Towards an Algebraic Account of the Intensional Hierarchy

Jim Laird
Dept. of Computer Science, University of Bath, UK

March 30, 2009

The Intensional Hierarchy

Game semantics has been succesfully used to model higher-order languages with many and various effects:

- Control (continuations, exceptions, coroutines).
- State (integer and higher-type references).
- Nondeterminism and concurrency.

Typically, each effect is associated with one or more constraints on games or strategies, which are relaxed to interpret the corresponding language feature.

Describing Models

General algebraic theories are useful for structuring our models:

- Models of higher-order and effectful functional programming: (CCCs, monads, premonoidal categories).
- Models of control (CPS, linear CPS, control categories).
- Models of linear logic/type theory.

These are very useful but...

Imperative effects are singular

Games models of imperative effects in particular are:

- rather concrete — e.g. given by composition with a complicated cell strategy.
- specifically gamey and not obviously related to other models of state (e.g. functor categories, state monads).
- tiresome or downright difficult to prove sound.
- similar, but slightly different, in each setting.

Objective

For each point in the intensional hierarchy, we aim to give:

1. A higher-order language with some combination of computational effects.
2. Some categorical structure, leading to an abstract model of (1).
3. One or more instances of (2) — e.g. a categories of games based on some combination of constraints on strategies.
4. A type theory, based directly on (2), which we can use to interpret and anlyse (1).

Sequoidal Categories

The basis for our account is a new operation on games (the sequoid).

- This captures dependence between events at an "atomic" level.
- It is instance of a general notion (action of a monoidal category) but used to describe structure which is specifically gamey.
- It fits with other useful structure: monoids, products, closed structure, monads, comonads.
- It crops up in different categories of games (AJM, HO/N, graph games...).

Motivations

- Abstract accounts of game semantics models (axiomatic proofs of soundness and completeness).
- Underlying logics and type theories for imperative effects:
  - Extension of Curry-Howard correspondence: what kind of proofs correspond to winning strategies which “cheat”?
  - Lifting theories of resource or interference control to a functional-imperative setting.
- New algebraic approaches to combining effects.

Action of a Monoidal Category

An *action* of a monoidal category $\mathcal{V}$ on a category $\mathcal{A}$ is given by:

- a functor $\_ \oslash \_ : \mathcal{A} \times \mathcal{V} \to \mathcal{A}$
- natural isomorphisms $A \oslash (X \otimes Y) \cong ((A \oslash X) \oslash Y)$ and $A \oslash I \cong A$.

satisfying appropriate conditions relative to the associativity/identity isos for $\mathcal{V}$.
More abstractly: an action of $\mathcal{V}$ on $\mathcal{A}$ is a monoidal functor from $\mathcal{V}$ to $[\mathcal{A}, \mathcal{A}]$.

Sequoidal Categories

A *sequoidal category* is given by:

- Two monoidal categories $\mathcal{C}$, $\mathcal{D}$.
- A monoidal functor $J : \mathcal{C} \to \mathcal{D}$.
- An action of $\mathcal{D}$ on $\mathcal{C}$.
- A natural transformation $w : (J\_ \otimes \_) \to J(\_ \oslash \_)$

E.g. the categories of pointed sets and functions, and pointed strict functions, with the monoid being the product, the monoidal functor being inclusion, and the action being left-strict product.

The AJM Sequoid

In an AJM setting, the sequoid is just like the tensor, except that *play starts on the left*:

- $A \otimes B = (M_A + M_B, [\lambda_A, \lambda_B], P_A \| P_B)$
- $A \oslash B = (M_A + M_B, [\lambda_A, \lambda_B], P_A \| \!\! \lfloor P_B)$

where:

- $P_A \| P_B$ is the set of interleavings of (tagged) plays (we may or may not require this to respect alternation),
- $P_A \| \!\! \lfloor P_B = \{s \in P_A \| P_B \mid s \restriction A = \varepsilon \implies s \restriction B = \varepsilon\}$ is *left-merge*.

Sequoidal Categories of Games

For various combinations of constraints (history-freeness, local alternation, alternation) we may construct a category of games and strategies $\mathcal{G}$ with:

- a monoidal subcategory of *strict* ($\bot$-preserving) strategies $\mathcal{G}_S$.
- action of $\_ \oslash \_$ and $w : A \otimes B \rightarrow A \oslash B$ by restriction of $\_ \otimes \_$.

Distributive Products

In history free games, information cannot flow between the components of $A \oslash B$.
In the "history sensitive" categories of games (with Cartesian product), the natural transformation:

$$\langle \pi_1 \oslash \mathrm{id}_C, \pi_2 \oslash \mathrm{id}_C \rangle : (A \times B) \oslash C \to (A \oslash C) \times (B \oslash C)$$

is an isomorphism.
The inverse — $\delta : (A \oslash C) \times (B \oslash C) \to (A \times B) \oslash C$ — allows information about play in $A \times B$ (which component Opponent chose) to be used in C.

A Binary Reference Cell

Let $\mathrm{assign(n)} : 1 \rightarrow \mathsf{com} \oslash \mathsf{bin} = (\mathsf{skip} \otimes \overline{n}); w$ for $n \in \{0, 1\}$
Then $\langle \mathrm{assign}(0), \mathrm{assign}(1) \rangle; \delta$ acts as a "one-shot" binary reference cell:

$$\begin{array}{ccccc} 1 & \rightarrow & \mathsf{com} \times \mathsf{com} & \oslash & \mathsf{bin} \\ & & O_n & & \\ & & P_n & & \\ & & & & O \\ & & & & P_n \end{array}$$

Linear Function Extensionality

The sequoidal categories of games have the closure property:

$$\mathcal{G}_S(A \oslash B, C) \cong \mathcal{G}_S(A, B \multimap C)$$

Without the "visibility condition", the natural transformation $A \oslash (B \multimap o) \to (A \multimap B) \multimap o$ is an isomorphism.
i.e. we can decompose any "continuation" of $A \multimap B$ into a continuation of B, left-merged with a value of A.

Coroutines

We can use linear function extensionality to perform coroutining between procedures:

$$((\mathrm{com} \oslash o) \multimap B) \otimes ((A \otimes \mathrm{com}) \multimap o) \rightarrow A \multimap B$$

O

P

O

P

O

P

Commuting Adjoints

If we do not require interleaving of plays in $\otimes, \oslash, \multimap$ to respect alternation, then the functors $\_ \oslash A : \mathcal{G}_S \to \mathcal{G}_S$ and $A \multimap \_ : \mathcal{G}_S \to \mathcal{G}_S$ *commute*.
Thus they are both right and left adjoints of each other!

Higher-order References

Composing the unit of the new adjunction — $\eta_{\mathsf{com}} : \mathsf{com} \to (A \multimap \mathsf{com}) \oslash A$ — with $\mathsf{skip} : 1 \to \mathsf{com}$ gives a one-shot *higher-order* reference cell:

$$
\begin{array}{ccccccc}
1 & \to & (A & \multimap & \mathsf{com}) & \oslash & A \\
 & & & & O & & \\
 & & & & P & & \\
 & & & & & & O \\
 & & P & & & & \\
 & & \vdots & & & & \vdots
\end{array}
$$

A Trace Operator

Commuting adjoints give us a trace operator
trace : $\mathcal{G}(A \otimes C, B \otimes C) \to \mathcal{G}(A, B)$ for free:

$$\text{trace}(f) = \Lambda(f; w); \epsilon_B$$

This gives the basis for a calculus for describing these strategies.

A traced calculus

Types:

$$S, T ::= B \mid S \multimap T \mid S \times T \mid S \oslash T$$

Terms-in-context of the form $\Gamma \vdash t : T; \Delta$ are interpreted as morphisms from $[\![\Gamma]\!]$ to $[\![T]\!] \oslash [\![\Delta]\!]$.

- Name hiding binds an input to an output name: $\nu x.t$ is the trace operator.
- Name-abstraction, application and pairing use the fact that $\_\oslash$ commutes with $\multimap$ and $\times$.
- We can output on a fresh name, and left-merge with another term: from $\Gamma \vdash s : A, \Delta$ and $\Gamma' \vdash t : B, \Delta'$, derive $\Gamma, \Gamma' \vdash s \lfloor\!\lfloor \overline{x}\langle t\rangle : A; \Delta, B, \Delta'$

Concurrency

We have symmetric monoidal categories of non-alternating games and strategies $\mathcal{G}$ (in which Opponent always starts) and $\widehat{G}$ (in which Player can start). $\mathcal{G}$ has all the sequoidal structure given so far, plus an adjunction:

$$\widehat{\mathcal{G}}(JA, B) \cong \mathcal{G}(A, o \oslash B)$$

This corresponds to adding parallel composition to our calculus: from $\Gamma \vdash s : o; \Delta$ and $\Gamma' \vdash t : o; \Delta'$, derive:
$\Gamma, \Gamma' \vdash s|t : o; \Delta, \Delta'$

Beyond Linearity

In order to interpret the duplication of arguments, we may:

- Introduce a linear exponential comonad (!).
- Move to a Hyland-Ong style setting with pointers.

In each case the sequoid gives a nice analysis.

The Sequoidal Comonad

Suppose we have a cpo-enriched sequoidal category with distributive products and the decomposition property:

$$A \otimes B \cong (A \oslash B) \times (B \oslash A)$$

For example: alternating, history sensitive games.
Then any fixedpoint (*minimal invariant*) $!A$ for the functor $A \oslash \_$ is a monoidal comonad.
The AJM ad Hyland exponentials are instances.

Sequoidal structure on justified games

- Constraints on arenas/strategies include *innocence*, *visibility*, *local alternation* and *alternation*.
- In each case we have monoidal categories $\mathcal{G}$ of games and strategies, and $\mathcal{G}_S$ of games and *well-opened, strict and linear* strategies, with a monoidal functor $! : \mathcal{G}_S \to \mathcal{G}$ (thread duplication).
- This factors through the inclusion of $\mathcal{G}_S$ into the (Cartesian closed) category of games and well-opened strategies.
- Pointers from initial B-moves in $A \oslash B$ go to initial A-moves — i.e. $A \oslash B = B^{\perp} \Rightarrow A$.

A traced calculus for HO games

- For each category of games, the functors $\_ \oslash A$ and $A \Rightarrow \_$ are commuting adjoints (so the product distributes).
- In each case, this yields a "pseudo-trace" operator from $\mathcal{G}_W(A \times C, B \oslash C)$ to $\mathcal{G}_W(A, B)$.
- Thus for each case we have a model of a fragment of the π-calculus (sufficient to embed the λ-calculus with fixedpoints).
- Imposing constraints controls how threads are combined — e.g. in the alternating (only) case we have an adjunction $\mathcal{G}(A, B) \cong \mathcal{G}_W(A, B \oslash B)$.
- We can express the relevant programming feature in each fragment (and prove soundness).

Further Directions

- We can characterise sequoidal categories and their structure using higher-order category theory. What does this tell us?
- What happens when we combine our functional-imperative type theories and models with rules for e.g. resource, interference or access control?
- We can think of our (recursion-free) type theories as proof systems for various flavours of “sequoidal logic” — an extension of Curry-Howard. What are the *winning strategies* corresponding to proofs? What are the the proof theoretic properties of this logic?