

Using category theory to organize and operate on information

David I. Spivak

Mathematics Department
Massachusetts Institute of Technology

McGill Applied Mathematics Seminar
October 2, 2017

Information integration

- Information:
 - It's constantly being generated;
 - It arises from multiple sources and perspectives;
 - It must be integrated to solve larger problems.

Information integration

- Information:
 - It's constantly being generated;
 - It arises from multiple sources and perspectives;
 - It must be integrated to solve larger problems.
- Information integration:
 - Putting things together.
 - Making connections, drawing analogies.
 - Finding common structures.

Information integration

- Information:
 - It's constantly being generated;
 - It arises from multiple sources and perspectives;
 - It must be integrated to solve larger problems.
- Information integration:
 - Putting things together.
 - Making connections, drawing analogies.
 - Finding common structures.
- Composition, similar to integration:
 - Putting components together.
 - Interconnecting complex systems.
 - Finding common structures across scales.

How category theory fits in

Integration and composition: we do this all the time.

- Driving in the car.
- Listening to a talk.
- Or in your work.
 - Stats and linear algebra.
 - Analysis and algorithms.

How category theory fits in

Integration and composition: we do this all the time.

- Driving in the car.
- Listening to a talk.
- Or in your work.
 - Stats and linear algebra.
 - Analysis and algorithms.
- CT is a discipline that focuses on *integration* as its *subject*.

How category theory fits in

Integration and composition: we do this all the time.

- Driving in the car.
- Listening to a talk.
- Or in your work.
 - Stats and linear algebra.
 - Analysis and algorithms.
- CT is a discipline that focuses on *integration* as its *subject*.
 - Translates between disciplines.
 - Classifies relationships between objects.
 - Manages what is preserved vs. lost in translation.
 - Focuses on composition: putting things together.

Why apply category theory?

Today we have more complex problems:

- internet of things,
- cyber-physical systems,
- globalization.
- Disparate, large-scale systems coming together.

Why apply category theory?

Today we have more complex problems:

- internet of things,
- cyber-physical systems,
- globalization.
- Disparate, large-scale systems coming together.

We need a theory dedicated to integration.

Why apply category theory?

Today we have more complex problems:

- internet of things,
- cyber-physical systems,
- globalization.
- Disparate, large-scale systems coming together.

We need a theory dedicated to integration.

Category theory can help organize this complexity.

- CT's organizational power is well-known in pure math.
- I've been pushing it into other fields.
- Engineering, CS, materials sci., manufacturing, oceanography, cog-sci.

Why apply category theory?

Today we have more complex problems:

- internet of things,
- cyber-physical systems,
- globalization.
- Disparate, large-scale systems coming together.

We need a theory dedicated to integration.

Category theory can help organize this complexity.

- CT's organizational power is well-known in pure math.
- I've been pushing it into other fields.
- Engineering, CS, materials sci., manufacturing, oceanography, cog-sci.
- NASA example: need accurate, timely decision amid great complexity.

Plan of the talk

I'll discuss CT as mathematics for organizing information.

- Main thrust: “composition”, putting things together.
- Formal notion: *operads*, a general framework for composition.
 - I'll give examples and sketch a definition.
 - Operads: the mathematical framework for the talk.

Plan of the talk

I'll discuss CT as mathematics for organizing information.

- Main thrust: “composition”, putting things together.
- Formal notion: *operads*, a general framework for composition.
 - I'll give examples and sketch a definition.
 - Operads: the mathematical framework for the talk.
- Application of CT to studying systems:
 - Systems of equations: a new numerical method.
 - Information systems: combine data from disparate sources.
 - Dynamical systems: compositional analyses.

What I mean by composition

Composition is **arranging** many **objects** together to make one **object**.

What I mean by composition

Composition is **arranging** many **objects** together to make one **object**.

If a situation has any sort of composition, maybe you have an operad.

What I mean by composition

Composition is **arranging** many **objects** together to make one **object**.

If a situation has any sort of composition, maybe you have an operad.

An operad consists of:

- A collection of **objects** $X, Y, \dots$,
- And ways to **arrange** them, $\varphi: X_1, \dots, X_k \rightarrow Y$,
- Such that arrangements can be **nested** inside each other.

What I mean by composition

Composition is **arranging** many **objects** together to make one **object**.

If a situation has any sort of composition, maybe you have an operad.

An operad consists of:

- A collection of **objects** $X, Y, \dots$,
- And ways to **arrange** them, $\varphi: X_1, \dots, X_k \rightarrow Y$,
- Such that arrangements can be **nested** inside each other.

Slightly more formal definition to come.

Operads are everywhere

Operads are used unconsciously in many fields.

- Electrical engineering: “wiring diagrams”.
- Design: “set-based design”.
- Computer programming: “data flow”.
- Natural language processing: “grammars”.
- Materials science: “hierarchical materials”.

Operads are everywhere

Operads are used unconsciously in many fields.

- Electrical engineering: “wiring diagrams”.
- Design: “set-based design”.
- Computer programming: “data flow”.
- Natural language processing: “grammars”.
- Materials science: “hierarchical materials”.

Let's bring operads to the fore.

- There's a common theme in the way we think.
- Operads structure this sort of thinking.
- With mathematical structure, we can go much further.

Operads are everywhere

Operads are used unconsciously in many fields.

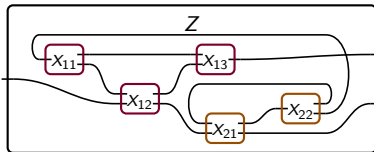
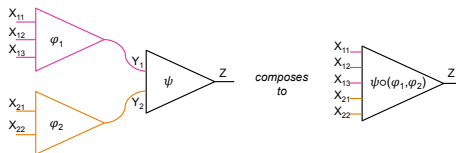
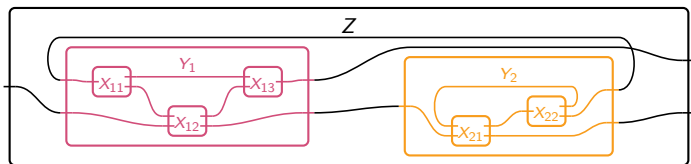
- Electrical engineering: “wiring diagrams”.
- Design: “set-based design”.
- Computer programming: “data flow”.
- Natural language processing: “grammars”.
- Materials science: “hierarchical materials”.

Let's bring operads to the fore.

- There's a common theme in the way we think.
- Operads structure this sort of thinking.
- With mathematical structure, we can go much further.

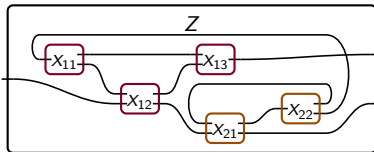
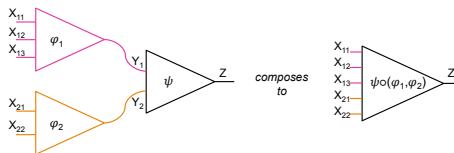
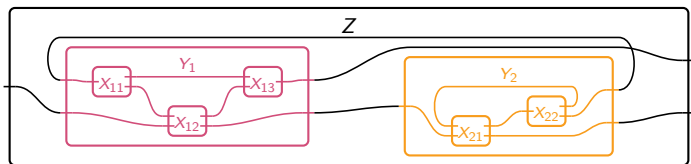
Let's look for **objects**, **arrangements**, and **nesting** in some examples.

Operad 1: wiring diagrams



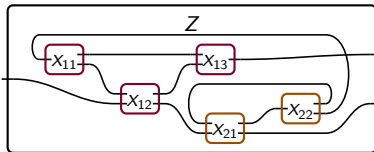
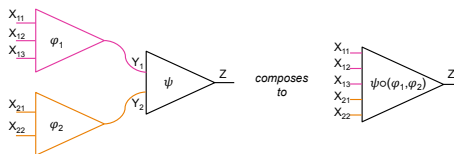
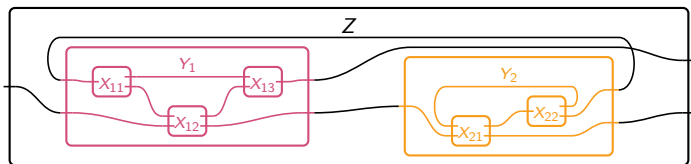
Objects: boxes with ports.

Operad 1: wiring diagrams



Objects: boxes with ports. Arrangements: wiring diagrams.

Operad 1: wiring diagrams



Objects: boxes with ports. Arrangements: wiring diagrams. Nesting: nesting.

Formal definition of operad

An operad $\mathcal{O}$ consists of

- A set $\text{Ob}(\mathcal{O})$, elements of which are called *objects*.
- For objects $X_1, \dots, X_k, Y \in \text{Ob}(\mathcal{O})$, a set

$$\text{Mor}_{\mathcal{O}}(X_1, \dots, X_k; Y)$$

Its elements are called *morphisms* or *arrangements* of $X_1, \dots, X_k$ in Y .
An arrangement $\varphi \in \text{Mor}_{\mathcal{O}}(X_1, \dots, X_k; Y)$ may be denoted

$$\varphi: X_1, \dots, X_k \rightarrow Y.$$

- For each object $X \in \text{Ob}(\mathcal{O})$, an identity arrangement $\text{id}_X: (X) \rightarrow X$.
- A composition, or *nesting* formula, e.g.,

$$\psi \circ (\varphi_1, \dots, \varphi_k): (X_{i,j}) \xrightarrow{\varphi_i} (Y_i) \xrightarrow{\psi} Z.$$

These are required to satisfy well-known “unital” and “associative” laws.

Operad 1: WDs again

An operad $\mathcal{W}$ for composing wiring diagrams:

- Object $X \in \mathcal{W}$: any possible box-with-ports.



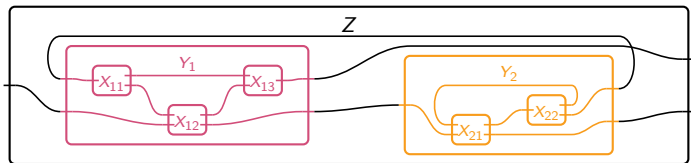
Operad 1: WDs again

An operad $\mathcal{W}$ for composing wiring diagrams:

- Object $X \in \mathcal{W}$: any possible box-with-ports.



- Arrangement $\phi: X_1, \dots, X_k \rightarrow Y$ in $\mathcal{W}$: any wiring of X 's in Y .
- Nesting: the facts about how wiring diagrams fit inside each other.



One operad $\mathcal{W}$ comprises all this.

Operad 2: hierarchical protein materials

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

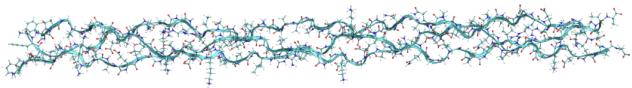
- A **protein** is an **arrangement** of simpler **proteins**.
 - There are “atomic” proteins: amino acids.
 - Protein materials include your skin: stretchable, breathable, waterproof.
 - Materials scientists would *love* to make materials like this.

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

Operad 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: amino acids.
 - Protein materials include your skin: stretchable, breathable, waterproof.
 - Materials scientists would *love* to make materials like this.
- Assemble new proteins from old:
 - arrange in series or parallel (H-bonds), or
 - arrange in helices, double helices, any conceivable curve, etc.



- Collagen has a **nested** structure: it is an array, each fiber of which is a triple helix, each strand of which is a helix, each unit of which is an amino acid.¹

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

More operads: Grammars and recipes

Context-free grammars are “free” operads.

$\langle \text{sentence} \rangle$	$::=$	$\langle \text{noun-phrase} \rangle \langle \text{verb-phrase} \rangle$
$\langle \text{noun-phrase} \rangle$	$::=$	$\langle \text{pronoun} \rangle \mid \langle \text{proper-noun} \rangle \mid \langle \text{determiner} \rangle \langle \text{nominal} \rangle$
$\langle \text{nominal} \rangle$	$::=$	$\langle \text{noun} \rangle \mid \langle \text{noun} \rangle \langle \text{nominal} \rangle$
$\langle \text{verb-phrase} \rangle$	$::=$	$\langle \text{verb} \rangle \mid \langle \text{verb} \rangle \langle \text{noun-phrase} \rangle \mid \langle \text{verb} \rangle \langle \text{prep-phrase} \rangle$
$\langle \text{prep-phrase} \rangle$	$::=$	$\langle \text{preposition} \rangle \langle \text{noun-phrase} \rangle$

More operads: Grammars and recipes

Context-free grammars are “free” operads.

$\langle \text{sentence} \rangle$	$::=$	$\langle \text{noun-phrase} \rangle \langle \text{verb-phrase} \rangle$
$\langle \text{noun-phrase} \rangle$	$::=$	$\langle \text{pronoun} \rangle \mid \langle \text{proper-noun} \rangle \mid \langle \text{determiner} \rangle \langle \text{nominal} \rangle$
$\langle \text{nominal} \rangle$	$::=$	$\langle \text{noun} \rangle \mid \langle \text{noun} \rangle \langle \text{nominal} \rangle$
$\langle \text{verb-phrase} \rangle$	$::=$	$\langle \text{verb} \rangle \mid \langle \text{verb} \rangle \langle \text{noun-phrase} \rangle \mid \langle \text{verb} \rangle \langle \text{prep-phrase} \rangle$
$\langle \text{prep-phrase} \rangle$	$::=$	$\langle \text{preposition} \rangle \langle \text{noun-phrase} \rangle$

Recipes are also operads.

- Combine sub-recipes to make a recipe.
- The outline for this talk as an operad:
 - **Objects**: points I want to make.
 - **Arrangements**: putting points together to make a bigger point.
 - **Nesting**: the well-known outline structure.

Operads and their algebras

One more formal notion to discuss: algebras.

Operads and their algebras

One more formal notion to discuss: algebras.

Rules of composition vs. stuff being composed.

Operads and their algebras

One more formal notion to discuss: algebras.

Rules of composition vs. stuff being composed.

- Operad : Group theory :: Algebras : Groups.
- Operad : Ring theory :: Algebras : Rings.

Operads and their algebras

One more formal notion to discuss: algebras.

Rules of composition vs. stuff being composed.

- Operad : Group theory :: Algebras : Groups.
- Operad : Ring theory :: Algebras : Rings.

Operad = theory. Algebras = models.

Each operad has many algebras

Each operad $\mathcal{O}$ is a “theory of composition”.

- **Objects** $X, Y, \dots$: what *sorts* of elements in this theory?
- **Arrangements** ϕ, ψ : what are the operations?
- **Nesting**: what kind of laws?

Each operad has many algebras

Each operad $\mathcal{O}$ is a “theory of composition”.

- **Objects** $X, Y, \dots$: what *sorts* of elements in this theory?
- **Arrangements** ϕ, ψ : what are the operations?
- **Nesting**: what kind of laws?

The $\mathcal{O}$ -algebras are the models of theory $\mathcal{O}$.²

- An $\mathcal{O}$ -algebra A says what’s actually being composed.
 - To each **object** X : a set $A(X)$ of elements.
 - To each **arrangement** ϕ : an k -ary operation $A(\phi)$.
 - If $\phi: X_1, \dots, X_k \rightarrow Y$ is an arrangement,
 - Then $A(\phi): A(X_1) \times \dots \times A(X_k) \rightarrow A(Y)$ is a function.
 - To each **nesting**: a law in A .

²Technically, an algebra is a functor $A: \mathcal{O} \rightarrow \mathbf{Set}$.

Each operad has many algebras

Each operad $\mathcal{O}$ is a “theory of composition”.

- **Objects** $X, Y, \dots$: what *sorts* of elements in this theory?
- **Arrangements** ϕ, ψ : what are the operations?
- **Nesting**: what kind of laws?

The $\mathcal{O}$ -algebras are the models of theory $\mathcal{O}$.²

- An $\mathcal{O}$ -algebra A says what’s actually being composed.
 - To each **object** X : a set $A(X)$ of elements.
 - To each **arrangement** ϕ : an k -ary operation $A(\phi)$.
 - If $\phi: X_1, \dots, X_k \rightarrow Y$ is an arrangement,
 - Then $A(\phi): A(X_1) \times \dots \times A(X_k) \rightarrow A(Y)$ is a function.
 - To each **nesting**: a law in A .

Next, I’ll explain what algebras on an operad look like.

²Technically, an algebra is a functor $A: \mathcal{O} \rightarrow \mathbf{Set}$.

Example: the operad for monoids

Monoids are groups without inverses.

- Any group G is a monoid.
- Natural numbers $\mathbb{N}$.
- $(n \times n)$ -matrices form a monoid under multiplication.

Example: the operad for monoids

Monoids are groups without inverses.

- Any group G is a monoid.
- Natural numbers $\mathbb{N}$.
- $(n \times n)$ -matrices form a monoid under multiplication.

There is one operad $\mathcal{O}$ whose algebras are monoids.

Example: the operad for monoids

Monoids are groups without inverses.

- Any group G is a monoid.
- Natural numbers $\mathbb{N}$.
- $(n \times n)$ -matrices form a monoid under multiplication.

There is one operad $\mathcal{O}$ whose algebras are monoids.

$\mathcal{O}$ has one object $\bullet$ and one k -ary morphism for each k : “ $*$ ”.

Example: the operad for monoids

Monoids are groups without inverses.

- Any group G is a monoid.
- Natural numbers $\mathbb{N}$.
- $(n \times n)$ -matrices form a monoid under multiplication.

There is one operad $\mathcal{O}$ whose algebras are monoids.

$\mathcal{O}$ has one object $\bullet$ and one k -ary morphism for each k : “ $*$ ”.

- To the object $\bullet$, a set $A(\bullet)$.
- To the k -ary morphism, a function $A(*) : A(\bullet) \times \cdots \times A(\bullet) \rightarrow A(\bullet)$.
 - Multiply k -many elements of a group G .
 - Add k -many numbers.
 - Multiply k -many $(n \times n)$ -matrices.

Example: the operad for monoids

Monoids are groups without inverses.

- Any group G is a monoid.
- Natural numbers $\mathbb{N}$.
- $(n \times n)$ -matrices form a monoid under multiplication.

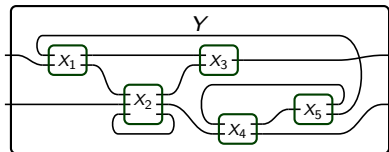
There is one operad $\mathcal{O}$ whose algebras are monoids.

$\mathcal{O}$ has one object $\bullet$ and one k -ary morphism for each k : “ $*$ ”.

- To the object $\bullet$, a set $A(\bullet)$.
- To the k -ary morphism, a function $A(*) : A(\bullet) \times \cdots \times A(\bullet) \rightarrow A(\bullet)$.
 - Multiply k -many elements of a group G .
 - Add k -many numbers.
 - Multiply k -many $(n \times n)$ -matrices.
- The operad laws (unitality, associativity) guarantee the monoid laws.

Multiple algebras for wiring diagrams operad

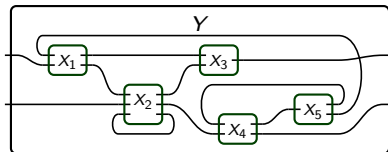
Recall operad $\mathcal{W}$ of all boxes X and WDs, $\phi: X_1, \dots, X_k \rightarrow Y$.³



³ Imagine each port / wire labeled by a topological space: the signal space.

Multiple algebras for wiring diagrams operad

Recall operad $\mathcal{W}$ of all boxes X and WDs, $\phi: X_1, \dots, X_k \rightarrow Y$.³

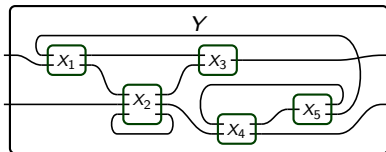


As previously: there are many different $\mathcal{W}$ -algebras $A: \mathcal{W} \rightarrow \mathbf{Set}$.

³ Imagine each port / wire labeled by a topological space: the signal space.

Multiple algebras for wiring diagrams operad

Recall operad $\mathcal{W}$ of all boxes X and WDs, $\phi: X_1, \dots, X_k \rightarrow Y$.³



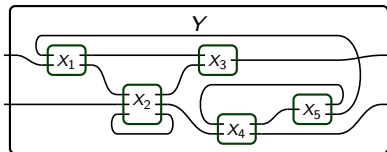
As previously: there are many different $\mathcal{W}$ -algebras $A: \mathcal{W} \rightarrow \mathbf{Set}$.

- Open dynamical systems: machines with ports.
 - $A(X)$: the set of all DS's with input-output shape X .
 - $A(\phi)$: a certain variable-sharing formula.

³ Imagine each port / wire labeled by a topological space: the signal space.

Multiple algebras for wiring diagrams operad

Recall operad $\mathcal{W}$ of all boxes X and WDs, $\phi: X_1, \dots, X_k \rightarrow Y$.³



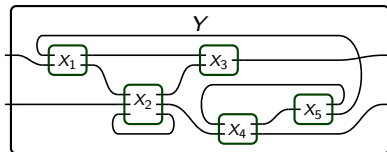
As previously: there are many different $\mathcal{W}$ -algebras $A: \mathcal{W} \rightarrow \mathbf{Set}$.

- Open dynamical systems: machines with ports.
 - $A(X)$: the set of all DS's with input-output shape X .
 - $A(\phi)$: a certain variable-sharing formula.
- In fact, dynamical systems comprise several algebras:
 - A_1 : continuous DS's (ODE's with time-varying parameters).
 - A_2 : discrete DS's (on a discrete clock).
 - A_3 : hybrid DS's (cyber-physical systems).

³ Imagine each port / wire labeled by a topological space: the signal space.

Multiple algebras for wiring diagrams operad

Recall operad $\mathcal{W}$ of all boxes X and WDs, $\phi: X_1, \dots, X_k \rightarrow Y$.³



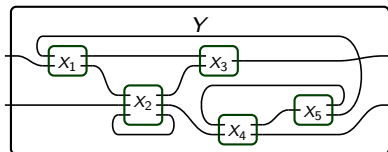
As previously: there are many different $\mathcal{W}$ -algebras $A: \mathcal{W} \rightarrow \mathbf{Set}$.

- Open dynamical systems: machines with ports.
 - $A(X)$: the set of all DS's with input-output shape X .
 - $A(\phi)$: a certain variable-sharing formula.
- In fact, dynamical systems comprise several algebras:
 - A_1 : continuous DS's (ODE's with time-varying parameters).
 - A_2 : discrete DS's (on a discrete clock).
 - A_3 : hybrid DS's (cyber-physical systems).
- But there's a much simpler “algebraic” algebra...

³ Imagine each port / wire labeled by a topological space: the signal space.

Tensors are a $\mathcal{W}$ -algebra

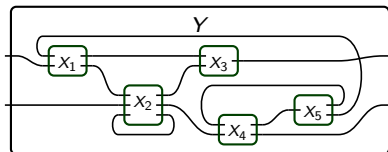
We said $\mathcal{W}$ is modeled by dynamical systems.



⁴ Imagine each wire labeled by a natural number: the dimension.

Tensors are a $\mathcal{W}$ -algebra

We said $\mathcal{W}$ is modeled by dynamical systems.



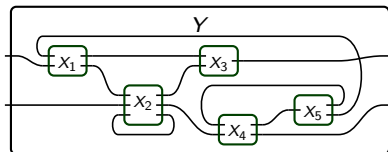
But $\mathcal{W}$ is also modeled by tensors (not dynamic at all).⁴

- Box = tensor format ($T \in V_1 \otimes \cdots \otimes V_n$).
- Wiring diagram = tensor network.
- Contract along shared wires to “compose”.

⁴ Imagine each wire labeled by a natural number: the dimension.

Tensors are a $\mathcal{W}$ -algebra

We said $\mathcal{W}$ is modeled by dynamical systems.



But $\mathcal{W}$ is also modeled by tensors (not dynamic at all).⁴

- Box = tensor format ($T \in V_1 \otimes \cdots \otimes V_n$).
- Wiring diagram = tensor network.
- Contract along shared wires to “compose”.

Later: a relationship between dynamical systems and tensors...

- ... which led me to a numerical method for solving systems.
- The pixel array method, up next.

⁴ Imagine each wire labeled by a natural number: the dimension.

Detailed look at a real-world application

Separately plot the solutions to equations: $f(x, w) = 0$ and $g(w, y) = 0$.

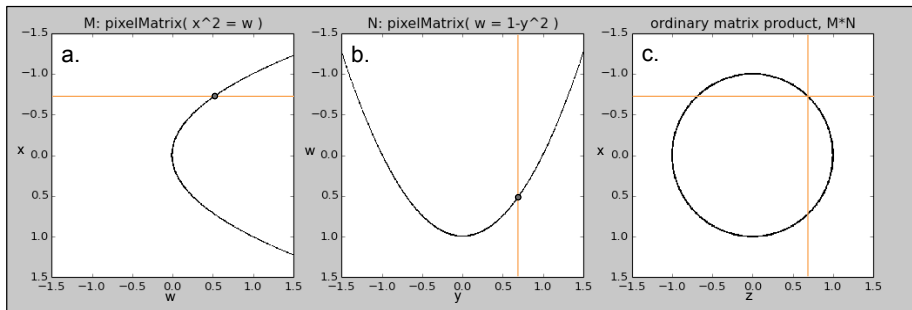
- Plot each in a bounding box, e.g. $[-1.5, 1.5]$.
- Consider plots as matrices of on/off pixels (booleans).
- Multiply matrices to solve system.

Detailed look at a real-world application

Separately plot the solutions to equations: $f(x, w) = 0$ and $g(w, y) = 0$.

- Plot each in a bounding box, e.g. $[-1.5, 1.5]$.
- Consider plots as matrices of on/off pixels (booleans).
- Multiply matrices to solve system.

Example: $x^2 = w$ and $w = 1 - y^2$.



A more complex example

The following eq's are not differentiable, nor even defined everywhere.

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

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

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

Q: For what values of w and z does a simultaneous solution exist? ⁵

⁵Spivak; Dobson; Kumari; Wu (2016) "Pixel Arrays: A fast and elementary method for solving nonlinear systems". <https://arxiv.org/abs/1609.00061>

A more complex example

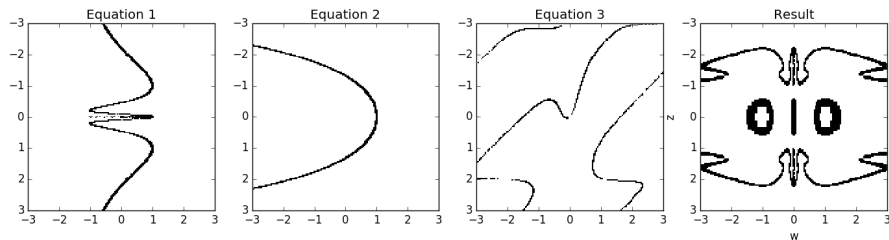
The following eq's are not differentiable, nor even defined everywhere.

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

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

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

Q: For what values of w and z does a simultaneous solution exist? ⁵



⁵Spivak; Dobson; Kumari; Wu (2016) "Pixel Arrays: A fast and elementary method for solving nonlinear systems". <https://arxiv.org/abs/1609.00061>

Equations and wiring diagrams

Consider an arbitrary system of equations having the following form:

$$f_1(\mathbf{t}, u, \mathbf{v}) = 0$$

$$f_2(\mathbf{v}, w, x) = 0$$

$$f_3(u, w, x, y) = 0$$

$$f_4(x, \mathbf{z}) = 0$$

Bold variables are those we want to *expose*; others are *unexposed*.

Equations and wiring diagrams

Consider an arbitrary system of equations having the following form:

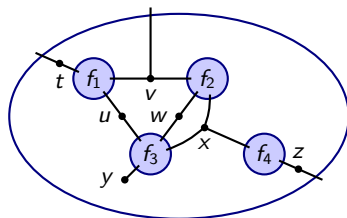
$$f_1(\mathbf{t}, u, \mathbf{v}) = 0$$

$$f_2(\mathbf{v}, w, x) = 0$$

$$f_3(u, w, x, y) = 0$$

$$f_4(x, z) = 0$$

Bold variables are those we want to *expose*; others are *unexposed*.

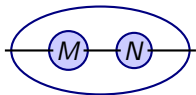


Said another way, we want $\{(t, v, z) \mid \exists u, w, x, y : f_1 = f_2 = f_3 = f_4 = 0\}$.

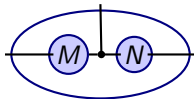
Example wiring diagrams for named operations

Some famous matrix products as wiring diagrams:

Multiplication: MN



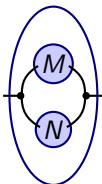
Khatri-Rao: $M \odot N$



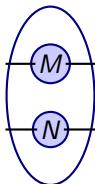
Trace: $\text{Tr}(M)$



Hadamard: $M \circ N$



Kronecker: $M \otimes N$

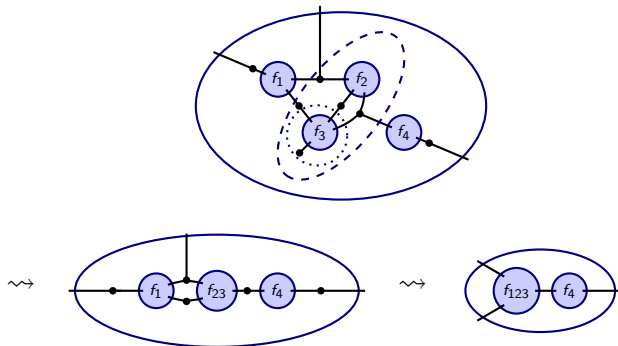


Marginalize: $\sum_i M_{i,j}$



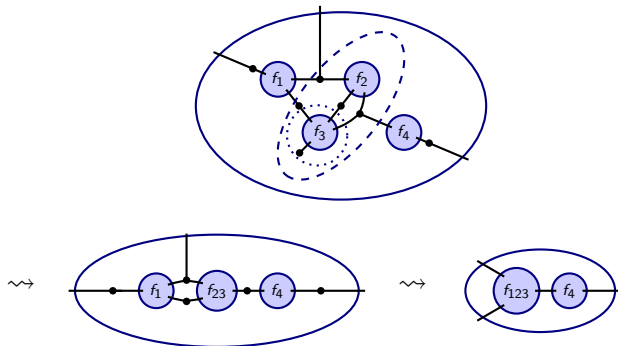
Clustering for speed

To solve systems of equations fast, cluster:



Clustering for speed

To solve systems of equations fast, cluster:



The operad associative law lets you choose the order.

- Order of contracting edges doesn't affect solution.
- But it does affect speed.

Speed test: apples and oranges

The inputs and outputs of the PA method are different than Newton's.

	Pixel Array method	Newton's method
Inputs:	a range for each variable	a good initial guess
Outputs:	all solutions for some variables	one solution in all variables.

- So it's hard to compare the speed of PA with Newton. (?)

Speed test: apples and oranges

The inputs and outputs of the PA method are different than Newton's.

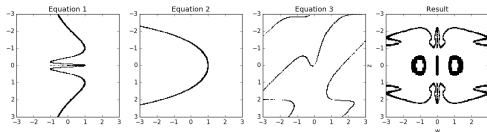
	Pixel Array method	Newton's method
Inputs:	a range for each variable	a good initial guess
Outputs:	all solutions for some variables	one solution in all variables.

- So it's hard to compare the speed of PA with Newton. (?)
- Our speed test assumes you want to produce “all solutions” in range.

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

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

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



- We iterated Newton over all boxes in the grid, each as an initial guess.

Speed test: apples and oranges

The inputs and outputs of the PA method are different than Newton's.

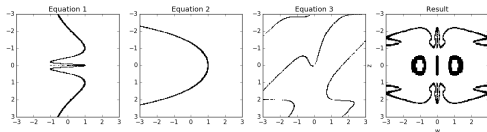
	Pixel Array method	Newton's method
Inputs:	a range for each variable	a good initial guess
Outputs:	all solutions for some variables	one solution in all variables.

- So it's hard to compare the speed of PA with Newton. (?)
- Our speed test assumes you want to produce “all solutions” in range.

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

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

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

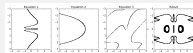


- We iterated Newton over all boxes in the grid, each as an initial guess.

Above case: PA was over 7200x faster.

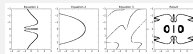
- PA: 1.5 seconds; Newton: we stopped Julia's NLSolve after 3 hours.
- PA was faster in every partially-decomposable system we tried.

Accuracy of the pixel array method



Output accuracy = input accuracy.

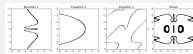
Accuracy of the pixel array method



Output accuracy = input accuracy.

- Start with a plot for each equation $f(x) = 0$ in system.
 - Fact: every cts function is uniformly cts in a bounding box.
 - This sets up a relationship between mesh-size δ and tolerance ϵ .

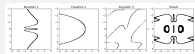
Accuracy of the pixel array method



Output accuracy = input accuracy.

- Start with a plot for each equation $f(x) = 0$ in system.
 - Fact: every cts function is uniformly cts in a bounding box.
 - This sets up a relationship between mesh-size δ and tolerance ϵ .
 - Plotting method: sample in center c of pixel, mark if $|f(c)| < \epsilon$.
 - This gives two accuracy guarantees:
 - If $f(x) = 0$ anywhere in the pixel then it is marked.
 - If pixel is marked then $|f(x)| < 2\epsilon$ everywhere in the pixel.

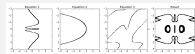
Accuracy of the pixel array method



Output accuracy = input accuracy.

- Start with a plot for each equation $f(x) = 0$ in system.
 - Fact: every cts function is uniformly cts in a bounding box.
 - This sets up a relationship between mesh-size δ and tolerance ϵ .
 - Plotting method: sample in center c of pixel, mark if $|f(c)| < \epsilon$.
 - This gives two accuracy guarantees:
 - If $f(x) = 0$ anywhere in the pixel then it is marked.
 - If pixel is marked then $|f(x)| < 2\epsilon$ everywhere in the pixel.
- PA method: multiply plots as tensors.
- Result (solution) has the same guarantees:
 - If there is a solution in a pixel, it is marked.
 - If a pixel is marked, then the pixel is 2ϵ -close to a solution.

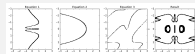
Other selling points of the PA method



Other selling points.

- Simple to implement: try it yourself in minutes.

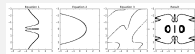
Other selling points of the PA method



Other selling points.

- Simple to implement: try it yourself in minutes.
- Works more generally:
 - Use relations instead, e.g. $f(x) \leq 0$.
 - Use arbitrary data sets as plots.

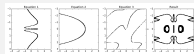
Other selling points of the PA method



Other selling points.

- Simple to implement: try it yourself in minutes.
- Works more generally:
 - Use relations instead, e.g. $f(x) \leq 0$.
 - Use arbitrary data sets as plots.
- Replace Booleans $\{T, F\}$ with any ordered *semiring* R , e.g. $\mathbb{R}$.
 - Example: $([0, \infty], \min, \max)$, “save the ϵ ’s.”
 - Monotonic: zoom in and out.
 - Multilinear: amenable to tensor decompositions.

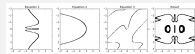
Other selling points of the PA method



Other selling points.

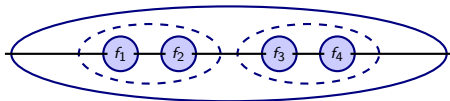
- Simple to implement: try it yourself in minutes.
- Works more generally:
 - Use relations instead, e.g. $f(x) \leq 0$.
 - Use arbitrary data sets as plots.
- Replace Booleans $\{T, F\}$ with any ordered *semiring* R , e.g. $\mathbb{R}$.
 - Example: $([0, \infty], \min, \max)$, “save the ϵ ’s.”
 - Monotonic: zoom in and out.
 - Multilinear: amenable to tensor decompositions.
- Robust: don’t need perfect plots.

Other selling points of the PA method



Other selling points.

- Simple to implement: try it yourself in minutes.
- Works more generally:
 - Use relations instead, e.g. $f(x) \leq 0$.
 - Use arbitrary data sets as plots.
- Replace Booleans $\{T, F\}$ with any ordered *semiring* R , e.g. $\mathbb{R}$.
 - Example: $([0, \infty], \min, \max)$, “save the ϵ ’s.”
 - Monotonic: zoom in and out.
 - Multilinear: amenable to tensor decompositions.
- Robust: don’t need perfect plots.
- Compositional: solve one piece now, use it later.



The PA idea is not particular to equations

Data often comes in the form of relations $R \subseteq A \times B \times C$.



A	B	C
1	r	3.77
1	r	3.84
1	s	2.04
2	r	4.90
2	s	2.18
.	.	.
.	.	.
.	.	.
4	s	6.18



	Attrib.	a_1	a_2	a_3	a_4
Obj.					
o_1			•	•	
o_2		•	•	•	
o_3				•	•
o_4				•	
o_5		•		•	
o_6				•	•
o_7			•	•	•

The PA idea is not particular to equations

Data often comes in the form of relations $R \subseteq A \times B \times C$.



A diagram showing a blue circle labeled 'R'. Three arrows point into it from the right, top, and bottom, labeled 'A', 'B', and 'C' respectively.

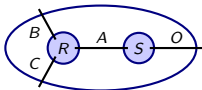
A	B	C
1	r	3.77
1	r	3.84
1	s	2.04
2	r	4.90
2	s	2.18
.	.	.
.	.	.
.	.	.
4	s	6.18



A diagram showing a blue circle labeled 'S'. Two arrows point into it from the left and right, labeled 'A' and 'O' respectively.

Attrib.	a_1	a_2	a_3	a_4
Obj.				
o_1		•	•	
o_2	•	•	•	
o_3			•	•
o_4				
o_5	•		•	
o_6			•	•
o_7		•	•	•

Relations form an $\mathcal{W}$ -algebra.



"Find all (b, c, o) for which a match exists in A "

The PA idea is not particular to equations

Data often comes in the form of relations $R \subseteq A \times B \times C$.



A diagram showing a blue circle labeled 'R' with three lines extending from it, labeled 'A', 'B', and 'C'.

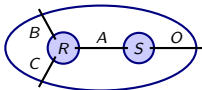
A	B	C
1	r	3.77
1	r	3.84
1	s	2.04
2	r	4.90
2	s	2.18
.	.	.
.	.	.
.	.	.
4	s	6.18



A diagram showing a blue circle labeled 'S' with two lines extending from it, labeled 'A' and 'O'.

Attrib.	a_1	a_2	a_3	a_4
Obj.				
o_1		•	•	
o_2	•	•	•	
o_3			•	•
o_4			•	
o_5	•		•	
o_6			•	•
o_7		•	•	•

Relations form an $\mathcal{W}$ -algebra.

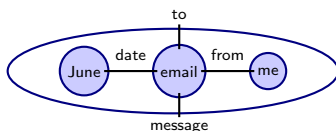


"Find all (b, c, o) for which a match exists in A "

I've co-founded a CT-based data integration company.

The value of a mathematical language

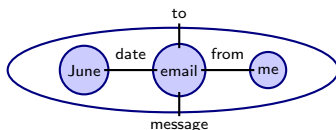
The operad structure is everywhere:



- Bridge formal math and casual users.
- The same formalism, many uses.
 - Data (relations) form a $\mathcal{W}$ -algebra.
 - So do matrices.
 - So do “compositional neural networks”.
 - So do dynamical systems.

The value of a mathematical language

The operad structure is everywhere:



- Bridge formal math and casual users.
- The same formalism, many uses.
 - Data (relations) form a $\mathcal{W}$ -algebra.
 - So do matrices.
 - So do “compositional neural networks”.
 - So do dynamical systems.
- Category theory considers relationships between these algebras.
 - “Compositional mappings”; like group homomorphisms.
 - This is our final topic.

Compositional mappings

In group theory: study groups using group homomorphisms.

- $f: G \rightarrow G', \quad f(g_1 * g_2) = f(g_1) *' f(g_2).$
- Lets you retain something, while dropping something.
- Controlled release of information.

Compositional mappings

In group theory: study groups using group homomorphisms.

- $f: G \rightarrow G', \quad f(g_1 * g_2) = f(g_1) *' f(g_2).$
- Lets you retain something, while dropping something.
- Controlled release of information.

Generalize to any operad $\mathcal{O}$.

- Study $\mathcal{O}$ -algebras using homomorphisms.
- $f: A \rightarrow A', \quad f \circ A(\phi) = A'(\phi) \circ f.$
- I call these “compositional” mappings/analyses.

Compositional mappings

In group theory: study groups using group homomorphisms.

- $f: G \rightarrow G', \quad f(g_1 * g_2) = f(g_1) *' f(g_2).$
- Lets you retain something, while dropping something.
- Controlled release of information.

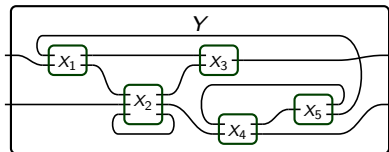
Generalize to any operad $\mathcal{O}$.

- Study $\mathcal{O}$ -algebras using homomorphisms.
- $f: A \rightarrow A', \quad f \circ A(\phi) = A'(\phi) \circ f.$
- I call these “compositional” mappings/analyses.

Compositional analysis is future-proof analysis.

- Example: *Average* is not compositional; *sum* and *count* are.
- Compositionality ensures your work can be reused.

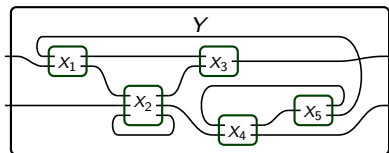
Compositional mappings of open dynamical systems



Recall the algebras of open dynamical systems.

- Continuous, discrete, hybrid.
- Each gives a different algebra.
- Continuous and discrete are subalgebras of hybrid.

Compositional mappings of open dynamical systems



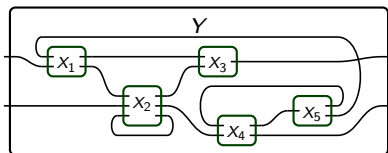
Recall the algebras of open dynamical systems.

- Continuous, discrete, hybrid.
- Each gives a different algebra.
- Continuous and discrete are subalgebras of hybrid.

“Subalgebra” is the first type of compositional mapping.

Compositional mappings

Suppose there's a continuous DS in each box:

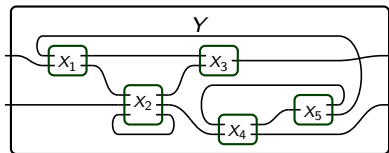


Two choices; what's the difference?

- Choice 1: compose the continuous systems and discretize the result;
- Choice 2: discretize each and then compose the discrete systems.

Compositional mappings

Suppose there's a continuous DS in each box:



Two choices; what's the difference?

- Choice 1: compose the continuous systems and discretize the result;
- Choice 2: discretize each and then compose the discrete systems.

“Discretization is compositional”: you get the same result either way.

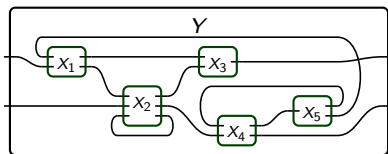
- This requires proof: arbitrary wiring diagrams, arbitrary DS's.
- I proved it for Euler;⁶ my student proved it for Runge Kutta.⁷

⁶Spivak “The steady states of dynamical systems”. *arXiv* 1512.00802.

⁷Ngotiaoco “Compositionality of the Runge-Kutta Method”. *arXiv* 1707.02804.

Bifurcation diagrams are compositional

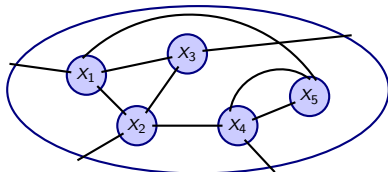
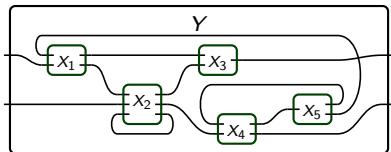
One last compositional mapping: bifurcation diagrams.



Bifurcation diagram: plot of steady states.

Bifurcation diagrams are compositional

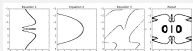
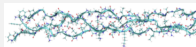
One last compositional mapping: bifurcation diagrams.



Bifurcation diagram: plot of steady states.

- Steady states: solution to $\dot{x} = 0$.
- Do this for each DS in a wiring diagram.
 - Get a system of equations.
 - Their plots are bifurcation diagrams.
 - Compose these plots using PA method.
- Bifurcation diagrams are compositional.
- Steady states of whole = simultaneous steady states of parts.

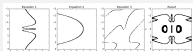
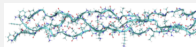
Summary



Category theory, and operads, as a framework for information integration.

- Operads give a framework for composing things of any sort.
 - Object (interface), morphism (arrangement), composition (nesting).
 - Discussed wiring diagrams, protein materials, grammars, recipes.
 - Operad = “theory of composition”; algebras = models.

Summary



Category theory, and operads, as a framework for information integration.

- Operads give a framework for composing things of any sort.
 - Object (interface), morphism (arrangement), composition (nesting).
 - Discussed wiring diagrams, protein materials, grammars, recipes.
 - Operad = “theory of composition”; algebras = models.

Go further by adding mathematical structure to complex systems.

- Solving systems of equations: pixel array method.
- Information integration: combine data from disparate sources.
- Interconnect dynamic systems: NASA work.
- Discretization of dynamical systems: you choose the order.

Outlook

Big data and deep learning: so hot right now.

- For good cause: it's been wildly successful and surpassed expectations.
- But surpassing expectations isn't generally sustainable.

Outlook

Big data and deep learning: so hot right now.

- For good cause: it's been wildly successful and surpassed expectations.
- But surpassing expectations isn't generally sustainable.

A quieter rumbling with category theory....

- Verizon and Facebook use functional programming.
- Airbus, Amgen, and Dossault use CT as a modeling language.
- DARPA increasingly calls for CT and “compositional mathematics”.
- NIST is considering CT as a standard.

Outlook

Big data and deep learning: so hot right now.

- For good cause: it's been wildly successful and surpassed expectations.
- But surpassing expectations isn't generally sustainable.

A quieter rumbling with category theory....

- Verizon and Facebook use functional programming.
- Airbus, Amgen, and Dossault use CT as a modeling language.
- DARPA increasingly calls for CT and “compositional mathematics”.
- NIST is considering CT as a standard.

Moving past perception: category theory?

- I'm no expert, but deep learning seems analogous to perception.
- Recognizing patterns without really being able to explain how.
- To go further, need language and tools to integrate knowledge.
- Category theory is the natural choice.

Questions and comments are welcome!