Game-enriched categories

Paul Blain Levy

University of Birmingham

April 16, 2024

Outline

1. Brief introduction to game semantics
2. Ouch, we have $m \times n$ categories of games
3. Hyland's co-Kleisli category of games
4. Game-enrichment gives $m + n$ jobs

A term denotes a strategy

We want to model a higher-order programming language, some version of typed λ-calculus.

A term denotes a strategy

We want to model a higher-order programming language, some version of typed λ-calculus.

Game semantics provides a way to achieve this.

A term denotes a strategy

We want to model a higher-order programming language, some version of typed λ-calculus.

Game semantics provides a way to achieve this.

Each type A denotes an arena, written $[\![A]\!]$.
This is a kind of game-board.

A term denotes a strategy

We want to model a higher-order programming language,
some version of typed λ-calculus.

Game semantics provides a way to achieve this.

Each type A denotes an arena, written $[\![A]\!]$.
This is a kind of game-board.

Each term $\Gamma \vdash M : A$ denotes a strategy, written $[\![M]\!]$.
This is for a suitable game that depends on the arenas $[\![\Gamma]\!]$ and $[\![A]\!]$.

A term denotes a strategy

We want to model a higher-order programming language,
some version of typed λ-calculus.

Game semantics provides a way to achieve this.

Each type A denotes an arena, written $[\![A]\!]$.
This is a kind of game-board.

Each term $\Gamma \vdash M : A$ denotes a strategy, written $[\![M]\!]$.
This is for a suitable game that depends on the arenas $[\![\Gamma]\!]$ and $[\![A]\!]$.

Play alternates between Proponent (the program) and Opponent (the environment).

A term denotes a strategy

We want to model a higher-order programming language,
some version of typed λ-calculus.

Game semantics provides a way to achieve this.

Each type A denotes an arena, written $[\![A]\!]$.
This is a kind of game-board.

Each term $\Gamma \vdash M : A$ denotes a strategy, written $[\![M]\!]$.
This is for a suitable game that depends on the arenas $[\![\Gamma]\!]$ and $[\![A]\!]$.

Play alternates between Proponent (the program) and Opponent (the environment).

Moves can be Questions or Answers.

Example

In call-by-value λ-calculus with recursion and side-effects, here is a term:

$$
\begin{array}{l}
\mathtt{f} : \mathtt{bool} \to \mathtt{bool} \vdash \\
\quad \mathtt{if\ f(true)} \\
\quad \mathtt{then\ false} \\
\quad \mathtt{else\ if\ f(true)\ then\ true\ else\ diverge} \\
\qquad : \mathtt{bool}
\end{array}
$$

Example

In call-by-value λ-calculus with recursion and side-effects, here is a term:

$$
\begin{array}{l}
\texttt{f} : \texttt{bool} \to \texttt{bool} \vdash \\
\quad \texttt{if f(true)} \\
\quad \texttt{then false} \\
\quad \texttt{else if f(true) then true else diverge} \\
\qquad : \texttt{bool}
\end{array}
$$

Example play:

PQ	OA	PQ	OA	PA
f(true)	false	f(true)	true	true

Example

In call-by-value λ-calculus with recursion and side-effects, here is a term:

$$\begin{array}{l} \texttt{f} : \texttt{bool} \to \texttt{bool} \vdash \\ \quad \texttt{if f(true)} \\ \quad \texttt{then false} \\ \quad \texttt{else if f(true) then true else diverge} \\ \qquad : \texttt{bool} \end{array}$$

Example play:

PQ	OA	PQ	OA	PA
f(true)	false	f(true)	true	true

The strategy is deterministic but partial, i.e. can diverge.

Game categories for PL semantics

It is common practice to organise a game model into a category.
Here are some examples:

1. The category of arenas and OP-visible strategies.
   Useful for modelling Idealized Algol.
   (Abramsky and McCusker)

Game categories for PL semantics

It is common practice to organise a game model into a category.
Here are some examples:

1. The category of arenas and OP-visible strategies.
   Useful for modelling Idealized Algol.
   (Abramsky and McCusker)
2. The category of arenas and strategies.
   Useful for modelling general references.
   (Abramsky, Honda and McCusker)

Game categories for PL semantics

It is common practice to organise a game model into a category.
Here are some examples:

1. The category of arenas and OP-visible strategies.
   Useful for modelling Idealized Algol.
   (Abramsky and McCusker)
2. The category of arenas and strategies.
   Useful for modelling general references.
   (Abramsky, Honda and McCusker)
3. The category of strong nominal arenas and equivariant strategies.
   Useful for modelling good general references.
   (Murawski and Tzevelekos)

Obligations

To set up each category of games, we have to

- define the objects
- define the homsets
- define composition
- prove associativity.

Obligations

To set up each category of games, we have to

- define the objects
- define the homsets
- define composition
- prove associativity.

Note

Depending on the specific categorical structure we want,

there may be more operations on strategies, and more equations to prove.

For example, unital laws.

Beyond determinism

Extend each language with nondeterministic choice.

Or with probabilistic choice.

Beyond determinism

Extend each language with nondeterministic choice.

Or with probabilistic choice.

Then the categories need to be adapted.

Same objects, more morphisms.

Deterministic / nondeterministic / probabilistic strategies

Deterministic strategy

A set of finite plays that satisfies a determinacy condition.

Nondeterministic strategy

- To model may-testing, drop the determinacy condition.

Deterministic / nondeterministic / probabilistic strategies

Deterministic strategy

A set of finite plays that satisfies a determinacy condition.

Nondeterministic strategy

- To model may-testing, drop the determinacy condition.
- For must-testing, include some divergences and infinite traces. (Harmer and McCusker)

Deterministic / nondeterministic / probabilistic strategies

Deterministic strategy

A set of finite plays that satisfies a determinacy condition.

Nondeterministic strategy

- To model may-testing, drop the determinacy condition.
- For must-testing, include some divergences and infinite traces. (Harmer and McCusker)
- To model infinite trace equivalence, include all infinite traces. (Levy)

Deterministic / nondeterministic / probabilistic strategies

Deterministic strategy

A set of finite plays that satisfies a determinacy condition.

Nondeterministic strategy

- To model may-testing, drop the determinacy condition.
- For must-testing, include some divergences and infinite traces. (Harmer and McCusker)
- To model infinite trace equivalence, include all infinite traces. (Levy)
- Can model bisimilarity using an element of a final coalgebra.

Probabilistic strategy

- For trace equivalence, a probabilistic strategy associates a real to each finite play. (Danos and Harmer)

Deterministic / nondeterministic / probabilistic strategies

Deterministic strategy

A set of finite plays that satisfies a determinacy condition.

Nondeterministic strategy

- To model may-testing, drop the determinacy condition.
- For must-testing, include some divergences and infinite traces. (Harmer and McCusker)
- To model infinite trace equivalence, include all infinite traces. (Levy)
- Can model bisimilarity using an element of a final coalgebra.

Probabilistic strategy

- For trace equivalence, a probabilistic strategy associates a real to each finite play. (Danos and Harmer)
- Can model bisimilarity using an element of a final coalgebra.

Summary

Basic model	Deterministic	May	Must	Infinite trace	...
OP-visible strategies	•	•	•	•	
General strategies	•	•	•	•	
Equivariant strategies	•	•	•	•	

Summary

Basic model	Deterministic	May	Must	Infinite trace	...
OP-visible strategies	•	•	•	•	
General strategies	•	•	•	•	
Equivariant strategies	•	•	•	•	

Need to construct $m \times n$ categories (for m rows and n columns).

Each requires homsets, composition, associativity etc.

Summary

Basic model	Deterministic	May	Must	Infinite trace	...
OP-visible strategies	•	•	•	•	
General strategies	•	•	•	•	
Equivariant strategies	•	•	•	•	

Need to construct $m \times n$ categories (for m rows and n columns).

Each requires homsets, composition, associativity etc.

Can't we do this just once for each row?

The category $\mathcal{G}_!$

We define a cartesian closed category.

The category $\mathcal{G}_!$

We define a cartesian closed category.

(It's the co-Kleisli category for the Hyland comonad.)

The category $\mathcal{G}_!$

We define a cartesian closed category.

(It's the co-Kleisli category for the Hyland comonad.)

An object is a forest, representing a game where Opponent starts.

The category $\mathcal{G}_!$

We define a cartesian closed category.

(It's the co-Kleisli category for the Hyland comonad.)

An object is a forest, representing a game where Opponent starts.

A morphism $A \longrightarrow B$ is a strategy for the following game B^A.

- Opponent starts playing B.
- Proponent can initiate an A-thread any number of times.
- Only Proponent can switch between threads.

Nondeterministic and probabilistic variations

By varying the notion of strategy, we have nondeterministic variations

$$\mathcal{G}_!^{\mathsf{may}} \quad \mathcal{G}_!^{\mathsf{must}} \quad \mathcal{G}_!^{\mathsf{inftrace}} \quad \mathcal{G}_!^{\mathsf{bisim}}$$

and probabilistic variations

$$\mathcal{G}_!^{\mathsf{probtrace}} \quad \mathcal{G}_!^{\mathsf{probbisim}}$$

Nondeterministic and probabilistic variations

By varying the notion of strategy, we have nondeterministic variations

$$\mathcal{G}_!^{\mathsf{may}} \quad \mathcal{G}_!^{\mathsf{must}} \quad \mathcal{G}_!^{\mathsf{inftrace}} \quad \mathcal{G}_!^{\mathsf{bisim}}$$

and probabilistic variations

$$\mathcal{G}_!^{\mathsf{probtrace}} \quad \mathcal{G}_!^{\mathsf{probbisim}}$$

Each is cartesian closed
and has $\mathcal{G}_!$ embedded as a cartesian closed wide subcategory.

Nondeterministic and probabilistic variations

By varying the notion of strategy, we have nondeterministic variations

$$\mathcal{G}_!^{\mathsf{may}} \quad \mathcal{G}_!^{\mathsf{must}} \quad \mathcal{G}_!^{\mathsf{inftrace}} \quad \mathcal{G}_!^{\mathsf{bisim}}$$

and probabilistic variations

$$\mathcal{G}_!^{\mathsf{probtrace}} \quad \mathcal{G}_!^{\mathsf{probbisim}}$$

Each is cartesian closed
and has $\mathcal{G}_!$ embedded as a cartesian closed wide subcategory.

Setting these up is n jobs.

Proposal

Recall we have $m = 3$ categories for modelling deterministic languages.

1. arenas and OP-visible strategies
2. arenas and strategies
3. strong nominal arenas and equivariant strategies.

Proposal

Recall we have $m = 3$ categories for modelling deterministic languages.

1. arenas and OP-visible strategies
2. arenas and strategies
3. strong nominal arenas and equivariant strategies.

Calling them "categories" is an understatement.

Proposal

Recall we have $m = 3$ categories for modelling deterministic languages.

1. arenas and OP-visible strategies
2. arenas and strategies
3. strong nominal arenas and equivariant strategies.

Calling them "categories" is an understatement.

They are $\mathcal{G}_!$-enriched categories. (I hope.)

Obligations: objects and hom-games

- Objects are arenas, as before.
- For arenas A and B, we give a hom-game $\mathcal{C}(A, B)$ whose strategies are precisely the morphisms from A to B.

Obligations: objects and hom-games

- Objects are arenas, as before.
- For arenas A and B, we give a hom-game $\mathcal{C}(A, B)$ whose strategies are precisely the morphisms from A to B.

Example

In the case of OP-visible strategies, $\mathcal{C}(A, B)$ is the pointer-game on B^A where both Players must obey the visibility constraint.

Obligations: composition and associativity

For arenas A, B, C, composition is no longer an operation on strategies
but a single strategy:
a $\mathcal{G}_!$-morphism $\mathcal{C}(A,B) \times \mathcal{C}(B,C) \longrightarrow \mathcal{C}(A,C)$.

Obligations: composition and associativity

For arenas A, B, C, composition is no longer an operation on strategies
but a single strategy:
a $\mathcal{G}_!$-morphism $\mathcal{C}(A,B) \times \mathcal{C}(B,C) \longrightarrow \mathcal{C}(A,C)$.

For arenas A, B, C, D, associativity is no longer a universally quantified equation
but a single equation:
left-associated and right-associated composition are the same strategy.

Obligations: composition and associativity

For arenas A, B, C, composition is no longer an operation on strategies
but a single strategy:
a $\mathcal{G}_!$-morphism $\mathcal{C}(A, B) \times \mathcal{C}(B, C) \longrightarrow \mathcal{C}(A, C)$.

For arenas A, B, C, D, associativity is no longer a universally quantified equation
but a single equation:
left-associated and right-associated composition are the same strategy.

Traditional composition and associativity follow.

Variation: nondeterministic strategies

For each of our $m = 3$ rows, we want

- a $\mathcal{G}_!^{\mathsf{may}}$-enriched category
- a $\mathcal{G}_!^{\mathsf{must}}$-enriched category
- ...

Variation: nondeterministic strategies

For each of our $m = 3$ rows, we want

- a $\mathcal{G}_!^{\text{may}}$-enriched category
- a $\mathcal{G}_!^{\text{must}}$-enriched category
- ...

So we still have $m \times n$ jobs, right?

Change of base: the principle

Given a (lax) monoidal functor $\mathcal{V} \to \mathcal{W}$,

any $\mathcal{V}$-enriched category becomes a $\mathcal{W}$-enriched one with the same objects.

Change-of-base: application

Since we have an embedding $\mathcal{G}_! \to \mathcal{G}_!^{\mathsf{may}}$

any $\mathcal{G}_!$-enriched category gives a $\mathcal{G}_!^{\mathsf{may}}$-enriched one with the same objects.

Change-of-base: application

Since we have an embedding $\mathcal{G}_! \to \mathcal{G}_!^{\mathsf{may}}$

any $\mathcal{G}_!$-enriched category gives a $\mathcal{G}_!^{\mathsf{may}}$-enriched one with the same objects.

Since we have an embedding $\mathcal{G}_! \to \mathcal{G}_!^{\mathsf{must}}$

any $\mathcal{G}_!$-enriched category gives a $\mathcal{G}_!^{\mathsf{must}}$-enriched one with the same objects.

Change-of-base: application

Since we have an embedding $\mathcal{G}_! \to \mathcal{G}_!^{\mathsf{may}}$

any $\mathcal{G}_!$-enriched category gives a $\mathcal{G}_!^{\mathsf{may}}$-enriched one with the same objects.

Since we have an embedding $\mathcal{G}_! \to \mathcal{G}_!^{\mathsf{must}}$

any $\mathcal{G}_!$-enriched category gives a $\mathcal{G}_!^{\mathsf{must}}$-enriched one with the same objects.

And so on.

Change-of-base: application

Since we have an embedding $\mathcal{G}_! \to \mathcal{G}_!^{\mathsf{may}}$

any $\mathcal{G}_!$-enriched category gives a $\mathcal{G}_!^{\mathsf{may}}$-enriched one with the same objects.

Since we have an embedding $\mathcal{G}_! \to \mathcal{G}_!^{\mathsf{must}}$

any $\mathcal{G}_!$-enriched category gives a $\mathcal{G}_!^{\mathsf{must}}$-enriched one with the same objects.

And so on.

Hom-games, composition, associativity etc. come for free.

Summary

For each row, we define a $\mathcal{G}_!$-enriched category.

Summary

For each row, we define a $\mathcal{G}_!$-enriched category.

For each column, we define a category $\mathcal{H}_!$ (actually a co-Kleisli category) and an identity-on-objects functor $\mathcal{G}_! \to \mathcal{H}_!$.

Summary

For each row, we define a $\mathcal{G}_!$-enriched category.

For each column, we define a category $\mathcal{H}_!$ (actually a co-Kleisli category)
and an identity-on-objects functor $\mathcal{G}_! \to \mathcal{H}_!$.

Overall, this amounts to $m + n$ tasks.

Take-home message

Don't say "category" when you actually have a game-enriched category.

Take-home message

Don't say "category" when you actually have a game-enriched category.

Work harder to tell the whole story.

Take-home message

Don't say "category" when you actually have a game-enriched category.

Work harder to tell the whole story.

This will pay off.