J, \oslash(\_)^\dagger)$ is a sequoidal CCC.

We now explain why our sequoidal categories of games cannot arise from a premonoidal adjunction. Observe that in any sequoidal closed category, if $J : \mathcal{L} \to \mathcal{C}$ has a left adjoint $G : \mathcal{C} \to \mathcal{L}$, then G is isomorphic to the functor $GI \oslash \_$, since we have $\mathcal{L}(GA, B) \cong \mathcal{C}(A, JB) \cong \mathcal{C}(I, A \multimap JB) \cong \mathcal{L}(GI, A \Rightarrow B) \cong \mathcal{L}(GI \oslash A, B)$. There can be no such functor from $\mathcal{G}$ into $\mathcal{G}_l$: it is straightforward to show that if there is a surjective map from strategies on A to linear strategies on $A \multimap A$, then A is either the empty game (terminal object) or has a single playable move: in neither case is $A \oslash \_$ left adjoint to the inclusion $J : \mathcal{G}_l \to \mathcal{G}$.

Thus the sequoidal structure in our games does not arise from an adjunction. In fact, more can be said: we can describe a CPS-style construction which recovers all premonoidal adjunction models which embed in a given sequoidal closed category and show that there is no such embedding into our category of games — i.e. we cannot apply constraints to strategies to carve out a non-trivial adjunction model.

Definition 3.11 An embedding of sequoidal categories from $(\mathcal{C}, \mathcal{L}', \oslash', J')$ to $(\mathcal{C}, \mathcal{L}, \oslash, J)$ is a strong functor of $\mathcal{C}$-actions $H : (\mathcal{L}', \oslash') \to (\mathcal{L}, \oslash)$ which is fully faithful and such that $J' = HJ$.

For any sequoidal closed category, $(\mathcal{C}, \mathcal{L}, \oslash, J)$, and object X in $\mathcal{L}$, let $\mathcal{L}_X$ be the category in which objects are pairs (A^+, A^-) of objects of $\mathcal{C}$, and morphisms from (A^+, A^-) to (B^+, B^-) are morphisms from $A^- \multimap X \oslash A^+$ to $B^- \multimap X \oslash B^+$ in $\mathcal{L}$.

Thus we may define the sequoidal category $(\mathcal{C}, \mathcal{L}_X, \oslash_X, J_X)$ in which $(A^+, A^-) \oslash_X B \triangleq (A^+ \otimes B, A^-)$ and $J_X(A^+, A^-) \triangleq J(A^- \multimap X \oslash A^+)$. This is sequoidal closed: $\oslash_X$ has a dual, $B \multimap_X (A^+, A^-) \triangleq (A^+, A^- \otimes B)$ such that $J_X(B \multimap_X (A^+, A^-)) \cong B \multimap J(A^+, A^-)$.

$(\mathcal{C}, \mathcal{L}_X, \oslash_X, J_X)$ embeds in $(\mathcal{C}, \mathcal{L}, \oslash, J)$: there is a fully faithful (identity on morphisms), strong functor of $\mathcal{C}$-actions $H_X : \mathcal{L}_X \to \mathcal{L}$ sending (A^+, A^-) to $A^- \multimap X \oslash A^+$, such that $J_X = JH_X$

We may also define (symmetric) premonoidal structure on $\mathcal{L}_X$ — $(A^+, A^-) \odot_X (B^+, B^-) = (A^+ \otimes B^+, A^- \otimes B^-)$, with unit (I, I) — and a functor $G_X : \mathcal{C} \to \mathcal{L}_X$ sending A to (A, I) and $f : A \to B$ to $X \oslash f : (A, I) \to (B, I)$. We note the following facts:

(i) If $G_X : \mathcal{C} \to \mathcal{L}_X$ is premonoidal left adjoint to $J_X : \mathcal{L}_X \to \mathcal{C}$ then $(\mathcal{C}, \mathcal{L}_X, J_X, G_X)$ is a premonoidal adjunction model embedding in $\mathcal{L}$.

(ii) If $\mathcal{L}'$ is a premonoidal adjunction model with an embedding $H : \mathcal{L}' \to \mathcal{L}$ into $(\mathcal{C}, \mathcal{L}, J, \oslash)$ then $\mathcal{L}'$ is equivalent to $\mathcal{L}_{HI}$, and so G_{HI} is premonoidal left adjoint to J_{HI}.

As argued above, if $\mathcal{L}_X$ is a premonoidal adjunction model then $\mathcal{L}(X, X) \equiv \mathcal{C}(1, X)$, which is impossible in our sequoidal category of games — thus by (ii), the latter cannot contain (by embedding) a premonoidal adjunction model.

3.3 Denotational Semantics

We can now define the interpretation of our type theory in a sequoidal CCC, $(\mathcal{C}, \mathcal{L}, \oslash, J)$. We interpret types as objects of $\mathcal{L}$ — by fixing a denotation for the ground type(s) and letting $[\![S \to T]\!] = J[\![S]\!] \multimap [\![T]\!]$, $[\![S \oslash T]\!] = [\![S]\!] \oslash J[\![T]\!]$ and $[\![\Pi_{i<n} T_i]\!] = \Pi_{i<n}[\![T]\!]$. Terms $x_1 : R_1, \ldots, x_m : R_n \vdash t : T; y_1 : S_1, \ldots, y_n : S_n$ are interpreted as morphisms from $J[\![R_1]\!] \times \ldots J[\![R_m]\!]$ to $J([\![T]\!] \oslash J[\![S_1]\!] \times \ldots \times J[\![S_n]\!])$ in $\mathcal{C}$.

The interpretation of variable invocation, abstraction, application, tupling and projection is based on the standard model of the simply-typed λ-calculus with products in a Cartesian closed category, together with isomorphisms $\theta_{A,B,X} : A \multimap J(X \oslash B) \cong J((A \multimap X) \oslash B)$ and $\zeta_{X,Y,A} : J(X \times Y \oslash A) \cong J(X \oslash A) \times J(Y \oslash A)$ derived from duality ($\_ \oslash A$ and $A \multimap \_$ preserve limits). *Co-abstraction* and *co-application* derive from the strength isomorphism $\lambda : X \oslash (B \times C) \cong (X \oslash B) \oslash C$, and (as proposed above) *definition* from the sequoidal structure and *hiding* from the trace structure.

To prove that our equational theory is sound with respect to this interpretation — i.e. if $\Gamma \vdash t =_{\mathcal{T}} t' : T; \Delta$ then $[\![\Gamma \vdash t : T; \Delta]\!] = [\![\Gamma \vdash t' : T; \Delta]\!]$ — we first extend our semantics to interpret *typed* head contexts as morphisms in the category $\mathcal{L}$.

Definition 3.12 We write $\Gamma; S \vdash L : T; \Delta$ if $\Gamma, \bullet : S \vdash L[\bullet] : T; \Delta$ — i.e. the type on the left of the turnstile represents the type of the holes in the head context.

We may substitute either terms or head contexts into the holes in a head context: such substitution is assumed to be non-capturing unless indicated otherwise, and agrees with the typing rules — i.e. if $\Gamma \vdash t : S; \Delta'$ and $\Gamma; S \vdash L : T; \Delta$ then $\Gamma; S \vdash L[t] : T; \Delta, \Delta'$. Moreover, we may interpret head contexts $\Gamma; S \vdash L : T; \Delta$ as morphisms from $[\![T]\!] \oslash [\![\Gamma]\!]$ to $[\![S]\!] \oslash [\![\Delta]\!]$ in $\mathcal{L}$ such that for any term $\Gamma \vdash t; \Delta'$:

$$[\![L[t]]\!]_{\mathcal{C}} = \delta_\Gamma; ([\![t]\!]_{\mathcal{C}}\{\Gamma\}); J([\![L]\!]_{\mathcal{L}} \oslash \Delta')$$

We now prove soundness based on the properties of a sequoidal CCC:

- The $\beta\eta$ equivalences for $\to, \times$ are standard for a CCC, and for $\oslash$ follow from interpretation of co-abstraction and co-application as natural isomorphisms.
- $[\![L[\nu x.t]]\!] = [\![\nu x.L[t]]\!]$ and $[\![L[s\{x := t\}]]\!] = [\![L[s]\{x := t\}]\!]$ follow from naturality of the trace operator and definition operation, respectively.
- $[\![\nu x.s\{x := t\}]\!] = [\![s]\!]$ (where x is not free in s) follows from the vanishing property of the trace and $\nu x.t\{x := s\} = \nu x.t[s/x]\{x := s\}$ and $[\![\nu x.t\{y := x\}]\!] = [\![t[\overline{y}/\overline{x}]]\!]$ (where x is not free in t) follow from the dinaturality of the trace.

Proposition 3.13 (Soundness) *If* $\Gamma \vdash s =_{\mathcal{T}} t; \Delta$ *then* $[\![\Gamma \vdash s; \Delta]\!] = [\![\Gamma \vdash t; \Delta]\!]$ *in every sequoidal CCC.*

To prove that our equational theory is *complete* — i.e. if $[\![\Gamma \vdash s : T; \Delta]\!] = [\![\Gamma \vdash t : T; \Delta]\!]$ in every sequoidal CCC then $s =_{\mathcal{T}} t$ — we will construct a term model: a sequoidal CCC $(\mathcal{L}_{\mathcal{T}}, \mathcal{C}_{\mathcal{T}}, J_{\mathcal{T}}, \oslash_{\mathcal{T}})$. The Cartesian closed category $\mathcal{C}_{\mathcal{T}}$ has types as objects: morphisms from S to T are $\mathcal{T}$-equivalence classes of closed terms of type $S \to T$: the composition of $s : R \to S$ and $t : S \to T$ is $\lambda x.t\,(s\,x)$ and the identity on T is $\lambda x.x$. This defines a CCC [11].

The category $\mathcal{L}_{\mathcal{T}}$ has types as objects: morphisms from S to T in $\mathcal{L}_{\mathcal{T}}$ are $\mathcal{T}$-equivalence classes of (closed) head contexts $\_; S \vdash L : T; \_$ (composition of $L : R \to S$ with $L' : S \to T$ is $L'[L] : R \to T$ and the identity on each object S is the context $S \vdash [\_] : S$). The product of types is a Cartesian product in $\mathcal{L}_{\mathcal{T}}$.

The (identity-on-objects) functor $J : \mathcal{L}_{\mathcal{T}} \to \mathcal{C}_{\mathcal{T}}$ sends the head context $L : S \to T$ to the term $\lambda x.L[x]$. This preserves products, and is a strict morphism of actions into the internal hom action on $\mathcal{C}_{\mathcal{T}}$ from the $\mathcal{C}^{op}$-action $(\mathcal{L}_{\mathcal{T}}, \to)$ which sends S, T to $S \to T$, and the morphisms $t : S \to S', L : T \to T'$ to $\lambda x.L[\_\,(t\,x)] : (S' \to T) \to (S \to T')$.

Lemma 3.14 *$\_ \to \_ : \mathcal{C}_{\mathcal{T}}^{op} \times \mathcal{L}_{\mathcal{T}} \to \mathcal{L}_{\mathcal{T}}$ has a dual action.*

Proof. For any T, the functor $T \to \_$ has a left and right adjoint, the functor $\_ \oslash T : \mathcal{L}_{\mathcal{T}} \to \mathcal{L}_{\mathcal{T}}$ sending $L : S \to S'$ to $\overline{\lambda x}.L[\_\,\overline{x}] : S \oslash T \to S' \oslash T$. The counits and units of the duality are:

$$(\nu x.[(\_\overline{x})\,x], \lambda x.\overline{\lambda y}.[\_]\{y := x\}) : T \to \_ \dashv \_ \oslash T$$
$$(\nu x.[(\_\,x)\,\overline{x}], \overline{\lambda x}.\lambda y.[\_]\{x := y\}) : \_ \oslash T \dashv T \to \_$$

□

Thus we have defined a sequoidal CCC, yielding an interpretation of each term-in-context $x_1 : R_1, \ldots, x_m : R_m \vdash t : T; y_1 : S_1, \ldots, y_n : S_n$ as a closed term $[\![t]\!]_{\mathcal{T}} : R_1 \times \ldots \times \to R_m \to T \oslash S_1 \times \ldots \times S_n$, which we may prove (by induction) is equivalent to t (up to uncurrying) in the equational theory:

Lemma 3.15 *$t =_{\mathcal{T}} \nu z.([\![t]\!]_{\mathcal{T}}\,\langle x_1, \ldots, x_n \rangle)\,\overline{z}\{y_1 := z.1\} \ldots \{y_n := z.n\}$.*

So if $[\![s]\!]_{\mathcal{T}} = [\![t]\!]_{\mathcal{T}}$ then $s =_{\mathcal{T}} t$, and we have established that our equational theory is complete with respect to interpretation in a sequoidal CCC.

Theorem 3.16 (Completeness) *If $[\![\Gamma \vdash t_1 : T; \Delta]\!] = [\![\Gamma \vdash t_2 : T; \Delta]\!]$ holds in every sequoidal CCC, then $t_1 =_{\mathcal{T}} t_2 : T$.*

4 From Global To Local State

In a sequoidal CCC $(\mathcal{C}, \mathcal{L}, J, \oslash)$, we represent an object of type T with *global state* of type S — i.e. taking an input state and producing an output state of type S — as a morphism into $(S \multimap T \oslash S)$. For any object S of $\mathcal{C}$, the adjunction between $S \multimap \_$ and $\_ \oslash S$ resolves the monad $\S_S = S \multimap (\_ \oslash S)$ on $\mathcal{L}$ (isomorphic to $(S \to \_) \oslash S$). Moreover, we have a natural transformation $S \multimap \omega_{S,A} : S \to (JX \times A) \to S \multimap J(X \oslash A) \cong J(S \multimap X \oslash A)$ making $(J, S \multimap \omega_S)$ a *lax morphism of monads* from $\S_S$ to the usual state monad $S \multimap (\_ \times S)$ for $\mathcal{C}$. In other words, we can construct objects using the state monad and convert to a sequoidal representation of state: in our calculus, this operation corresponds to the term $\lambda f.\lambda x.\overline{\lambda y}.(f\,x).1\{y := (f\,x).2\} : (S \to T \times S) \to S \to T \oslash S$.

However, $\S_S$ allows operations on the state not available to the state monad: it satisfies isomorphisms including $\S_S \S_T A \cong \S_{S \times T} A$, $\S_S(A \times B) \cong \S_S(A) \times \S_S(B)$, and $\S_S(A \multimap B) \cong A \multimap \S_S B$. The last of these allows values to be passed out of static scope, which is key to straightforward manipulation of higher-order states. Our calculus provides a convenient language to express this by representing objects with global state of type S as terms $\Gamma, x : S \vdash t : T; y : S$: the function which discards its input state, writes its argument to the output state and returns the identity is $x : S \vdash \lambda z.\lambda a.a\{y := z\} : S \to B \to B; y : S$, while $x : S \vdash x\{y := x\} : S; y : S$, returns one copy of the output state and passes on another.

Last but not least, a morphism to $S \multimap T \oslash S$ corresponds to a coalgebra for the sequoid, allowing its global state to be made local by encapsulation, as we now explain.

4.1 Final Coalgebras for the Sequoid

The sequoidal CCCs of games in [10,3,5] are based on a construction of $!A$ — the *cofree commutative comonoid* on A — from a final coalgebra for the sequoid in those categories.

Definition 4.1 Given a functor $F : \mathcal{C} \to \mathcal{C}$, a *final coalgebra* for F is a terminal object in the category of F-coalgebras. In other words, an F-coalgebra — a pair (A, α) of an object A of $\mathcal{C}$ and a morphism $\alpha : A \to FA$ — such that for any F-coalgebra $(B, \beta : B \to FB)$ there is a unique morphism of F-coalgebras from (B, β)

to (A, α) — that is, a unique morphism $(\!|\beta|\!) : B \to A$ in $\mathcal{C}$ (the "anamorphism" of β) for which the following diagram commutes:

$$\begin{tikzcd}
B \arrow[r, "\beta"'] \arrow[d, "(\!|\beta|\!)"] & FB \arrow[d, "F(\!|\beta|\!)"] \\
A \arrow[r, "\alpha"] & FA
\end{tikzcd}$$

Given a cartesian sequoidal category with such a final coalgebra for the functor $J(A \oslash \_) : \mathcal{C} \to \mathcal{C}$ for each A, we can define a morphism $m :!A{\otimes}!B \to!(A \times B)$ by taking the anamorphism of the coalgebra:

$$!A{\otimes}!B \overset{\langle \alpha\otimes !B, !A\otimes\beta\rangle}{\longrightarrow} (J(A{\oslash}!A){\otimes}!B) \times (J(B{\oslash}!B){\otimes}!A \overset{\omega\times\omega}{\longrightarrow} (A{\oslash}!A){\oslash}!B) \times ((B{\oslash}!B){\oslash}!A \cong (A \times B) \oslash (!A{\otimes}!B)$$

In [5] it was shown that in a cartesian sequoidal category where this map, and the terminal morphism $t : I \to 1$ are isomorphisms, the comonoid $(!A, !\delta; m^{-1} :!A \to!A{\otimes}!A, t_{!A} :!A \to I)$ is the *cofree commutative comonoid* on JA. Which is to say that there is a morphism $\mathsf{der}_A :!A \to A = \alpha; J(\oslash t_A)$ giving a natural equivalence between morphisms into A and comonoid morphisms into $(!A, !\delta; m^{-1}, t_{!A})$. From a sequoidal closed cartesian category $(\mathcal{C}, \mathcal{L}, J, \oslash)$ we may thus define a sequoidal CCC $(\mathcal{C}_!, \mathcal{L}, !, \_{\oslash}!\_)$, where $\mathcal{C}_!$ consist of objects of $\mathcal{L}$, with morphisms from A to B being comonoid morphisms from $!A$ to $!B$.

As discussed in [5], in addition to a model of linear logic, this *sequoidal exponential* gives a recipe for interpreting imperative objects with local state:

- Take a morphism $\sigma : S \to J(A \oslash S)$ in $\mathcal{C}$ representing an object with access to global state
- $(!S, \sigma)$ is a $J(A \oslash \_)$-coalgebra in $\mathcal{C}$ and therefore has a unique anamorphism $(\!|\sigma|\!) : S \to!A$ taking an initial state to the cofree commutative comonoid on A: the comonoid structure characterizes shared access to $(\!|\sigma|\!)$, which passes hidden state between successive invocations.

However, this recipe depends on the decomposition of our sequoidal CCC into a model of linear logic. To reflect this decomposition in our calculus would represent a substantial syntactic overhead. Moreover, it does not correspond to the structure of our sequoidal CCC of Hyland-Ong games (nor the interpretation of higher-order store within it). Here, we show that we may give a coalgebraic interpretation of higher-order store without a linear decomposition by considering terminal objects in a category in which objects are still $J(A \oslash \_)$-coalgebras, but morphisms between them are certain natural transformations.

First, we give the syntactic/operational account by extending the calculus defined in Section 2 with an *encapsulated definition* operation which abstracts the global state of an object by binding it to a variable and passing the state between successive invocations of the variable while hiding it from the outside world. We will show that this may be interpreted as the "anamorphism" operation in our category of $J(A \oslash \_)$-coalgebras.

Definition 4.2 We extend the grammar of raw terms of our calculus to allow expressions of the form $r\{x := \varepsilon(s, t)\}$ (the definition of x in r as the *encapsulation* of t with initial state s). This has the typing rule:

$$\frac{\Gamma{\vdash}r{:}R;\Delta \qquad \Gamma{\vdash}s{:}S;\_ \qquad \Gamma,{\vdash}t{:}S{\to}T{\oslash}S;\_}{\Gamma{\vdash}r\{x{:=}\varepsilon(s,t)\}{:}R;\Delta,x{:}T}$$

Each invocation of x fetches a copy of t, passes in either the initial state s (on first invocation) or (subsequently) the state from the previous invocation and uses a fresh name to pass the output state generated by t to the next invocation. Thus encapsulated definition satisfies the *unfolding rule*:

$$\nu x.L[x\{x := \varepsilon(s, t)\}] =_{\mathcal{T}^+} \nu x.\nu b.L[(t\, s)\, \overline{b}\{x := \varepsilon(b, t)\}]$$

4.2 Representing Local Variables

Given terms $t_1, \ldots t_n$ such that $\Gamma, y : S \vdash t_i : T_i; z : S$, the encapsulated definition $\_\{x := \varepsilon(s, \lambda y.\overline{\lambda z}.\langle t_1, \ldots, t_n\rangle)\}$ binds x to an object which acts as an interface to a hidden local state of type S, shared between the methods $t_1, \ldots, t_n$ — i.e. the only operations on the state are those specified by selecting a method by projection and applying it to some arguments. Our key example is the higher-order reference cell storing values of type S, which (as proposed in [19]) may be represented as a product of its two methods — writing and reading to global state of type S — giving a term $\mathtt{cell}_S : S \to (\mathtt{var}[S]) \oslash S \triangleq \lambda y.\lambda\overline{z}\langle\lambda a.\lambda b.b\{z := a\}, y\{z := y\}\rangle$, where $\mathtt{var}[S] \triangleq (S \to B \to B) \times S$. By encapsulating this, we define a new variable declaration $\mathtt{ref}_S : S \to (\mathtt{var}[S] \to$

$T) \to T$, which applies its second argument to a new cell initialized with its first argument:

$$\mathtt{ref}_S \triangleq \lambda u.\lambda f.\nu c.(f\ c)\{c := \varepsilon(u, \mathtt{cell})\}$$

Using the unfolding rule, we can prove within the equational theory that this has the expected read-write behaviour. Writing $\mathsf{set}(a)$ and $\mathtt{get}(a)$ for $a.1$ and $a.2$:
$\mathtt{ref}_S\, s\, \lambda a.L[\mathsf{set}(a)\ s] =_{\mathcal{T}^+} \nu c.L[c.1\ t]\{c := \varepsilon(s, \mathtt{cell})\}$
$=_{\mathcal{T}^+} \nu c.\nu b.E[((\mathtt{cell}\ s)\ \overline{b}).1\ t]\{c := \varepsilon(b, \mathtt{cell})\}$
$=_{\mathcal{T}^+} \nu c.\nu b.L[(\lambda x.\lambda z.z\{b := x\})\ t]\{c := \varepsilon(b, \mathtt{cell})\}$
$=_{\mathcal{T}^+} \nu c.\nu b.L[\lambda z.z\{b := t\}]\{c := \varepsilon(b, \mathtt{cell})\}$
$=_{\mathcal{T}^+} \nu c.L[\lambda z.z]\{c := \varepsilon(t, \mathtt{cell})\} =_{\mathcal{T}^+} \mathtt{ref}_S\, t\, \lambda a.L[\lambda z.z]$
$\mathtt{ref}_S\, t\, \lambda a.L[\mathtt{get}(a)] =_{\mathcal{T}^+} \nu c.L[c.2]\{c := \varepsilon(t, \mathtt{cell})\}$
$=_{\mathcal{T}^+} \nu c.\nu b.L[((\mathtt{cell}\ t)\ \overline{b}).2]\{c := \varepsilon(b, \mathtt{cell})\}$
$=_{\mathcal{T}^+} \nu c.\nu b.L[t\{b := t\}]\{c := \varepsilon(b, \mathtt{cell})\}$
$=_{\mathcal{T}^+} \nu c.L[t]\{c := \varepsilon(t, \mathtt{cell})\} = \mathtt{ref}_S\, t\, \lambda a.L[t]$.

4.3 Coinduction

As illustrated in the example of the reference cell above, the unfolding rule describes the finitary behaviour of encapsulated objects. However, it is typically insufficient to prove equivalences between such objects which involve infinite (or unbounded) behaviour. To fill this gap we extend our theory with a rule which will enable us to give coinductive proofs of such equivalences. Essentially, this rule says that any environment which contains a definition of a variable x is equivalent to an encapsulated definition for x which has the same unfolding behaviour.

Definition 4.3 The *environment contexts* are given by the grammar: $\mathcal{E} := [\_] \mid \mathcal{E}\{x := t\} \mid \mathcal{E}\{x := \varepsilon(y, t)\} \mid \nu x.\mathcal{E}$

That is, an environment is a context consisting of a single hole followed by a sequence of (immutable and encapsulated) definitions, inside a sequence of name hidings. An environment context is a head context such that if $\Gamma; S \vdash \mathcal{E} : S; \Delta$ then $\Gamma; T \vdash \mathcal{E} : T; \Delta$ for any type T, so we may type $\mathcal{E}$ with the judgment $\Gamma \vdash \mathcal{E}; \Delta$.

We may now define the coinduction rule, which necessarily takes a different form from the axiomatic presentation of the rest of the equational theory.

Definition 4.4 The equational theory of our calculus ($=_{\mathcal{T}^+}$) is the least congruence which contains the axioms of Table 2 and the unfolding rule, and is is closed under the following coinduction rule.

For any term $\Gamma \vdash t : S \to (T \oslash S)$ *and environment* $\Gamma, u : S \vdash \mathcal{E}; x : T$*:*
if $\Gamma \vdash \nu x.\mathcal{E}[x\{y := x\}] =_{\mathcal{T}^+} \nu v.\mathcal{E}[y/x, v/u][(t\ u)\ \overline{v}]$ *then* $\mathcal{E} =_{\mathcal{T}^+} \_\{x := \varepsilon(u, t)\}$.

4.4 Examples

As a simple application of the coinduction rule, we show that encapsulated definition of an object which discards its state is equivalent to immutable definition.

Lemma 4.5 *If* $x, \overline{y} \notin FV(t)$ *then* $\_\{x := \varepsilon(s, \lambda a.\lambda\overline{b}.t\{b := a\})\} =_{\mathcal{T}^+} \_\{x := t\}$.

Proof. $\nu x.x\{y := x\}\{x := t\} =_{\mathcal{T}^+} t\{y := t\} =_{\mathcal{T}^+} \nu v.t\{v := u\}\{y := t\}$
$=_{\mathcal{T}^+} \nu v.((\lambda a.\lambda\overline{b}.t\{b := a\}\{y := t\})\ u)\ \overline{v}\{y := t\}$ and so this follows directly from the coinduction rule. □

As a slightly more involved example, we may show that a cell storing objects of product type can be macro-expressed using two cells to store each component of the product. Let $y : T_1 \times T_2 \vdash C[\_]; x : \mathtt{var}[T_1 \times T_2]$ be the environment:
$\nu c_1.\nu c_2.\_\{x := \langle \lambda a.(\mathsf{set}(c_1)\ a.1); (\mathsf{set}(c_2)\ a.2), \langle \mathtt{get}(c_1), \mathtt{get}(c_2)\rangle\}\{c_1 := \varepsilon(y.1, \mathtt{cell}_{T_1})\}\{c_2 := \varepsilon(y.2, \mathtt{cell}_{T_2})\}$

Proposition 4.6 $C[\_] =_{\mathcal{T}^+} \_\{x := \varepsilon(y, \mathtt{cell}_{T_1 \times T_2})\}$

Proof. We show that $\nu x.C[x\{y := x\}.1\ v] =_{\mathcal{T}^+} C[y/x, v/u][\lambda z.z]$ and $\nu x.C[x\{y := x\}.2\ v] =_{\mathcal{T}^+} C[y/x, v/u][u]$, and may therefore apply the coinduction rule. □

We may use state encapsulation to implement any stateful objects, not just reference cells. Typically, there may be a variety of ways to represent the internal state of such objects — e.g. a queue might be implemented using two stacks, or using an array with pointers — but hiding this state can make these implementations observably equivalent. A useful principle for proving such equivalences is to identify a term $s : S \to S'$ which relates the input state of one definition to the output state of the other in the following sense.

Definition 4.7 Given terms $t : S \to T \oslash S$, $t : S' \to T \oslash S'$, and $s : S \to S'$, define $t\mathcal{R}(s)t'$ if:

$$t'\ (s\,a)\,\overline{c} =_{\mathcal{T}^+} \nu b.(t\,a)\,\overline{b}\{c := s\,b\}$$

Proposition 4.8 *If* $t\mathcal{R}(s)t'$ *then* $\_\{x := \varepsilon(y,t)\} =_{\mathcal{T}^+} \_\{x := \varepsilon(s\,y,t')\}$

Proof. $\nu x.x\{x := \varepsilon(s\,y,t')\}$
$=_{\mathcal{T}^+} \nu x.\nu c.(t'\ (s\,y))\,\overline{c}\{x := \varepsilon(c,t')\}$
$=_{\mathcal{T}^+} \nu x.\nu c.\nu b.(t\,y)\,\overline{b}\{c := s\,b\}\{x := \varepsilon(c,t')\}$
$=_{tl} \nu x.\nu b.(t\,y)\,\overline{b}\{x := \varepsilon(s\,b,t')\}$.
So by the coinduction rule, $\_\{x := \varepsilon(y,t)\} =_{\mathcal{T}^+} \_\{x := \varepsilon(s\,y,t')\}$ as required. □

5 Coalgebraic Sequoidal Categories

In this section, we axiomatize the properties of a sequoidal CCC which allow us to interpret encapsulated definition in such a way that the equational theory is sound and complete. These are motivated by the construction of sequoidal CCCs of games using the final coalgebra for the sequoid, but reformulated in a way which makes sense in a sequoidal CCC without an explicit decomposition into a model of linear logic — as in the case of HO-style games. As suggested in the previous section, we will do this by taking morphisms between $J(A \oslash \_)$ coalgebras on B and C to be certain natural transformations from $\_ \oslash B$ to $\_ \oslash C$.

Definition 5.1 A sequoidal transformation σ between (objects) A and B in a sequoidal closed category is a natural transformation $\sigma : \_ \oslash A \to \_ \oslash B$ such that the following diagrams commute (for each C):

$$\begin{tikzcd}
C \multimap (\_ \oslash A) \arrow[r, "\theta_{A,C}"] \arrow[d, "C \multimap \sigma"] & (C \multimap \_) \oslash A \arrow[d, "\sigma(C \multimap \_)"] \\
C \multimap (\_ \oslash B) \arrow[r, "\theta_{B,C}"] & (C \multimap \_) \oslash B
\end{tikzcd}
\qquad
\begin{tikzcd}
(\_ \oslash A) \oslash C \arrow[r, "\gamma_{A,C}"] \arrow[d, "\sigma \oslash C"] & (\_ \oslash C) \oslash A \arrow[d, "\sigma(\_ \oslash C)"] \\
(\_ \oslash B) \oslash C \arrow[r, "\gamma_{B,C}"] & (\_ \oslash C) \oslash B
\end{tikzcd}$$

In any sequoidal CCC we can compose a sequoidal transformation $\sigma : \_ \oslash A \to \_ \oslash B$ with a morphism $f \in \mathcal{C}(B, JC)$ using the the trace operator: we define $\Phi(\sigma, f) \in \mathcal{C}(A, JC)$ to be the trace of:

$$A \otimes B \cong B \otimes A \overset{f \otimes A}{\longrightarrow} JC \otimes A \overset{\omega_{C,A}}{\longrightarrow} J(C \oslash A) \overset{J(\sigma_C)}{\longrightarrow} J(C \oslash B)$$

Definition 5.2 A $J(A \oslash \_)$*-coalgebra transformation* between coalgebras $(B, \beta : B \to J(A \oslash B))$ and $(C, \gamma : C \to J(A \oslash C))$ is a sequoidal transformation $\sigma : \_ \oslash B \to \_ \oslash C$ such that $\Phi(\sigma, \gamma) = \beta; J(\sigma_A) : B \to J(A \oslash C)$.

Observe that $\Phi(\sigma_1; \sigma_2, f) = \Phi(\sigma_1, \Phi(\sigma_2, f))$ and $\Phi(I, f) = f$, and thus the $J(A \oslash \_)$-coalgebras and their transformations form a category. We can now define the property required for a sequoidal CCC to interpret encapsulated definition.

Definition 5.3 A sequoidal CCC is coalgebraic if, for each $\mathcal{L}$-object A, the coalgebra $(JA, \delta; \omega_{A,JA} : JA \to J(A \oslash JA))$ is a terminal object in the category of $J(A \oslash \_)$-coalgebras and sequoidal transformations.

In other words, for any morphism $f : S \to J(A \oslash S)$ there is a unique sequoidal natural transformation $(\![f]\!)$ from S to JA such that

$$\Phi((\![f]\!), \delta; \omega_{A,JA}) = f; J(\![f]\!)_A : S \to J(A \oslash JA)$$

The key to showing that our sequoidal CCC of games is sequoidal coalgebraic is the following observation.

Lemma 5.4 *The monoidal functor* $\oslash$ *from* $\mathcal{G}$ *to* $\mathcal{G}_l^{\mathcal{G}_l}$ *is fully faithful.*

Proof. This follows from the observations:

- The map $o \oslash \_ : \mathcal{G}(A, B) \to \mathcal{G}_l(o \oslash A, o \oslash B)$, where o is the game with a single Opponent move, is a bijection (since $o \oslash \_$ just adds a single initial move to each arena, and a pair of opening moves to each sequence in a strategy).
- For any distinct linear strategies $\sigma, \sigma' : C \oslash A \to C \oslash B$ there exists a (linear) strategy $\rho : C \to o$ such that $\sigma; (\rho \oslash B) \neq \sigma'; (\rho \oslash B)$. (We find a minimal sequence in σ but not σ' (w.lo.g.), project its odd prefixes onto C, and obtain a linear strategy $\rho : C \to o$ by adding an opening move in o.)

□

In other words, sequoidal transformations from A to B just correspond to (not generally well-opened) strategies on $A \Rightarrow B$. Moreover, for any morphisms $\sigma : A \to B$ and $\tau : B \to C$ $\Phi((\_ \oslash \sigma), \tau)$ is the parallel composition plus hiding of σ with τ.

Proposition 5.5 *The sequoidal CCC of games, $(\mathcal{G}_w, \mathcal{G}_l, J, \_ \oslash (\_)^\dagger)$ is coalgebraic.*

Proof. It suffices to show that for each $J(A \oslash \_)$-coalgebra $f : S \to J(A \oslash S)$ in $\mathcal{G}_w$, there exists a unique morphism $\llparenthesis f \rrparenthesis : S \to A$ in $\mathcal{G}$ such that $\llparenthesis f \rrparenthesis; \delta; \omega_{A,JA} = f; (J(A \oslash \llparenthesis f \rrparenthesis))$.

Observe that composition with $\delta; \omega_{A,JA} : A \to A \oslash A$ is a natural isomorphism $\phi : \mathcal{G}(\_, A) \to \mathcal{G}_w(\_, A \oslash A)$. Thus we may define $\llparenthesis f \rrparenthesis$ as the $\subseteq$-least fixed point (minimal invariant) of the map sending $g : S \to A$ to $\phi_S^{-1}(f; J(A \oslash g))$. □

We now give a formal semantics of encapsulated definition in a coalgebraic sequoidal CCC and show that it is sound and complete. Given a term $\Gamma \vdash t : S \to T \oslash S$, the morphism $\llbracket \Gamma \vdash t \rrbracket^\ddagger \triangleq \langle \Lambda^{-1}\llbracket t \rrbracket, \pi_l \rangle; \omega_{\llbracket T \rrbracket \oslash \llbracket S \rrbracket, \llbracket \Gamma \rrbracket} : (J\llbracket S \rrbracket \times \llbracket \Gamma \rrbracket) \to \llbracket T \rrbracket \oslash (J\llbracket S \rrbracket \times \llbracket \Gamma \rrbracket)$ is a coalgebra for the functor $J(\llbracket T \rrbracket \oslash \_)$, and therefore we may take $\_\{x := \varepsilon(y, t)\}$ to denote the unique $J(\llbracket T \rrbracket \oslash \_)$-coalgebra transformation from it into $\delta_{\llbracket T \rrbracket}; \omega_{\llbracket T \rrbracket, \llbracket T \rrbracket} : J\llbracket T \rrbracket \to J(\llbracket T \rrbracket \oslash J\llbracket T \rrbracket) = \llbracket x : T \vdash \lambda\overline{y}.\{y := x\} \rrbracket$. Soundness of the unfolding rule follows directly from this definition.

To show that the coinduction rule is sound, we observe that we may interpret each typed environment $\Gamma \vdash \mathcal{E}; \Delta$ as a sequoidal transformation from $\_ \oslash J\llbracket \Gamma \rrbracket$ to $\_ \oslash J\llbracket \Delta \rrbracket$ (since the trace, immutable and encapsulated definition operations are all natural and commute with $\oslash$ and $\multimap$). In particular, this agrees with the interpretation of $\mathcal{E}$ as a head context — i.e. $\llbracket \Gamma \vdash \mathcal{E}; \Delta \rrbracket_{\llbracket T \rrbracket} = \llbracket \Gamma; T \vdash \mathcal{E} : T; \Delta \rrbracket$ at each type T.

Thus an environment $\Gamma \vdash \mathcal{E}; x : T$ such that $\llbracket \nu x.\mathcal{E}[x\{y := x\}] \rrbracket = \nu b.\mathcal{E}[y/x, v/u][(t\, u)\, \overline{b}]$ denotes a $J(\llbracket T \rrbracket \oslash \_)$-coalgebra transformation from $\llbracket \Gamma \vdash t : \rrbracket^\ddagger$ to $\llbracket \Gamma \vdash x\{y := x\} \rrbracket^\ddagger$ and is therefore equal to $\llbracket \Gamma \vdash \_\{x := \varepsilon(u, t)\}; x : T \rrbracket$ by the uniqueness of the terminal map. So we have shown that our equational theory is sound:

Proposition 5.6 *If $\Gamma \vdash s =_{\mathcal{T}^+} t : T; \Delta$ then $\Gamma \vdash \llbracket \Gamma \vdash s : T; \Delta \rrbracket = \llbracket \Gamma \vdash t : T; \Delta \rrbracket$.*

5.1 Completeness

We prove completeness of the equational theory by extending the term model defined in Section 3 with encapsulated definition, and showing that this yields a coalgebraic sequoidal CCC and thus a model of our calculus in which every term is equivalent to its denotation.

We need to show that in our term model over the extended calculus, for each type T the coalgebra $\lambda x : S.\lambda\overline{y}.x\{y := x\} : T \to T \oslash T$ is terminal in the category of $J(T \oslash \_)$-coalgebras and their transformations. Given a $J(T \oslash \_)$-coalgebra in $\mathcal{C}_\mathcal{T}$ — a term $t : S \to T \oslash S$ — the family of head contexts $\lambda\overline{x}.\nu y.[\_\,\overline{y}]\{x := \varepsilon(y, t)\} : R \oslash S \to R \oslash T$ is a sequoidal natural transformation.

This is a $J(T \oslash \_)$-coalgebra transformation from t into $\lambda x : S.\lambda\overline{y}.x\{y := x\}$ by the unfolding rule: it remains to establish that this is the unique such. Any $J(T \oslash \_)$-coalgebra transformation from t into $\lambda x : S.\lambda\overline{y}.x\{y := x\}$ which is the denotation of an environment context $\mathcal{E}$ satisfies $\nu x.\mathcal{E}[x\{y := x\}] =_\mathcal{T} \nu b.\mathcal{E}[b/a][y/x][(t\, a)\, \overline{b}]$ by definition and is therefore equal to $\_\{x := \varepsilon(a, t)\}$ by the coinduction rule. It therefore suffices to show that in fact every sequoidal natural transformation on our term model is the denotation of an environment context.

Lemma 5.7 *For any sequoidal transformation $\sigma : \_ \oslash S \to \_ \oslash T$ there exists an environment $x : S \vdash \mathcal{E} : \overline{y} : T$ such that $\sigma_R = \lambda\overline{y}.\nu x.\mathcal{E}[\_\,\overline{x} : S]$ for each R.*

Proof. By strong normalization for the simply-typed λ-calculus, $x : S; R \vdash (\sigma_R[\lambda\overline{z}.\_\{z := x\}]\, \overline{y}) : R; y : T$ is equivalent to a β-normal form: observe that if R is a ground type then this normal form must be an environment context $x : S \vdash \mathcal{E}; y : T$.

We now show that $\sigma_R = \lambda\overline{y}.\nu x.\mathcal{E}[\_\,\overline{x} : S]$ for every R by structural induction on R. (Since e.g. $\sigma_{R_1 \to R_2}[\_] =_{\mathcal{T}^+} \lambda a.\sigma_{R_1 \to R_2}[\_]\, a =_{\mathcal{T}^+} \lambda a.\sigma_{R_2}[\_\, a]$ by the commutativity property of σ with $\to$). □

Thus we have shown that the encapsulation of a coalgebra morphism is unique up to the equational theory and hence that:

Proposition 5.8 *The term model is a coalgebraic sequoidal CCC.*

It is straightforward to show that Proposition 3.15 extends to terms with encapsulated definition and hence that the equational theory of the calculus is sound and complete with respect to interpretation in a coalgebraic sequoidal CCC.

Theorem 5.9 *$s =_{\mathcal{T}^+} t$ if and only if $\llbracket \Gamma \vdash s : T; \Delta \rrbracket = \llbracket \Gamma \vdash t : T; \Delta \rrbracket$ in every coalgebraic sequoidal CCC.*

References

[1] Martin Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lèvy. Explicit substitutions. In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages, San Francisco, California*, pages 31–46. ACM, 1990.

[2] S. Abramsky, K. Honda and G. McCusker. A fully abstract games semantics for general references. In *Proceedings of LICS '98*. IEEE Press, 1998.

[3] M. Churchill, J. Laird, and G. McCusker. Imperative programs as proofs via game semantics. In *Proceedings of twenty-sixth Symosium on Logic in Computer Science (LICS) '11*. IEEE press, 2011.

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

[5] W. J. Gowers and J. Laird. Sequoidal categories and transfinite games: A coalgebraic approach to stateful objects in game semantics. 2017.

[6] Bart Jacobs and Erik Poll. Coalgebras and monads in the semantics of java. *Theoretical Computer Science*, 291:329 –349, 2003.

[7] L. J. Jagadeesan and R. Jagadeesan. Causality and true concurrency: A data-flow analysis of the pi-calculus. In *Proceedings of AMAST '95*, 1995.

[8] G. Janelidze and G. M. Kelly. A note on actions of a monoidal category. *Theory and Applications of Categories*, 9(4):61 – 91, 2001.

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

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

[11] J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic.* Number 7 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1986.

[12] P. B. Levy. *Call-By-Push-Value.* Semantic Structures in Computation. Kluwer, 2004.

[13] J. Longley. Some programming languages suggested by game models. In *Proceedings of MFPS XXV*. ENTCS, 2009.

[14] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part i. *I AND II. INFORMATION AND COMPUTATION*, 100, 1989.

[15] R. E. Møgelberg and S. Staton. Linearly-used state in models of call-by-value. In *Proceedings of Fourth International Conference on Algebra and Coalgebra in Computer Science (CALCO 2011)*, number 6859 in Lecture Notes in Comput. Sci., pages 293–313, 2011.

[16] E. Moggi. Computational lambda-calculus and monads. Technical Report ECS-LFCS-88-66, University of Edinburgh Department of Computer Science, 1988.

[17] François Pottier. Hiding local state in direct style: a higher-order anti-frame rule. In *Twenty-Third Annual IEEE Symposium on Logic In Computer Science (LICS'08)*, pages 331–340, June 2008.

[18] J. Power and E. Robinson. Premonoidal categories and notions of computation. *Mathematical Structures in Computer Science*, 11, 1993.

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

[20] J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249:3 – 80, 2000.

[21] J. J. M. M. Rutten. A coinductive calculus of streams. *Math. Struct. in Comp. Science*, 15:93 – 147, 2005.