An Operadic Approach to Compositionality

David I. Spivak

dspivak@math.mit.edu
Mathematics Department
Massachusetts Institute of Technology

Presented on 2016/12/05
at the Compositionality Workshop,
Simons Institute for the Theory of Computing

Outline

Outline

Composition

Composition is assembling many things together to make one thing.

Composition

Composition is assembling many things together to make one thing.

- One says that Y is composed of pieces $X_1, \ldots, X_n$.
  - We should specify not only the pieces, but also their arrangement.
  - We could denote an arrangement $\varphi \colon X_1, \ldots, X_n \to Y$.
  - Arrangements can be nested inside each other.

Composition

Composition is assembling many things together to make one thing.

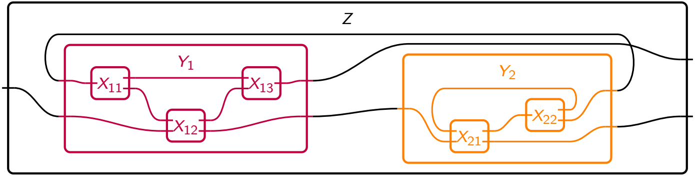
- One says that Y is composed of pieces $X_1, \ldots, X_n$.
  - We should specify not only the pieces, but also their arrangement.
  - We could denote an arrangement $\varphi\colon X_1, \ldots, X_n \to Y$.
  - Arrangements can be nested inside each other.
- This leads naturally to the notion of operad $\mathcal{O}$, which specifies:
  - the set of possible things $X, Y, \ldots$;
  - the set of arrangements φ, ψ by which one thing is composed of many;
  - how nesting works $\psi \circ (\varphi_1, \ldots, \varphi_n)$.

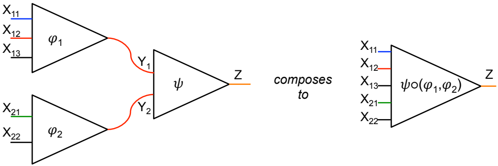
Composition

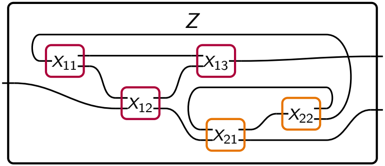
Composition is assembling many things together to make one thing.

- One says that Y is composed of pieces $X_1, \ldots, X_n$.
  - We should specify not only the pieces, but also their arrangement.
  - We could denote an arrangement $\varphi\colon X_1, \ldots, X_n \to Y$.
  - Arrangements can be nested inside each other.
- This leads naturally to the notion of operad $\mathcal{O}$, which specifies:
  - the set of possible things $X, Y, \ldots$;
  - the set of arrangements φ, ψ by which one thing is composed of many;
  - how nesting works $\psi \circ (\varphi_1, \ldots, \varphi_n)$.
- Note: by *operad*, I mean what is usually called "colored operad".

Picturing arrangements φ, ψ of things X, Y, Z







Syntax and semantics

- An operad $\mathcal{O}$ specifies a *theory of composition*.
  - $\mathcal{O}$ specifies various kinds of things and how they can be arranged.
  - These are the sorts and the operations in our theory $\mathcal{O}$ of composition.

Syntax and semantics

- An operad $\mathcal{O}$ specifies a *theory of composition*.
  - $\mathcal{O}$ specifies various kinds of things and how they can be arranged.
  - These are the sorts and the operations in our theory $\mathcal{O}$ of composition.
- Functorial semantics: a *model of* $\mathcal{O}$ is a functor $M\colon \mathcal{O} \to \mathbf{Set}$.
  - For every sort $X \in \mathcal{O}$, we have a set $M(X)$ of things of that sort.
  - For every arrangement $\varphi\colon X_1, \ldots, X_n \to Y$ in $\mathcal{O}$, we have a function
  $$M(\varphi)\colon M(X_1) \times \cdots \times M(X_n) \to M(Y).$$
    - Given a tuple $(x_1, \ldots, x_n) \in M(X_1) \times \cdots \times M(X_n)$,
    - and a rule φ for assembling them,
    - we obtain some new $\varphi(x_1, \ldots, x_n) \in M(Y)$.
- I think this is a reasonable formalism for the term *composition*.

What is compositionality?

In my lexicon, it is *attributes* and *analyses* that can be compositional.

- An attribute is like a projection onto a simpler space.
    - One attribute of an ODE is its set of steady states (subset of $\mathbb{R}^n$).
    - One attribute of a function is whether it's injective (Boolean).

What is compositionality?

In my lexicon, it is *attributes* and *analyses* that can be compositional.

- An attribute is like a projection onto a simpler space.
  - One attribute of an ODE is its set of steady states (subset of $\mathbb{R}^n$).
  - One attribute of a function is whether it's injective (Boolean).
- Formally, suppose given an operad $\mathcal{O}$ and a model $M\colon \mathcal{O} \to \mathbf{Set}$.
  - By a *compositional analysis*, I mean a system of attributes for $\mathcal{O}$.
  - It consists of an $\mathcal{O}$-model N and a natural transformation $A\colon M \to N$.
  - To each sort $X \in \mathcal{O}$, we have an attribute $A_X\colon M(X) \to N(X)$.

What is compositionality?

In my lexicon, it is *attributes* and *analyses* that can be compositional.

- An attribute is like a projection onto a simpler space.
  - One attribute of an ODE is its set of steady states (subset of $\mathbb{R}^n$).
  - One attribute of a function is whether it's injective (Boolean).
- Formally, suppose given an operad $\mathcal{O}$ and a model $M\colon \mathcal{O} \to \mathbf{Set}$.
  - By a *compositional analysis*, I mean a system of attributes for $\mathcal{O}$.
  - It consists of an $\mathcal{O}$-model N and a natural transformation $A\colon M \to N$.
  - To each sort $X \in \mathcal{O}$, we have an attribute $A_X\colon M(X) \to N(X)$.
  - *Compositionality*: for any arrangement φ and things $x_1, \ldots, x_n$, we have

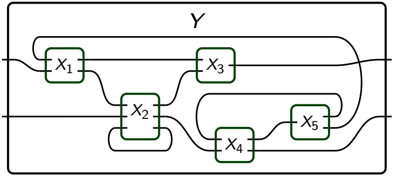
$$A_Y\big(M(\varphi)(x_1, \ldots, x_n)\big) = N(\varphi)\big(A_{X_1}(x_1) \ldots, A_{X_n}(x_n)\big)$$

- Compositionality of A means the following two things are the same:
  - composing pieces in the model, then projecting via attribute A
  - projecting each piece via attribute A, then composing their images.
- Summary: "analyzing commutes with assembling."

Example 1: steady states of dynamical systems

Taking steady states is a compositional analysis of dynamical systems.

- There is an operad $\mathcal{W}$ for composing dynamical systems.

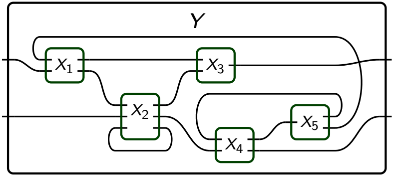


- We'll discuss a model $\mathrm{DS}\colon \mathcal{W} \to \textbf{Set}$ of "dynamical systems".

Example 1: steady states of dynamical systems

Taking steady states is a compositional analysis of dynamical systems.

- There is an operad $\mathcal{W}$ for composing dynamical systems.



- We'll discuss a model $\mathrm{DS}\colon \mathcal{W} \to \mathbf{Set}$ of "dynamical systems".
- There is a compositional analysis $A\colon \mathrm{DS} \to \mathrm{Mat}$.
  - Here, $\mathrm{Mat}\colon \mathcal{W} \to \mathbf{Set}$ is the model of matrices.
  - A assigns to each dynamical system its matrix of steady states.
  - Compute steady states of composite system by matrix arithmetic.

Example 2: hierarchical protein materials

There is an operad $\mathcal{M}$ for composing hierarchical protein materials.

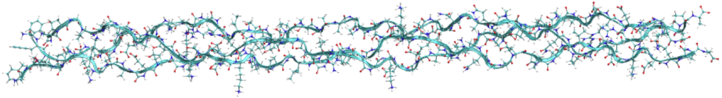
- A protein is an arrangement of simpler proteins.
  - There are atomic proteins, namely amino acids.
  - Protein materials include your skin: stretchable, breathable, waterproof.
  - (Computer MD versions of) proteins are a model, $\mathrm{Prot}\colon \mathcal{M} \to \mathbf{Set}$.[1]

[1] Giesa, T.; Jagadeesan, R.; Spivak, D.I.; Buehler, M.J. (2015) "Matriarch: a Python library for materials architecture." *ACS Biomaterials Science & Engineering*.

Example 2: hierarchical protein materials

There is an operad $\mathcal{M}$ for composing hierarchical protein materials.

- A protein is an arrangement of simpler proteins.
  - There are atomic proteins, namely amino acids.
  - Protein materials include your skin: stretchable, breathable, waterproof.
  - (Computer MD versions of) proteins are a model, $\mathrm{Prot}\colon \mathcal{M} \to \mathbf{Set}$.[1]
- A new protein can be assembled from a finite set of proteins
  - arranged in series or parallel, or
  - arranged in helices, double helices, any conceivable curve, etc.

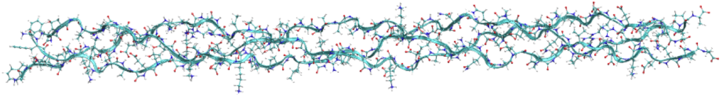


[1] Giesa, T.; Jagadeesan, R.; Spivak, D.I.; Buehler, M.J. (2015) "Matriarch: a Python library for materials architecture." *ACS Biomaterials Science & Engineering*.

Example 2: hierarchical protein materials

There is an operad $\mathcal{M}$ for composing hierarchical protein materials.

- A protein is an arrangement of simpler proteins.
  - There are atomic proteins, namely amino acids.
  - Protein materials include your skin: stretchable, breathable, waterproof.
  - (Computer MD versions of) proteins are a model, $\mathrm{Prot}\colon \mathcal{M} \to \mathbf{Set}$.[1]
- A new protein can be assembled from a finite set of proteins
  - arranged in series or parallel, or
  - arranged in helices, double helices, any conceivable curve, etc.



- A compositional analysis would be "incredibly" useful in mat. sci.:
  - Example: assign a value, e.g. strength or toughness, to each protein
  - with a formula for composing strengths according to any arrangement.
  - Even if not perfectly compositional, it would be highly valuable.

[1] Giesa, T.; Jagadeesan, R.; Spivak, D.I.; Buehler, M.J. (2015) "Matriarch: a Python library for materials architecture." *ACS Biomaterials Science & Engineering*.

Plan of the talk

Here's the plan for the rest of the talk.

- Discuss operads of string diagrams. In particular:
  - monoids and categories,
  - traced monoidal categories,
  - hypergraph categories.
- Exemplify compositional analyses: steady states of dynamic systems.
  - Define dynamical system (continuous, discrete).
  - Define their steady states and show how they "compose like matrices".
- Conclude with a few more words on compositionality (or lack thereof).

Outline

String diagrams

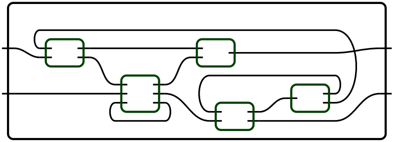
- String diagrams are attributed to Penrose, Joyal, Street, Verity, etc.
  - They give us a visual tool for solving algebra problems.
  - Peter Selinger's survey of graphical languages is fun and helpful.

String diagrams

- String diagrams are attributed to Penrose, Joyal, Street, Verity, etc.
  - They give us a visual tool for solving algebra problems.
  - Peter Selinger's survey of graphical languages is fun and helpful.
- How operads come into play:
  - We can organize the string diagrams for a doctrine as an operad $\mathcal{O}$.
  - The connection between string diagrams and their meaning is a functor $M\colon \mathcal{O} \to \mathbf{Set}$.

String diagrams

- String diagrams are attributed to Penrose, Joyal, Street, Verity, etc.
  - They give us a visual tool for solving algebra problems.
  - Peter Selinger's survey of graphical languages is fun and helpful.
- How operads come into play:
  - We can organize the string diagrams for a doctrine as an operad $\mathcal{O}$.
  - The connection between string diagrams and their meaning is a functor $M\colon \mathcal{O} \to \mathbf{Set}$.
- Below is an example string diagram for traced monoidal categories.

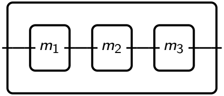


- We want to encode such diagrams as mathematical objects.

String diagrams for monoids

It is well-known that the terminal operad $\mathcal{T}$ is the theory of monoids.

- $\mathcal{T}$ has one object $*$, and one n-ary morphism for every n.
- A model of T is (as always) a functor $M\colon \mathcal{T} \to \mathbf{Set}$.
  - It assigns to the unique object $*$ a set $M := M(*)$.
  - It assigns an operation $M^n = M \times \cdots \times M \to M$ for every n.

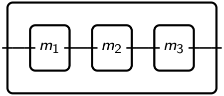


  - The composition formula in $\mathcal{T}$ ensures the associativity and unitality.

String diagrams for monoids

It is well-known that the terminal operad $\mathcal{T}$ is the theory of monoids.

- $\mathcal{T}$ has one object $*$, and one n-ary morphism for every n.
- A model of $\mathcal{T}$ is (as always) a functor $M\colon \mathcal{T} \to \mathbf{Set}$.
  - It assigns to the unique object $*$ a set $M := M(*)$.
  - It assigns an operation $M^n = M \times \cdots \times M \to M$ for every n.



  - The composition formula in $\mathcal{T}$ ensures the associativity and unitality.
- One can think of this as an "unbiased" perspective on monoids:
  - $\mathcal{T}$ gives us all the operations (n-ary multiplication) on equal footing,
  - in contrast to the usual approach: two generators, unit and mult.

String diagrams for categories: monoids + labels

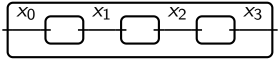
String diagrams focus on morphisms, not objects.

- This is the downside of using operads: they are parametric on objects.
  - So there is no operad for categories.
  - For any set of objects Λ, there is an operad for Λ-categories.
  - I'd like a nice way to deal with this, but haven't settled on anything.

String diagrams for categories: monoids + labels

String diagrams focus on morphisms, not objects.

- This is the downside of using operads: they are parametric on objects.
  - So there is no operad for categories.
  - For any set of objects Λ, there is an operad for Λ-categories.
  - I'd like a nice way to deal with this, but haven't settled on anything.
- Choose Λ. Define $\mathcal{O}_\Lambda$ as the following operad.
  - Its objects are pairs $X = (x_1, x_2) \in \Lambda^2$, drawn x_1  x_2.
  - Its n-ary morphisms $X_1, \ldots, X_n \to Y$ are tuples $(x_0, \ldots, x_n) \in \Lambda^{n+1}$,
  - ... such that $X_i = (x_{i-1}, x_i)$ and $Y = (x_0, x_n)$.



- A model $\mathcal{C} \colon \mathcal{O}_\Lambda \to \mathbf{Set}$ is an ("unbiased") category with objects Λ:
  - $\mathcal{C}$ assigns a set $\mathcal{C}(x_1, x_2)$ to each object $(x_1, x_2) \in \Lambda^2$
  - and assigns a "composition formula" to each compatible string of such.
- Next: there's a similar but more interesting story for traced categories.

Traced monoidal categories are models of $\mathcal{C}ob$

Modulo string labels, the operad for traced monoidal categories is $\mathcal{C}ob$:[2]

Theorem

There is an equivalence of categories: $\mathbf{Fun}(\mathcal{C}ob, \mathbf{Set}) \simeq \mathbf{TrCat}$.

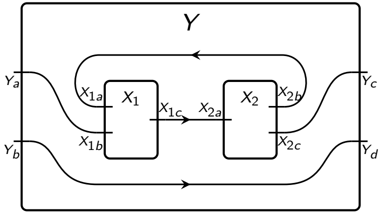
[2] Spivak, DI; Schultz, P; Rupel, D. (2017) "String diagrams for traced and compact categories are oriented 1-cobordisms". To appear in *Journal of Pure and Applied Algebra*.

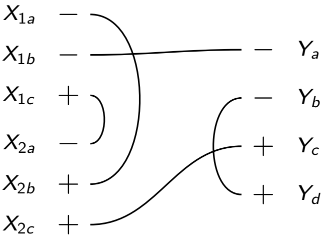
Traced monoidal categories are models of $\mathcal{C}ob$

Modulo string labels, the operad for traced monoidal categories is $\mathcal{C}ob$:[2]

Theorem

There is an equivalence of categories: $\mathbf{Fun}(\mathcal{C}ob, \mathbf{Set}) \simeq \mathbf{TrCat}$.





[2] Spivak, DI; Schultz, P; Rupel, D. (2017) "String diagrams for traced and compact categories are oriented 1-cobordisms". To appear in *Journal of Pure and Applied Algebra*.

Hypergraph categories

Another doctrine seems useful in applications: "hypergraph categories".

- The usual definition of *hypergraph category* is a bit "involved":
    - It is a symmetric monoidal category $\mathcal{C}$ in which
    - each object is equipped with the structure of a monoid and comonoid
    - that satisfy several additional axioms.

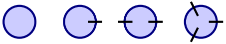
Hypergraph categories

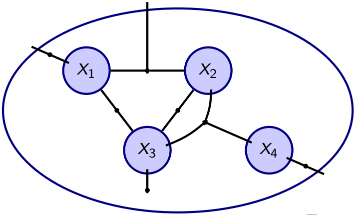
Another doctrine seems useful in applications: "hypergraph categories".

- The usual definition of *hypergraph category* is a bit "involved":
  - It is a symmetric monoidal category $\mathcal{C}$ in which
  - each object is equipped with the structure of a monoid and comonoid
  - that satisfy several additional axioms.

But the concept is quite easy from the perspective of string diagrams.

- As indicated by the name, string diagrams are *hypergraphs*.
- Pictorially, $\mathcal{H}$ is the operad with these objects and morphisms:

objects:  etc.

morphisms: 

Operad $\mathcal{H} = \mathcal{C}ospan$ and hypergraph categories

Let's give a more formal description of the operad $\mathcal{H}$.

- Different authors could mean slightly different things. Main issue:
- Can an edge in a hypergraph be incident to *zero* vertices?
  - If yes, then $\mathcal{H} = \mathcal{C}ospan$.
  - If no, then $\mathcal{H} = \mathcal{C}orel$. (This is the definition I used above; see Fong.)

Operad $\mathcal{H} = \mathcal{C}ospan$ and hypergraph categories

Let's give a more formal description of the operad $\mathcal{H}$.

- Different authors could mean slightly different things. Main issue:
- Can an edge in a hypergraph be incident to *zero* vertices?
  - If yes, then $\mathcal{H} = \mathcal{C}ospan$.
  - If no, then $\mathcal{H} = \mathcal{C}orel$. (This is the definition I used above; see Fong.)
- Either way, objects are finite sets (the set of ports) 
  - Morphisms are either cospans $X_1 \sqcup \cdots \sqcup X_n \to L \leftarrow Y$
  - or jointly surjective cospans $X_1 \sqcup \cdots \sqcup X_n \sqcup Y \twoheadrightarrow L$.

Operad $\mathcal{H} = \mathcal{C}ospan$ and hypergraph categories

Let's give a more formal description of the operad $\mathcal{H}$.

- Different authors could mean slightly different things. Main issue:
- Can an edge in a hypergraph be incident to *zero* vertices?
  - If yes, then $\mathcal{H} = \mathcal{C}ospan$.
  - If no, then $\mathcal{H} = \mathcal{C}orel$. (This is the definition I used above; see Fong.)
- Either way, objects are finite sets (the set of ports) 
  - Morphisms are either cospans $X_1 \sqcup \cdots \sqcup X_n \to L \leftarrow Y$
  - or jointly surjective cospans $X_1 \sqcup \cdots \sqcup X_n \sqcup Y \twoheadrightarrow L$.
- Examples of hypergraph categories:
  - Baez, Fong: Passive linear circuits. $\mathrm{PLC}\colon \mathcal{H} \to \mathbf{Set}$.
  - The category of relations: $\mathrm{Rel}\colon \mathcal{H} \to \mathbf{Set}$.
  - Similar: The category of arrays (i.e. tensors): $\mathrm{Arr}\colon \mathcal{H} \to \mathbf{Set}$.

Arrays as models $\mathrm{Arr}\colon \mathcal{H} \to \mathbf{Set}$

Setup: let k be a semi-ring. We'll consider arrays with entries in k.

- We need to add labels to the strings, namely finite sets.
  - For convenience, identify finite sets with their cardinalities in $\mathbb{N}$.
  - So an object $X \in \mathcal{H}$ is a finite set P and function $X\colon P \to \mathbb{N}$.
  - Define $\overline{X} := \prod_{p \in P} X(p)$.

Arrays as models $\mathrm{Arr}\colon \mathcal{H} \to \mathbf{Set}$

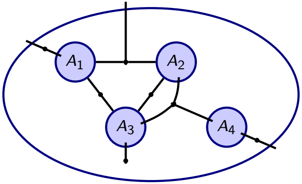
Setup: let k be a semi-ring. We'll consider arrays with entries in k.

- We need to add labels to the strings, namely finite sets.
  - For convenience, identify finite sets with their cardinalities in $\mathbb{N}$.
  - So an object $X \in \mathcal{H}$ is a finite set P and function $X\colon P \to \mathbb{N}$.
  - Define $\overline{X} := \prod_{p \in P} X(p)$.
- To each X we assign the set $\mathrm{Arr}(X) := \{A\colon \overline{X} \to k\}$.
  - So if $P = 1, 2$ and $X(1) = m$ and $X(2) = n$ then
  - $\overline{X} = m \times n$ and $\mathrm{Arr}(X)$ is the set of $m \times n$ matrices.

Arrays as models $\mathrm{Arr}\colon \mathcal{H} \to \mathbf{Set}$

Setup: let k be a semi-ring. We'll consider arrays with entries in k.

- We need to add labels to the strings, namely finite sets.
  - For convenience, identify finite sets with their cardinalities in $\mathbb{N}$.
  - So an object $X \in \mathcal{H}$ is a finite set P and function $X\colon P \to \mathbb{N}$.
  - Define $\overline{X} := \prod_{p \in P} X(p)$.
- To each X we assign the set $\mathrm{Arr}(X) := \{A\colon \overline{X} \to k\}$.
  - So if $P = 1, 2$ and $X(1) = m$ and $X(2) = n$ then
  - $\overline{X} = m \times n$ and $\mathrm{Arr}(X)$ is the set of $m \times n$ matrices.
- A cospan, as drawn below, specifies an array multiplication formula.



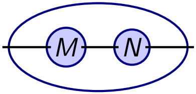
Example wiring diagrams for named operations

A single array multiplication formula returns famous matrix products.

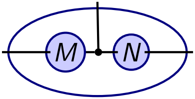
Example wiring diagrams for named operations

A single array multiplication formula returns famous matrix products.

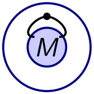
Multiplication: MN



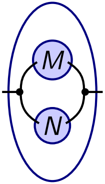
Khatri-Rao: $M \odot N$



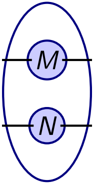
Trace: $\mathrm{Tr}(M)$



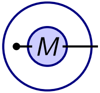
Hadamard: $M \circ N$



Kronecker: $M \otimes N$



Marginalize: $\sum_i M_{i,j}$



The real role of operads

For each type of string diagram, there is a corresponding operad $\mathcal{O}$.

- Operad functors allow you to change the string diagram type.
  - These generate free-forgetful adjunctions.
  - For example, the adjunction between categories and traced categories.

The real role of operads

For each type of string diagram, there is a corresponding operad $\mathcal{O}$.

- Operad functors allow you to change the string diagram type.
  - These generate free-forgetful adjunctions.
  - For example, the adjunction between categories and traced categories.
- Working directly with doctrines is often preferable to using operads.
  - People understand faster w/o operads. (Traced cat vs. $\mathcal{C}ob$-model).
  - You don't have to worry about object labels.

The real role of operads

For each type of string diagram, there is a corresponding operad $\mathcal{O}$.

- Operad functors allow you to change the string diagram type.
  - These generate free-forgetful adjunctions.
  - For example, the adjunction between categories and traced categories.
- Working directly with doctrines is often preferable to using operads.
  - People understand faster w/o operads. (Traced cat vs. $\mathcal{C}ob$-model).
  - You don't have to worry about object labels.
- On the other hand, there are sometimes reasons to prefer operads:
  - With operads, you aren't restricted to looking at named doctrines.
    - E.g. traced cats without identities are perfect for dynamical systems.
    - String diagrams may be more basic than their generators and relations.

The real role of operads

For each type of string diagram, there is a corresponding operad $\mathcal{O}$.

- Operad functors allow you to change the string diagram type.
  - These generate free-forgetful adjunctions.
  - For example, the adjunction between categories and traced categories.
- Working directly with doctrines is often preferable to using operads.
  - People understand faster w/o operads. (Traced cat vs. $\mathcal{C}ob$-model).
  - You don't have to worry about object labels.
- On the other hand, there are sometimes reasons to prefer operads:
  - With operads, you aren't restricted to looking at named doctrines.
    - E.g. traced cats without identities are perfect for dynamical systems.
    - String diagrams may be more basic than their generators and relations.
  - It gives an unbiased presentation, which can be nice to have.
  - (Subjective) Engineers seem to find the perspective compelling.
    - They like the idea of building one thing out of many.
    - And they seem to understand string diagrams faster than gens/rels.

Outline

Discrete and continuous dynamical systems

Dynamical systems are machines that take input, change state, and produce output.[3]

- They usually come in one of two flavors: discrete and continuous.
- All of our dynamical systems are open: they can interact with others.

[3] Spivak, DI (2015) "The steady states of coupled dynamical systems compose according to matrix arithmetic". Available online: http://arxiv.org/abs/1512.00802

Discrete and continuous dynamical systems

Dynamical systems are machines that take input, change state, and produce output.[3]

- They usually come in one of two flavors: discrete and continuous.
- All of our dynamical systems are open: they can interact with others.

Let $(X^{\mathrm{in}}, X^{\mathrm{out}})$ be a pair of sets (resp. manifolds). $X^{\mathrm{in}} \boxed{} X^{\mathrm{out}}$

[3] Spivak, DI (2015) "The steady states of coupled dynamical systems compose according to matrix arithmetic". Available online: `http://arxiv.org/abs/1512.00802`

Discrete and continuous dynamical systems

Dynamical systems are machines that take input, change state, and produce output.[3]

- They usually come in one of two flavors: discrete and continuous.
- All of our dynamical systems are open: they can interact with others.

Let $(X^{\text{in}}, X^{\text{out}})$ be a pair of sets (resp. manifolds). X^{in} X^{out}

Definition

A discrete (resp. continuous) dynamical system is a tuple $(S, f^{\text{upd}}, f^{\text{rdt}})$.

- S is a set (resp. manifold) of *states*;
- $f^{\text{upd}}\colon X^{\text{in}} \times S \to S$ (resp. $f^{\text{upd}}\colon X^{\text{in}} \times S \to TS$) is a function;
- $f^{\text{rdt}}\colon S \to X^{\text{out}}$ is function.

[3] Spivak, DI (2015) "The steady states of coupled dynamical systems compose according to matrix arithmetic". Available online: http://arxiv.org/abs/1512.00802

Composing dynamical systems

Dynamical systems can be composed, almost like in a traced category.

- But not quite. If you allow identities, you can't have feedback.
- Related to the *traced ideals* of Abramsky, Blute, Panangaden.

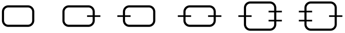
Composing dynamical systems

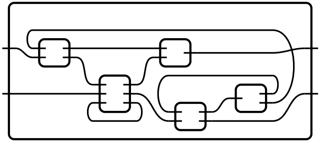
Dynamical systems can be composed, almost like in a traced category.

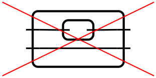
- But not quite. If you allow identities, you can't have feedback.
- Related to the *traced ideals* of Abramsky, Blute, Panangaden.
- Something like the following should be true:
  - If $\mathcal{C}$ is a Sym.Mon.Cat, there is an operad $\mathcal{W}_{\mathcal{C}}$ such that
  - the category of traced ideals in $\mathcal{C}$ is equivalent to $\mathbf{Fun}(\mathcal{W}_{\mathcal{C}}, \mathbf{Set})$.
  - $\mathcal{W}_{\mathcal{C}}$ is the left class of an orthogonal factorization system on $\mathcal{C}ob_{/\mathrm{Ob}(\mathcal{C})}$.
- Dynamical systems form a traced ideal in this sense.
  - Letting $\mathcal{W}$ be the operad $\mathcal{W}_{\mathbf{Set}}$ (resp. $\mathcal{W}_{\mathbf{Man}}$),
  - discrete (resp. continuous) dynamical systems is a model $\mathcal{W} \to \mathbf{Set}$.

Wiring diagrams for dynamical systems

There is an operad $\mathcal{W}$ whose objects and morphisms look like this:

objects:  etc.

morphisms: 

- $\mathcal{W} \subseteq \mathcal{C}ob$ is a suboperad, missing only "passing wires" 
- Think of $\mathcal{W}$ as modeling "traced categories without identities".

Steady states are a compositional analysis

Let $\mathcal{W}$ be the operad of wiring diagrams as on the previous slide.

- We said that dynamical systems form a model, $\mathrm{DS}\colon \mathcal{W} \to \mathbf{Set}$.

Steady states are a compositional analysis

Let $\mathcal{W}$ be the operad of wiring diagrams as on the previous slide.

- We said that dynamical systems form a model, $\mathrm{DS}\colon \mathcal{W} \to \mathbf{Set}$.
- There is a *compositional analysis* for dynamical systems, in matrices.
  - That is, we have a model $\mathrm{Mat}\colon \mathcal{W} \to \mathbf{Set}$ and
  - a natural transformation $\mathsf{Stst}\colon \mathrm{DS} \to \mathrm{Mat}$, given by "steady states".
  - Mat consists of matrices in $k = \{0, 1\}$. (Other k's also work.)

Steady states are a compositional analysis

Let $\mathcal{W}$ be the operad of wiring diagrams as on the previous slide.

- We said that dynamical systems form a model, $\mathrm{DS}\colon \mathcal{W} \to \textbf{Set}$.
- There is a *compositional analysis* for dynamical systems, in matrices.
  - That is, we have a model $\mathrm{Mat}\colon \mathcal{W} \to \textbf{Set}$ and
  - a natural transformation $\mathsf{Stst}\colon \mathrm{DS} \to \mathrm{Mat}$, given by "steady states".
  - Mat consists of matrices in $k = \{0, 1\}$. (Other k's also work.)
- How steady states of an $(X^{\mathrm{in}}, X^{\mathrm{out}})$-dynamical system form a matrix:
  - Let $F = (S, f^{\mathrm{upd}}, f^{\mathrm{rdt}})$ be the dynamical system and $M := \mathsf{Stst}(F)$.
  - M's entries are indexed by $\overline{X^{\mathrm{in}}} \times \overline{X^{\mathrm{out}}}$. Given $(x, y) \in \overline{X^{\mathrm{in}}} \times \overline{X^{\mathrm{out}}}$,
    - the *steady states* at (x, y) is $\{s \in S \mid f^{\mathrm{upd}}(x, s) = s \text{ and } f^{\mathrm{rdt}}(s) = y\}$.
    - $M(x, y) \in \{0, 1\}$ is 0 iff the set of steady states is empty.
  - For continuous systems, such a matrix is called the *bifurcation diagram*.

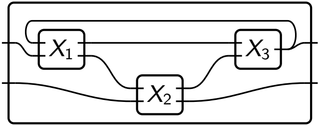
Steady states are a compositional analysis

Let $\mathcal{W}$ be the operad of wiring diagrams as on the previous slide.

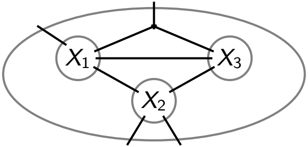
- We said that dynamical systems form a model, $\mathrm{DS}\colon \mathcal{W} \to \mathbf{Set}$.
- There is a *compositional analysis* for dynamical systems, in matrices.
  - That is, we have a model $\mathrm{Mat}\colon \mathcal{W} \to \mathbf{Set}$ and
  - a natural transformation $\mathsf{Stst}\colon \mathrm{DS} \to \mathrm{Mat}$, given by "steady states".
  - Mat consists of matrices in $k = \{0, 1\}$. (Other k's also work.)
- How steady states of an $(X^{\mathrm{in}}, X^{\mathrm{out}})$-dynamical system form a matrix:
  - Let $F = (S, f^{\mathrm{upd}}, f^{\mathrm{rdt}})$ be the dynamical system and $M := \mathsf{Stst}(F)$.
  - M's entries are indexed by $\overline{X^{\mathrm{in}}} \times \overline{X^{\mathrm{out}}}$. Given $(x, y) \in \overline{X^{\mathrm{in}}} \times \overline{X^{\mathrm{out}}}$,
    - the *steady states* at (x, y) is $\{s \in S \mid f^{\mathrm{upd}}(x, s) = s \text{ and } f^{\mathrm{rdt}}(s) = y\}$.
    - $M(x, y) \in \{0, 1\}$ is 0 iff the set of steady states is empty.
  - For continuous systems, such a matrix is called the *bifurcation diagram*.
- These steady state matrices compose according to the same $\mathcal{W}$.

Stepping back

- Dynamical systems compose according to an operad $\mathcal{W}$.
- Arrays compose according to an operad $\mathcal{H}$.
- There's an operad functor $U\colon \mathcal{W} \to \mathcal{H}$.

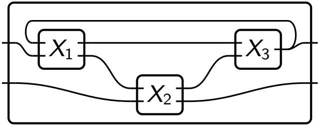


$\mapsto$

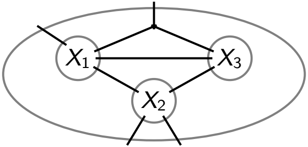


Stepping back

- Dynamical systems compose according to an operad $\mathcal{W}$.
- Arrays compose according to an operad $\mathcal{H}$.
- There's an operad functor $U\colon \mathcal{W} \to \mathcal{H}$.



$\mapsto$



Steady states map dynamical systems to arrays via U:

$$
\begin{tikzcd}
\mathcal{W} \arrow[rr, "U"] \arrow[dr, "\mathrm{DS}"'] \arrow[rr, phantom, shift right=5, "\overset{\mathsf{Stst}}{\Rightarrow}"] & & \mathcal{H} \arrow[dl, "\mathrm{Arr}"] \\
& \mathbf{Set} &
\end{tikzcd}
$$

Outline

Back to compositionality

Here's what we've been saying:

- An analysis is a way of viewing things of some type, $A\colon M \to N$.
- Suppose the things x can be arranged (φ's) to create new things.
  - The analysis A is compositional if it commutes with φ's.
  - That is, there is an isomorphism $N(\varphi)A(x) \cong A(M(\varphi)(x))$

Back to compositionality

Here's what we've been saying:

- An analysis is a way of viewing things of some type, $A\colon M \to N$.
- Suppose the things x can be arranged (φ's) to create new things.
  - The analysis A is compositional if it commutes with φ's.
  - That is, there is an isomorphism $N(\varphi)A(x) \cong A(M(\varphi)(x))$
- But what if A is merely lax, i.e. a map $N(\varphi)A(x) \to A(M(\varphi)(x))$.
  - We could call the difference a *generative effect*.
  - A is like an estimate, and the effect comes from "inexactness" of A.
  - Elie Adam (MIT) has a cohomological theory of generative effects.
  - E.g., if A is left exact, recover $A(M(\varphi))$ from cohomology of $N(\varphi)(A)$.

Summary

In this talk, we discussed the following:

- A general definition of composition and compositionality.
  - Composition is building one thing out of many.
  - An analysis is compositional when it commutes with composition.
  - How it'd be nice to have compositional analyses of materials.

Summary

In this talk, we discussed the following:

- A general definition of composition and compositionality.
  - Composition is building one thing out of many.
  - An analysis is compositional when it commutes with composition.
  - How it'd be nice to have compositional analyses of materials.
- Operads describe string diagrams of known categorical doctrines.
  - Monoids and categories: basically the terminal operad, plus labels.
  - Traced monoidal categories: the operad *Cob* of oriented cobordisms.
  - Hypergraph categories: the operad *Cospan* or *Corel*.

Summary

In this talk, we discussed the following:

- A general definition of composition and compositionality.
  - Composition is building one thing out of many.
  - An analysis is compositional when it commutes with composition.
  - How it'd be nice to have compositional analyses of materials.
- Operads describe string diagrams of known categorical doctrines.
  - Monoids and categories: basically the terminal operad, plus labels.
  - Traced monoidal categories: the operad *Cob* of oriented cobordisms.
  - Hypergraph categories: the operad *Cospan* or *Corel*.
- An example compositional analysis: steady states of dyn. systems.
  - Discrete and continuous dynamical systems have steady states.
  - These can be arranged into matrices and operated on as such.

Summary

In this talk, we discussed the following:

- A general definition of composition and compositionality.
  - Composition is building one thing out of many.
  - An analysis is compositional when it commutes with composition.
  - How it'd be nice to have compositional analyses of materials.
- Operads describe string diagrams of known categorical doctrines.
  - Monoids and categories: basically the terminal operad, plus labels.
  - Traced monoidal categories: the operad *Cob* of oriented cobordisms.
  - Hypergraph categories: the operad *Cospan* or *Corel*.
- An example compositional analysis: steady states of dyn. systems.
  - Discrete and continuous dynamical systems have steady states.
  - These can be arranged into matrices and operated on as such.

Thanks for inviting me to speak!

More on steady states and the pixel array method

Discrete vs. continuous dynamical systems

Computing steady states: how well does this work in practice?

- For discrete DS's this works well, exponentially reducing complexity.

Discrete vs. continuous dynamical systems

Computing steady states: how well does this work in practice?

- For discrete DS's this works well, exponentially reducing complexity.
- For continuous DS's, the "matrices" have continuum-many entries.
    - For example, suppose we have a DS on a box $\mathbb{R} - \boxed{} - \mathbb{R}$.
    - Then the steady state matrix is a function $\mathbb{R} \times \mathbb{R} \to \{0, 1\}$.
    - This relation is usually called the *bifurcation diagram* of the DS.

Discrete vs. continuous dynamical systems

Computing steady states: how well does this work in practice?

- For discrete DS's this works well, exponentially reducing complexity.
- For continuous DS's, the "matrices" have continuum-many entries.
  - For example, suppose we have a DS on a box $\mathbb{R} - \boxed{} - \mathbb{R}$.
  - Then the steady state matrix is a function $\mathbb{R} \times \mathbb{R} \to \{0, 1\}$.
  - This relation is usually called the *bifurcation diagram* of the DS.
- Given a wiring diagram, we must do matrix arithmetic on such beasts.
  - Calculating global steady states is tantamount to solving a system of relations.
  - This is hard in general. Generally people use Newton's method.
  - But the matrix arithmetic idea suggests another approach: pixelating.

Simple example

For simplicity, suppose we have equations $f(x, w) = 0$ and $g(w, y) = 0$.

- We plot them in some range $[-1.5, 1.5]$ using a certain pixel size.
- The plots are matrices M, N whose entries are on/off pixels.
- M and N are now *finite boolean matrices* corresponding to f and g.

Simple example

For simplicity, suppose we have equations $f(x, w) = 0$ and $g(w, y) = 0$.

- We plot them in some range $[-1.5, 1.5]$ using a certain pixel size.
- The plots are matrices M, N whose entries are on/off pixels.
- M and N are now *finite boolean matrices* corresponding to f and g.

Multiplying these two matrices MN yields...

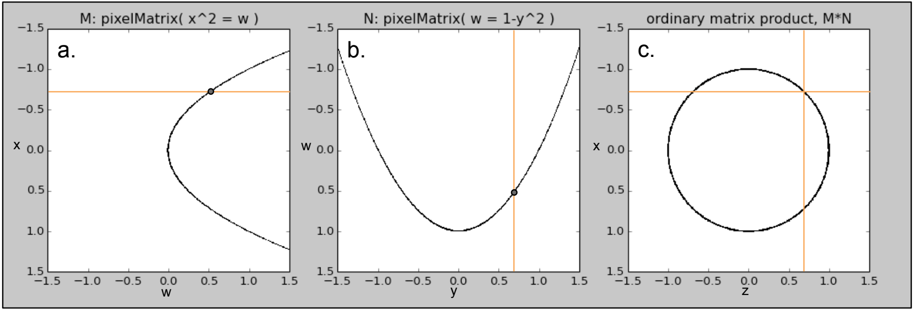
Simple example

For simplicity, suppose we have equations $f(x, w) = 0$ and $g(w, y) = 0$.

- We plot them in some range $[-1.5, 1.5]$ using a certain pixel size.
- The plots are matrices M, N whose entries are on/off pixels.
- M and N are now *finite boolean matrices* corresponding to f and g.

Multiplying these two matrices MN yields the simultaneous solution.

- For example, plot equations $x^2 = w$ and $w = 1 - y^2$, and multiply.



A more complex example

Here's a more complex example:

$$\cos\left(\ln(z^2 + 10^{-3}x)\right) - x + 10^{-5}z^{-1} = 0 \qquad \text{(Equation 1)}$$

$$\cosh(w + 10^{-3}y) + y + 10^{-4}w = 2 \qquad \text{(Equation 2)}$$

$$\tan(x + y)(x - 2)^{-1}(x + 3)^{-1}y^{-2} = 1 \qquad \text{(Equation 3)}$$

Q: For what values of w and z does a simultaneous solution exist?[4]

[4] Spivak, DI; Dobson, MRC; Kumari, S. (2016) "Pixel Arrays: A fast and elementary method for solving nonlinear systems". http://arxiv.org/pdf/1609.00061v1.pdf

A more complex example

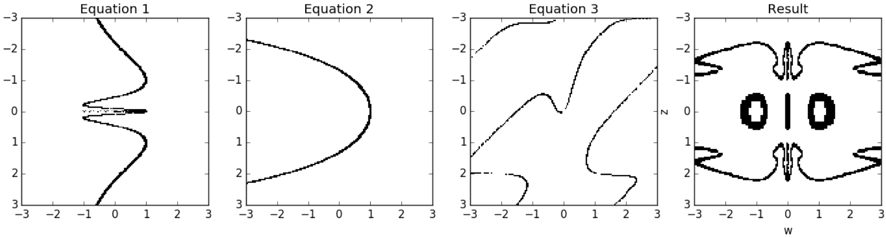
Here's a more complex example:

$$\cos\left(\ln(z^2+10^{-3}x)\right) - x + 10^{-5}z^{-1} = 0 \qquad \text{(Equation 1)}$$

$$\cosh(w+10^{-3}y) + y + 10^{-4}w = 2 \qquad \text{(Equation 2)}$$

$$\tan(x+y)(x-2)^{-1}(x+3)^{-1}y^{-2} = 1 \qquad \text{(Equation 3)}$$

Q: For what values of w and z does a simultaneous solution exist?[4]

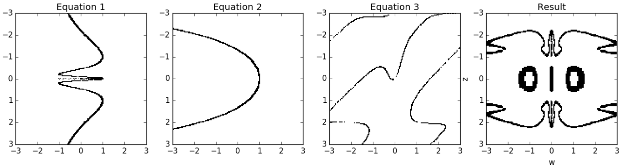


[4] Spivak, DI; Dobson, MRC; Kumari, S. (2016) "Pixel Arrays: A fast and elementary method for solving nonlinear systems". http://arxiv.org/pdf/1609.00061v1.pdf

Pixel array method

I call this the *pixel array method*.

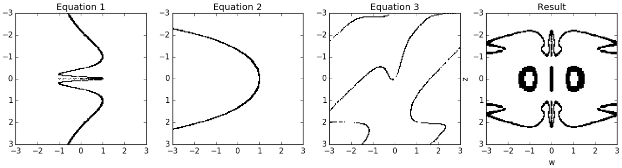
- We can solve systems by plotting and multiplying arrays.
- This gives good results, both in terms of speed and accuracy.
  - We also see the whole solution set, which could be quite useful.



Pixel array method

I call this the *pixel array method*.

- We can solve systems by plotting and multiplying arrays.
- This gives good results, both in terms of speed and accuracy.
  - We also see the whole solution set, which could be quite useful.



- Upshot: we can actually find steady states of systems of systems.
  - For discrete dynamical systems, it works on the nose.
  - For continuous ones, we use pixel arrays.
  - It's an estimate, but it converges and we can bound the error.