Thinking about modularity in networks

David I. Spivak

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

Presented on 2015/04/03
at the University of Pennsylvania
Complex Systems Seminar

Outline

1 **Introduction**
- Motivation
- Introducing operads

2 Recognizing a connection

3 Defining operads

4 Applications of operads

5 Networks of networks

6 Conclusion

Motivation

- At all times we seem to find ourselves stuck between:
  - the opportunities and obligations that we feel we must fulfill, and
  - the imperfect resources we're provided.
- This is true at many scales and in many domains.
  - It's true at my scale; I hope to fulfill expectations, given what I have.
  - It's true at the scale of my organs, and my cells.
  - It's true in a corporation or government agency.
- "Stretching it", it's even true for mechanical properties of materials:
  - If force applied on one end and it's not transferred across,
  - the material "fails". It is no longer one material.

The ubiquity of a strategy

- In all these systems, there seems to be a common strategy:
  - The role of the parts is to distribute the load assigned to the whole.
  - In this way you can transfer obligations effectively to your peers
  - (without destroying those connections by “asking too much”).
- The scale-invariance and ubiquity of this strategy is compelling.
  - A well-designed material has a way of distributing load.
  - So does someone with good time management skills.
  - Your brain distributes its current problem among its subsystems.
- There are many domains which display this kind of scale-invariance.

The promise of fractals

- I recall my father telling me about a kind of “fractal fever”.
- In the 1980s scientists were very interested in fractals, e.g., in:
  - Plants (a single leaf or broccoli).
  - Rivers, faults, and vasculature.
  - Stock market fluctuations.
- Scientists wanted to use fractals as a conceptual tool for explaining phenomena.

It didn't quite work for everyone; why?

- Fractals are a little too special: the machinery is too limited.
- Scientists from whom the analogy was compelling couldn't always produce:
  - shapes with fractional dimension,
  - repeated patterns, no matter how far you zoom in,
  - iterated functions or recurrence relations to generate their phenomena.
- Fractals are always about space and geometry.
- The inspirational and compelling idea wasn't completely realized.
  - Unlike fractals, the cases of interest weren't always geometric objects.
  - For example, the scale-invariance of obligations vs. resources isn't.

Operads describe similar phenomena

- I believe the promise of fractals may still be realized by operads.
- By "the promise of fractals" I roughly mean:
  - a mathematical formalism for understanding self-similarity across scales.
- An operad $\mathcal{O}$ is a collection of operations, which can be combined.
  - Operads can reproduce fractals as fixed points of operations on $\mathbb{C}$.
  - But operads are much more flexible than fractals.
  - They're not just about geometry and contraction mappings.
- Operads are the mathematics of modularity.
  - Modules can be combined according to the operations in $\mathcal{O}$.
  - The result is a new module, ready to be further put in combination.

Plan of the talk

- I'm leaving fractals aside; they were just motivation.
- I want to explain operads: how they might be interesting to scientists.
- Here's the plan:
    - Discuss a better running example: recipes.
    - Give the formal definition of operads.
    - Provide a couple different examples: materials and networks.
    - Conclude the talk.
- The main theme will be modularity:
    - Building up complex systems by combining subsystems.

Outline

1 Introduction

2 **Recognizing a connection**
- The operadic nature of recipes
- Applying pure math

3 Defining operads

4 Applications of operads

5 Networks of networks

6 Conclusion

Recipes

Here's a recipe for impressing ones new friend:

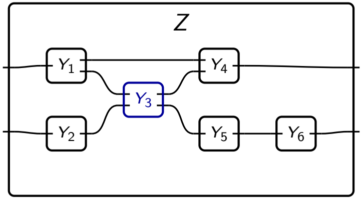
- Invite them over.
- Prepare before they arrive.
  - Make sure the house is clean.
  - Cook a fancy dinner.
    - Find a recipe that people say is good.
    - Go to the store to get ingredients.
    - Follow the recipe.
  - Think of a few things to talk about with the guests.
- When they arrive:
  - Offer them a drink.
  - (etc.)

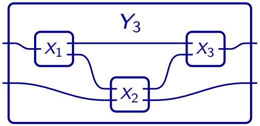
What's operadic about recipes

- A recipe is built out of steps which are themselves sub-recipes.
- These sub-recipes can be done in series, or in parallel.
- It has to do with zooming and chunking.
  - Can we zoom in forever and see recipes all the way down?
  - Maybe, but that's not a necessary part of being an operad.
  - What's necessary is that you can zoom out.
  - You can put recipes together (series and parallel); the result is a recipe.
- Put together a recipe for batter and one for frosting, and make a cake.

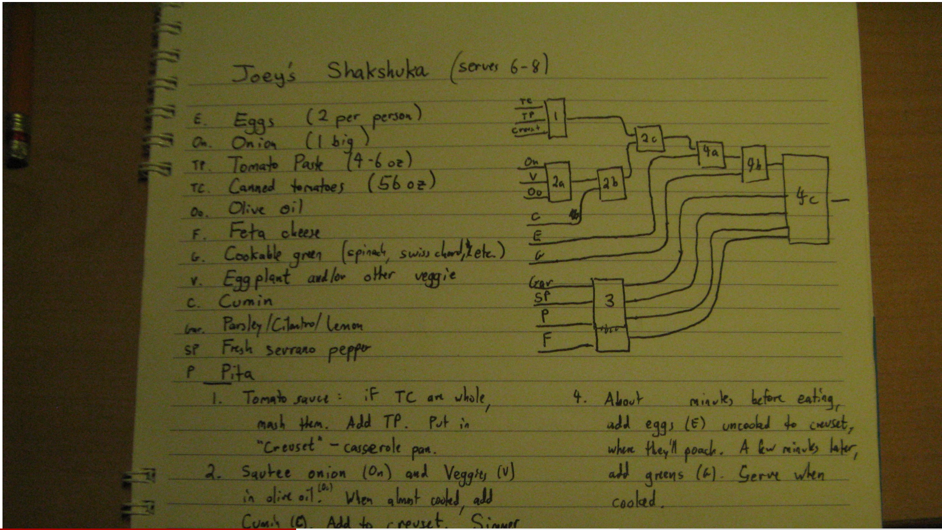
A picture of a recipe

- On the left you see a recipe for Z.
- The steps are $Y_1, \ldots Y_6$.
  - Some have a specific order: step Y_1 must be done before Y_3.
  - Others don't: step Y_4 can be done in any order with Y_5 and Y_6.
- We can elaborate on the details of Y_3, to see how it's implemented.
  - Shown on the right: note it has the correct number of in/out ports.
  - If you substitute it in, you'll replace module Y_3 with X_1, X_2, X_3.





Example: Shakshuka!



Joey's Shakshuka (serves 6-8)

E.	Eggs (2 per person)
On.	Onion (1 big)
TP.	Tomato Paste (4-6 oz)
TC.	Canned tomatoes (56 oz)
Oo.	Olive oil
F.	Feta cheese
G.	Cookable green (spinach, swiss chard, etc.)
V.	Eggplant and/or other veggie
C.	Cumin
Gar.	Parsley/Cilantro/Lemon
SP	Fresh serrano pepper
P	Pita

1. Tomato sauce: if TC are whole, mash them. Add TP. Put in "Creuset" - casserole pan.
2. Sautee onion (On) and Veggies (V) in olive oil$^{(Oo)}$. When almost cooked, add Cumin (C). Add to creuset. Simmer

4. About minutes before eating, add eggs (E) uncooked to creuset, where they'll poach. A few minutes later, add greens (G). Serve when cooked.

Category Theory

- Operads are a sub-discipline of category theory (CT).
- Since its invention in the 1940s, CT has revolutionized math.
  - It is able to connect disparate disciplines into a unified framework.
  - It abstracts common themes from algebra, topology, and logic.
  - It's the key to accessing the world of pure math.
- Category theory has been applied outside of math as well.
  - Computer science (functional programming, databases),
  - Physics (Feynman diagrams, quantum information theory).

Applied category theory

- Operads, like all of CT, was invented for its use in pure math.
- The notion of “modular systems” fits naturally into this framework.
- I’m speaking to you in the very early stages of this application.
    - I don’t yet know all the ways in which operads will be useful.
    - But operads have demonstrated their power in pure math.
    - And pure math has demonstrated its utility in science.
- Future progress will be driven by collaborations.

Outline

1 Introduction

2 Recognizing a connection

3 Defining operads
- A framework for modularity
- Formal definition of operads
- Example of composition

4 Applications of operads

5 Networks of networks

6 Conclusion

An operad is an "abstract modular environment"

- I will define operads formally in a few slides.
- An operad $\mathcal{O}$ is a framework for any sort of modularity.
- To specify $\mathcal{O}$ is to specify:
  - The set of module types (or interfaces) you'll consider.
  - The ways that modules can be put together to form larger ones.
  - How nesting works. (Usually feels obvious, but it must be specified.)
- Recipes, as we discussed, fits this description:
  - A module type is a box with input and output channels (ingredients).
  - Boxes are put together by connecting ingredient supply to demand.
  - Nesting is accomplished by expanding a step as a recipe of its own.

What is an operad? An overview

- An operad consists of a few interlocking components, including:
  1. A set of *objects*, a.k.a. module types, interfaces, or building blocks.
  2. A set of *morphisms*, a.k.a., arrangements or building instructions.
  3. A formula for *composition*, a.k.a, nesting or instruction composition.
- Objects, morphisms, and compositions are the heart and soul of CT.

Formal definition of operad

An operad $\mathcal{O}$ consists of

- A set $\mathrm{Ob}(\mathcal{O})$, elements of which are called *objects*, or interfaces.
- For interfaces $X_1, \ldots, X_n, Y \in \mathrm{Ob}(\mathcal{O})$, a set

  $$\mathrm{Hom}_{\mathcal{O}}(X_1, \ldots, X_n; Y)$$

  Its elements are called *morphisms* or arrangements of $X_1, \ldots, X_n$ in Y. An arrangement $\varphi \in \mathrm{Hom}_{\mathcal{O}}(X_1, \ldots, X_n; Y)$ may be denoted

  $$\varphi \colon (X_1, \ldots, X_n) \to Y.$$

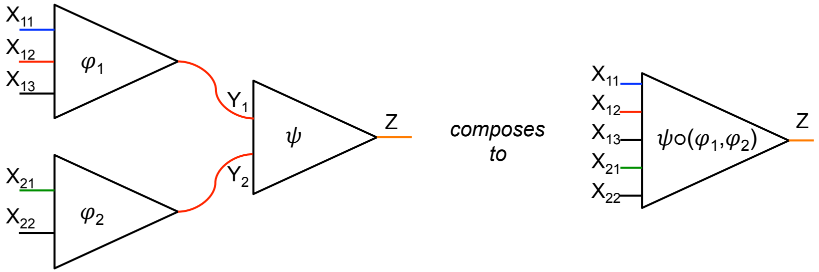
- For each object $X \in \mathrm{Ob}(\mathcal{O})$, an identity arrangement $\mathrm{id}_X \colon (X) \to X$
- A composition, or nesting formula, e.g.,

  $$\psi \circ (\varphi_1, \ldots, \varphi_n) \colon (X_{i,j}) \xrightarrow{\varphi_i} (Y_i) \xrightarrow{\psi} Z.$$

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

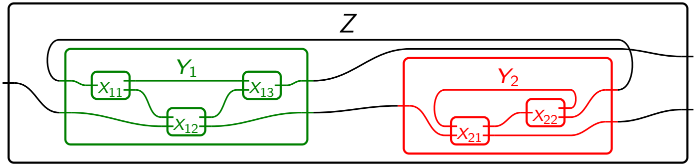
Another way to see it

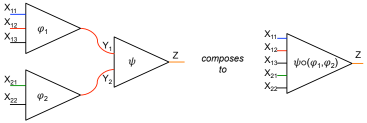
- Often the objects in operad are shown as colors.
- The morphisms are many-input, one-output relationships.
- They can be composed:

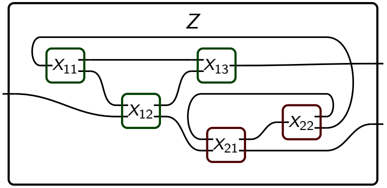


- Here, ψ represents an arrangement of a Y_1 and a Y_2 to make a Z.

Example: composition of networks







Outline

1 Introduction

2 Recognizing a connection

3 Defining operads

4 Applications of operads
- Potential domains of application
- Materials architecture

5 Networks of networks

6 Conclusion

Potential domains of application

Operads might organize how we think about a variety of applied problems:

- Potential applications to:
  - Manufacturing processes,
  - Signaling networks in systems biology,
  - Neural circuits.
- A successful collaboration: applying operads in materials science.
- Plan for remainder of talk:
  - We'll switch gears and discuss the materials case in some detail.
  - Then we'll wind down with networks.

A problem in materials design

- In materials science, one tries to fabricate new better materials.
- An emerging paradigm in materials design: control at all levels.
  - Old idea: take known macro-materials and combine them in new ways.
  - New idea: design from the ground up, fine-tuning at all levels.
- Hierarchical protein materials offer the ability to do that.
  - Proteins are machines that do every task in your body.
  - They also form your waterproof breathable stretchable skin.
  - We can control the amino acid sequences using genetic engineering.
  - By controlling the structure at all levels, we get fine-tuned results.
- The problem is, we have no idea what we're doing.
  - Through experimental trial and error, we slowly learn how things work.
  - Recently, people are simulating protein materials to learn faster.

We designed a tool called Matriarch

- The process for simulating hierarchical protein materials is tedious.
  - Because it's such a new field, there is a lack of organization.
  - People program amino-acid placement by hand.
  - Compromise equilibration-time efficiency for programming efficiency.
- We developed a tool for creating hierarchical protein materials.
- It is called *Matriarch*, standing for materials architecture.
- And (of course) it is based on operads.

The operadic model of Matriarch

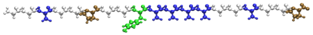
Let $\mathcal{M}$ denote the operad for Matriarch.

- The objects (building blocks) in $\mathcal{M}$ are proteins.
    - These start with amino acids, but include everything you can build.
    - They are differentiated according to their bondable interface.
- The morphisms (building instructions) in $\mathcal{M}$ are commands such as:
    - 1-ary: `reverse`, `rigidMotion`, `twist`,
    - 2-ary: `attach`, `space`, `overlay`,
    - *n*-ary: `makeArray`, `attachSeries`, `spaceSeries`.
    - Compositions: `helix`, `collagen` — these are nested operations.
- The composition (nesting) is straightforward.
    - You keep building materials of higher and higher complexity.
    - And then putting the results together (using the above commands).
    - The result is a new building block of higher complexity.

Sample architectures

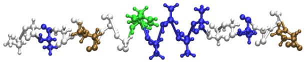
a

Strand1 = chain(seq1)



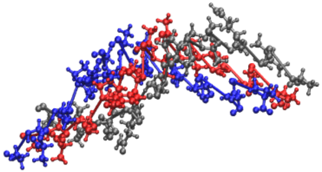
b

Hel1 = helix(Strand1, 1.0, 5.0)



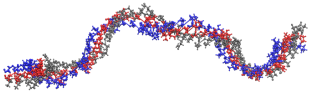
c

TH = collagen(Strand1, Strand2)



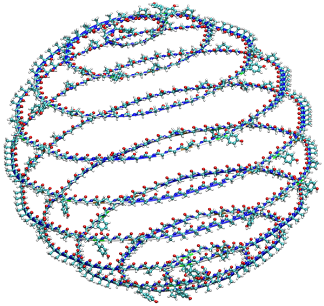
d

Worm = twist(attachSeries(TH,5), W)



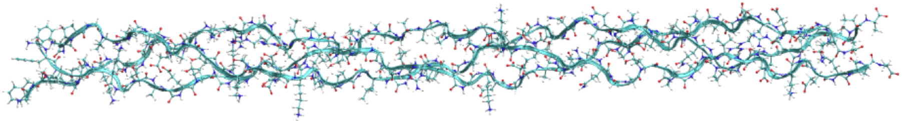
e

Apple = twist(Strand3, SSFunc)



Example of materials architecture: collagen

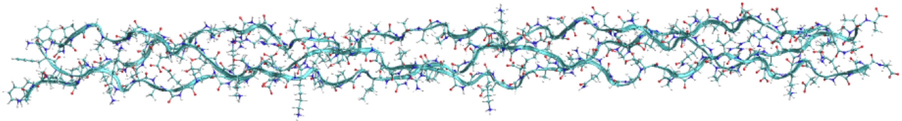
- Collagen is the most common protein in mammals.
- Its design is hierarchical.



```
a1             =   chain(seq1)
a2             =   chain(seq2)
hel1           =   helix(a1, rad=1.5, pitch=9.5, handed=L)
hel2           =   helix(a2, rad=1.5, pitch=9.5, handed=L)
helhel1        =   helix(hel1, rad=4, pitch=85, handed=R)
helhel2        =   helix(hel2, rad=4, pitch=85, handed=R)
helhel1rot     =   rigidMotion(helhel1, rotate=120, shift=2.8)
helhel2rot     =   rigidMotion(helhel2, rotate=240, shift=-5.6)
tropocollagen  =   overlay(helhel1, helhel1rot, helhel2rot)
collagen       =   makeArray(tropocollagen,1000,1000,distance=8.1)
```

Example of materials architecture: collagen

- A fibril of collagen is an array of tropocollagen molecules.
- Each molecule of tropocollagen is a right-handed triple helix.
- Each of its three strands is a left-handed helix.
- Each of these individual helices is a chain of many amino acids.



```
a1              =   chain(seq1)
a2              =   chain(seq2)
hel1            =   helix(a1, rad=1.5, pitch=9.5, handed=L)
hel2            =   helix(a2, rad=1.5, pitch=9.5, handed=L)
helhel1         =   helix(hel1, rad=4, pitch=85, handed=R)
helhel2         =   helix(hel2, rad=4, pitch=85, handed=R)
helhel1rot      =   rigidMotion(helhel1, rotate=120, shift=2.8)
helhel2rot      =   rigidMotion(helhel2, rotate=240, shift=-5.6)
tropocollagen   =   overlay(helhel1, helhel1rot, helhel2rot)
collagen        =   makeArray(tropocollagen,1000,1000,distance=8.1)
```

Matriarch as a design tool

$$\texttt{attachSeries}\Big(\texttt{helix}(\textit{seq},\ \text{rad}{=}4,\ \text{pitch}{=}85), \texttt{copies} = 10\Big)$$

- We already said:
  - With Matriarch, it is easy to adjust protein material architecture.
  - Equilibration times are drastically reduced.
  - The equilibration is controlled: no wrong foldings.
- Just as important: The result is a human-understandable structure.
  - A set of descriptive commands to synthesize the material.
  - "Carve nature at its joints."
  - This, instead of a list of atomic coordinates, or a prose description.
  - Provides a good position from which to consider material design.
- Note: this includes parametric design, but not limited to it.
  - One optimizes a given product ("what's the best seq, rad, pitch?")
  - But hierarchical continuation is key: use it as a part in a bigger whole.

What did operads really do for us?

- Operads provided a design framework.
  - The Matriarch operad served as software specification for the program.
  - It efficiently translated user requirements into functional requirements.
  - Later change requests were easy to implement: the formalism is flexible.
- The result was a systematic approach to materials design.
  - Users can work with materials at multiple scales.
  - Once you've produced something, you can use it as a building block.

Outline

1 Introduction

2 Recognizing a connection

3 Defining operads

4 Applications of operads

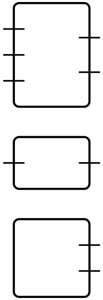
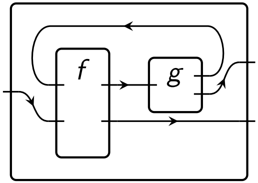
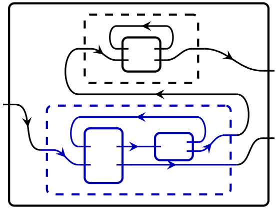
5 **Networks of networks**
- A zoo of operads
- Different wiring diagram operads
- Semantics of wiring diagrams

6 Conclusion

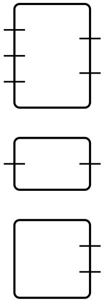
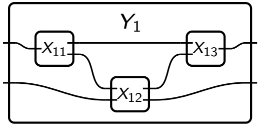
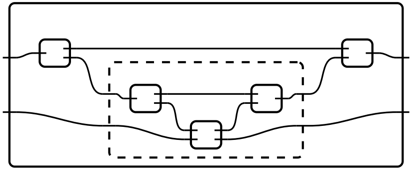
A zoo of operads

- There's a whole zoo of operads—very different animals.
  - The operad of networks looks pretty different from that of materials.
  - One involved wiring diagrams, the other involved `attach` and `twist`.
- The reason is that operads are just the rules of modularity.
  - If you can tell me your interfaces, arrangements, and nesting,
  - you probably have an operad.
  - Modularity is a very general phenomenon; it takes on many forms.
- Even just for wiring diagrams, there's a sub-zoo.

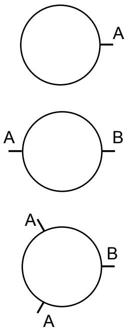
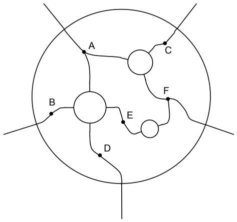
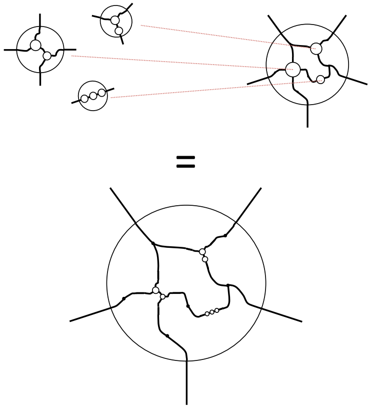
Directed wiring diagrams are modular

Interfaces	Arrangements	Nesting
		

And another: wiring diagrams without feedback

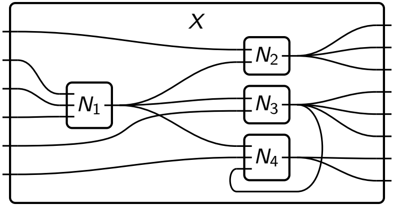
Interfaces	Arrangements	Nesting
		
		(Getting a sense of how fractals are a special case?)

Another modular notion of wiring diagram

Interfaces	Arrangements	Nesting
		

Semantics of wiring diagrams

- Each of these different wiring diagram types is its own operad.
- Different operads offer different semantics.
- Here's one that looks like neural pathways. It has splitting wires.



- If each box N_i represents a dynamical system, you can derive one for X.
    - So, if you have a model for neurons, you get one for communities.
    - You can create building blocks of community behaviors.
    - If you squint, you can see the similarity with the Matriarch program.

Outline

1 Introduction

2 Recognizing a connection

3 Defining operads

4 Applications of operads

5 Networks of networks

6 Conclusion

Conclusion

- Somehow, the human brain handles a huge range of problems.
  - Planning a wedding or a space mission.
  - Assembling Ikea furniture or architecting a house.
  - Understanding societies, or individual biology or psychology.
- In each case, the understanding comes from putting pieces together.
- There is a certain principle at work across many scales and domains.
  - Each system emerges out of interactions among its parts.
  - Parts can be chunked into sub-systems, which are again parts.
- Operads provide a language in which to consider such issues.

Thank you!