Compositional analyses of dynamical systems

FA9550-14-1-0031

David I. Spivak

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

Outline

E pluribus unum

Composition

Composition is assembling many things together to make one thing.[1]

[1] Color scheme and category-theoretic formalism:
"thing" = *object*, "arrangement" = *morphism*, "nesting" = *composition*.

Composition

Composition is assembling many things together to make one thing.[1]

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

[1] Color scheme and category-theoretic formalism:
"thing" = *object*, "arrangement" = *morphism*, "nesting" = *composition*.

Composition

Composition is assembling many things together to make one thing.[1]

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

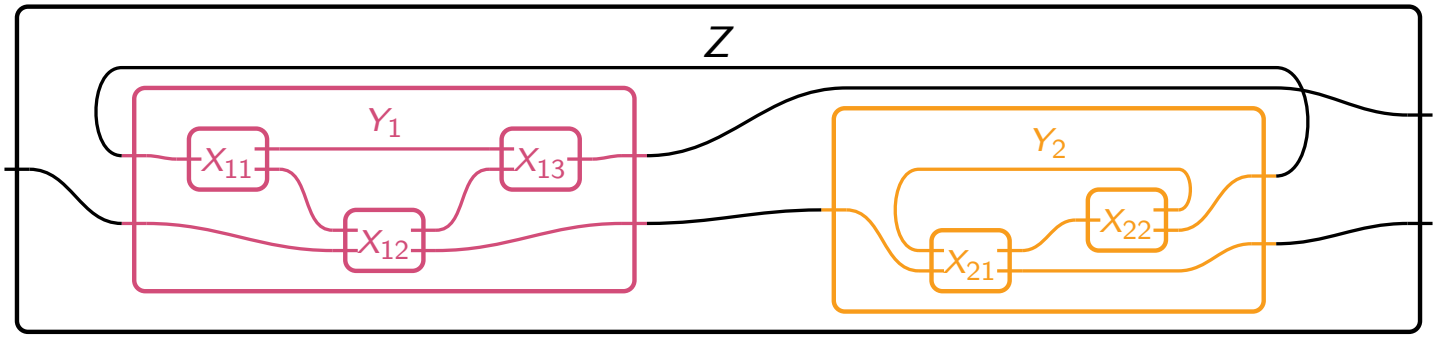
Let's give two very different examples, to see the scope.

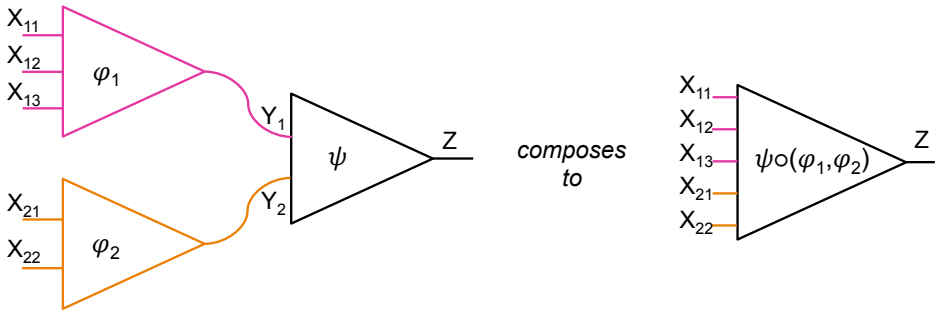
[1] Color scheme and category-theoretic formalism:
"thing" = *object*, "arrangement" = *morphism*, "nesting" = *composition*.

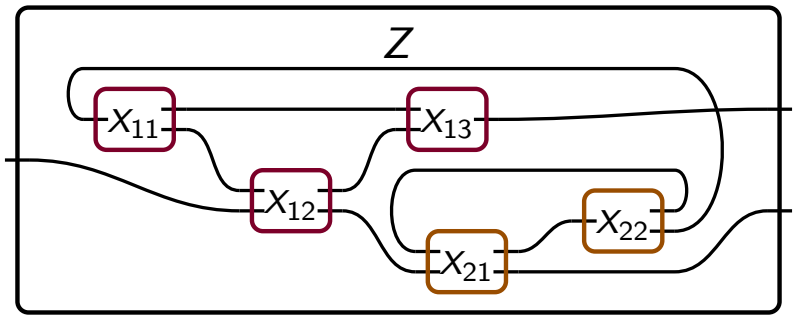
Arrangements 1: wiring diagrams

There is an operad $\mathcal{D}$ for composing hierarchical dynamical systems.

- Note the things, arrangements, and nesting here.







Arrangements 2: hierarchical protein materials

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

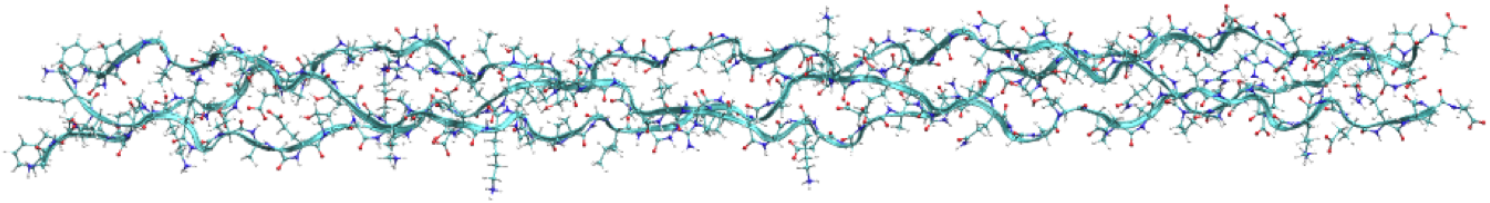
- A protein is an arrangement of simpler proteins.
  - There are "atomic" proteins, namely amino acids.
  - Protein materials include your skin: stretchable, breathable, waterproof.[2]

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

Arrangements 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.[2]
- 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.



- (Computer MD versions of) proteins model this operad.

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

Plan: compositional analyses of dynamic systems

So composition—one thing out of many—can be formalized by operads.

- There's also a mathematical definition of *compositional analysis*.
  - It's something we can compute, but which also composes well:
  - Component-level analyses can be combined to provide a system-level analysis.

Plan: compositional analyses of dynamic systems

So composition—one thing out of many—can be formalized by operads.

- There's also a mathematical definition of *compositional analysis*.
  - It's something we can compute, but which also composes well:
  - Component-level analyses can be combined to provide a system-level analysis.
- Why? Compositional analysis is future-proof analysis.
  - Example: *Average* is not compositional; *sum* and *count* are.
  - Always expect your machine to be used in a larger system.

Plan: compositional analyses of dynamic systems

So composition—one thing out of many—can be formalized by operads.

- There's also a mathematical definition of *compositional analysis*.
  - It's something we can compute, but which also composes well:
  - Component-level analyses can be combined to provide a system-level analysis.
- Why? Compositional analysis is future-proof analysis.
  - Example: *Average* is not compositional; *sum* and *count* are.
  - Always expect your machine to be used in a larger system.
- I will give several examples of compositional mappings and analyses.
  - Discretization of dynamical systems is compositional.
  - Steady state analyses are compositional.
- I'll also explain a few other compositional frameworks.
  - A new way of solving systems of non-linear equations.
  - A mathematical theory of collaborative design.

Outline

E pluribus unum

Operads and algebras: the syntax and semantics of composition

There is a category-theoretic object, called an operad.[3]

- Operads are a framework for composing systems of systems.
- Operads are a sort of syntax; their semantics are called algebras.
- Or think operad=interface logic; algebra=inhabitants.
- Some examples of compositional structures that operads can model:
  - Every recipe—a sequence of smaller recipes—is an operad. (Performance plans might constitute an algebra for it.)

[3] Using a new color scheme: operad=syntax, algebra=semantics.

Operads and algebras: the syntax and semantics of composition

There is a category-theoretic object, called an operad.[3]

- Operads are a framework for composing systems of systems.
- Operads are a sort of syntax; their semantics are called algebras.
- Or think operad=interface logic; algebra=inhabitants.
- Some examples of compositional structures that operads can model:
  - Every recipe—a sequence of smaller recipes—is an operad.
    (Performance plans might constitute an algebra for it.)
  - Every context-free grammar is an operad.
    (Attribute grammars might constitute an algebra for it.)

[3] Using a new color scheme: operad=syntax, algebra=semantics.

Operads and algebras: the syntax and semantics of composition

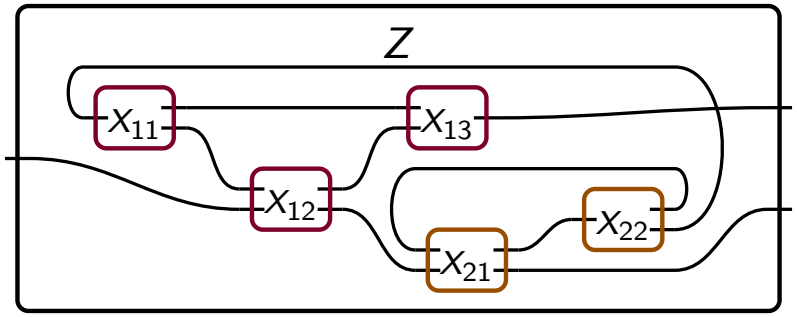
There is a category-theoretic object, called an operad.[3]

- Operads are a framework for composing systems of systems.
- Operads are a sort of syntax; their semantics are called algebras.
- Or think operad=interface logic; algebra=inhabitants.
- Some examples of compositional structures that operads can model:
  - Every recipe—a sequence of smaller recipes—is an operad.
    (Performance plans might constitute an algebra for it.)
  - Every context-free grammar is an operad.
    (Attribute grammars might constitute an algebra for it.)
  - Every sort of wiring diagram, as in electric circuits or signal flow.
    (We'll discuss several different algebras for wiring diagrams.)
- I'll describe operads of wiring diagrams next.

[3] Using a new color scheme: operad=syntax, algebra=semantics.

Multiple interpretations for the same operad

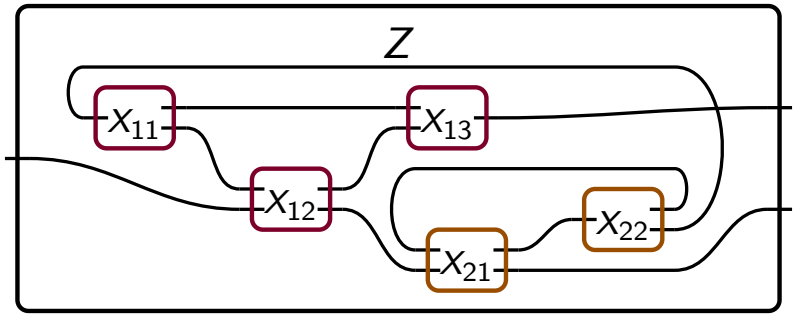
There is an operad of wiring diagrams.



But there are many different semantics for this operad.

Multiple interpretations for the same operad

There is an operad of wiring diagrams.

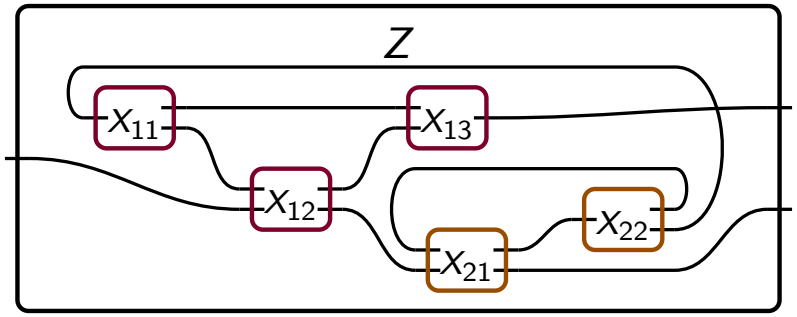


But there are many different semantics for this operad.

- Continuous dynamical systems compose according to these pictures.
- Discrete dynamical do too. Hybrid systems do too.

Multiple interpretations for the same operad

There is an operad of wiring diagrams.



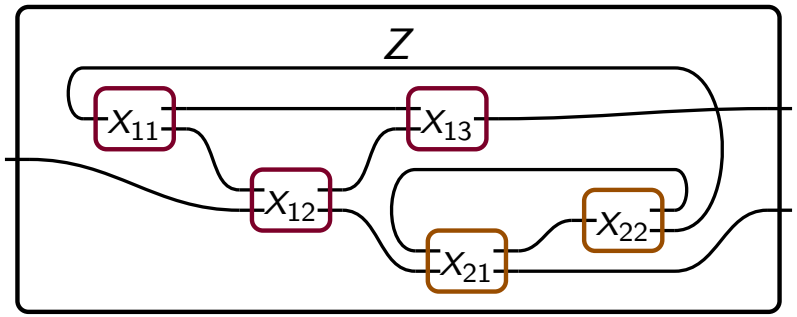
But there are many different semantics for this operad.

- Continuous dynamical systems compose according to these pictures.
- Discrete dynamical do too. Hybrid systems do too.
- Mathematicians recognized long ago that matrices (tensors) do too.
  - Sequential composition is "matrix multiplication".
  - Parallel composition is "Kronecker product".
  - Feedback is the block-trace operator.

Each of these semantics interpretations is formalized as an algebra.

Compositional mappings: translating between algebras

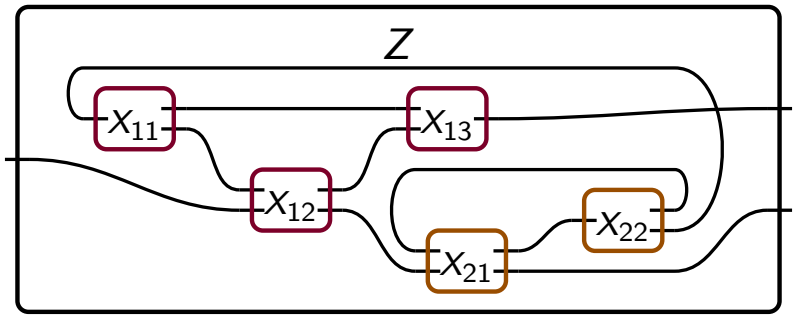
Suppose A and B are both algebras for the same operad $\mathcal{O}$.



- For example, A=continuous dynamics, B=discrete dynamics.

Compositional mappings: translating between algebras

Suppose A and B are both algebras for the same operad $\mathcal{O}$.

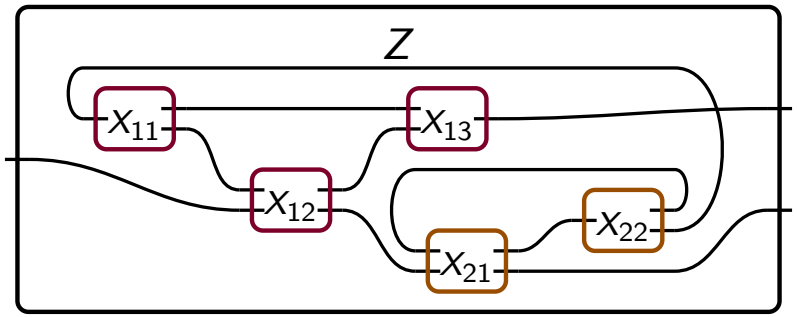


- For example, A=continuous dynamics, B=discrete dynamics.
- A comp'l mapping (algebra morphism) $A \to B$ does the following:
  - For each box, it translates each A-inhabitant to a B-inhabitant.
  - For each wiring diagram, it satisfies an equation:

translate-then-compose = compose-then-translate.

Compositional mappings: translating between algebras

Suppose A and B are both algebras for the same operad $\mathcal{O}$.



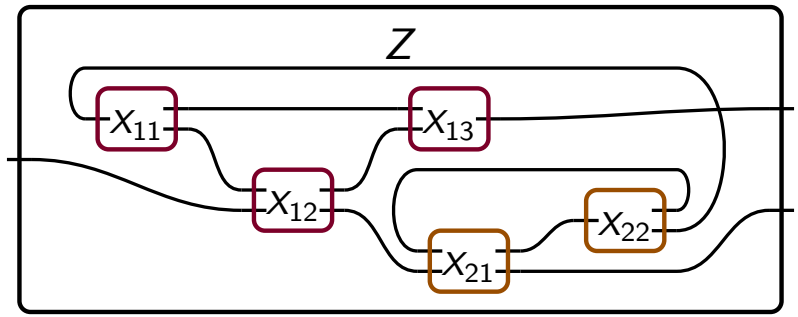
- For example, A=continuous dynamics, B=discrete dynamics.
- A comp'l mapping (algebra morphism) $A \to B$ does the following:
  - For each box, it translates each A-inhabitant to a B-inhabitant.
  - For each wiring diagram, it satisfies an equation:

translate-then-compose = compose-then-translate.

Question: is "discretizing a continuous dynamical system" compositional?

Discretization is compositional

Rephrasing the question, suppose there is a continuous DS in each box:



You have the following choices; what's the difference between them?

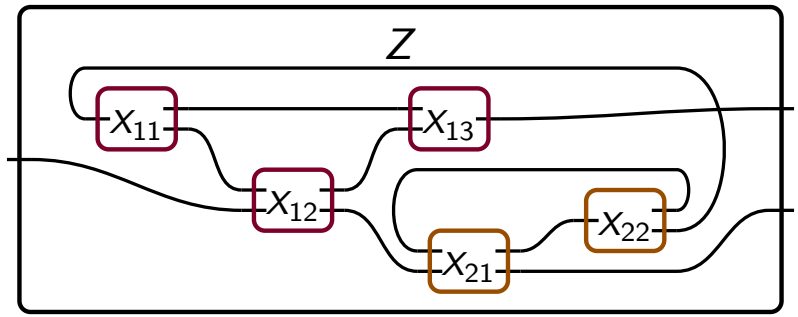
- Choice 1: compose the continuous systems and discretize the result;
- Choice 2: discretize each and then compose the discrete systems.

[4] Spivak "The steady states of dynamical systems". *arXiv* 1512.00802.

[5] Ngotiaoco "Compositionality of the Runge-Kutta Method". *arXiv* 1707.02804.

Discretization is compositional

Rephrasing the question, suppose there is a continuous DS in each box:



You have the following choices; what's the difference between them?

- 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.

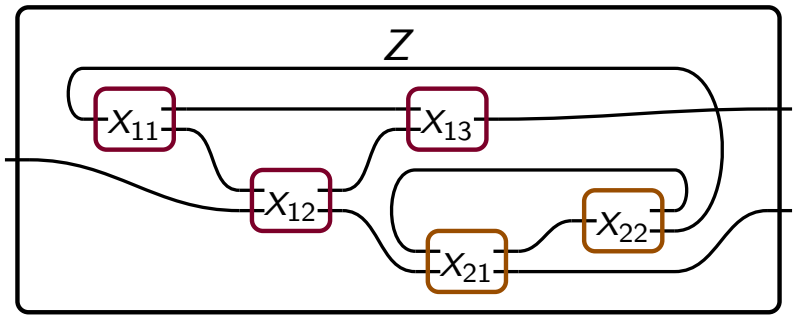
- This requires proof: arbitrary wiring diagrams, arbitrary DS's.
- I proved it for Euler;[4] my student proved it for Runge Kutta.[5]

[4] Spivak "The steady states of dynamical systems". *arXiv* 1512.00802.

[5] Ngotiaoco "Compositionality of the Runge-Kutta Method". *arXiv* 1707.02804.

Bifurcation diagrams are compositional

Suppose there's a DS (cont/disc/hybrid) in each box.

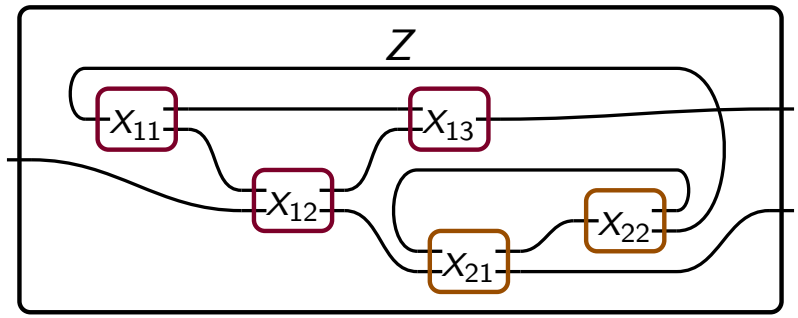


To each box, associate the "bifurcation diagram" of the DS.

- It's the n-dimensional plot of inputs/outputs that have a steady state.
- Pixelate it to form an n-dimensional tensor.
- We said that n-dim'l tensors are a semantics for wiring diagrams.

Bifurcation diagrams are compositional

Suppose there's a DS (cont/disc/hybrid) in each box.



To each box, associate the "bifurcation diagram" of the DS.

- It's the n-dimensional plot of inputs/outputs that have a steady state.
- Pixelate it to form an n-dimensional tensor.
- We said that n-dim'l tensors are a semantics for wiring diagrams.

"Bifurcation diagrams are compositional": get the same answer whether

- you first compose the DS's and then plot the bifurcation diagram; or
- you first plot the bifurcation diagrams, then compose these tensors.

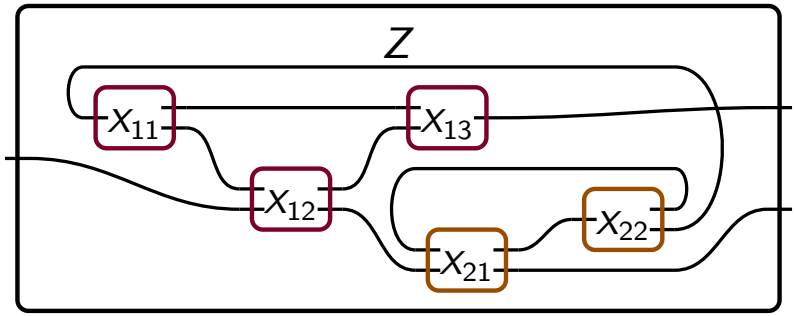
Outline

E pluribus unum

From steady states to systems of equations

Steady states of a DS are solutions to a system of equations, $\dot{\mathbf{x}} = 0$.

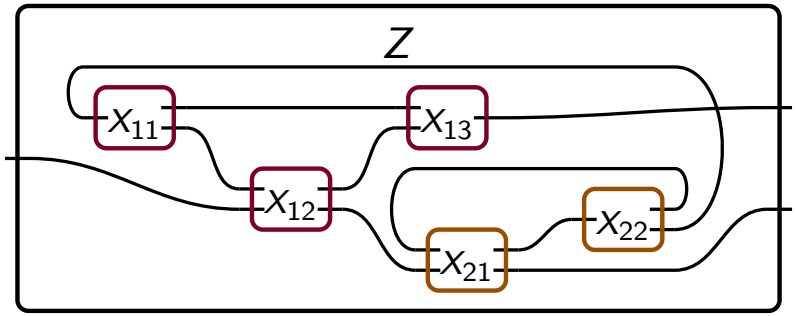
- For interconnected systems, the various systems are combined.



From steady states to systems of equations

Steady states of a DS are solutions to a system of equations, $\dot{\mathbf{x}} = 0$.

- For interconnected systems, the various systems are combined.



- Some variables are shared (as dictated by the wiring diagram).
- So what did "bifurcation diagrams are compositional" stuff tell us?
  - In fact, that was about plotting equations (as dots on a screen).
  - I said that to find steady states, multiply the plots as matrices.
  - Turns out this is really about solving systems of equations.
- So we're solving systems of *non-linear* equations by matrix arithmetic.

Imagining the pixel array method

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

- Plot each in its own bounding box, say in the range $[-1.5, 1.5]$.
- Consider the plots as matrices M, N whose entries are on/off pixels.
- That is, M and N are *boolean matrices* corresponding to f and g.

Imagining the pixel array method

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

- Plot each in its own bounding box, say in the range $[-1.5, 1.5]$.
- Consider the plots as matrices M, N whose entries are on/off pixels.
- That is, M and N are *boolean matrices* corresponding to f and g.

Multiplying these two matrices MN yields...

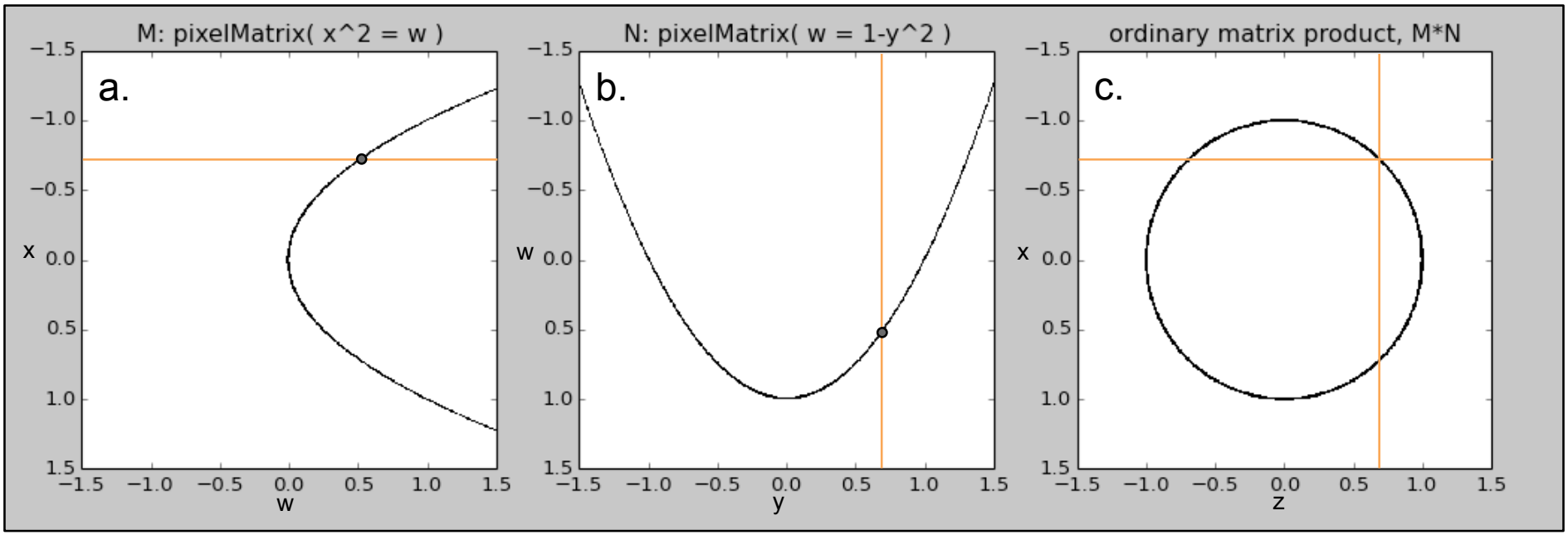
Imagining the pixel array method

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

- Plot each in its own bounding box, say in the range $[-1.5, 1.5]$.
- Consider the plots as matrices M, N whose entries are on/off pixels.
- That is, M and N are *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

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

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

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

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

Q: For what values of w and z does a simultaneous solution exist? [6]

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

A more complex example

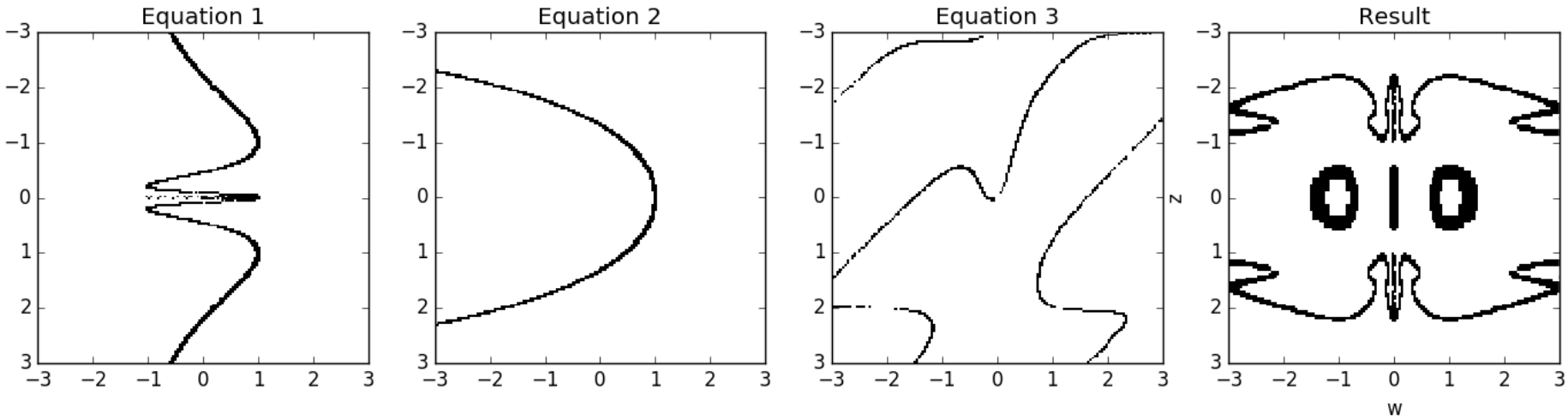
The following eq's are not differentiable, nor even defined everywhere.

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

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

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

Q: For what values of w and z does a simultaneous solution exist? [6]



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

Outline

E pluribus unum

Mathematical co-design

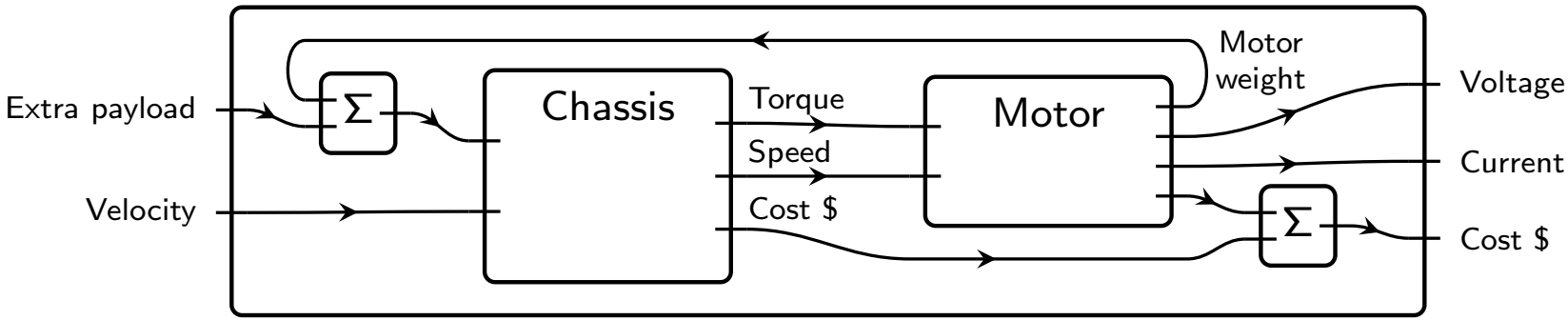
Designing complex systems of systems is collaborative.

- Many teams working on the same project: independent or dependent?
  - Each team's internal decisions have impact on other teams.
  - Feedback loops are prevalent in design.

Mathematical co-design

Designing complex systems of systems is collaborative.

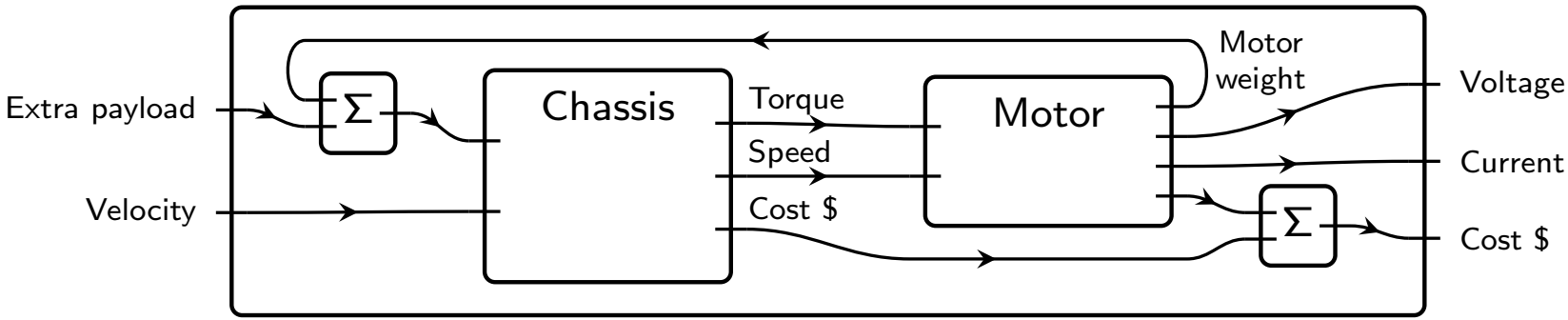
- Many teams working on the same project: independent or dependent?
  - Each team's internal decisions have impact on other teams.
  - Feedback loops are prevalent in design.
  - Example: a motor powers the chassis, which carries the motor.



Mathematical co-design

Designing complex systems of systems is collaborative.

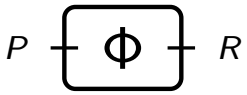
- Many teams working on the same project: independent or dependent?
  - Each team's internal decisions have impact on other teams.
  - Feedback loops are prevalent in design.
  - Example: a motor powers the chassis, which carries the motor.



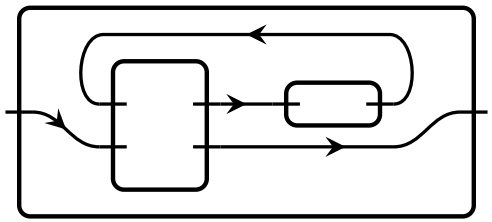
- There is a compositional theory of co-design (due to Andrea Censi).
  - Each box is a design problem.
  - Inside may be another interconnected system of design problems.

The underlying math of co-design

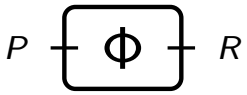
What is a design problem in the underlying math?



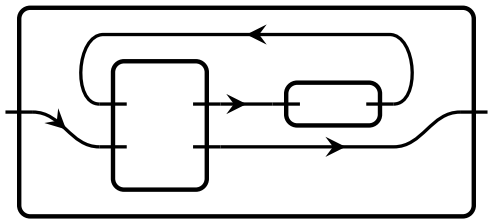
- Design is about feasibility: resources provided vs. resources required.
- Each resource (wire) is modeled as a poset, the order is by preference.
- Feasibility is a monotonic function $\Phi\colon P^{\mathrm{op}} \times R \to \{\mathtt{F}, \mathtt{T}\}$.



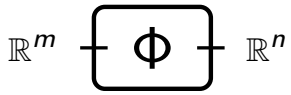
The underlying math of co-design

What is a design problem in the underlying math? 

- Design is about feasibility: resources provided vs. resources required.
- Each resource (wire) is modeled as a poset, the order is by preference.
- Feasibility is a monotonic function $\Phi\colon P^{\mathrm{op}} \times R \to \{\mathtt{F}, \mathtt{T}\}$.

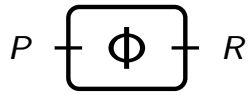


Potential application to reachability analysis for interconnected DS's.

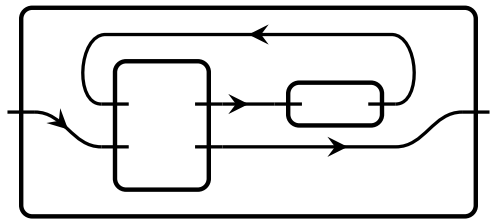
- Imagine an initialized DS in each box. 

- To each wire, associate a poset of open sets U in the manifold.

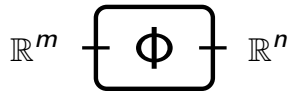
The underlying math of co-design

What is a design problem in the underlying math? 

- Design is about feasibility: resources provided vs. resources required.
- Each resource (wire) is modeled as a poset, the order is by preference.
- Feasibility is a monotonic function $\Phi\colon P^{\mathrm{op}} \times R \to \{\mathtt{F}, \mathtt{T}\}$.



Potential application to reachability analysis for interconnected DS's.

- Imagine an initialized DS in each box. 

- To each wire, associate a poset of open sets U in the manifold.
- To a box, need $\Phi(U_1, U_2) \in \{\mathtt{F}, \mathtt{T}\}$ for each $U_1 \subseteq \mathbb{R}^m$ and $U_2 \subseteq \mathbb{R}^n$.
- Put $\Phi(U_1, U_2) = \mathtt{T}$ iff, by keeping input in U_1, output is kept in U_2.

Combining these in a system, we get the overall reachability relation.

Outline

E pluribus unum

Summary

Operad: a mathematical framework defining a given sort of composition.

- What are your interfaces? What sorts of arrangements are allowed?
- We saw a few operads: mostly wiring diagrams.
- We also briefly touched on operads for proteins, recipes, grammars.

Summary

Operad: a mathematical framework defining a given sort of composition.

- What are your interfaces? What sorts of arrangements are allowed?
- We saw a few operads: mostly wiring diagrams.
- We also briefly touched on operads for proteins, recipes, grammars.

Algebra: the set of inhabitants for interfaces, and the combination rules.

- Focused on dynamical systems; briefly also tensors and co-design.
- Compositional mappings and analyses (discretization, bifurcation).
- Discussed pixel array method for solving systems.
- Mentioned co-design theory, and possible application to reachability.

Summary

Operad: a mathematical framework defining a given sort of composition.

- What are your interfaces? What sorts of arrangements are allowed?
- We saw a few operads: mostly wiring diagrams.
- We also briefly touched on operads for proteins, recipes, grammars.

Algebra: the set of inhabitants for interfaces, and the combination rules.

- Focused on dynamical systems; briefly also tensors and co-design.
- Compositional mappings and analyses (discretization, bifurcation).
- Discussed pixel array method for solving systems.
- Mentioned co-design theory, and possible application to reachability.

Operads offer a flexible framework for discussing compositionality.

Questions and comments welcome!