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

1. Introduction
2. Operads of string diagrams
3. Steady states are compositional
4. Conclusion
Outline

1 Introduction
   - What is compositionality?
   - Plan of the talk

2 Operads of string diagrams

3 Steady states are compositional

4 Conclusion
Composition

Composition is assembling many things together to make one thing.
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 $\varphi: X_1, \ldots, X_n \to Y$.
  - Arrangements can be nested inside each other.
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 $\varphi : X_1, \ldots, X_n \to Y$.
  - 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 $\varphi, \psi$ 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 $\varphi : X_1, \ldots, X_n \to Y$.
  - 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 $\varphi, \psi$ 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”.

David I. Spivak (MIT)  Operadic Approach to Compositionality  Presented on 2016/12/05
Picturing arrangements $\varphi, \psi$ of things $X, Y, Z$
Syntax and semantics

- An operad $O$ specifies a *theory of composition*.
  - $O$ specifies various *kinds of things* and how they can be *arranged*.
  - These are the *sorts* and the *operations* in our theory $O$ 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 $M : \mathcal{O} \rightarrow \textbf{Set}$.
  - For every sort $X \in \mathcal{O}$, we have a set $M(X)$ of things of that sort.
  - For every arrangement $\varphi : X_1, \ldots, X_n \rightarrow Y$ in $\mathcal{O}$, we have a function $M(\varphi) : M(X_1) \times \cdots \times M(X_n) \rightarrow M(Y)$.
    - Given a tuple $(x_1, \ldots, x_n) \in M(X_1) \times \cdots \times M(X_n)$,
    - and a rule $\varphi$ 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 $M: \mathcal{O} \to \text{Set}$.
  - 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 $A: M \to N$.
  - To each sort $X \in \mathcal{O}$, we have an attribute $A_X: M(X) \to N(X)$. 

Compositionality means the following two things are the same:

1. Composing pieces in the model, then projecting via attribute $A$
2. Projecting each piece via attribute $A$, then composing their images.

Summary: "analyzing commutes with assembling."
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 \( M: \mathcal{O} \to \text{Set} \).
  - 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 \( A: M \to N \).
  - To each sort \( X \in \mathcal{O} \), we have an attribute \( A_X: M(X) \to N(X) \).

Compositionality: for any arrangement \( \varphi \) and things \( x_1, \ldots, x_n \), we have

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

- 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 $\text{DS}: \mathcal{W} \to \text{Set}$ 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 $\text{DS}: \mathcal{W} \to \text{Set}$ of “dynamical systems”.
- There is a compositional analysis $A: \text{DS} \to \text{Mat}$.
  - Here, $\text{Mat}: \mathcal{W} \to \text{Set}$ 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, $\text{Prot}: \mathcal{M} \rightarrow \textbf{Set}$.\(^1\)

---

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, $\text{Prot}: \mathcal{M} \rightarrow \text{Set}$.\(^1\)

- 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.

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, $\text{Prot}: \mathcal{M} \to \text{Set}$.\(^1\)
- 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.

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

1 Introduction

2 Operads of string diagrams
   - String diagrams
   - Monoids and categories
   - Traced categories and cobordisms
   - Hypergraph categories
   - The real role of operads

3 Steady states are compositional

4 Conclusion
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 $M : \mathcal{O} \to \text{Set}$. 
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 $M : \mathcal{O} \to \text{Set}$.

- 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 $\ast$, and one $n$-ary morphism for every $n$.
- A model of $\mathcal{T}$ is (as always) a functor $M : \mathcal{T} \to \text{Set}$.
  - It assigns to the unique object $\ast$ a set $M := M(\ast)$.
  - 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 $T$ is the theory of monoids.

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

The composition formula in $T$ ensures the associativity and unitality.

One can think of this as an “unbiased” perspective on monoids:

- $T$ 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.

Choose Λ. Define $O_Λ$ as the following operad.

Its objects are pairs $(x_1, x_2) \in Λ^2$, drawn $x_1 \ x_2$.

Its $n$-ary morphisms $X_1, \ldots, X_n \to Y$ are tuples $(x_0, \ldots, x_n) \in Λ^{n+1}$, such that $X_i = (x_{i-1}, x_i)$ and $Y = (x_0, x_n)$.

A model $C: O_Λ \to \text{Set}$ is an (“unbiased”) category with objects Λ: $C$ assigns a set $C((x_1, x_2))$ to each object $(x_1, x_2) \in Λ^2$ and assigns a “composition formula” to each compatible string of such.

Next: there’s a similar but more interesting story for traced categories.
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 $\Lambda$, there is an operad for $\Lambda$-categories.
  - I’d like a nice way to deal with this, but haven’t settled on anything.

- Choose $\Lambda$. Define $O_\Lambda$ as the following operad.
  - Its objects are pairs $X = (x_1, x_2) \in \Lambda^2$, drawn $\xymatrix@!=2pc{ x_1 & x_2 }$.
  - 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 $C: O_\Lambda \to \text{Set}$ is an ("unbiased") category with objects $\Lambda$:
  - $C$ assigns a set $C(x_1, x_2)$ 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 \( \text{Cob} \)

Modulo string labels, the operad for traced monoidal categories is \( \text{Cob} \):\(^2\)

**Theorem**

*There is an equivalence of categories: \( \text{Fun}(\text{Cob}, \text{Set}) \cong \text{TrCat} \).*

**Traced monoidal categories are models of \( \text{Cob} \)**

Modulo string labels, the operad for traced monoidal categories is \( \text{Cob} \):^2

**Theorem**

*There is an equivalence of categories: \( \text{Fun}(\text{Cob}, \text{Set}) \cong \text{TrCat} \).*

---

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: $\circ$ $\circ$ $\circ$ $\circ$ etc.

  morphisms:
Operad $\mathcal{H} = \textit{Cospan}$ 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 $\mathcal{H} = \textit{Cospan}$.
  - If no, then $\mathcal{H} = \textit{Corel}$. (This is the definition I used above; see Fong.)
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 $\mathcal{H} = \text{Cospans}$.
    - If no, then $\mathcal{H} = \text{Corels}$. (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 $X_1 \sqcup \cdots \sqcup X_n \sqcup Y \to L$. 

Examples of hypergraph categories:

- Baez, Fong: Passive linear circuits. PLC: $\mathcal{H} \to \text{Set}$.
- The category of relations: $\text{Rel}: \mathcal{H} \to \text{Set}$.
- Similar: The category of arrays (i.e. tensors): $\text{Arr}: \mathcal{H} \to \text{Set}$.
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 $\mathcal{H} = \text{Cospa}$n.
    - If no, then $\mathcal{H} = \text{Corel}$. (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. PLC: $\mathcal{H} \to \text{Set}$.
  - The category of relations: $\text{Rel}: \mathcal{H} \to \text{Set}$.
  - Similar: The category of arrays (i.e. tensors): $\text{Arr}: \mathcal{H} \to \text{Set}$. 
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 $X : P \to \mathbb{N}$.
  - Define $\overline{X} := \prod_{p \in P} X(p)$. 

Arrays as models $\text{Arr} : \mathcal{H} \to \text{Set}$
Arrays as models $\text{Arr}: \mathcal{H} \to \text{Set}$

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 $X: P \to \mathbb{N}$.
- Define $\overline{X} := \prod_{p \in P} X(p)$.

- To each $X$ we assign the set $\text{Arr}(X) := \{A: \overline{X} \to k\}$.
  - So if $P = 1, 2$ and $X(1) = m$ and $X(2) = n$ then
  - $\overline{X} = m \times n$ and $\text{Arr}(X)$ is the set of $m \times n$ matrices.
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 $X : P \rightarrow \mathbb{N}$.
  - Define $\overline{X} := \prod_{p \in P} X(p)$.

- To each $X$ we assign the set $\text{Arr}(X) := \{A: \overline{X} \rightarrow k\}$.
  - So if $P = 1, 2$ and $X(1) = m$ and $X(2) = n$ then
  - $\overline{X} = m \times n$ and $\text{Arr}(X)$ 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.
A single array multiplication formula returns famous matrix products.

**Multiplication:** $MN$

**Khatri-Rao:** $M \odot N$

**Trace:** $\text{Tr}(M)$

**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.
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 $O$.

- 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. $Cob$-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 $O$.

- 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. $Cob$-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.
Steady states are compositional

Outline

1 Introduction

2 Operads of string diagrams

3 Steady states are compositional
   - Dynamical systems
   - Steady states

4 Conclusion
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.

Steady states are compositional

Dynamical systems

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). \(X^{\text{in}} \bigcirclearrowright X^{\text{out}}\)

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). \(X^{\text{in}} \sqcup X^{\text{out}}\)

**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;
- \(f^{\text{upd}} : X^{\text{in}} \times S \to S\) (resp. \(f^{\text{upd}} : X^{\text{in}} \times S \to TS\)) is a function;
- \(f^{\text{rdt}} : S \to X^{\text{out}}\) is function.

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 $\mathcal{W}_C$ such that the category of traced ideals in $\mathcal{C}$ is equivalent to $\text{Fun}(\mathcal{W}_C, \text{Set})$.
  - $\mathcal{W}_C$ is the left class of an orthogonal factorization system on $\text{Cob}_{/\text{Ob}(C)}$.
- Dynamical systems form a traced ideal in this sense.
  - Letting $\mathcal{W}$ be the operad $\mathcal{W}_{\text{Set}}$ (resp. $\mathcal{W}_{\text{Man}}$),
  - discrete (resp. continuous) dynamical systems is a model $\mathcal{W} \rightarrow \text{Set}$. 
There is an operad $\mathcal{W}$ whose objects and morphisms look like this:

- **Objects:**
  - $\square$, $\bigcirc$, $\bigotimes$, $\bigodot$, etc.

- **Morphisms:**
  - Diagram of wiring connections

- $\mathcal{W} \subseteq Cob$ 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, \( \mathcal{DS}: \mathcal{W} \to \text{Set} \).

\[ M(x, y) \in \{0, 1\} \] 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, $DS: \mathcal{W} \to \textbf{Set}$.
- There is a *compositional analysis* for dynamical systems, in matrices.
  - That is, we have a model $\text{Mat}: \mathcal{W} \to \textbf{Set}$ and
  - a natural transformation $\text{Stst}: DS \to \text{Mat}$, given by “steady states”.
  - $\text{Mat}$ 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, $\text{DS}: \mathcal{W} \to \text{Set}$.
- There is a *compositional analysis* for dynamical systems, in matrices.
  - That is, we have a model $\text{Mat}: \mathcal{W} \to \text{Set}$ and
  - a natural transformation $\text{Stst}: \text{DS} \to \text{Mat}$, given by “steady states”.
  - $\text{Mat}$ consists of matrices in $k = \{0, 1\}$. (Other $k$'s also work.)
- How steady states of an $(X^{\text{in}}, X^{\text{out}})$-dynamical system form a matrix:
  - Let $F = (S, f^{\text{upd}}, f^{\text{rdt}})$ be the dynamical system and $M := \text{Stst}(F)$.
  - $M$'s entries are indexed by $X^{\text{in}} \times X^{\text{out}}$. Given $(x, y) \in X^{\text{in}} \times X^{\text{out}}$,
    - the *steady states* at $(x, y)$ is $\{s \in S \mid f^{\text{upd}}(x, s) = s \text{ and } f^{\text{rdt}}(s) = y\}$.
    - $M(x, y) \in \{0, 1\}$ 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, $DS: \mathcal{W} \to \text{Set}$.
- There is a *compositional analysis* for dynamical systems, in matrices.
  - That is, we have a model $\text{Mat}: \mathcal{W} \to \text{Set}$ and
  - a natural transformation $\text{Stst}: DS \to \text{Mat}$, given by “steady states”.
  - $\text{Mat}$ consists of matrices in $k = \{0, 1\}$. (Other $k$’s also work.)
- How steady states of an $(X^{in}, X^{out})$-dynamical system form a matrix:
  - Let $F = (S, f^{upd}, f^{rdt})$ be the dynamical system and $M := \text{Stst}(F)$.
  - $M$’s entries are indexed by $\overline{X^{in}} \times \overline{X^{out}}$. Given $(x, y) \in \overline{X^{in}} \times \overline{X^{out}}$,
    - the steady states at $(x, y)$ is $\{s \in S \mid f^{upd}(x, s) = s$ and $f^{rdt}(s) = y\}$.
    - $M(x, y) \in \{0, 1\}$ 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 $\mathcal{W}$.
Steady states are compositional

Steady states

Stepping back

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

![Diagram showing compositional structure of dynamical systems and arrays with operad functors](image)
Dynamical systems compose according to an operad $\mathcal{W}$.  
Arrays compose according to an operad $\mathcal{H}$.  
There’s an operad functor $U: \mathcal{W} \to \mathcal{H}$.

Steady states map dynamical systems to arrays via $U$:  

$$
\mathcal{W} \xrightarrow{U} \mathcal{H} \\
\text{DS} \xrightarrow{\text{Stst}} \mathcal{H} \\
\text{Arr} \xrightarrow{\text{Set}} \text{Set}
$$
Outline

1. Introduction
2. Operads of string diagrams
3. Steady states are compositional
4. Conclusion
   - Compositionality vs. generative effects
   - Summary
Back to compositionality

Here’s what we’ve been saying:

- An analysis is a way of viewing things of some type, $A: M \to N$.
- Suppose the things $x$ can be arranged ($\varphi$’s) to create new things.
  - The analysis $A$ is compositional if it commutes with $\varphi$’s.
  - That is, there is an isomorphism $N(\varphi)A(x) \cong A(M(\varphi)(x))$
Here’s what we’ve been saying:

- An analysis is a way of viewing things of some type, \( A: M \to N \).
- Suppose the things \( x \) can be arranged (\( \phi \)’s) to create new things.
  - The analysis \( A \) is compositional if it commutes with \( \phi \)’s.
  - That is, there is an isomorphism \( N(\phi)A(x) \cong A(M(\phi)(x)) \)
- But what if \( A \) is merely lax, i.e. a map \( N(\phi)A(x) \to A(M(\phi)(x)) \).
  - 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 \( A(M(\phi)) \) from cohomology of \( N(\phi)(A) \).
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.
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 $\text{Cob}$ of oriented cobordisms.
  - Hypergraph categories: the operad $\text{Cospan}$ or $\text{Corel}$.

Thanks for inviting me to speak!

David I. Spivak (MIT)
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 \( \text{Cob} \) of oriented cobordisms.
  - Hypergraph categories: the operad \( \text{Cospa} \text{n} \) or \( \text{Cor} \text{el} \).

- 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.
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 \( \text{Cob} \) of oriented cobordisms.
  - Hypergraph categories: the operad \( \text{Cosp} \) or \( \text{Corel} \).

- 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.

For example, suppose we have a DS on a box $\mathbb{R}^2$. Then the steady state matrix is a function $\mathbb{R}^2 \rightarrow \{0, 1\}$. 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.
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 $\mathbb{R} \to \mathbb{R}$.
  - 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.
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 $\mathbb{R} \to \mathbb{R}$.
  - 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:

\[
\begin{align*}
\cos \left( \ln(z^2 + 10^{-3}x) \right) - x + 10^{-5}z^{-1} &= 0 \quad \text{(Equation 1)} \\
\cosh(w + 10^{-3}y) + y + 10^{-4}w &= 2 \quad \text{(Equation 2)} \\
\tan(x + y)(x - 2)^{-1}(x + 3)^{-1}y^{-2} &= 1 \quad \text{(Equation 3)}
\end{align*}
\]

Q: For what values of $w$ and $z$ does a simultaneous solution exist?\(^4\)

---

A more complex example

Here’s a more complex example:

\[
\cos \left( \ln(z^2 + 10^{-3}x) \right) - x + 10^{-5}z^{-1} = 0 \quad \text{(Equation 1)}
\]
\[
cosh(w + 10^{-3}y) + y + 10^{-4}w = 2 \quad \text{(Equation 2)}
\]
\[
tan(x + y)(x - 2)^{-1}(x + 3)^{-1}y^{-2} = 1 \quad \text{(Equation 3)}
\]

Q: For what values of \( w \) and \( z \) does a simultaneous solution exist?\(^4\)

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.
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.