Profunctors — what are they useful for?

Axel Kerinec, **Giulio Manzonetto**, Federico Olimpieri

giulio.manzonetto@lipn.univ-paris13.fr

LIPN, Université Sorbonne Paris Nord

September 29th, 2022

Differential λ-calculus  Relational Semantics

Differential λ-calculus

Relational Semantics

Taylor Expansion

Differential λ-calculus

Relational Semantics

Taylor Expansion

Differential Linear Logic

Resource Calculi

Differential Categories

Differential λ-calculus

Relational Semantics

Taylor Expansion

Differential Linear Logic

Resource Calculi

Differential Categories

Integral Categories

Tangent Categories

'Quantum' Categories

Differential λ-calculus

Relational Semantics

Taylor Expansion

Differential Linear Logic

Resource Calculi

Differential Categories

Integral Categories

Tangent Categories

'Quantum' Categories

Probabilistic Models

Non-deterministic Models

Quantum Models

Weighted Relational Models

Differential λ-calculus

Relational Semantics

Taylor Expansion

Differential Linear Logic

Resource Calculi

Probabilistic Models

Non-deterministic Models

Differential Categories

Quantum Models

Weighted Relational Models

Integral Categories

Tangent Categories

'Quantum' Categories

Joyal

Profunctorial Models

The Ancestors: Intersection Types

Intersection types:

$$\alpha, \beta \ ::= \xi \mid \omega \mid \alpha \to \beta \mid \alpha \wedge \beta$$

$\wedge$ is associative, commutative, idempotent ($\alpha \wedge \alpha = \alpha$) with $\alpha \wedge \omega = \alpha$.

Typing rules. Simply typed rules +

$$\frac{}{\Gamma \vdash_\wedge M : \omega} \qquad \frac{\Gamma \vdash_\wedge M : \alpha \wedge \beta}{\Gamma \vdash_\wedge M : \alpha} \qquad \frac{\Gamma \vdash_\wedge M : \alpha \wedge \beta}{\Gamma \vdash_\wedge M : \beta}$$

The operation $\wedge$ induces a subtyping relation $\alpha \leq \beta$, e.g.

$$\frac{\alpha' \leq \alpha \quad \beta \leq \beta'}{\alpha \to \beta \leq \alpha' \to \beta'}$$

Theorem M is typable with $\alpha \neq \omega \iff M$ is head-normalizable.

Type inference/inhabitation is undecidable $\rightsquigarrow$ impredicative techniques.

Filter Models (Barendregt-Coppo-Dezani'83)

$$[\![P]\!] \cong \{\alpha \mid \vdash_\wedge P : \alpha\} \in \text{Filters}$$

- Programs

```
[...]
let rec f x =
if (x = 0)or(x =1)
then 1
else f(x-2) + f(x-1)
[...]
```

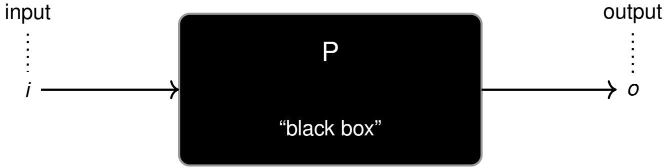
Filter Models (Barendregt-Coppo-Dezani'83)

$$[\![P]\!] \cong \{\alpha \mid \ \vdash_\wedge P : \alpha\} \in \text{Filters}$$

- Programs = Scott continuous functions

$$\underbrace{[\![P]\!]}_{\text{program}} : \quad \underbrace{\mathcal{D}}_{\text{input}} \quad \underbrace{\to}_{\text{computations}} \quad \underbrace{\mathcal{D}}_{\text{output}}$$

$$\underbrace{P}_{\text{program}} \quad \underbrace{n}_{\text{input}} \quad \underbrace{\rightsquigarrow}_{\text{computations}} \quad \underbrace{m}_{\text{output}}$$



Quantitative Semantics

Intensional view on Programs

Linear Logic allows to “open the box”. . .

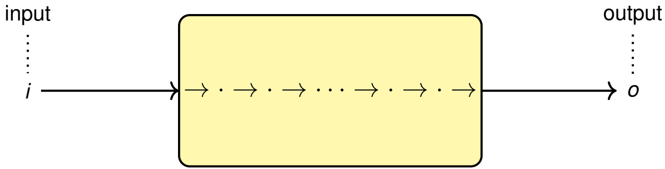
Quantitative Semantics

Intensional view on Programs

Linear Logic allows to "open the box"...

Quantitative Properties

- Number of steps to termination,
- Number of calls to the argument at runtime,
- Amount of resources used during the computation,
- Non-deterministic setting: number of "ways" to get the output.
- Probabilistic setting: the probability of getting the output.



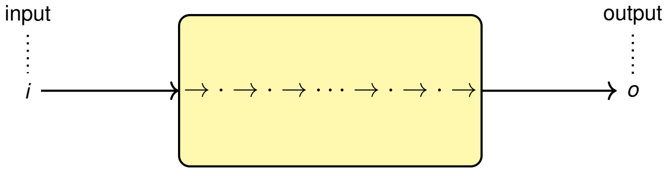
Quantitative Semantics

Intensional view on Programs

Linear Logic allows to “open the box”. . .

Quantitative Properties

- Number of steps to termination,
- Number of calls to the argument at runtime,
- Amount of resources used during the computation,
- Non-deterministic setting: number of “ways” to get the output.
- Probabilistic setting: the probability of getting the output.



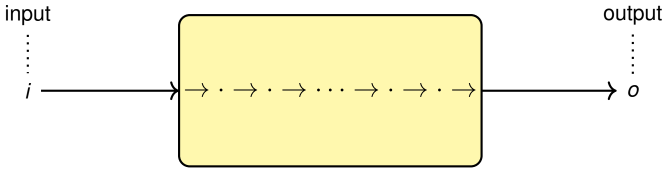
Quantitative Semantics

Intensional view on Programs

Linear Logic allows to “open the box”...

Quantitative Properties

- Number of steps to termination,
- Number of calls to the argument at runtime,
- Amount of resources used during the computation,
- Non-deterministic setting: number of “ways” to get the output.
- Probabilistic setting: the probability of getting the output.



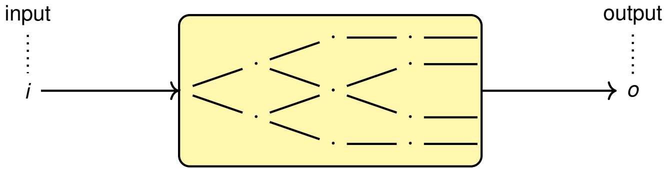
Quantitative Semantics

Intensional view on Programs

Linear Logic allows to “open the box”. . .

Quantitative Properties

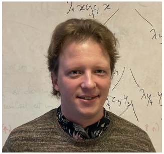
- Number of steps to termination,
- Number of calls to the argument at runtime,
- Amount of resources used during the computation,
- Non-deterministic setting: number of “ways” to get the output.
- Probabilistic setting: the probability of getting the output.



The Relational Semantics

Antonio Bucciarelli

Thomas Ehrhard



Flavien Breuvart

Domenico Ruoppolo

The Relational Semantics

The category **MRel** is the simplest quantitative model of Linear Logic:

- Data/Objects: sets
- Program/Morphism $A \to B$: relation from $\mathcal{M}_f(A)$ and B.

calls to the argument ... output

$$[\![P]\!] \subseteq \mathcal{M}_f(A) \times B$$

MRel is a Cartesian closed category, therefore a semantics of λ-calculus.

A. Bucciarelli, T. Ehrhard, G. Manzonetto:
Not Enough Points Is Enough. CSL 2007: 298-312

Relational type systems

Relational types:

$$\alpha, \beta \ ::= \ \xi \mid \mu \multimap \alpha$$

Multi-types:

$$\mu, \nu ::= [\alpha_1, \dots, \alpha_k] \qquad (k \in \mathbb{N})$$

Idea: Substitute intersection $\wedge$ by a LL tensor product $\otimes$ satisfying commutativity, associativity, neutrality, not idempotence $\alpha \otimes \alpha \neq \alpha$.

$$[\alpha_1, \dots, \alpha_k] = \alpha_1 \otimes \cdots \otimes \alpha_k$$

Relational type systems

Relational types:

$$\alpha, \beta \ ::= \ \xi \mid \mu \multimap \alpha$$

Multi-types:

$$\mu, \nu ::= [\alpha_1, \ldots, \alpha_k] \qquad (k \in \mathbb{N})$$

Typing rules

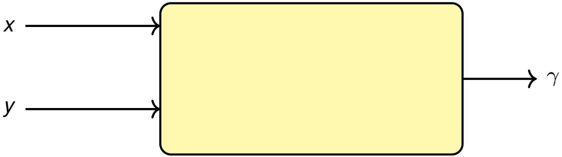
$$\frac{}{x : [\alpha] \vdash x : \alpha} \qquad \frac{\Gamma, x : \mu \vdash M : \alpha}{\Gamma \vdash \lambda x.M : \mu \multimap \alpha}$$

$$\frac{\Gamma_0 \vdash M : [\beta_1, \ldots, \beta_n] \multimap \alpha \quad \Gamma_1 \vdash N : \beta_1 \quad \cdots \quad \Gamma_n \vdash N : \beta_n}{\sum_{i=0}^{n} \Gamma_i \vdash MN : \alpha}$$

Intuitively...

What does it mean that

$$\vdash \lambda xy.M : [\alpha_1, \alpha_2, \alpha_3] \multimap [\beta] \multimap \gamma \quad ?$$



During its execution *M* is going to call

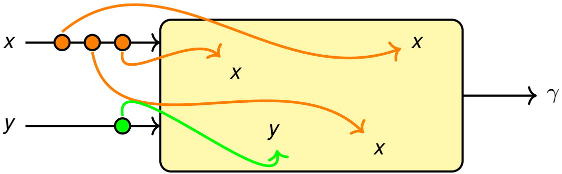
- 3 times its argument *x*, with type $\alpha_1, \alpha_2, \alpha_3$ (respectively);
- 1 time its argument *y*, with type β;

in order to produce a result of type γ.

Intuitively. . .

What does it mean that

$$\vdash \lambda xy.M : [\alpha_1, \alpha_2, \alpha_3] \multimap [\beta] \multimap \gamma \quad ?$$



During its execution *M* is going to call

- 3 times its argument *x*, with type $\alpha_1, \alpha_2, \alpha_3$ (respectively);
- 1 time its argument *y*, with type β;

in order to produce a result of type γ.

Quantitative features

Given a derivation Π, define

$$\#\Pi = \text{size of the derivation tree}$$

Quantitative Subject Reduction

If Π is a derivation of

$$\Gamma \vdash (\lambda x.M)N : \alpha$$

then there exists a derivation Π' of

$$\Gamma \vdash M\{N/x\} : \alpha$$

such that $\#\Pi' < \#\Pi$.

Quantitative properties

Theorem (De Carvalho'08). If P has type α then

1. P is head-normalizable;
2. it is possible to compute an upper bound $\#\alpha$ to the number of $\to_h$.

Theorem (Bucciarelli et al'18). Type inhabitation is decidable.

We actually have a model, so what?

Switching to denotational models we capture operational properties beyond termination of head-reduction.

$$[\![M]\!]^{\mathcal{D}} = \{(\Gamma, \alpha) \mid \Gamma \vdash M : \alpha\}$$

Theorem (Soundness)

$$M =_\beta N \quad \Rightarrow \quad [\![M]\!]^{\mathcal{D}} = [\![N]\!]^{\mathcal{D}}$$

More generally, we would like to understand the theory of a model $\mathcal{D}$.

Assume $[\![M]\!]^{\mathcal{D}} = [\![N]\!]^{\mathcal{D}}$,

what can we say about M, N in terms of their operational properties?

The Böhm Tree Semantics

Given a program M, its Böhm tree $\mathrm{BT}(M)$ is defined by:

- If M does not have a hnf, then

$$\mathrm{BT}(M) = \perp,$$

  where $\perp$ represents the undefined.

- Otherwise $M \twoheadrightarrow_h \lambda x_1 \ldots x_n.y\, M_1 \cdots M_k$ and

$$\mathrm{BT}(M) = \lambda x_1 \ldots x_n.y$$

$$\mathrm{BT}(M_1) \quad \cdots \quad \mathrm{BT}(M_k)$$

Example

$\mathrm{BT}(\mathbf{Y})$

$=$

$\lambda f.f$

f

f

f

$\vdots$

The Böhm Tree Semantics

Given a program M, its Böhm tree $\mathrm{BT}(M)$ is defined by:

- If M does not have a hnf, then

$$\mathrm{BT}(M) = \bot,$$

  where $\bot$ represents the undefined.

- Otherwise $M \twoheadrightarrow_h \lambda x_1 \dots x_n.y\, M_1 \cdots M_k$ and

$$\mathrm{BT}(M) = \lambda x_1 \dots x_n.y$$

$$\mathrm{BT}(M_1) \quad \cdots \quad \mathrm{BT}(M_k)$$

Example

$\mathrm{BT}(\mathbf{Y})$

$\shortparallel$

$\bot$

Approximation Theorem

$$\mathrm{BT}(M) = \bigsqcup \mathcal{A}(M)$$

where $\mathcal{A}(M) = \{A \mid M \twoheadrightarrow_\beta M' \;\&\; A \text{ finite approximant of } M'\}$

The Böhm Tree Semantics

Given a program M, its Böhm tree $\mathrm{BT}(M)$ is defined by:

- If M does not have a hnf, then

$$\mathrm{BT}(M) = \bot,$$

  where $\bot$ represents the undefined.

- Otherwise $M \twoheadrightarrow_h \lambda x_1 \dots x_n.y\, M_1 \cdots M_k$ and

$$\mathrm{BT}(M) = \lambda x_1 \dots x_n.y$$

$$\mathrm{BT}(M_1) \quad \cdots \quad \mathrm{BT}(M_k)$$

Example

$\mathrm{BT}(\mathbf{Y})$

$=$

$\lambda f.f$

$|$

$\bot$

Approximation Theorem

$$\mathrm{BT}(M) = \bigsqcup \mathcal{A}(M)$$

where $\mathcal{A}(M) = \{A \mid M \twoheadrightarrow_\beta M' \text{ \& } A \text{ finite approximant of } M'\}$

The Böhm Tree Semantics

Given a program M, its Böhm tree $\mathrm{BT}(M)$ is defined by:

- If M does not have a hnf, then

$$\mathrm{BT}(M) = \bot,$$

  where $\bot$ represents the undefined.

- Otherwise $M \twoheadrightarrow_h \lambda x_1 \dots x_n.y\, M_1 \cdots M_k$ and

$$\mathrm{BT}(M) = \lambda x_1 \dots x_n.y$$

$$\mathrm{BT}(M_1) \quad \cdots \quad \mathrm{BT}(M_k)$$

Example

$\mathrm{BT}(\mathbf{Y})$

$=$

$\lambda f.f$

f

$\bot$

Approximation Theorem

$$\mathrm{BT}(M) = \bigsqcup \mathcal{A}(M)$$

where $\mathcal{A}(M) = \{A \mid M \twoheadrightarrow_\beta M' \ \&\ A \text{ finite approximant of } M'\}$

The Böhm Tree Semantics

Given a program M, its Böhm tree $\mathrm{BT}(M)$ is defined by:

- If M does not have a hnf, then

$$\mathrm{BT}(M) = \bot,$$

  where $\bot$ represents the undefined.

- Otherwise $M \twoheadrightarrow_h \lambda x_1 \dots x_n.y\, M_1 \cdots M_k$ and

$$\mathrm{BT}(M) = \lambda x_1 \dots x_n.y$$

$$\mathrm{BT}(M_1) \quad \cdots \quad \mathrm{BT}(M_k)$$

Example

$\mathrm{BT}(\mathbf{Y})$

$=$

$\lambda f.f$

|

f

|

f

|

$\bot$

Approximation Theorem

$$\mathrm{BT}(M) = \bigsqcup \mathcal{A}(M)$$

where $\mathcal{A}(M) = \{A \mid M \twoheadrightarrow_\beta M' \ \&\ A \text{ finite approximant of } M'\}$

The Böhm Tree Semantics

Given a program M, its Böhm tree $\mathrm{BT}(M)$ is defined by:

- If M does not have a hnf, then

$$\mathrm{BT}(M) = \bot,$$

  where $\bot$ represents the undefined.

- Otherwise $M \twoheadrightarrow_h \lambda x_1 \ldots x_n.y\, M_1 \cdots M_k$ and

$$\mathrm{BT}(M) = \lambda x_1 \ldots x_n.y$$

$$\mathrm{BT}(M_1) \quad \cdots \quad \mathrm{BT}(M_k)$$

Example

$\mathrm{BT}(\mathbf{Y})$

$=$

$\lambda f.f$

$|$

f

$|$

f

$|$

f

$|$

$\bot$

Approximation Theorem

$$\mathrm{BT}(M) = \bigsqcup \mathcal{A}(M)$$

where $\mathcal{A}(M) = \{A \mid M \twoheadrightarrow_\beta M' \ \&\ A \text{ finite approximant of } M'\}$

The Böhm Tree Semantics

Given a program M, its Böhm tree $\mathrm{BT}(M)$ is defined by:

- If M does not have a hnf, then

$$\mathrm{BT}(M) = \bot,$$

  where $\bot$ represents the undefined.

- Otherwise $M \twoheadrightarrow_h \lambda x_1 \dots x_n.y\, M_1 \cdots M_k$ and

$$\mathrm{BT}(M) = \lambda x_1 \dots x_n.y$$

$$\mathrm{BT}(M_1) \quad \cdots \quad \mathrm{BT}(M_k)$$

Example

$\mathrm{BT}(\mathbf{Y})$

$=$

$\lambda f.f$

f

f

f

$\vdots$

The Böhm Tree Semantics

$$\mathcal{B} \vdash M = N \iff \mathrm{BT}(M) = \mathrm{BT}(N)$$

Typed vs Untyped occurrences

In the following derivation, the subterm Ω is not typed:

$$\dfrac{\dfrac{x : [[] \multimap \alpha] \vdash x : [] \multimap \alpha}{x : [[] \multimap \alpha] \vdash x\Omega : \alpha}}{\lambda x.x\Omega : [[] \multimap \alpha] \multimap \alpha}$$

This derivation is in typed normal form.

On the contrary, the redex occurrence $\mathrm{I}x = (\lambda y.y)x$ is typed in:

$$\dfrac{\dfrac{x : [[\alpha] \multimap \alpha] \vdash x : [\alpha] \multimap \alpha \qquad \dfrac{\vdash \mathrm{I} : [\alpha] \multimap \alpha \quad x : [\alpha] \vdash x : \alpha}{x : [\alpha] \vdash \mathrm{I}x : \alpha}}{x : [[\alpha] \multimap \alpha, \alpha] \vdash x\Omega : \alpha}}{\lambda x.x(\mathrm{I}x) : [[\alpha] \multimap \alpha, \alpha] \multimap \alpha}$$

Typed vs Untyped occurrences

In the following derivation, the subterm Ω is not typed:

$$\dfrac{\dfrac{x : [[] \multimap \alpha] \vdash x : [] \multimap \alpha}{x : [[] \multimap \alpha] \vdash x\Omega : \alpha}}{\lambda x.x\Omega : [[] \multimap \alpha] \multimap \alpha}$$

This derivation is in typed normal form.

On the contrary, the redex occurrence $\mathsf{Ix} = (\lambda \mathsf{y}.\mathsf{y})\mathsf{x}$ is typed in:

$$\dfrac{\dfrac{x : [[\alpha] \multimap \alpha] \vdash x : [\alpha] \multimap \alpha \qquad \dfrac{\vdash \mathsf{I} : [\alpha] \multimap \alpha \quad x : [\alpha] \vdash x : \alpha}{x : [\alpha] \vdash \mathsf{Ix} : \alpha}}{x : [[\alpha] \multimap \alpha, \alpha] \vdash x\Omega : \alpha}}{\lambda x.x(\mathsf{Ix}) : [[\alpha] \multimap \alpha, \alpha] \multimap \alpha}$$

Let Π be a derivation of $\Gamma \vdash M : \alpha$ in typed nf

Associate an approximant $A_\Pi \in \mathcal{A}$ s.t. $\Gamma \vdash A_\Pi : \alpha$ and $A_\Pi \sqsubseteq M$ by

$$x : [\alpha] \vdash x : \alpha \qquad\qquad \Rightarrow A_\Pi = x$$

$$\dfrac{\dfrac{\Pi'}{\Gamma, x : \sigma \vdash M : \alpha}}{\Gamma \vdash \lambda x.M : \sigma \multimap \alpha} \qquad\qquad \Rightarrow A_\Pi = \lambda x.A_{\Pi'}$$

$$\dfrac{\dfrac{\Pi_0}{\Gamma_0 \vdash M : [\beta_1, \dots, \beta_n] \multimap \alpha} \quad \dfrac{\Pi_i}{\Gamma_i \vdash N : \beta_i}}{\sum_{i=0}^{n} \Gamma_i \vdash MN : \alpha} \qquad\qquad \Rightarrow A_\Pi = A_{\Pi_0}(\bigsqcup_{i=1}^{n} A_{\Pi_i})$$

Note that $\bigsqcup_{i=1}^{0} A_i = \bot$

Antonio Bucciarelli, Delia Kesner, Simona Ronchi Della Rocca.
The Inhabitation Problem for Non-idempotent Intersection Types.
IFIP TCS 2014: 341-354

The Approximation Theorem

Theorem. For $M \in \Lambda$, we have

$$\Gamma \vdash M : \alpha \iff \exists A \in \mathcal{A}(M) \,.\, \Gamma \vdash A : \alpha$$

Proof. ($\Rightarrow$) Let Π be a derivation of $\Gamma \vdash M : \alpha$ not in typed nf.

- Then, by the Weighted Subject Reduction,

$$M = M_0 \to_{\text{typed}-\text{redex}} M_1 \twoheadrightarrow_{\text{typed}-\text{redex}} M_n = N$$

  and there exists a derivation Π' of $\Gamma \vdash N : \alpha$ in typed nf.
- Therefore, $\Gamma \vdash A_{\Pi'} : \alpha$ with $A_{\Pi'} \sqsubseteq N$.
- Since $M \twoheadrightarrow_\beta N$ and $A_{\Pi'} \sqsubseteq N$, we conclude $A_{\Pi'} \in \mathcal{A}(M)$.

($\Leftarrow$) Easy. □

F. Breuvart, G. Manzonetto, D. Ruoppolo: Relational Graph Models at Work. Log. Methods Comput. Sci. 14(3) (2018)

The Approximation Theorem and Its Consequences

Theorem (Semantic Approximation Theorem)

For $M \in \Lambda$, we have

$$[\![M]\!] = \bigcup_{A \in \mathcal{A}(M)} [\![A]\!]$$

Corollary 1. M has an hnf $\iff [\![M]\!] \neq \emptyset$

Proof.

- M has no hnf $\Rightarrow \mathcal{A}(M) = \{\bot\}$. Therefore, $[\![M]\!] = [\![\bot]\!] = \emptyset$.
- M does have a hnf $\Rightarrow M$ typable $\Rightarrow [\![M]\!] \neq \emptyset$.

Corollary 2. $\mathrm{BT}(M) = \mathrm{BT}(N) \quad \Rightarrow \quad [\![M]\!] = [\![N]\!]$.

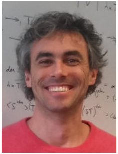
Proof.

$$\begin{aligned} [\![M]\!] &= \textstyle\bigcup_{A \in \mathcal{A}(M)} [\![A]\!], && \text{by Approximation Theorem,} \\ &= \textstyle\bigcup_{A \in \mathcal{A}(N)} [\![A]\!], && \text{by } \mathcal{A}(M) = \mathcal{A}(N), \\ &= [\![N]\!], && \text{by Approximation Theorem.} \end{aligned}$$

The Weighted Relational Semantics

Jim Laird

Guy McCusker



Michele Pagani

Idea: A relation R between sets A to B can be seen as

$$R \subseteq A \times B$$

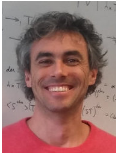
Replace **Bool** by an arbitrary (continuous) semi-ring:

$$R : A \times B \to \mathcal{R}$$

The Weighted Relational Semantics

Jim Laird

Guy McCusker



Michele Pagani

Idea: A relation R between sets A to B can be seen as

$$R : A \times B \to \mathbf{Bool}$$

Replace **Bool** by an arbitrary (continuous) semi-ring:

$$R : A \times B \to \mathcal{R}$$

The Profunctorial Semantics

Axel Kerinec

Federico Olimpieri

Higher-Order Generalization

Weighted relations are functions

$$R : A \times B \to \mathcal{R}$$

Rather than functions, we use functors:

$$F : \mathbf{A}^{op} \times \mathbf{B} \Rightarrow \mathbf{Set}$$

thus $\mathbf{A}, \mathbf{B}$ are categories.

Marcelo Fiore, Nicola Gambino, Martin Hyland, and Glynn Winskel.
The cartesian closed bicategory of generalised species of structures.
Journal of the London Mathematical Society 77, 1 (2008), 203–220.

Higher-Order Generalization

Weighted relations are functions

$$R : A \times B \to \mathcal{R}$$

Rather than functions, we use functors:

$$F : \mathbf{A}^{op} \times \mathbf{B} \Rightarrow \mathbf{Set}$$

thus $\mathbf{A}, \mathbf{B}$ are categories.

Marcelo Fiore, Nicola Gambino, Martin Hyland, and Glynn Winskel.
The cartesian closed bicategory of generalised species of structures.
Journal of the London Mathematical Society 77, 1 (2008), 203–220.

[Fake news] General enough to encompass models of quantum λ-calculus.

Michele Pagani, Peter Selinger, Benoît Valiron:
Applying quantitative semantics to higher-order quantum computing.
POPL 2014: 647-658

"Il ne faut pas avoir peur. . . "

— Thomas Ehrhard

Intuitively...

Set-theoretic	Category theoretic
sets	categories
functions	functors
equations	(natural) isomorphisms

$$\text{Relations} \quad \Rightarrow \quad \text{Profunctors}$$

$$[\![M]\!](\Gamma, \alpha) \cong \left\{ \dfrac{\begin{matrix}\Pi \\ \vdots\end{matrix}}{\Gamma \vdash M : \alpha} \right\}$$

Profunctorial Approximation Theorem

Soundness only holds "up to isomorphism":

$$M =_\beta N \quad \Rightarrow \quad [\![M]\!] \cong [\![N]\!].$$

Theorem (Approximation Theorem)

There is a natural isomorphism

$$\mathrm{appr}(M) : [\![M]\!] \cong [\![\mathrm{BT}(M)]\!]$$

Now the interpretation of a λ-term contains much more structure...

Profunctorial Approximation Theorem

Corollary. Characterization of the theory of the model:

$$\mathrm{BT}(M) = \mathrm{BT}(N) \iff [\![M]\!] \cong [\![N]\!].$$

Proof ($\Rightarrow$) As in the relational semantics.

($\Leftarrow$) Assume $[\![M]\!] \cong [\![N]\!]$ and $\mathrm{BT}(M) \neq \mathrm{BT}(N)$, towards a contradiction.

- Then $\mathrm{nf}([\![M]\!]) = \mathrm{nf}([\![N]\!])$, but there exists, say, $P \in \mathcal{A}(M) - \mathcal{A}(N)$.
- Take a derivation $\Pi \in \mathrm{nf}([\![M]\!]) = \mathrm{nf}([\![N]\!])$ such that $A_\Pi = P$.
- $\Pi \in [\![N']\!]$ for some N' such that $N \twoheadrightarrow_\beta N'$.
- We obtain $A_\Pi = P \leq_\perp N'$, thus $P \in \mathcal{A}(N)$. Contradiction. □

Axel Kerinec, Giulio Manzonetto, Federico Olimpieri.
Why Are Proofs Relevant in Proof-Relevant Bicategorical Models?
(Conditionally) accepted in POPL 2023.

Thinking in progress...

Decategorification Pseudofunctor (change of base)

$$\mathrm{Dec} : \mathrm{Prof} \to \mathrm{Polr}$$

where Polr = preorders and monotonic relations.

Theorem

$$\mathrm{Dec}([\![M]\!]) = [\![M]\!]^{\mathrm{MPolr}}$$

In general

$$\mathrm{Th}(\mathcal{D}^{\mathrm{Prof}}) \subseteq \mathrm{Th}(\mathrm{Dec}(\mathcal{D}^{\mathrm{Prof}}))$$

BUT we believe the results transfer:

- The model with an atom $\star$ and no equations induces as theory $\mathcal{B}$
- The model with $[\star] \to \star \simeq \star$ induces $\mathcal{H}^+$
- The model with $[] \to \star \simeq \star$ induces $\mathcal{H}^*$

That cannot be a coincidence.

Happy Birthday, Thomas!