A few bridges between operational and denotational semantics of programming languages

Soutenance d'habilitation à diriger les recherches

Tom Hirschowitz

November 17, 2017

Structure of the talk

- Trajectory.
- Bibliography (overview of contributions).
- Focus on one chapter of manuscript: shapely monads.

Outline

Early work

- PhD thesis on modular programming.

  Viewing programs as component assemblies.

- Further work on component-oriented programming.

  Modify modular structure at runtime.

Goal

Ensure safety!

Mathematical description of programming languages

Basic method

Structural operational semantics (SOS).

Presenting execution of a programming language as an

- inductively generated,
- labelled,
- binary

transition relation between programs.

Example (Synchronisation in the π-calculus)

$$\frac{P \xrightarrow{\bar{a}\langle m \rangle} P' \qquad Q \xrightarrow{a(m)} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

P sends message m on channel a,
Q receives m on a
$\implies P|Q$ does a silent transition to $P'|Q'$.

Behavioural equivalences

Important question in programming language research

When are two given programs equivalent?

Several answers: behavioural equivalences.

Important reasoning tool

Denotational semantics, a.k.a. models.

- In a sense close to model theory: interpret the syntax.
- E.g. (Scott), types as ordered sets, functions as monotone maps.
- Difficulty: no general notion of model!
  - fairly standard for purely functional languages,
  - for 'logical' languages as well,
  - hard work, e.g., for linear logic,
  - currently debated for type theory,
  - undefined in general.

Need of general results

- Mostly methods, little common theory.
- Especially in the interplay between SOS and variable binding.

E.g., $\forall x.A(x) = \forall y.A(y)$.

- So started looking around, learnt bits of proof theory, linear logic, and finally category theory.

Existing approaches I

Syntactic frameworks for SOS[1].

- Description of inductive generation process:

  $$\text{basic rules} \rightsquigarrow \text{transition relation.}$$

- General results under hypotheses, e.g., some behavioural equivalence (bisimilarity) is a congruence.
- No general notion of model.

[1] GSOS, de Simone, tyft/tyxt, PANTH, . . .

Existing approaches II

Outside SOS: graphical calculi.

- Programs are (kind of) graphs.
- Transitions given by local transformation rules.
- Examples:
  - Petri nets (Petri, 1962).
  - Proof nets (Girard, 1987), interaction nets (Lafont, 1990).
  - To a certain extent, bigraphs (Jensen and Milner, 2004).
  - Wire calculus (Sobociński, 2009).
- Description of (non-inductive) generation process.
- No general notion of model.
  E.g., took quite long to work out for proof nets[a]!

[a] Bierman. *On Intuitionistic Linear Logic*. PhD thesis, Cambridge, 1993.

Existing approaches III

Categorical frameworks (bialgebraic semantics (Fiore et al.), nominal logic (Pitts et al.), . . .).

- Description of inductive generation process under hypotheses.
- General results (as before).
- Specification: automatic notion of model.
- Confession: haven't really managed to appropriate these.

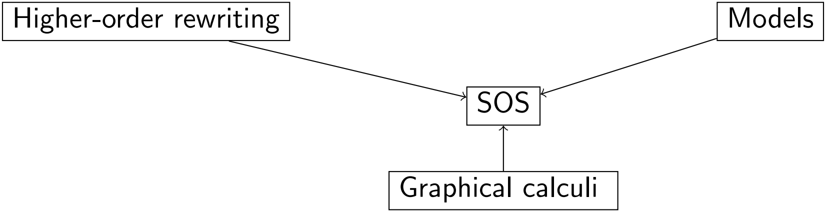
Long-term motivation

Reconcile theory and practice on these matters.

Difficulty

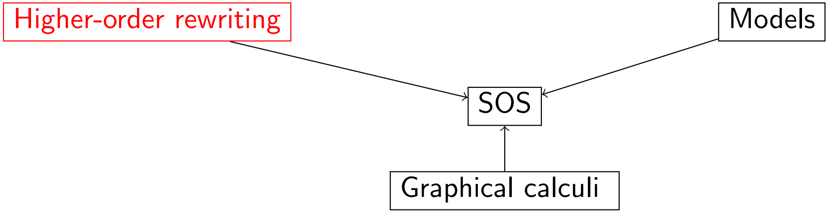
SOS is a wild territory.

Strategy: approach SOS from tamer settings.



Outline

Approaching SOS I: higher-order rewriting



Approaching SOS I

Higher-order rewriting (HOR):

- $\approx$ SOS for logic (vs. programming languages);
- main interest: determinism (vs. behavioural equivalences).

HOR as a SOS fragment

- No labels.
- Transition relation is a congruence (transitions may occur anywhere in the program).

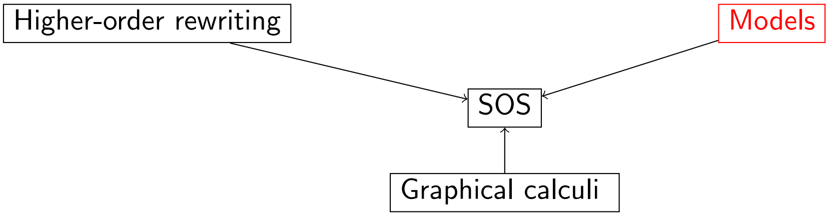
Chapter 3, published in LMCS (2013)

	Syntactic frameworks	HOR
Description of inductive generation	✓	✓
General notion of model	✗	✓

Generated transition relation = initial model.

Approaching SOS II: from models

- Missing from syntactic frameworks and graphical calculi: general notions of models.
- Idea:
  - start from existing notions of models (for instances of SOS);
  - try to generalise them to fragments of SOS.



Approaching SOS II

Game semantics

Interpret types as games and programs as innocent strategies.

Chapter 5 (with Eberhart, Pous, Seiller)

Recasting of innocence as a sheaf condition.

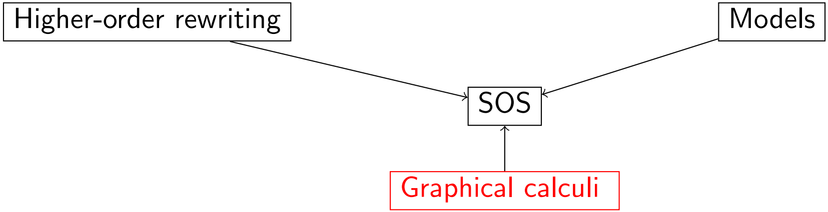
$\rightsquigarrow$ New, analogous models for two concurrent languages (CCS and π).

$\rightsquigarrow$ Abstract framework (playgrounds):

$$\begin{tikzcd} & \text{playground} \arrow[dl, squiggly] \arrow[dr, squiggly] & \\ \text{`SOS' transition relation} \arrow[rr, "\text{interpretation}"'] & & \text{innocent strategies.} \end{tikzcd}$$

- Covers the new models of CCS and π.
- Conjecture: also covers more standard models, e.g., of PCF.

Approaching SOS III: graphical calculi



Chapter 4 (with Garner): today's focus!

- Definition of 'graphical calculus'.
- Description of inductive generation process.
- General notion of model.
- Construction of initial model.
- Application to more standard mathematical structures:

Operads as the models of an adequate graphical calculus.

Outline

Mathematical motivation

Certain algebraic structures with

- obvious graphical intuition;
- tedious formal definition.

E.g., operads, properads, polycategories, PROPs, and variants.

Computer science motivation

Graphical calculi with

- obvious graphical intuition;
- tedious formal definition;
- involved or non-existent notion of model.

E.g., interaction nets, proof nets, bigraphs.

Contributions (with Garner)

- Make graphical intuition rigorous thanks to presheaf theory.
- $\rightsquigarrow$ Alternative definition of

  maths: the algebraic structure in question
  comp. sci.: a notion of model for the graphical calculus in question.
- View old definition as economical characterisation:

	old definition	new definition
statement	hard	easy
construction	easy	hard

Posing the problem categorically

$$\begin{tikzcd}
\text{presheaves} \arrow[r, rightsquigarrow] & \text{endofunctor } B \arrow[r, rightsquigarrow] & \text{monad } T \arrow[r, rightsquigarrow] & T\text{-algebras} \arrow[d, leftrightsquigarrow] \\
\text{pictures} \arrow[u, leftrightsquigarrow] & & & \text{algebraic structures}
\end{tikzcd}$$

Need to explain these terms, at least intuitively.

- Rightmost part: standard categorical approach to algebra.
- Just need to derive T from the pictures!

Outline

Categories

Definition

Objects, and morphisms between them.

Example

	Objects	Morphisms
Set	Sets	Functions
Mon	Monoids	Monoid homomorphisms
Grp	Groups	Group homomorphisms
...		

Functors

Definition

Functor = morphism of categories.

Example

$$
\begin{tikzcd}
\mathrm{Set} \arrow[r, bend left, "L"] & \mathrm{Mon} \arrow[l, bend left, "U"]
\end{tikzcd}
$$

- Action on objects:
  $L(X) = \sum_n X^n$
  $=$ sequences of elements of X,
  $=$ free monoid on X.
  Multiplication:
  $(x_1, \ldots, x_n), (x_{n+1}, \ldots, x_p) \mapsto (x_1, \ldots, x_p)$.
- Action on morphisms:
  $$\begin{array}{rcl} L(X \xrightarrow{f} Y) \colon L(X) & \to & L(Y) \\ (x_1, \ldots, x_n) & \mapsto & (f(x_1), \ldots, f(x_n)). \end{array}$$
- Other example:
  $U(M) = |M|$, carrier of M.

Monads

Definition

Monad = endofunctor + structure.

Example

$$\begin{tikzcd} \mathrm{Set} \arrow[loop above, "T"] \arrow[r, bend left, "L"] & \mathrm{Mon} \arrow[l, bend left, "U"] \end{tikzcd}$$

- Composite $T = U \circ L$.
- $T(X)$ = free monoid viewed as a set.
- T is a monad.

Crucial point I: algebraic structures = algebras for a monad

T-algebra

T-algebra = morphism $\begin{array}{c} T(X) \\ {\scriptstyle m}\big\downarrow \\ X \end{array}$ with easy conditions.

Example: previous T

- $T(X)$ = free monoid viewed as a set.
- So m maps sequences $(x_1, \ldots, x_n)$ to elements.
- Thought of as multiplication.

Example T-algebra:

$$\begin{array}{rcl} m\colon T(\mathbb{N}) & \to & \mathbb{N} \\ (n_1, \ldots, n_p) & \mapsto & \sum_i n_i . \end{array}$$

Morphisms of algebras

Morphisms of T-algebras

$$\begin{tikzcd} T(X) \arrow[r, "T(f)"] \arrow[d, "m"'] & T(Y) \arrow[d, "m'"] \\ X \arrow[r, "f"'] & Y \end{tikzcd}$$

- $f(m(x_1, \ldots, x_n)) = m'(f(x_1), \ldots, f(x_n))$.
- Morphism = structure-preserving map.

Proposition (in the monoids example)

T-algebras form a category T-Alg, equivalent to Mon.

Moral (standard, but very important!)

Algebraic structure (monoids) $\Leftarrow$ monad T.

T describes 'free' algebraic structures.

Other examples on sets

Algebraic structure	$T(X)$
Monoids	$\sum_n X^n$
Commutative monoids	$\sum_n X^n/\mathfrak{S}_n$
Rings, modules, algebras, ...	...
Complete semi-lattices	$\mathcal{P}(X)$

Non-example: fields, as there are no free fields over a set.

Outline

From pictures to presheaves

- Running example: (nonsymmetric, coloured) operads.
- Well-known case: T already known!
- Result specialises to: characterisation of T as a free shapely monad.

$$
\begin{tikzcd}
\text{family of presheaves} \arrow[r, squiggly] & \text{endofunctor } B \arrow[r, squiggly] & \text{monad } T \arrow[r, squiggly] & T\text{-algebras} \arrow[d, squiggly] \\
\text{pictures} \arrow[u, squiggly] & & & \text{algebraic structures}
\end{tikzcd}
$$

From pictures to presheaves

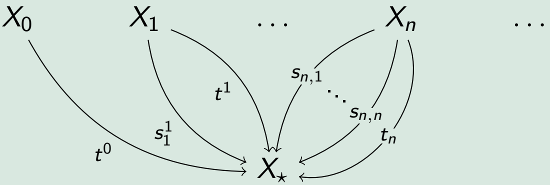
- Running example: (nonsymmetric, coloured) operads.
- Well-known case: T already known!
- Result specialises to: characterisation of T as a free shapely monad.

$$\begin{tikzcd}
\text{family of multigraphs} \arrow[r, rightsquigarrow] & \text{endofunctor } B \arrow[r, rightsquigarrow] & \text{monad } T \arrow[r, rightsquigarrow] & T\text{-algebras} \arrow[d, rightsquigarrow] \\
\text{pictures} \arrow[u, rightsquigarrow] & & & \text{algebraic structures}
\end{tikzcd}$$

Multigraphs

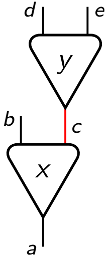
Multigraph X ≈ graph whose edges may have several sources.

Diagram



- $X_\star$: vertices;
- X_n: edges with n sources;
- $s_{n,i}(e)$: ith source of n-ary e;
- $t_n(e)$: target of e.

Example multigraph



- $X_\star = \{a, b, c, d, e\}$,
- $X_2 = \{x, y\}$,
- $X_n = \varnothing$ otherwise,
- $t_2(x) = x \cdot t = a$ (notation!),
- $x \cdot s_1 = b$, $x \cdot s_2 = c$, $y \cdot t = c$, $y \cdot s_1 = d$, $y \cdot s_2 = e$.

Category of multigraphs

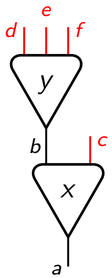
Morphism = map preserving target and sources.

Proposition

Multigraphs form a category MGph.

Intuitive definition

A *(nonsymmetric, coloured) operad* (in sets) $\mathcal{O}$ is a multigraph $\mathcal{O}$ with 'plugging', e.g., for all $x \in \mathcal{O}_2$ and $y \in \mathcal{O}_3$ with $y \cdot t = x \cdot s_1$, one may form



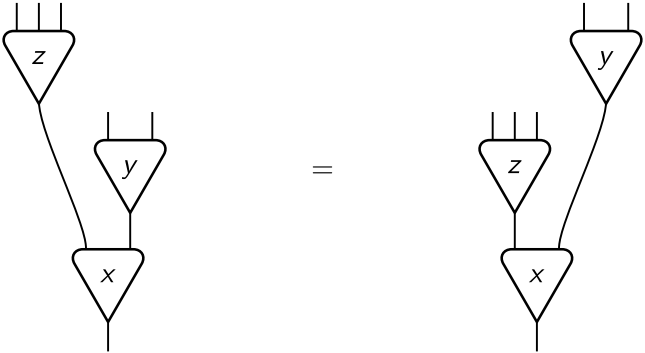
in $\mathcal{O}_4$.

Notation

Denoted by $x \circ_1^{2,3} y$.

Intuitive definition (cont'd)

Plugging should satisfy obvious graphical axioms, e.g.,



Dreadful glimpses of standard definition

Definition

A *(nonsymmetric, coloured) operad* (in sets) is

- a multigraph $\mathcal{O}$, together with
- for all m, n, i, $x \in \mathcal{O}_m$ and $y \in \mathcal{O}_n$ such that $x \cdot s_i = y \cdot t$, an element

$$x \circ_i^{m,n} y \in \mathcal{O}_{m+n-1};$$

- for all $a \in \mathcal{O}_\star$, an element $id_a \in \mathcal{O}_1$;
- satisfying axioms like

$$(x \circ_i^{m,n} y) \circ_j^{m+n-1,p} z = \begin{cases} (x \circ_j^{m,p} z) \circ_{i+p-1}^{m+p-1,n} y & (\text{if } j < i) \\ x \circ_i^{m,n+p-1} (y \circ_{j-i+1}^{n,p} z) & (\text{if } i \leq j < i+n) \end{cases}$$

for all $x \in \mathcal{O}_m$, $y \in \mathcal{O}_n$, $z \in \mathcal{O}_p$.

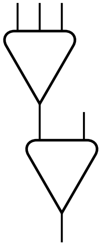
Outline

Endofunctors from multigraphs

$$\begin{tikzcd}
\text{family of multigraphs} \arrow[r, squiggly, red] & \text{endofunctor } B \arrow[r, squiggly] & \text{monad } T \arrow[r, squiggly] & T\text{-algebras} \arrow[d, squiggly] \\
\text{pictures} \arrow[u, squiggly] & & & \text{algebraic structures}
\end{tikzcd}$$

Crucial point II: arguments for composition = multigraph morphisms

- Recall the picture for composition in $\mathcal{O}$, on the right.
- View it as a multigraph, say X.



(Morphisms $X \to \mathcal{O}$) $\Leftrightarrow$ (choices of (x, y)):

- $x \in \mathcal{O}_2$ and $y \in \mathcal{O}_3$,
- such that $x \cdot s_1 = y \cdot t$.

= potential arguments for $\circ_1^{2,3}$ if it existed.

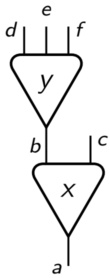
Arities

Definition (Basic arities)

- X is the arity of $\circ_1^{2,3}$.
- Obvious generalisation: $X_i^{m,n}$ is the arity of $\circ_i^{m,n}$.
- Similarly, arity of *id*: multigraph with just one vertex (wire).

Making sense of h_X-algebras

- Recall our example multigraph X on the right.
- Consider the functor $h_X \colon \mathrm{MGph} \to \mathrm{MGph}$ defined by:
  - $h_X(Y)_\star = Y_\star$,
  - $h_X(Y)_4 = \mathrm{MGph}(X, Y)$, the set of multigraph morphisms from X to Y,
  - $h_X(Y)_n = \varnothing$ for $n \neq 4$.
- So $h_X(Y)_4 = \{(x', y') \in Y_2 \times Y_3 \mid x' \cdot s_1 = y' \cdot t\}$.
- An algebra $h_X(Y) \to Y$ is determined by:
  - a multigraph Y,
  - plus a map $h_X(Y)_4 \to Y_4$, i.e.,
  - an interpretation of $\circ_1^{2,3}$!



Summary

Multigraph X $\rightsquigarrow$ functor which specifies an operation of arity X.

I.e., algebras have such an operation.

The monad from derived arities

$$\begin{tikzcd}
\text{family of}\\ \text{multigraphs} \arrow[r, squiggly] \arrow[rr, squiggly, bend left] & \text{endofunctor } B \arrow[r, squiggly] & \text{monad } T \arrow[r, squiggly] & T\text{-algebras} \arrow[d, squiggly] \\
\text{pictures} \arrow[u, squiggly] & & & \text{algebraic structures}
\end{tikzcd}$$

Graphical definition of operads

Need to define arities for all derived operations:

Definition

Let $\mathcal{T}_n$ denote the class of planar trees with n leaves.

Define $T \colon \mathrm{MGph} \to \mathrm{MGph}$ by:

- $T(Y)_\star = Y_\star$,
- $T(Y)_n = \sum_{X \in \mathcal{T}_n} \mathrm{MGph}(X, Y)$, the set of multigraph morphisms from some n-ary tree X to Y.

Lemma

The functor T is a monad on MGph.

Theorem

Operads are equivalent to T-algebras.

Outline

Generating monads

$$\begin{array}{ccccccc} \text{family of} & & & & & & \\ \text{multigraphs} & \rightsquigarrow & \text{endofunctor } B & \rightsquigarrow & \text{monad } T & \rightsquigarrow & T\text{-algebras} \\ \updownarrow & & & & & & \updownarrow \\ \text{pictures} & & & & & & \text{algebraic structures} \end{array}$$

- Goal: generate T automatically from basic arities.

  - Compositions $X_i^{n,m}$.
  - Identities I_a.

Signature for operads

Definition

Let $\mathcal{B}_n$ denote the set of basic arities with n leaves.

Intuition: filiform trees of depth 2.

Define $B \colon \mathsf{MGph} \to \mathsf{MGph}$ by:

- $B(Y)_\star = Y_\star$,
- $B(Y)_n = \sum_{X \in \mathcal{B}_n} \mathsf{MGph}(X, Y)$, the set of multigraph morphisms from some n-ary basic arity X to Y.

Question: how to generate T from B?

Naive attempt

Well-known correspondence

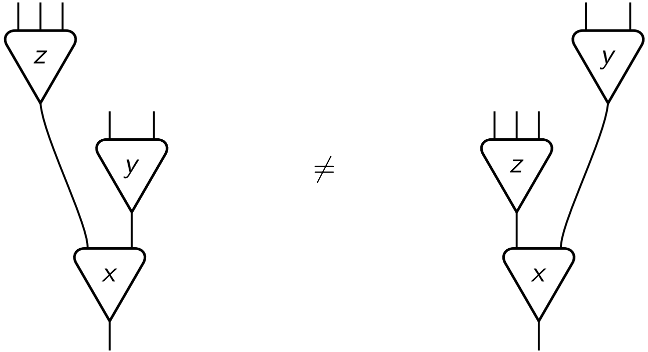
$$\begin{tikzcd} \text{Endofunctors on MGph} \arrow[r, bend left, "\mathcal{M}"] & \text{Monads on MGph.} \arrow[l, bend left, "\mathcal{U}"] \end{tikzcd}$$

Miss!

$\mathcal{M}(B) \ncong T$.

Reason

$\mathcal{M}(B)$-algebras do not satisfy any of the axioms!



Which monads do enforce them? Shapely ones!

Shapely monads

Subcategory

$$\mathsf{Framed}(\mathsf{MGph}) \subseteq \mathsf{Cell}(\mathsf{MGph}) \subseteq \mathsf{Analytic}(\mathsf{MGph}) \subseteq \mathsf{Endo}(\mathsf{MGph}).$$

- Stable under composition.
- Has a terminal object $\top$, automatically a monad.

Definition

Shapely = subfunctor of $\top$ in $\mathsf{Framed}(\mathsf{MGph})$.
Graphical calculus = shapely monad.

Intuition: at most one operation of each arity.

Generation result

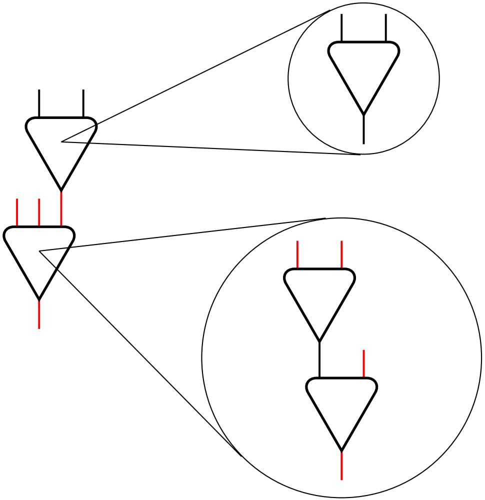
Theorem

$T = \bigcup_n (id \cup B)^{\cdot n}$ is the free shapely monad over B.

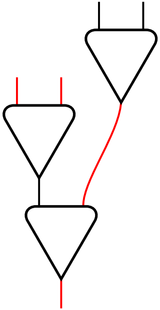
$B \cdot B$ denotes the image of $B \circ B$: $B \circ B \twoheadrightarrow B \cdot B \hookrightarrow \top$.

Illustration of $B \cdot B$

$B \circ B \quad \twoheadrightarrow \quad B \cdot B$



$\mapsto$



General result

- Consider any presheaf category with a subterminal object $\top$.

At most one morphism from any object to $\top$.

- Consider $\top$-analytic functors, i.e., analytic functors with a map to $\top$.
- Suppose they are stable under composition.
- Example: framed endofunctors.

Definition

Shapely functor = subfunctor of $\top$.

Theorem

The free shapely monad on a shapely endofunctor B is $\bigcup_n (id \cup B)^{\cdot n}$.

Applications

- Characterisation of the monads for polycategories, properads, PROPs, etc, as free shapely monads.
- Definition of free shapely monads for interation nets and fragments of proof nets.

Conclusion

- Sketched several approaches to mathematising programming language research.
- Rather diverse contributions.
- Still lots of work to do to reconcile theory and practice!

Thanks!

Shapely functors: intuition

- Restrict to functors with at most one operation per arity.
- There should be one 'full' such functor $\top$, with one operation for each possible arity.
- This functor $\top$ should be a monad.
- Selecting basic arities $\Leftrightarrow$ picking a subfunctor $B \subseteq \top$.
- Generating $T \approx \bigcup_n (id \cup B)^{\cdot n}$, the smallest submonad of $\top$ containing B.

Shapely functors: strategy

Find a subcategory $\mathcal{C}$ of Endo(MGph)

- stable under composition and
- having a terminal object $\top$.

I.e., such that $\forall C \in \mathcal{C}, \exists!$ morphism $C \to \top$.

Indeed:

- $\top$ automatically a monad *via* $\top \circ \top \to \top$;
- can then generate $\bigcup_n B^n$ amongst subfunctors of $\top$.

Towards shapely functors I: analytic functors

Subcategory Analytic(MGph) $\subseteq$ Endo(MGph) of functors s.t.

$$T(Y)_n = \sum_{x \in T(1)_n} \mathrm{MGph}(A(x), Y)/G(x)$$

where

- $A(x)$ is the arity of x,
- $G(x) \triangleleft \mathfrak{S}_{A(x)}$ is a subgroup of the automorphism group of $A(x)$.
- Generalisation of Joyal's analytic endofunctors on sets.

Miss again!

- Does have a terminal object.
- Not stable under composition.

Towards shapely functors II: cellular functors

Subcategory $\mathsf{Cell}(\mathsf{MGph}) \subseteq \mathsf{Analytic}(\mathsf{MGph}) \subseteq \mathsf{Endo}(\mathsf{MGph})$.

Miss again!

- Stable under composition.
- No terminal object!