Axioms for Sequentiality, State and Concurrency

Jim Laird:
University of Bath, UK

UCL, September 2023

Motivations and Background

Understanding a key Abramsky insight - how local state is implicit in the *behaviour* of strategies, determined by their *history*.

- Coalgebraic methods can be used to abstract (hide) the explicit state in a system.
- These principles may be used to construct (sub)programs with local state by encapsulating their global state.
- To have a semantics, we need to understand how these objects compose — how to represent shared access, how to pass higher-order values...
- ... by relating them to categorical structure, both general (symmetric monoidal closure, cofree commutative comonoids).
- and specific to a (game) semantics of global and local state.

This work

Use these ideas to extend:

- the simply-typed λ-calculus with an operation for encapsulating state and a coinductive theory for reasoning about it
- its CCC semantics to a *sound* and *complete* categorical model of state encapsulation.
- Extending to objects in concurrent calculis (e.g. the π-calculus).

Final Coalgebras

A final coalgebra for $F : \mathcal{C} \to \mathcal{C}$ is a coalgebra $(B, \beta : B \to FB)$ such that for any F-coalgebra $(S, \sigma : S \to FS)$ there is a unique coalgebra morphism ("anamorphism") $(\![\sigma]\!) : S \to B$.

$$\begin{array}{ccc} S & \xrightarrow[\sigma]{} & FS \\ \downarrow{\scriptstyle (\![\sigma]\!)} & & \downarrow{\scriptstyle F(\![\sigma]\!)} \\ B & \xrightarrow{\beta} & FB \end{array}$$

From global state...

Consider the endofunctor $FX = A \times X$ on the category of sets. A coalgebra $(S, \sigma : S \to A \times S)$ represents an object with globally accessible state: given an input state $s \in S$, return a value $a \in A$ and an output state $s' \in S$.

… to local state

F has a *final coalgebra* $(A^\omega, \alpha : A^\omega \to A \times A^\omega)$, where $\alpha(aw) = \langle a, w \rangle$.
Its anamorphism sends an initial state to a "stream" of copies of σ which pass their output state as the input to the next instance. — e.g. the anamorphism of $\lambda x.\langle x, \neg x \rangle : \texttt{Bool} \to \texttt{Bool} \times \texttt{Bool}$ sends $\texttt{tt}$ to the sequence $\texttt{tt}, \texttt{ff}, \texttt{tt}, \texttt{ff}, \texttt{tt}, \texttt{ff}, \ldots$.

Games and the cofree commutative comonoid

To build a compositional semantics from stateful objects we need structure to share access to them.
In the symmetric monoidal category of (AJM-style) two player games, morphisms are given by sets of *plays* (finite, alternating sequences), where:

- A play in $A \otimes B$ is an interleaving of (tagged) plays in A and B.
- A play in the *cofree commutative comonoid* $!A$ is an interleaving of plays in finitely many copies of A (i.e. $A \otimes A \otimes \ldots$), ignoring tags (we can just pick them in order).

Games and the cofree commutative comonoid

To build a compositional semantics from stateful objects we need structure to share access to them.
In the symmetric monoidal category of (AJM-style) two player games, morphisms are given by sets of *plays* (finite, alternating sequences), where:

- A play in $A \otimes B$ is an interleaving of (tagged) plays in A and B.
- A play in the *cofree commutative comonoid* $!A$ is an interleaving of plays in finitely many copies of A (i.e. $A \otimes A \otimes \ldots$), ignoring tags (we can just pick them in order).

We may equip $!A$ with morphisms to duplicate ($\delta : !A \to !A \otimes !A$), discard ($\eta : !A \to I$) and derelict ($\mathrm{der} : !A \to A$), and for any $f : !A \to B$, a unique *comonoid morphism* $f^\dagger : !A \to !B$ such that $f^\dagger ; \mathrm{der} = f$.

The Free Comonoid as a Final Coalgebra

This cofree commutative comonoid structure is derivable coalgebraically from the *sequoid* functor $\oslash$:
A play in $A \oslash B$ is an interleaving of plays in A and B *which starts in* A.
Thus $!A \cong A \oslash !A$ — moroever, this isomorphism is a final coalgebra for the functor $FX = A \oslash X$

Writing Stateful Programs

We now have a recipe for encapsulating an object with global state $\sigma : S \to A \oslash S$, to an object with local state — its anamorphism $(\![\sigma]\!) : S \to !A$ — with a commutative comonoid structure allowing shared access.

Writing Stateful Programs

We now have a recipe for encapsulating an object with global state $\sigma : S \to A \oslash S$, to an object with local state — its anamorphism $(\![\sigma]\!) : S \to !A$ — with a commutative comonoid structure allowing shared access. **However**

- Linear types are a big syntactic overhead.
- They are unnecessary in the most successful games models of state (based on HO games, in which pointers replace indices in the !).
- We can capture *higher-order* state [Abramsky, Honda and McCusker, LICS 1998] (test-of-time award) using further properties of the sequoid.

Sequoidal CCCs

$(\mathcal{C}, \mathcal{L}, \oslash, J)$ is a sequoidal CCC if:

1. $\mathcal{C}$ is a CCC and $\mathcal{L}$ is a category with finite products.
2. $(\mathcal{L}, \oslash)$ is a $\mathcal{C}$-action (monoidal functor from $\mathcal{C}$ to $\mathcal{L}^{\mathcal{L}}$) with a *dual* $\mathcal{C}^{op}$ action (i.e. right-and-left adjoint) $(\mathcal{L}, \Rightarrow)$.
3. $J : \mathcal{L} \to \mathcal{C}$ preserves products and sends $(\mathcal{L}, \Rightarrow)$ to the internal hom of $\mathcal{C}$.

Derived constructions:

- A lax morphism of actions: $\Lambda^{-1}(J\eta_{X,B}) : JX \times B \to J(X \oslash B)$ (where η is the unit of $B \Rightarrow \_ \dashv \_ \oslash B$)
- A sequoidal trace operator: given $f : A \times B \to J(X \oslash B)$, define $\mathrm{tr}^B_{A,X}(f) : A \to JX = \Lambda(f); J(\epsilon_{B,X})$, where $\epsilon_{B,X} : B \Rightarrow (X \oslash B) \to X$ is the co-unit of $B \Rightarrow \_ \dashv \_ \oslash B$.

Examples

Compact closed models of linear logic — given a cartesian closed category $\mathcal{C}$, a compact closed category with products, $(\mathcal{L}, \otimes, I)$, and a monoidal functor $J : \mathcal{L} \to \mathcal{C}$ with a (monoidal) left adjoint $! : \mathcal{C} \to \mathcal{L}$: we can define $A \oslash X \triangleq !A \otimes X$.

e.g.

- $\mathcal{C}$ is the category of sets and functions, and $\mathcal{L}$ is the category of sets and relations,
- $J : \mathcal{L} \to \mathcal{C}$ is the powerset functor and $\oslash : \mathcal{L} \times \mathcal{C} \to \mathcal{L}$ is the action sending X, A to $X \times A$ — $(\mathcal{P}(X \times A) \cong \mathcal{P}(X)^A)$.

Examples

Compact closed models of linear logic — given a cartesian closed category $\mathcal{C}$, a compact closed category with products, $(\mathcal{L}, \otimes, I)$, and a monoidal functor $J : \mathcal{L} \to \mathcal{C}$ with a (monoidal) left adjoint $! : \mathcal{C} \to \mathcal{L}$: we can define $A \oslash X \triangleq !A \otimes X$.

e.g.

- $\mathcal{C}$ is the category of sets and functions, and $\mathcal{L}$ is the category of sets and relations,
- $J : \mathcal{L} \to \mathcal{C}$ is the powerset functor and $\oslash : \mathcal{L} \times \mathcal{C} \to \mathcal{L}$ is the action sending X, A to $X \times A$ — $(\mathcal{P}(X \times A) \cong \mathcal{P}(X)^A)$.

Not all sequoidal CCCs arise in this way — e.g. *Hyland-Ong Games* (without visibility condition):

- $\mathcal{C}$ is the CCC of HO games and "single-threaded strategies"; $\mathcal{L}$ is its subcategory of strict, linear strategies.
- $J : \mathcal{L} \to \mathcal{C}$ is inclusion and $A \oslash B = \overline{B} \Rightarrow A$ — every initial move in B has a causal pointer into an initial move in A — their polarities are flipped in $B \Rightarrow A$, but not $B \oslash A$.

Global State

For any object S of $\mathcal{C}$ we have a monad $\S_S = S \Rightarrow (\_ \oslash S)$ on $\mathcal{L}$:

- $(J, S \Rightarrow \omega_{S,A} : S \Rightarrow (JX \times A) \to S \Rightarrow J(X \oslash A) \cong J(S \Rightarrow X \oslash A))$ is a *lax morphism of monads* from the usual state monad $S \Rightarrow (\_ \times S)$ for $\mathcal{C}$, inducing read and write operations.
- $\S_S \S_T A \cong \S_{S \times T} A$, $\S_S(A \times B) \cong \S_S(A) \times \S_S(B)$, and $\S_S(A \Rightarrow B) \cong A \Rightarrow \S_S B$. (stored values can be passed out of static scope)

Coalgebraic Encapsulation

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

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

- Define a category $\mathcal{C}_A$ in which objects are $J(A \oslash \_)$ coalgebras, and morphisms from (B, β) to (C, γ) are natural transformations $\sigma : \_ \oslash B \to \_ \oslash C$ such that $\Phi(\sigma, \gamma) = \beta; J(\sigma_A) : B \to J(A \oslash C)$.
- We require that $J(\delta_A); \omega_{A,A} : JA \to J(A \oslash JA)$ is a terminal object in $\mathcal{C}_A$.

In HO-style games, a natural transformation from $\_ \oslash A$ to $\_ \oslash B$ just corresponds to a "multithreaded" strategy from A to B, and Φ composes it with a single-threaded strategy.

A Type Theory for Sequoidal CCCs

We add to the simply-typed λ-calculus with products:

- *Definition* — $s\{x := t\}$ (doesn't bind x)
- *Declaration* — $\nu x.t$ (does bind x)
- *Co-abstraction* — $\overline{\lambda x}.t$ — and *co-application* — $t\,\overline{y}$.

Types are generated by the grammar

$$S, T := B \mid \Pi_{i<n} S_i \mid S \to T \mid T \oslash S$$

Some typing rules

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

$$\frac{\Gamma \vdash t : T ; \Delta, x : S}{\Gamma \vdash \overline{\lambda x} . t : T \oslash S ; \Delta} x \notin \Gamma \qquad \frac{\Gamma \vdash s : S \oslash T ; \Delta}{\Gamma \vdash s \, \overline{x} : S ; \Delta, x : T} x \notin \Delta$$

Denotational Semantics

Interpret $\Gamma \vdash t : T; \Delta$ as a morphism $[\![t]\!]^{\Gamma}_{\Delta} : [\![\Gamma]\!] \to J([\![T]\!] \oslash [\![\Delta]\!])$ in a sequoidal CCC.

- Definition interpreted by the lax morphism of actions from $\times$ to $\oslash$.

$$[\![s\{x := t\}]\!] = \langle [\![s]\!]^{\Gamma}_{\Delta}, [\![t]\!]^{\Gamma} \rangle ; \omega$$

- Declaration is given by the sequoidal trace operator:

$$[\![\nu x.t]\!] = \mathrm{tr}([\![t]\!]^{\Gamma,x}_{\Delta,x})$$

Equational Theory

$$
\begin{aligned}
(\overline{\lambda x}.t)\,\overline{y} &=_{\mathcal{T}} t[y/x] \\
\overline{\lambda x}.(t\,\overline{x}) &=_{\mathcal{T}} t \;(x \notin FV(t)) \\
\nu x.L[t] &=_{\mathcal{T}} L[\nu x.t] \;(x \notin FV(L[\_]) \\
L[t\{x := s\}] &=_{\mathcal{T}} L[t]\{x := s\} \\
\nu x.t\{x := s\} &=_{\mathcal{T}} \nu x.t[s/x]\{x := s\} \\
\nu x.t\{y := x\} &=_{\mathcal{T}} t[y/x] \;(x \notin FV(t)) \\
\nu x.t\{x := s\} &=_{\mathcal{T}} t \;(x \notin FV(t))
\end{aligned}
$$

where

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

These are sound and complete for interpretation in a sequoidal CCC.

Encapsulated Definition

We extend our calculus with encapsulated definition:

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

Example: the reference cell — $\mathtt{new}\,(x := s)\,\mathtt{in}\,t \triangleq$

$$\nu x.t\{x := \varepsilon(s, \lambda b.\overline{\lambda c}.\langle \lambda y.\lambda z.z\{c := y\}, b\{c := b\}\rangle)$$

The *unfolding rule*:

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

establishes the correct read-write behaviour.

Coinduction Rule

An *environment context* is a sequence of definitions and declarations:

$$\mathcal{E} := [\_] \mid \mathcal{E}\{x := t\} \mid \mathcal{E}\{x := \varepsilon(y, t)\} \mid \nu x.\mathcal{E}$$

Coinduction Rule:

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

This soundly and completely axiomatizes the encapsulation operator in the semantics.

Coinduction Rule

An *environment context* is a sequence of definitions and declarations:

$$\mathcal{E} := [\_] \mid \mathcal{E}\{x := t\} \mid \mathcal{E}\{x := \varepsilon(y, t)\} \mid \nu x.\mathcal{E}$$

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 $\mathcal{E} =_{\mathcal{T}^+} \_\{x := \varepsilon(u, t)\}$.

This soundly and completely axiomatizes the encapsulation operator in the semantics.
It may be used to prove that two implementations of an object with hidden state but the same observable behaviour are equivalent. E.g. given $t : S \to A \oslash S$ and $t' : S \to A \oslash S'$, and a *state transformation* $f : S \to S'$ such that

$$\lambda x \lambda \overline{y}.\nu z.(t\, x)\, \overline{z}\{y := f\, z\} = \lambda x.t'\,(f\, x)$$

Then $\_\{x := \varepsilon(u, t)\} =_{\mathcal{T}^+} \_\{x := \varepsilon(f\, u, t')\}$.

Computational Sequoidal Categories

We have an effect (state): we want to model call-by-value, monad types etc (and coproducts).
Obvious step (adjoint sequoidal CCC) ask for $J : \mathcal{L} \to \mathcal{C}$ to have a left adjoint, $\Sigma : \mathcal{C} \to \mathcal{L}$ (cf. adjunction models of cbpv).
But then $\Sigma A \cong \Sigma 1 \oslash A$, since
$\mathcal{L}(\Sigma 1 \oslash A, B) \equiv \mathcal{L}(\Sigma 1, A \Rightarrow B) \cong \mathcal{C}(1, A \Rightarrow B) \cong \mathcal{C}(A, B)$.
Weaker requirement —- a functor $J' : \mathcal{L}' \to \mathcal{C}'$ with a left adjoint which *factorizes* into $J' = H; J; K$, where $H : \mathcal{L}' \to \mathcal{L}$ and $K : \mathcal{C} \to \mathcal{C}'$.

Adjoint Sequoidal CCCs and Concurrency

- If the monad $T : \mathcal{C} \to \mathcal{C}$ induced by an adjoint sequoidal CCC is symmetric monoidal, then its Kleisli category is compact closed, giving a compact closed model of linear logic.
- These are models of the π-calculus (with only replicated input) [Sakayori and Tsukada, 2019].
- In HO games, models in which $J : \mathcal{L} \to \mathcal{C}$ has a left adjoint — $o \oslash \_$, where o is the one-move game — are interleaved or concurrent games: the co-unit $o \oslash A \to A$ is a "spawn" strategy.
- The induced monad is symmetric monoidal —- in "synchronous" games the monoidal strength $(o \oslash A) \times (o \oslash B) \to o \oslash (A \times B)$ decomposes into left and right strengths.
- Adding encapsulated definition is equivalent in expressiveness to the π-calculus. (But with a co-inductive theory of concurrent objects with local state.)