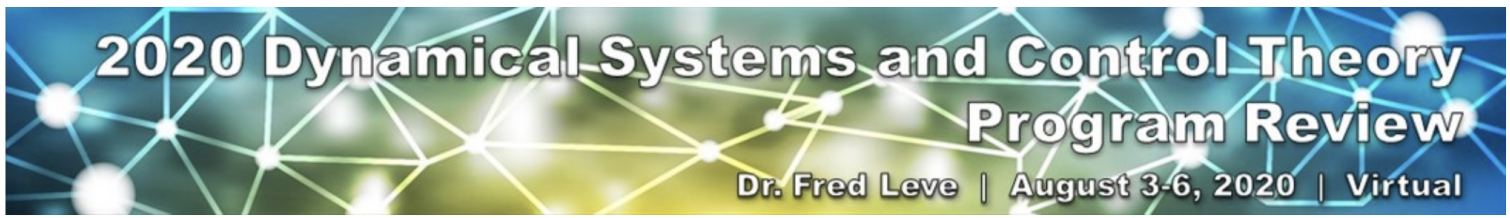
Addressing hypercomplexity: The ACT pipeline from abstraction to application

David I. Spivak
(Joint with Brendan Fong, Paolo Perrone, Evan Patterson)

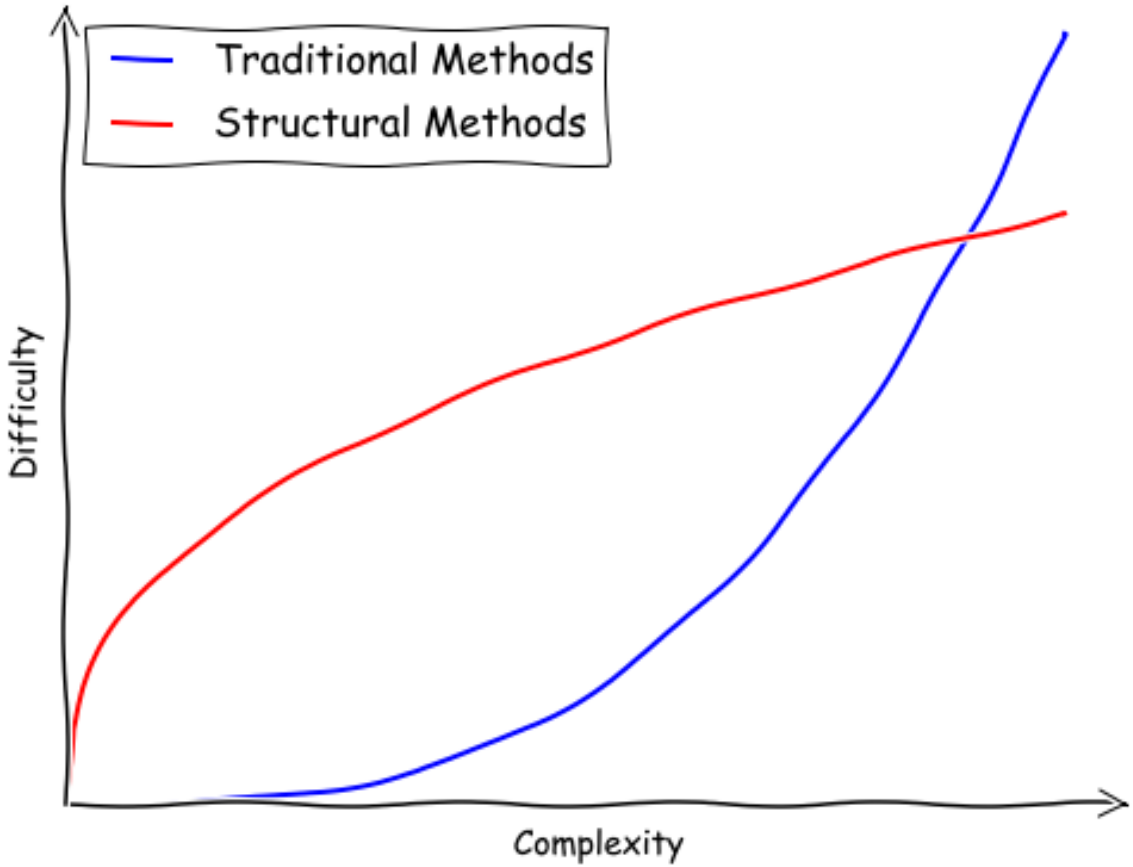
Department of Mathematics
Massachusetts Institute of Technology



Outline

What's wrong?

As problem complexity increases, good organization becomes crucial.



(Courtesy of Spencer Breiner, NIST)

The problem is hypercomplexity

The world is becoming more complex.

- We need to deal with more and more diverse sorts of interaction.
- Rather than closed dynamical systems, everything is open.
- The state of one system affects those of systems in its environment.

Examples are everywhere.

- Designing anything requires input from more diverse stakeholders.
- Software is developed by larger teams, touching a common codebase.
- Scientists need to use experimental data obtained by their peers.
  - (But the data structure and assumptions don't simply line up.)

We need to handle this new complexity; traditional methods are faltering.

How might we deal with hypercomplexity?

Organize the problem spaces effectively, as objects in their own right.

- Find commonalities in the various problems we work on.
- Consider the ways we build complex problems out of simple ones.
- Think of *problem spaces and solutions* as mathematical entities.

With that in hand, how do you solve a hypercomplex problem?

- Migrating subproblems to groups who can solve them.
- Take the returned solutions and fit them together.
- Ensure in advance that these solutions will compose.
  - Do this according the mathematical problem-space structure.

Applied category theory

Applied category theory is a growing field.

- Current well-known applications
  - Pure math, where it's been revolutionary.
  - Quantum compilers (Cambridge Quantum Computing)
  - Database integration (Conexus AI)
  - High assurance programming (Haskell)
- Less-known applications
  - Robotics design (Andrea Censi's co-design language)
  - Model-aware scientific computing (Semanticmodels.jl)

We can use ACT to structure problem / solution spaces. Example:

- "Indexed monoidal double categories" for open dynamical systems.
- Organizes trajectories, steady states, periodic orbits, ...
- ... and what happens to them as we wire open systems together.

The problem holding ACT back

Applied category theory is very abstract.

- This is both a blessing and a curse.
  - Blessing: it applies to a very large swath of science and tech.
  - Curse: it requires restarting from the ground up.
- Coming out of pure math, there hasn't been sufficient drive for tools.
- We need more "applied applied category theory".

We're still at the very beginning of what's possible.

ACT Pipeline: from concrete to theoretical

To bring category theory to the real world, we must bridge the gap.

- We need powerful abstractions—pure theory—to guide applications.
- We need real-world applications to guide theory.

My group aims to span this spectrum.

- Find common core abstractions, aiming for elegance and simplicity.
- Apply it to the real world and see what else is needed.

We work with mathematicians, scientists, and engineers of all sorts.

Plan of the talk

- Discuss some recent work of our group;
- Discuss some early work in building tools for ACT;
- Conclude.

Outline

Perception pipeline for robotics applications

Joint with Pasquale Antonante and Luca Carlone (LIDS @ MIT).

- Domain: robotics and autonomous vehicle perception.
- Problem: integrate information from many different subsystems.
  - LIDAR, IMU, Camera, high-definition map, GPS.
  - What happens when they disagree?
- Previous solution: diagnosability graph to detect faults.
- Our solution: use temporal type theory to bring in time.
  - Get a graph for every *interval* rather than single slice of time.
  - We proved that this improves fault detection and diagnosability.

Not a huge result, but both sides benefited from knowledge exchange.

Temporal type theory and stochastic processes

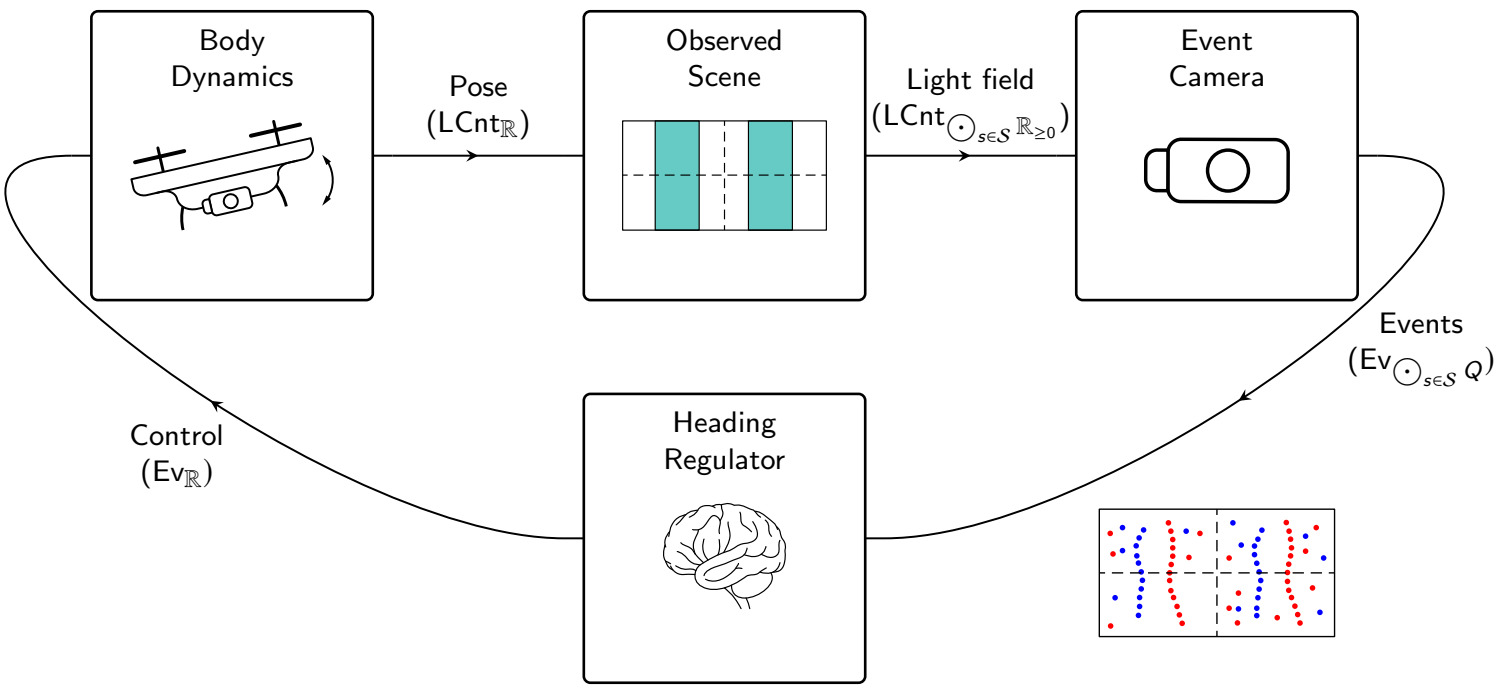
- Previously we developed a sheaf-theoretic system for general behavior.
  - Embed differential equations, inclusions, etc.
  - Discrete systems, hybrid systems, etc.
  - Delays, non-determinism, etc.
  - A model-agnostic "big tent" for anything that behaves in time.
  - A dependent type theory and logic for combining behaviors.

  Joint with Tobias Fritz (Perimeter Institute)
- Wanted to include measure theory, to consider behavioral tendencies.
  - State axioms of probability valuations using the internal language.
  - (Used the space of compacts, as in Helene Frankowska's talk.)
  - Result: a probability theory takes place in the local type theory.
  - The result generalizes stochastic processes to any spacetime.

Event-based systems

Joint with Gioele Zardini, Andrea Censi, Emilio Frazzoli (IDSC @ ETH)

- Used temporal type theory to combine the parts of an UAV.
- In particular: event-based system, e.g. event cameras.
- Any finite number of different events in a given time interval.



Outline

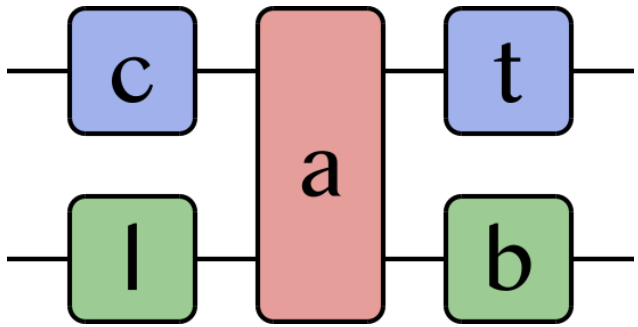
How to implement scientific and engineering models

Joint work with Evan Patterson (Stanford)

- **Numerical code**: C/C++/Fortran or indirectly via Python, R, ...
  - no high-level representations: new model $\implies$ new code
  - implementation is challenging and error prone
- **Domain-specific languages**: Simulink, Stan, TensorFlow, ...
  - easier to build new models, but only within confines of the DSL
  - abstractions and algorithms cannot be shared across domains
- **Computer algebra systems**: Mathematica, Maple, ...
  - powerful symbolic representations, but limited numerics
  - mainly confined to calculus and algebra

Catlab: a fresh approach to scientific modeling

This trichotomy is a false one.



Catlab.jl is a new framework for applied category theory:

- Written in **Julia**, which solves the "two-language problem":
  - A modern, high-level programming language
  - Just-in-time compiler generates fast code for numerical computing
- Catlab is a **hybrid system**:
  - Abstractions from category theory allow functionality to be shared between different domains and embedded DSLs
  - Symbolic and diagrammatic computing for categorical algebra
  - Numerical code generation and integration with Julia ecosystem

Case study: $\mathcal{C}$-sets as a data structure

A *$\mathcal{C}$-set* (copresheaf) is a functor $\mathcal{C} \to$ **Set**.

Things that are $\mathcal{C}$-sets: graphs, simplicial sets, Petri nets, wiring diagrams, databases with domain-specific schemas, ...

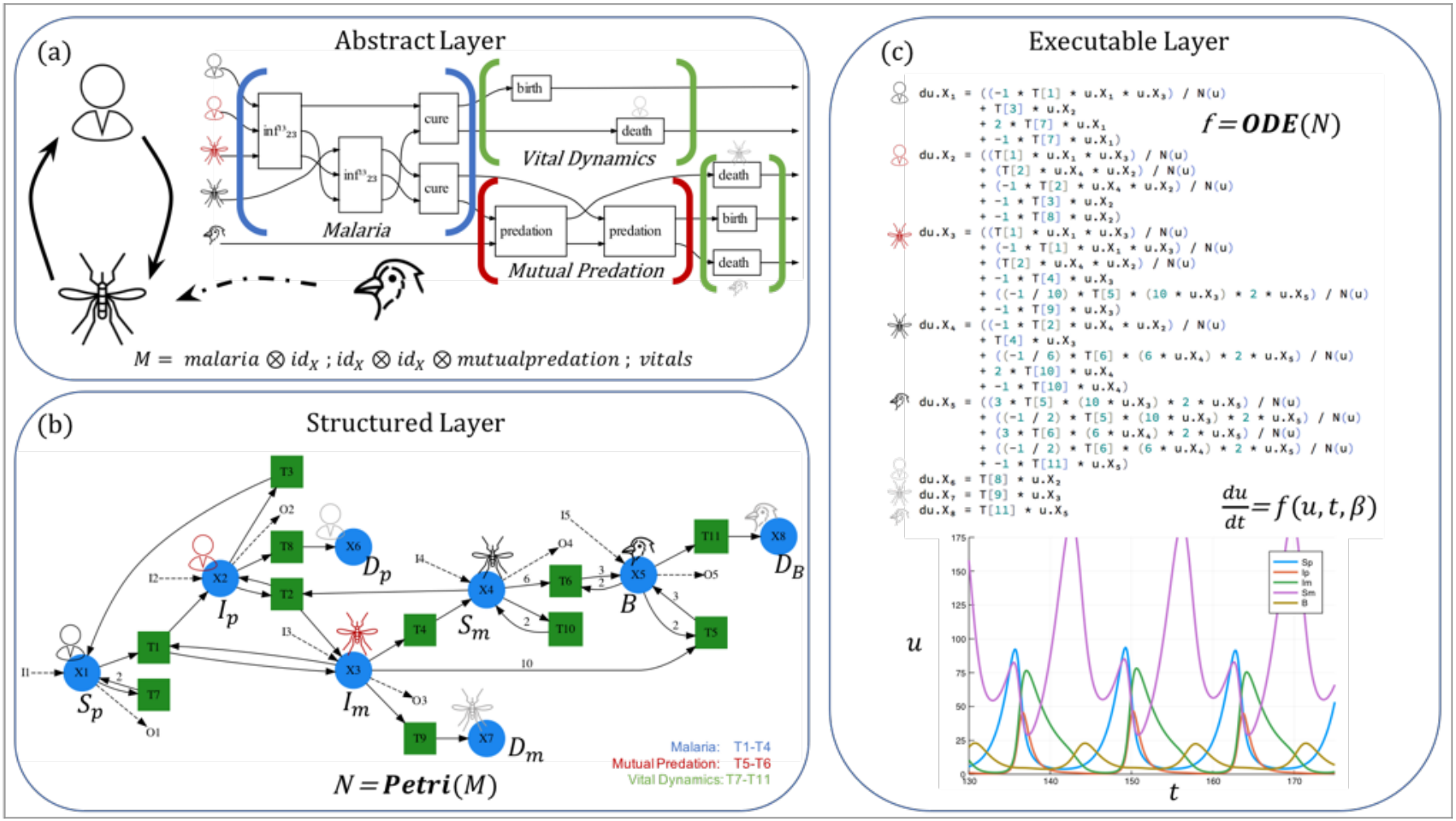
```
@present GraphSchema(FreeCategory) begin
  V::Ob
  E::Ob
  src::Hom(E,V)
  tgt::Hom(E,V)
end
Graph = CSetType(GraphSchema, index=[:src,:tgt])
```

$$\mathcal{C} = \left\{ \begin{array}{c} E \\ \mathsf{src} \downarrow\downarrow \mathsf{tgt} \\ V \end{array} \right\}$$

$$\textbf{Graph} = \textbf{Fun}(\mathcal{C}, \textbf{Set})$$

- Catlab generates efficient data structure for any schema $\mathcal{C}$
- Costless abstraction: high-level specification w/o performance penalty

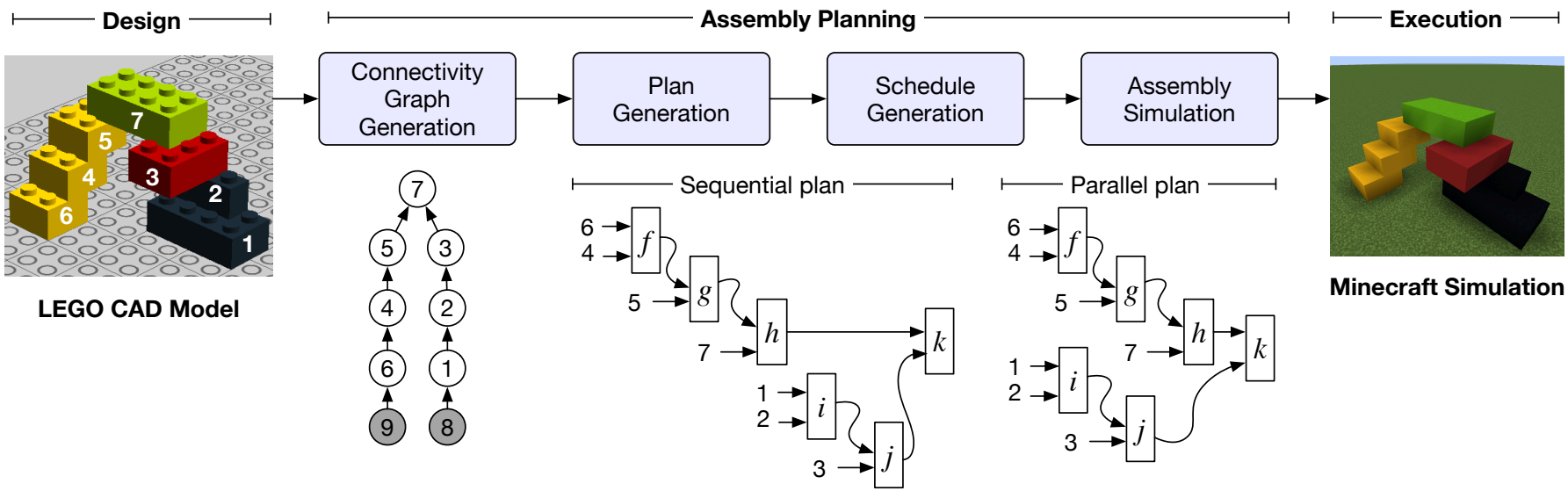
Application: Petri nets and epidemiological models



(Fairbanks, Halter, et al 2019-20: arXiv:1907.03536, arXiv:2005.04831)

Further applications

- String diagrams for assembly planning
  (Master, Patterson, et al 2019: arXiv:1909.10475)



- Semantic models of data science code
  (Patterson, Baldini, et al 2018: arXiv:1807.05691)

DoD sponsored programs

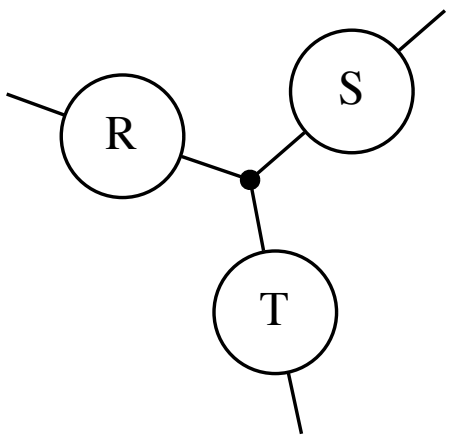
Catlab is now being used in several DoD sponsored programs:

- AFOSR: *Dynamical Systems and Control* (this program)
- ONR: *Extracting, Explaining, and Estimating Information in Sonar Data* (E3ISD)
- DARPA: *Automating Scientific Knowledge Extraction* (ASKE), Phases 1-3
- DARPA: *Artificial Social Intelligence for Successful Teams* (ASIST)
- DARPA: *Computable Models* (COMP Mods), Phase 1

Case study: Undirected wiring diagrams

Undirected wiring diagrams describe the composition of

- behavior contracts for dynamical systems
- pixel arrays (discretized continuous relations)
- tables in a SQL database, by joins
- tensor networks in quantum physics and chemistry
  - Steady states—organized by parameters and outputs—is a functor from dynamical systems networks to tensor networks.



The same category-theoretic Julia code handles all these applications.

Undirected wiring diagrams as C-sets

$$C = \left\{ P_{\text{out}} \xrightarrow{\text{junction}_{\text{out}}} J \xleftarrow{\text{junction}} P \xrightarrow{\text{box}} B \right\}$$

$$\mathbf{UWD} = \mathbf{Fun}(C, \mathbf{Set})$$

```
@present SchemaUWD(FreeCategory) begin
  Box::Ob
  Port::Ob
  OuterPort::Ob
  Junction::Ob

  box::Hom(Port,Box)
  junction::Hom(Port,Junction)
  outer_junction::Hom(OuterPort,Junction)
end
UntypedUWD = CSetType(SchemaUWD, index=[:box,:junction,:outer_junction])
```

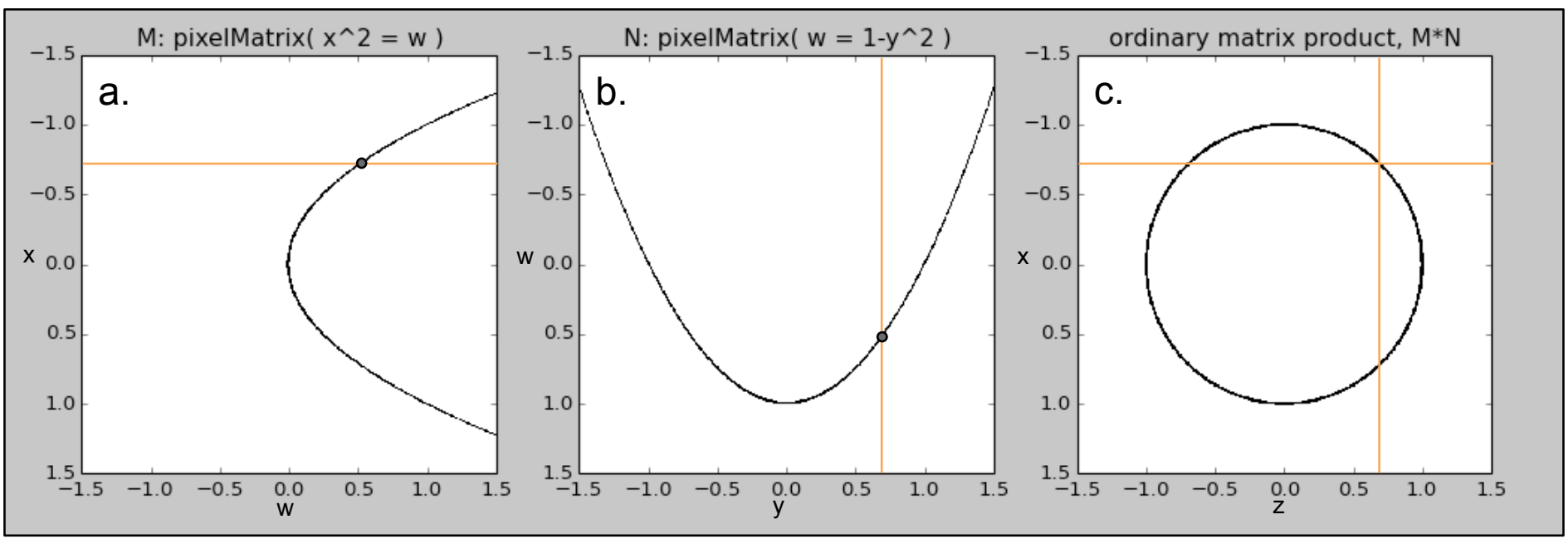
Another look at matrix multiplication

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

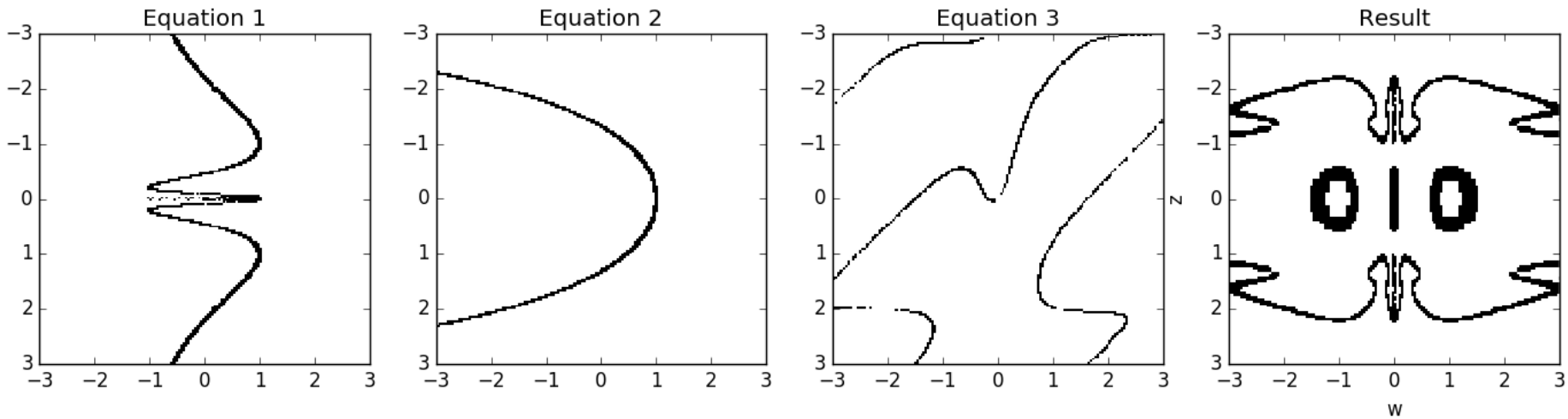
The following eq's are not differentiable, nor even defined everywhere.

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

$$\cosh(w + 10^{-3}y) + y + 10^{-4}w - 2 = 0 \qquad \text{(Equation 2)}$$

$$\tan(x + y)(x - 2)^{-1}(x + 3)^{-1}y^{-2} - 1 = 0 \qquad \text{(Equation 3)}$$

Q: For what values of w and z does a simultaneous solution exist? [1]

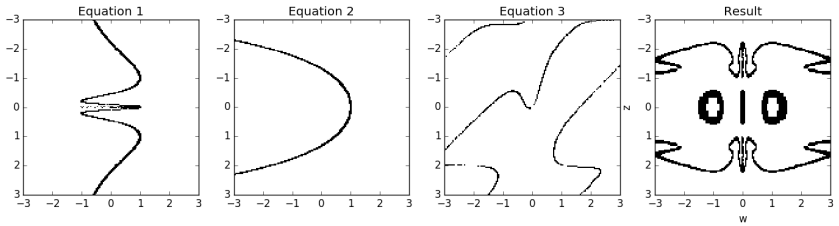


[1] Spivak, DI; Dobson, MRC; Kumari, S. (2016) "Pixel Arrays: A fast and elementary method for solving nonlinear systems". `http://arxiv.org/pdf/1609.00061v1.pdf`

Selling points

The Pixel Array method has the following features:

- it approximates *all solutions in a given bounding box*;
- it's *much faster* than quasi-Newton methods for finding "all solutions" to partially decomposable systems;
- it introduces *no false negatives*;
- it works for *non-differentiable* or even *discontinuous* functions;
- it's *not iterative* and requires no initial guess, in contrast with quasi-Newton methods;
- it's *functorial* with respect to steady states/periodic orbits of interconnected dynamical systems; and
- it *provides insights*, by showing the whole solution set at once.



Pixel arrays in Catlab

Leverage Julia ecosystem for tensors:

- Tullio.jl and other packages provide generic tensor computing
- Catlab provides `BoolRig` type for boolean-valued tensors

$$
\begin{aligned}
R_1(w,x,y) &\iff \tan(y+w) + \exp(x) = 2 \\
R_2(v,x,y) &\iff x^3 + \cos(\log(y^2)) = 1.5v \\
R_3(v,w,z) &\iff w + z + 10^{-1}v = 0.5
\end{aligned}
$$

```
@tullio R1[i,j,k] := (abs(tan(y[k] + w[i]) + exp(x[j]) - 2) < ϵ) |> BoolRig
@tullio R2[i,j,k] := (abs(x[j]^3 + cos(log(y[k]^2)) - 1.5v[i]) < ϵ) |> BoolRig
@tullio R3[i,j,k] := (abs(w[j] + z[k] + v[i]/10 - 0.5) < ϵ) |> BoolRig
```

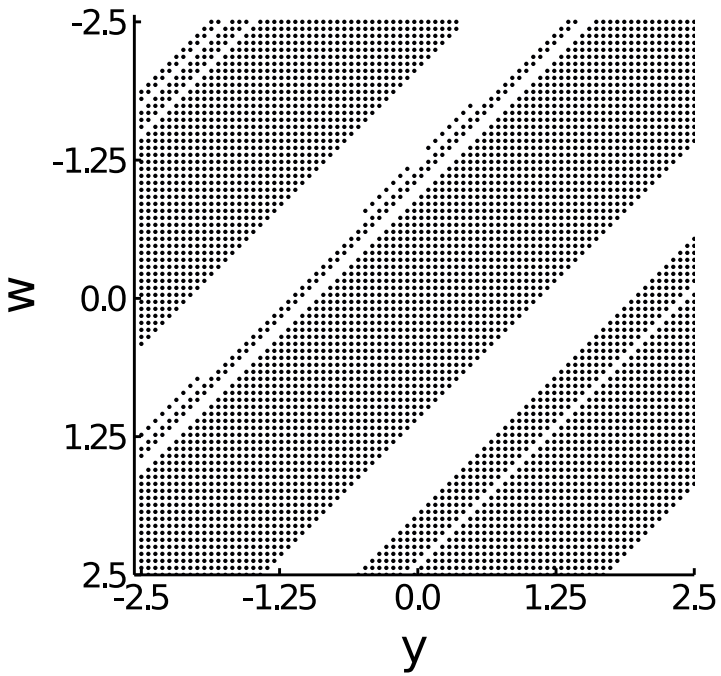
Contraction of pixel arrays

Generate Julia expression in Einstein notation:

```
compile_tensor_expr(diagram)
```

Output can be used with Tullio.jl and other packages:

```
:(out[w, y] = R1[w, x, y] * R2[v, x, y] * R3[v, w, z]) # output of above
```



Outline

Summary

We look for abstractions that will be reusable across domains.

- This lets us work with a large number of different disciplines.
- Factor hypercomplex problems; use functors to hand off fragments.

But until now, much of this has remained stuck in academic papers.

- The ACT community has talked about fascinating possibilities.
- With tooling like CQL, UMAP, and Catlab we show it actually works.
- AFOSR funding has been integral to establishing ACT, and we won't disappoint.

Thanks; comments and questions welcome!

Relevant group papers July 2019 – 2020

- Antonante; **Spivak**; Carlone (2020) "Monitoring and Diagnosability of Perception Systems". `https://arxiv.org/abs/2005.11816`
- **Fong, Spivak** (2019) *An Invitation to Applied Category Theory: Seven Sketches in Compositionality*. Cambridge University Press.
- **Fong**, Myers, **Spivak** (2020) "Behavioral Mereology: A Modal Logic for Passing Constraints", *Proceedings of the 3rd International Conference on Applied Category Theory*.
- **Fong**, Sarazola (2020) "A recipe for black box functors" *Theory and Applications of Categories* Vol. 35, No. 26, pp 979-1011.
- **Fong, Spivak, Tuyéras** (2019) "Backprop as Functor: A compositional perspective on supervised learning". *LICS*.
- Fritz, **Perrone**, Rischel, Gonda (2020) "Stochastic dominance and the Blackwell theorem in Markov categories." *To appear*
- Genovese, F.; **Spivak** (2020) "A Categorical Semantics for Guarded Petri Nets". *ICGT*.
- Patterson, E.; **Spivak**; Vagner, D. (2020) "Wiring diagrams as normal forms for computing in symmetric monoidal categories". *Proceedings of the 3rd International Conference on Applied Category Theory*.
- **Spivak** (2020) "Poly: An abundant categorical setting for mode-dependent dynamics."
- Zardini G.; **Spivak**; Censi, A; Frazzoli, E. (2020) "A Compositional Sheaf-Theoretic Framework for Event-Based Systems". *Proceedings of the 3rd International Conference on Applied Category Theory*.