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

1. Trajectory
2. Bibliography (summary of contributions)
3. Motivation
4. Preliminaries
5. Operads
6. Graphical operads
7. Shapely monads
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 $\pi$-calculus)

\[
\begin{align*}
P & \xrightarrow{\bar{a}\langle m \rangle} P' \\
Q & \xrightarrow{a(m)} Q'
\end{align*}
\]

\[
P | Q \xrightarrow{\tau} P' | Q'
\]

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

\[ \forall x. A(x) = \forall y. A(y) \]

- 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:**
  
  basic rules \(\rightsquigarrow\) 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\(^a\)!

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.
SOS is a wild territory.

Strategy: approach SOS from tamer settings.
Outline

① Trajectory

② Bibliography (summary of contributions)

③ Motivation

④ Preliminaries

⑤ Operads

⑥ Graphical operads

⑦ Shapely monads
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)

<table>
<thead>
<tr>
<th></th>
<th>Syntactic frameworks</th>
<th>HOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description of inductive generation</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>General notion of model</td>
<td>x</td>
<td>✓</td>
</tr>
</tbody>
</table>

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.
⇝ New, analogous models for two concurrent languages (CCS and $\pi$).
⇝ Abstract framework (playgrounds):

```
playground

'SOS' transition relation ➞ interpretation ➞ innocent strategies.
```

- Covers the new models of CCS and $\pi$.
- 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

1. Trajectory
2. Bibliography (summary of contributions)
3. Motivation
4. Preliminaries
5. Operads
6. Graphical operads
7. Shapely monads
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.
- 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:

<table>
<thead>
<tr>
<th></th>
<th>old definition</th>
<th>new definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>statement</td>
<td>hard</td>
<td>easy</td>
</tr>
<tr>
<td>construction</td>
<td>easy</td>
<td>hard</td>
</tr>
</tbody>
</table>
Posing the problem categorically

\[ \text{presheaves} \rightsquigarrow \text{endofunctor } B \rightsquigarrow \text{monad } T \rightsquigarrow \text{T-algebras} \]

Need to explain these terms, at least intuitively.

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

1. Trajectory
2. Bibliography (summary of contributions)
3. Motivation
4. Preliminaries
5. Operads
6. Graphical operads
7. Shapely monads
Categories

Definition

Objects, and morphisms between them.

Example

<table>
<thead>
<tr>
<th></th>
<th>Objects</th>
<th>Morphisms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set</td>
<td>Sets</td>
<td>Functions</td>
</tr>
<tr>
<td>Mon</td>
<td>Monoids</td>
<td>Monoid homomorphisms</td>
</tr>
<tr>
<td>Grp</td>
<td>Groups</td>
<td>Group homomorphisms</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
**Functors**

**Definition**

Functor = morphism of categories.

**Example**

- **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:**
  \[ L(X \xrightarrow{f} Y) : L(X) \to L(Y) \]
  \[ (x_1, \ldots, x_n) \mapsto (f(x_1), \ldots, f(x_n)). \]

- **Other example:**
  \[ U(M) = |M|, \text{ carrier of } M. \]
Monads

Definition

\textbf{Monad} = \textit{endofunctor} + \textit{structure}.

Example

\begin{itemize}
  \item Composite \( T = U \circ L \).
  \item \( T(X) = \text{free monoid viewed as a set} \).
  \item \( T \) is a monad.
\end{itemize}
Crucial point I: algebraic structures $\equiv$ algebras for a monad

$T$-algebra

$T$-algebra $\equiv$ morphism $m$ with easy conditions.

Example: previous $T$

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

Example $T$-algebra: $m: T(\mathbb{N}) \rightarrow \mathbb{N}$

$$(n_1, \ldots, n_p) \mapsto \sum_i n_i.$$
Morphisms of $T$-algebras

$T(X) \xrightarrow{T(f)} T(Y)$

$X \xrightarrow{m} \Downarrow m$

$Y \xrightarrow{f} \Downarrow m'$

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

Proposition (in the monoids example)

$T$-algebras form a category $T$-$\text{Alg}$, equivalent to $\text{Mon}$.

Moral (standard, but very important!)

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

$T$ describes ‘free’ algebraic structures.
Other examples on sets

<table>
<thead>
<tr>
<th>Algebraic structure</th>
<th>$T(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monoids</td>
<td>$\sum_n X^n$</td>
</tr>
<tr>
<td>Commutative monoids</td>
<td>$\sum_n X^n / S_n$</td>
</tr>
<tr>
<td>Rings, modules, algebras, . . .</td>
<td>. . .</td>
</tr>
<tr>
<td>Complete semi-lattices</td>
<td>$\mathcal{P}(X)$</td>
</tr>
</tbody>
</table>

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

① Trajectory
② Bibliography (summary of contributions)
③ Motivation
④ Preliminaries
⑤ Operads
⑥ Graphical operads
⑦ Shapely monads
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.

family of presheaves $\leadsto$ endofunctor $B \leadsto$ monad $T \leadsto\leadsto$ $T$-algebras

pictures $\leadsto$ algebraic structures
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.

family of
multigraphs $\leadsto$ endofunctor $B \leadsto$ monad $T \leadsto T$-algebras
pictures $\leadsto$ algebraic structures
Multigraph $X \approx$ graph whose edges may have several sources.

Diagram

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

**Multigraph** $X \approx$ graph whose edges may have several sources.
Example multigraph

- $X_\star = \{a, b, c, d, e\}$,
- $X_2 = \{x, y\}$,
- $X_n = \emptyset$ 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 \( \text{MGph} \).
Intuitive definition

A \emph{(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 \( \mathcal{O}_4 \).

\[
\begin{tikzpicture}
  \node at (0,0) {\( x \)};
  \node at (0,-1.5) {\( a \)};
  \node at (0,-3) {\( y \)};
  \node at (0,-4.5) {\( b \)};
  \node at (0,-6) {\( c \)};
  \node at (0,-7.5) {\( d \)};
  \node at (0,-9) {\( e \)};
  \node at (0,-10.5) {\( f \)};

  \draw (0,0) -- (0,-1.5);
  \draw (0,-1.5) -- (0,-3);
  \draw (0,-3) -- (0,-4.5);
  \draw (0,-4.5) -- (0,-6);
  \draw (0,-6) -- (0,-7.5);
  \draw (0,-7.5) -- (0,-9);
  \draw (0,-9) -- (0,-10.5);
\end{tikzpicture}
\]

Notation

Denoted by \( x \circ_{1,3}^{2,3} y \).
Plugging should satisfy obvious graphical axioms, e.g.,

\[ z 
\begin{array}{c}
\downarrow \\
\text{y} \\
\downarrow \\
x
\end{array} \] 
\[ = \] 
\[ y 
\begin{array}{c}
\downarrow \\
z \\
\downarrow \\
x
\end{array} \]
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 $a \in \mathcal{O}_*$, an element $id_a \in \mathcal{O}_1$;
- satisfying axioms like

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

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

1. Trajectory
2. Bibliography (summary of contributions)
3. Motivation
4. Preliminaries
5. Operads
6. Graphical operads
7. Shapely monads
Endofunctors from multigraphs

family of multigraphs $\leadsto$ endofunctor $B \leadsto$ monad $T \leadsto$ $T$-algebras

pictures $\leadsto$ algebraic structures
Crucial point II:
arguments for composition = multigraph morphisms

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

$\text{(Morphisms } X \to O) \iff (\text{choices of } (x, y)):$
  - $x \in O_2$ and $y \in O_3$,
  - such that $x \cdot s_1 = y \cdot t$.

$\Rightarrow$ potential arguments for $\circ_{1,3}^{2,3}$ if it existed.
Arities

Definition (Basic arities)

- $X$ is the arity of $\circ_{1}^{2,3}$.
- Obvious generalisation: $X^{m,n}_{i}$ 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 $h_X : \text{MGph} \to \text{MGph}$ defined by:
  - $h_X(Y)_* = Y_*$,
  - $h_X(Y)_4 = \text{MGph}(X, Y)$, the set of multigraph morphisms from $X$ to $Y$,
  - $h_X(Y)_n = \emptyset$ 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 $\circ_{1,3}^{2,3}$!

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

I.e., algebras have such an operation.
The monad from derived arities

family of multigraphs \(\sim\) endofunctor \(B\) \(\sim\) monad \(T\) \(\sim\) \(T\)-algebras

pictures \(\Downarrow\) algebraic structures
Graphical definition of operads

Need to define arities for all derived operations:

**Definition**

Let $T_n$ denote the class of planar trees with $n$ leaves.

Define $T : \text{MGph} \rightarrow \text{MGph}$ by:

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

**Lemma**

*The functor $T$ is a monad on $\text{MGph}$.***

**Theorem**

*Operads are equivalent to $T$-algebras.*
Outline

1. Trajectory
2. Bibliography (summary of contributions)
3. Motivation
4. Preliminaries
5. Operads
6. Graphical operads
7. Shapely monads
Generating monads

- Goal: generate $T$ automatically from basic arities.

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

Let $B_n$ denote the set of basic arities with $n$ leaves.

Intuition: filiform trees of depth 2.

Define $B : \text{MGph} \to \text{MGph}$ by:

- $B(Y)_* = Y_*$,
- $B(Y)_n = \sum_{X \in B_n} \text{MGph}(X, Y)$, 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

\[
\begin{array}{ccc}
\text{Endofunctors on } \mathbf{MGph} & \xrightleftharpoons{\mathcal{M}} & \text{Monads on } \mathbf{MGph}.
\end{array}
\]

Miss!

\[\mathcal{M}(B) \not\cong T.\]
\(\mathcal{M}(B)\)-algebras do not satisfy any of the axioms!

Which monads do enforce them? **Shapely** ones!
**Shapely monads**

Subcategory

\[
\text{Framed}(\text{MGph}) \subseteq \text{Cell}(\text{MGph}) \subseteq \text{Analytic}(\text{MGph}) \subseteq \text{Endo}(\text{MGph}).
\]

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

**Definition**

Shapely = subfunctor of \( \top \) in Framed(MGph).

Graphical calculus = shapely monad.

Intuition: at most one operation of each arity.
Generation result

**Theorem**

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

\( B \cdot B \) denotes the **image** of \( B \circ B : B \circ B \rightarrow B \cdot B \leftarrow T \).
Illustration of $B \cdot B$
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 $\bigcup_n (id \cup B)^n$. 
Applications

- Characterisation of the monads for polycategories, properads, PROPs, etc, as free shapely monads.
- Definition of free shapely monads for interaction 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 $\iff$ picking a subfunctor $B \subseteq \top$.
- Generating $T \approx \bigcup_n (id \cup B)^n$, the smallest submonad of $\top$ containing $B$. 
Shapely functors: strategy

Find a subcategory $\mathcal{C}$ of $\text{Endo}(\text{MGph})$
- stable under composition and
- having a terminal object $\top$.

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

Indeed:
- $\top$ automatically a monad via $\top \circ \top \rightarrow \top$;
- can then generate $\bigcup_n B^n$ amongst subfunctors of $\top$. 
Towards shapely functors I: analytic functors

Subcategory \( \text{Analytic}(\text{MGph}) \subseteq \text{Endo}(\text{MGph}) \) of functors s.t.

\[
T(Y)_n = \sum_{x \in T(1)_n} \text{MGph}(A(x), Y)/G(x)
\]

where

- \( A(x) \) is the arity of \( x \),
- \( G(x) \triangleleft \mathcal{G}_{A(x)} \) 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 $\text{Cell}(\text{MGph}) \subseteq \text{Analytic}(\text{MGph}) \subseteq \text{Endo}(\text{MGph})$.

Miss again!

- Stable under composition.
- No terminal object!