A category-theoretic approach to materials design

David I. Spivak

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

Presented on 2015/01/26
at the 8th Design Theory SIG Paris Workshop
Session: Mathematics and Design Theory

Informatics for design

- To understand politics, follow the money.
- To understand design, what should you follow?
- It has been proposed that you should follow the information.
  - What is the structure of the information being used?
  - How is information being combined?
  - How is information being communicated and translated?
  - How is information guiding decisions?
- What kind of "accounting" is needed to follow the information?
  - To record the flow of money we use a spreadsheet.
  - How do we record the flow of information through a design process?
  - How to do this at scale?

Mathematical informatics

- Mathematical informatics is the mathematics of information structures.
  - Databases,
  - Architecture plans,
  - Grammars,
  - Programming languages.
- Of all mathematical fields, **category theory** seems to fit best.
- So we beautify the name and consider *Categorical Informatics*.

Example: Mathematical conception of a database

- Databases are information-bearing structures.

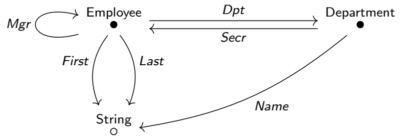
Employee				
Id	**First**	**Last**	**Mgr**	**Dpt**
101	David	Hilbert	103	q10
102	Bertrand	Russell	102	x02
103	Alan	Turing	103	q10

Department		
Id	**Name**	**Secr**
q10	Sales	101
x02	Production	102

- The *structure* of a database is: a *graph with path equations*, $\mathcal{G}$.

Employee.*Mgr*.*Dpt*=Employee.*Mgr*
Department.*Secr*.*Dpt*=Department

$\mathcal{G}$:=



- The data itself, as well as queries, etc., have a compatible formalism.

Toward a proper science of design

In any special doctrine of nature there can be only as much proper science as there is mathematics therein. –Immanuel Kant

- Mathematical theories of design
  - Several have been proposed, e.g., topology CDP, set theory C-K.
  - Could a theory of design be *verified* mathematically (as in physics)?
- Proposal: use categorical informatics.
  - Formalize information itself as a mathematical object (think databases).
  - Follow the information through a given design process.
  - Roughly, I'm offering a mathematically-verifiable description of K.

Categorical informatics

- Of all math fields, why is category theory best suited for informatics?
  - CT was invented to build bridges between mathematical fields.
  - Since its invention in 1940, it has civilized much of mathematics.
  - Topology (orig.), algebra (Grothendieck), logic & set theory (Lawvere)
  - It has moved into computer science, linguistics, and physics.
- Category theory is the mathematics of structures.
  - Sets, vector spaces, algebraic equations: all are structured in this sense.
  - Layering structure, and caring for its interoperability is what CT is for.

High-level view of the talk

- I will discuss an abstract perspective on modular design.
  - Modular pieces: *building blocks*.
  - Methods for combining them: *building instructions*.
  - Put these together to get a *modular design environment*.
  - Operads: a category-theoretic framework for modular design.
- I will present one concrete application of operads in modular design.
  - The modular design environment of "hierarchical protein materials".
  - This is just the first finished application of the basic idea.
  - Please keep in mind how it can generalize to other environments.
- I will say how operads may be used to store artifact theories.

What do I mean by artifact theory?

By *artifact theory*, I mean the following.

- As designers create new structures, they use information.
  - How smaller pieces can be put together to form complex structures.
  - How behavioral bounds on small lead to behavioral bounds on large.
- The former is an understanding of architectural design patterns.
- The latter is experience with structure-function relationships.
- Designers also create new information.
  - They may discover new architectural design patterns.
  - They may discover new structure-function relationships.
  - Through experimentation, new theories of "how things work" emerge.

The artifact theory is the evolving knowledge held by the design team.

Talk outline

For the remainder of this talk, I will discuss:

- A project we did in materials science: software called *Matriarch*.
  - Matriarch stands for Materials Architecture.
  - How Matriarch helps with the materials design process.
  - Joint work with Tristan Giesa, Ravi Jagadeesan, and Markus Buehler.
- Category theory in pictures.
  - Focus on *operads*, a framework within CT, useful for design.
  - How the materials science picture generalizes to other design problems.
- An organizational framework for artifact theories.
  - How to organize information for designers to use in their process?
  - A systematic mathematical approach to organizing this information.
  - E.g., querying the database: *"I want to build something with properties XYZ. What are my options?"*

Materials design

- What we want: High-quality, environmentally-friendly materials.
  - Old idea: high-quality macro requires high-quality micro.
  - New idea: high-quality macro is achievable with cheap, abundant micro.
- Examples: silk, collagen.
  - These materials are made by animals by eating widely-available food.
  - The micro is cheap and abundant, but the result has excellent qualities.
  - Silk is stronger than steel; collagen is used in bone, skin, cartilage, etc.
  - These materials are assemblies of simple (amino acid) building blocks.
- How to mimic these amazing materials and fit them to our needs?
  - In the wet lab, you investigate their hierarchical structures.
  - If you need to modify something, you'll want to use computers.

Computational modeling

- A new 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.
- This requires a massive amount of computation.
  - You can't do all this in a wet lab.
  - Simulation allows you to play around with micro-structures.
  - "This amino acid is preventing what you want; can we get rid of it?"
  - Molecular dynamics (MD) simulators are used to run experiments.
- The current state of computational materials design.
  - There does not currently exist a general tool to create new microstructures.
  - You have to do everything (place atoms and bonds) by hand.
  - This is extremely tedious, and leads to problematic work-arounds.

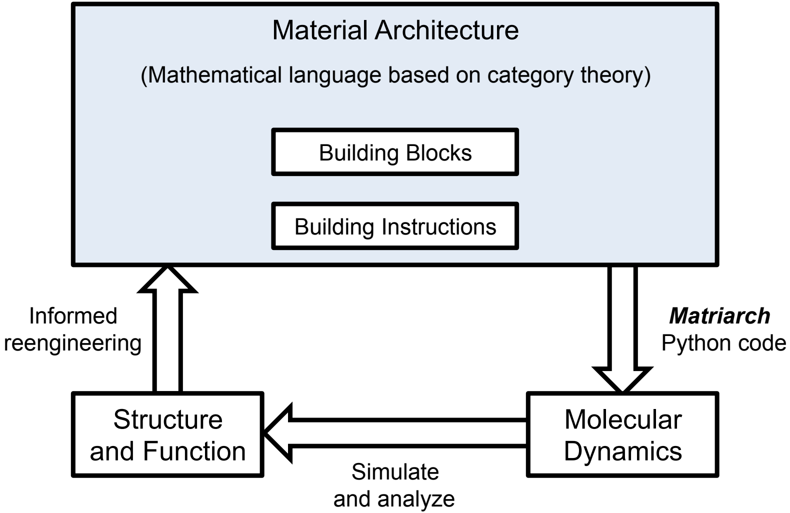
The challenges to overcome

- The dilemma: spend time programming or equilibrating??
  - If you want to save labor time, you place atoms into straight chains.
  - But these take forever to equilibrate (settle into place).
  - Moreover, they may equilibrate to the wrong shape (local minimum).
- Computational modelers develop tricks.
  - Tiny pieces of code (scripts) that work for what they need now.
  - These scripts are difficult to share, reuse, and explain.
- All these problems can be solved simultaneously.
  - Make a language to synthesize hierarchical structures.
  - Atoms placed near their final positions reduces equilibration time.
  - It is much easier to communicate in this language than in scripts.

Materials architecture

- What is materials architecture?
  - Building blocks: proteins, from amino acids to collagen.
  - Building instructions: forming new bb's out of old.
  - Hierarchical materials are built by combining these into programs.
- In general, an modular design environment will be
  - Your set of building blocks,
  - Your set of building instructions.
  - The "space" of architectures that can be assembled in this way.

Summary of the Matriarch design process



Building blocks and building instructions in Matriarch

Building blocks:

- 20 standard amino acids, plus proline (for creating collagen).
- Users can import their own building blocks from PDB.

Building instructions:

- `attach`,
- `space`,
- `overlay`,
- `reverse`,
- `rigidMotion`,
- `twist`,
- `makeArray`.

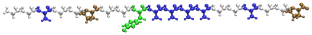
Matriarch programs:

- Any combination of building instructions applied to building blocks

Sample architectures

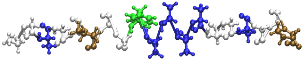
a

Strand1 = chain(seq1)



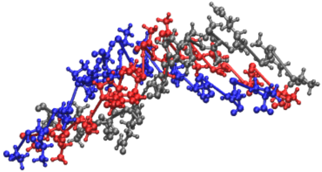
b

Hel1 = helix(Strand1, 1.0, 5.0)



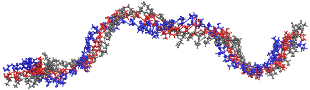
c

TH = collagen(Strand1, Strand2)



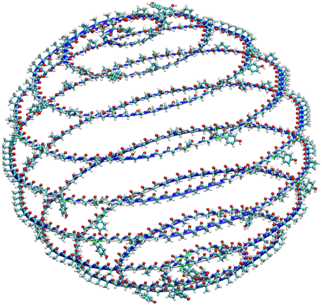
d

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



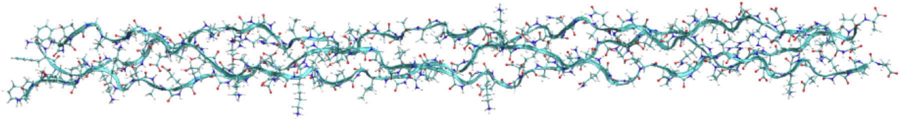
e

Apple = twist(Strand3, SSFunc)



Example of materials architecture: collagen

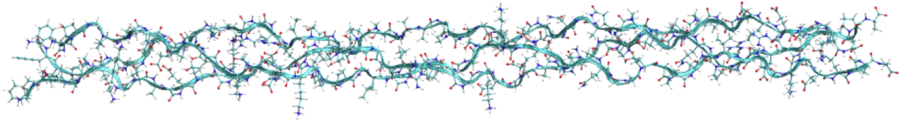
- Collagen is the most common protein in mammals.
- Its design is hierarchical.
  - A fibril of collagen is an array of tropocollagen molecules.
  - Each tropocollagen molecule is a right-handed triple helix.
  - Each of its three strands is a left-handed helix.
  - Each of these individual helices is 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)

Materials architecture

- A fibril of collagen is an array of tropocollagen molecules.
- Each tropocollagen module 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

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

- 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.
  - It provides a better position from which to build an artifact theory.
- 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's this got to do with category theory?

- We said Matriarch was built "using" category theory.
- But why is CT necessary?
- Isn't the Matriarch idea fairly simple and intuitive?
- Answer: category theory led to this intuitive design.

Category theory was the software specification

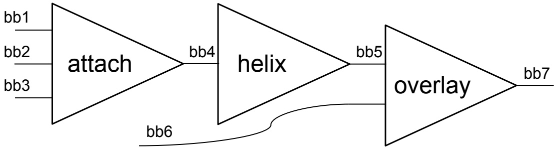
- We first understood the architecture problem using *operads*.
- Here's what to think of when you hear the word "operad":
    - Building blocks,
    - Building instructions,
    - Guaranteed equivalences between different programs you can create.

$$\texttt{reverse}(\texttt{attach}(\texttt{x},\texttt{y})) = \texttt{attach}(\texttt{reverse}(y),\texttt{reverse}(x))$$

- The code then followed the mathematical (operadic) specification.
    - Goguen proposed category theory as a software specification language.
- Note: you don't need to know operads to work with Matriarch.

Explaining operads (a very rough sketch)

- What is the relation between operads and category theory?
  - Analogy: the relation between line integrals and calculus.
  - They are important for any expert to know, but it's just one piece.
  - They are useful for certain problems.
  - They are a generalization of the founding idea.
- Categories are worlds of things and connections from one to another.
- Operads are worlds of things and connections from many to another.
  - A Matriarch command uses many, say X_1, X_2, X_3, to build one Y.



How operads are useful in design

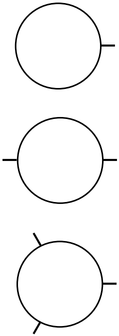
- Operads can be thought of as the language of assembly.
  - Or, modular design environments.
- As said above, an operad $\mathcal{O}$ consists of:
  - a set of building blocks,
  - a set of building instructions,
  - a set of guaranteed equivalences between different programs, e.g.,

$$\texttt{reverse}(\texttt{attach}(\mathtt{x},\mathtt{y})) = \texttt{attach}(\texttt{reverse}(y),\texttt{reverse}(x))$$

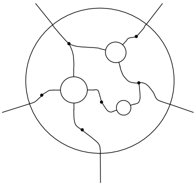
- It is a language for thinking about building complex from simple.

Wiring diagrams type 1: interconnected cells

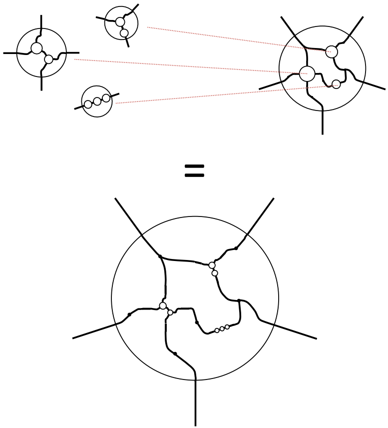
Building blocks



Building instructions

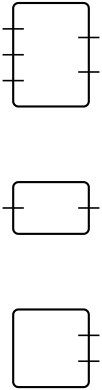


Programs

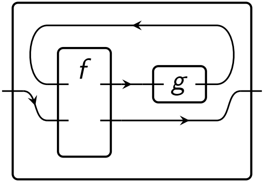


Wiring diagrams 2: processes

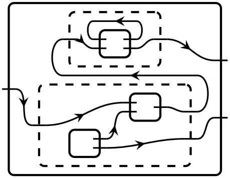
Building blocks



Building instructions



Programs



Operad mappings

- Recall from above:
  - Matriarch uses an operad $\mathcal{M}$ to encodes material architecture.
  - Other operads are useful for other design spaces, e.g., process design.
  - Still other operads are useful in pure math, e.g., **Set**.
- The last math we should discuss today: *Operad mappings*.
  - Operad mappings allow us to translate between design spaces.
  - Example: the *processes* operad maps to the *cells* operad.
  - Operad mappings allow us to change the operad and predict its effect.
  - Operad mappings allow us to connect into the mathematical universe.
- What is an operad mapping $\mathcal{M} \to \mathcal{N}$? It is a formula, which
  - sends building blocks in $\mathcal{M}$ to building blocks in $\mathcal{N}$,
  - sends building instructions in $\mathcal{M}$ to building instructions in $\mathcal{N}$,
  - ensures that all guarantees in $\mathcal{M}$ also hold in $\mathcal{N}$.
- We will use operad mappings soon, but let's get back to the point.

How operads may further support design

Where are we now?

- I have discussed operads in general.
- I have also shown our specific application: Matriarch.

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

- Matriarch aids the materials designer by:
  - speeding up equilibration times,
  - providing a language for creating and thinking about structures.
- In the future, I want it to hold the *artifact theory* for materials.

Artifact theories as information structures

- Designers typically use diverse information sources to solve problems.
  - First-hand knowledge about what works.
  - Discussions with other designers, engineers, and users.
  - Catalogs of information on relevant subjects.
  - Together, all this defines their artifact theory, and it evolves.
- My goal is to find the structure that this information likes to live in.
  - By what scheme should the artifact theory be arranged?
  - What is the *mathematical shape* of this information?
- I believe that operads provide this structure, this information-shape.

Operads and artifact theories

How to understand the evolving artifact theory mathematically?

- Suppose given an operad $\mathcal{O}$ for a design space.
- $\mathcal{O}$ encodes the known building blocks and building instructions.
- What if we want to add a new building block or building instruction?
  - This is represented by a new, slightly bigger operad $\mathcal{O}'$.
  - And, importantly, an operad mapping $\mathcal{O} \to \mathcal{O}'$ comparing them.
  - The mapping allows import of old knowledge to new structure.
- But how does the knowledge *about* building blocks tie in?
  - For each building block x, designers consider certain metrics.
  - For example, they may consider strength and toughness of materials.
  - They consider how building instructions affect the metrics.
- I think the knowledge can be stored in an operad mapping $\mathcal{O} \xrightarrow{K} \mathbf{Set}$.
  - K holds the set of metric values (strength=3, toughness=5) for blocks
  - and our knowledge about the affect of building instructions on metrics.

What we can hope for

Here is a possible future design environment.

- There is a set of building blocks and building instructions.
  - This is formalized by a mathematical object called an operad $\mathcal{O}$.
  - But like Matriarch, no one has to know that.
  - The operad $\mathcal{O}$, as well as all math written below, sit in the background.
- Designers use $\mathcal{O}$ to create custom architectures.
  - Building blocks, building instructions provide a well-formed language.
- The knowledge $K\colon \mathcal{O} \to \mathbf{Set}$ is like the state of a database.
  - Predict behavior of new designs using knowledge $K\colon \mathcal{O} \to \mathbf{Set}$.
  - It can be queried: *"I want to build something with properties XYZ. What are my options?"*
  - Update the operad $\mathcal{O}$ or the knowledge K through experimentation.

Summary

- One way to study design is to consider how designers use information.
  - With a mathematical underpinning, this study can be more scientific.
- I discussed Matriarch and operads.
  - Matriarch allows the assembly of novel protein materials.
  - These can be tested in molecular dynamics simulators.
  - An operad encodes a set of building blocks and building instructions.
  - These can be used to assemble arbitrarily complex architectures.
- Mappings between operads will allow us to:
  - Communicate between different design teams;
  - Import knowledge from other design problems;
  - Add new building blocks or instructions in a given design space;
  - Add to our given knowledge about structure-function relationships.

Thanks for the invitation to speak!