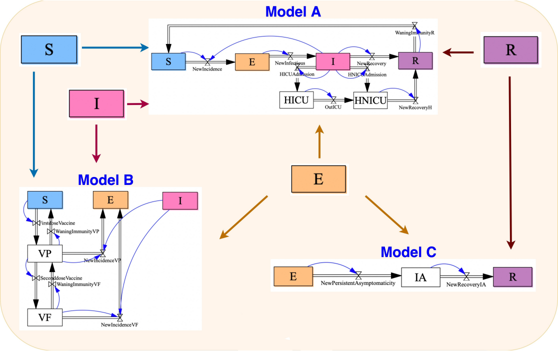
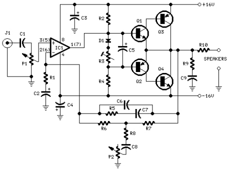
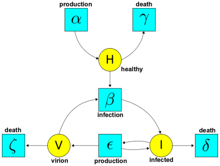
MODELING WITH
APPLIED CATEGORY THEORY

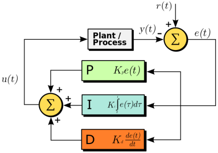


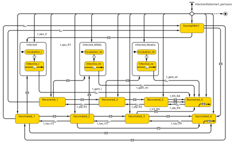
John Baez

In many areas of science and engineering, people describe systems using different kinds of *network diagrams*:

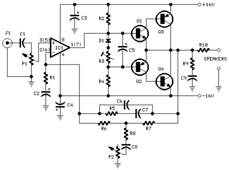


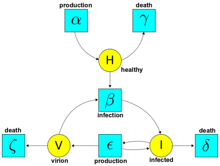


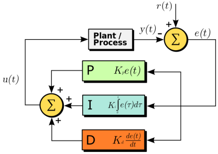


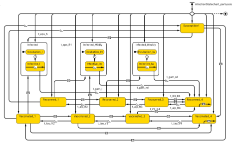


In many areas of science and engineering, people describe systems using different kinds of *network diagrams*:



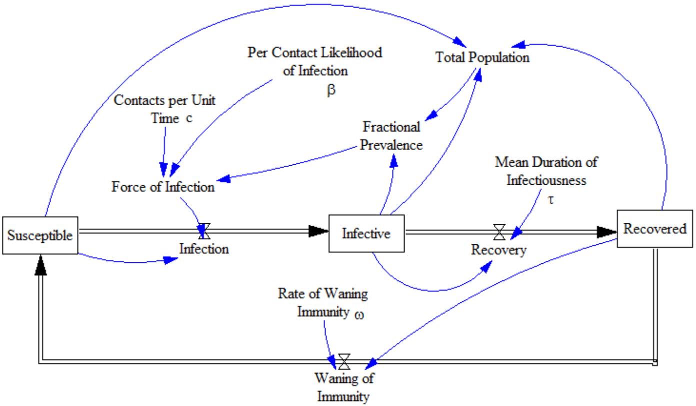






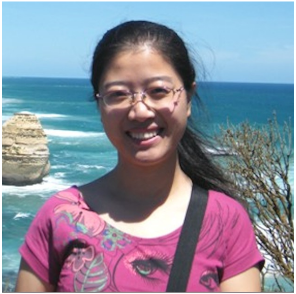
Category theory provides a unified way to work with these in software.

For example: in “System Dynamics”, dynamical systems are modeled using “stock and flow diagrams”:



These diagrams are now widely used in economics, population biology, epidemiology, etc.

There is a community of epidemiologists who use stock-flow diagrams to model the spread of disease. This includes my collaborators Nate Osgood and Xiaoyan Li, who did COVID modeling for the government of Canada.

Stock and flow modeling is often done using software called AnyLogic. It's powerful, but it has several big problems:

- It has no support for *"composing"* models: that is, taking several smaller models and putting them together to form a larger model.

Stock and flow modeling is often done using software called AnyLogic. It's powerful, but it has several big problems:

- It has no support for *"composing"* models: that is, taking several smaller models and putting them together to form a larger model.
- It has no support for *"functorial semantics"*: that is, the systematic separation of a model from the choice of how we run the model.

Stock and flow modeling is often done using software called AnyLogic. It's powerful, but it has several big problems:

- It has no support for *"composing"* models: that is, taking several smaller models and putting them together to form a larger model.
- It has no support for *"functorial semantics"*: that is, the systematic separation of a model from the choice of how we run the model.
- It has no support for *collaboratively* building models.

Stock and flow modeling is often done using software called AnyLogic. It's powerful, but it has several big problems:

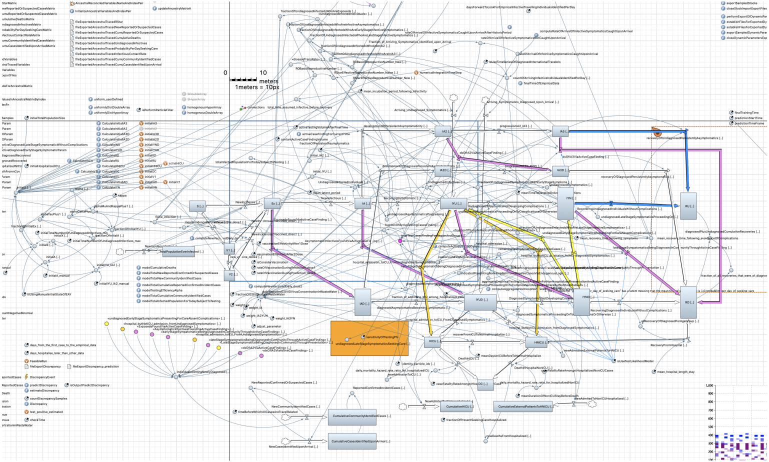
- It has no support for *"composing"* models: that is, taking several smaller models and putting them together to form a larger model.
- It has no support for *"functorial semantics"*: that is, the systematic separation of a model from the choice of how we run the model.
- It has no support for *collaboratively* building models.
- It is not *free* and not *open-source!*

Our new software aims to solve all these problems.

This software has been developed by Evan Patterson, James Fairbanks, Owen Lynch, Sophie Libkind, Kris Brown, Nathaniel D. Osgood, Xiaoyan Li, Eric Redekopp and others:

- AlgebraicJulia: an ecosystem of mathematical software written in Julia.
- CatLab.jl: a framework for applied category theory within AlgebraicJulia.
- StockFlow.jl: software for building and running stock and flow models, based on CatLab.
- ModelCollab: a web-based front end for using StockFlow.jl collaboratively.
- AlgebraicPetri.jl: software for building and running Petri net models, based on CatLab.
- AlgebraicABMs.jl: software for agent based modeing, based on CatLab.

The ability to *compose* models is crucial because realistic models are complicated and built out of many smaller parts. Here is Osgood and Li's COVID model used by the government of Canada:



Compositionality and Functorial Semantics

Very roughly:

- *"composing"* models is taking two or more smaller models and putting them together to form a larger model.
- *"functorial semantics"* is the systematic separation of a model from the choice of how we extract information from it, treating each as a mathematical structure in its own right.

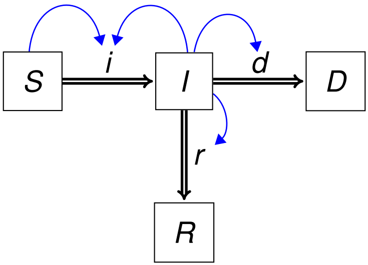
Compositionality

Models of systems may be constructed individually and then coupled together to build larger models. To achieve this, each model is not merely a piece of code, but a crisply defined mathematical structure designed from the start to be combined with other models.

Technically, this is done by treating models as “morphisms” in a “category”, and the coupling of models as “composition” of these morphisms. Thus, this approach to model design is known as **compositionality**.

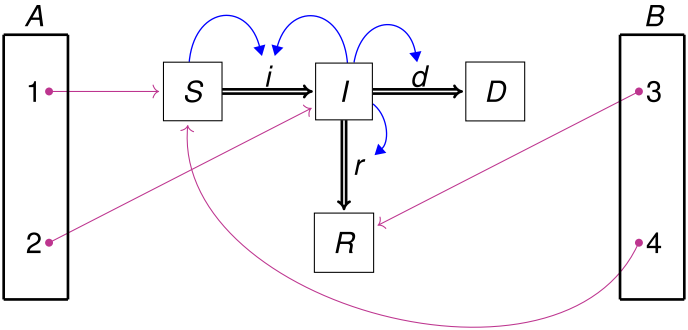
A (very simple sort of) **stock-flow diagram** consists of

- a finite set of **stocks**,
- arrows between stocks called **flows**,
- arrows from stocks to flows called **links**,
- a **flow function** $\phi_f \colon \mathbb{R}^n \to \mathbb{R}$ for each flow f.



$$\phi_i \colon \mathbb{R}^2 \to \mathbb{R}, \quad \phi_r \colon \mathbb{R} \to \mathbb{R}, \quad \phi_d \colon \mathbb{R} \to \mathbb{R}$$

An **open stock-flow diagram** is a stock-flow diagram equipped with maps $L\colon A \to \text{Stocks}$, $R\colon B \to \text{Stocks}$ for some finite sets A, B.



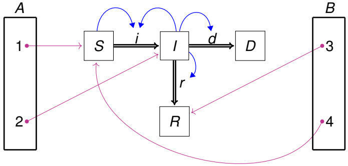
$$\phi_i : \mathbb{R}^2 \to \mathbb{R}, \quad \phi_r : \mathbb{R} \to \mathbb{R}, \quad \phi_d : \mathbb{R} \to \mathbb{R}$$

We call this an open stock-flow diagram **from A to B** and write it as $A \xrightarrow{F} B$.

We can **compose** open stock-flow diagrams $A \xrightarrow{F} B$ and $B \xrightarrow{G} C$ by "gluing them together along B".

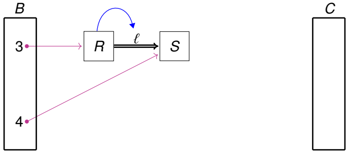
We get an open stock-flow diagram called $A \xrightarrow{GF} C$.

$$A \xrightarrow{F} B$$



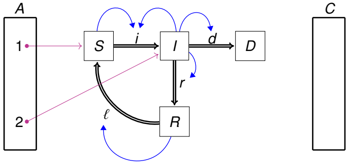
$$\phi_i : \mathbb{R}^2 \to \mathbb{R}, \quad \phi_r : \mathbb{R} \to \mathbb{R}, \quad \phi_d : \mathbb{R} \to \mathbb{R}$$

$$B \xrightarrow{G} C$$



$$\phi_\ell : \mathbb{R} \to \mathbb{R}$$

$$A \xrightarrow{GF} C$$



$$\phi_i : \mathbb{R}^2 \to \mathbb{R}, \quad \phi_r : \mathbb{R} \to \mathbb{R}, \quad \phi_d : \mathbb{R} \to \mathbb{R}, \phi_\ell : \mathbb{R} \to \mathbb{R}$$

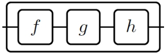
Since composition is associative:

$$(HG)F = H(GF)$$

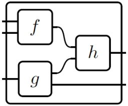
we say there is a **category** StockFlow with:

- finite sets as objects,
- open stock-flow diagrams as morphisms,
- composition of morphisms defined as above.

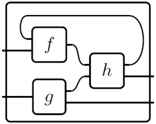
There are also fancier kinds of categories:



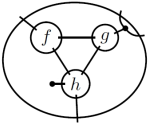
category



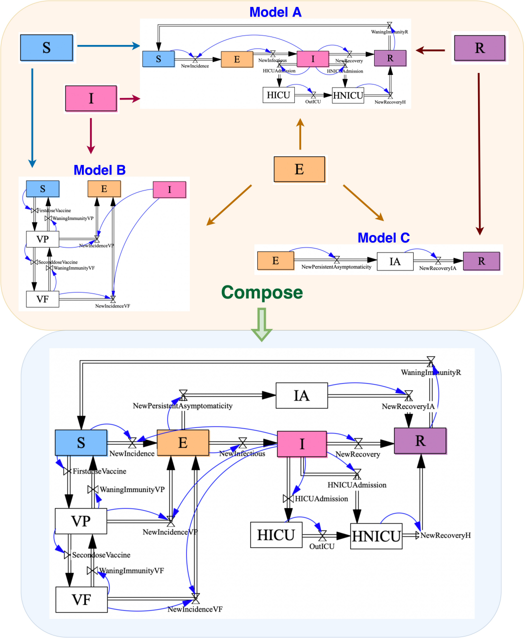
monoidal category



traced monoidal category



hypergraph category

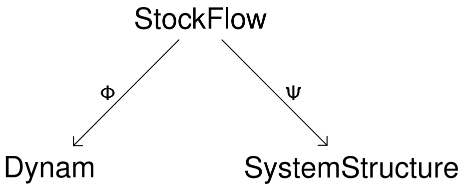


Model A
S
I
R
E
Model B
Model C
Compose
S
E
I
R
NewIncidence
NewInfectious
NewRecovery
HICUAdmission
HNICUAdmission
HICU
HNICU
OutICU
NewRecoveryH
WaningImmunityR
VP
VF
FirstdoseVaccine
WaningImmunityVP
NewIncidenceVP
SeconddoseVaccine
WaningImmunityVF
NewIncidenceVF
IA
NewPersistentAsymptomaticity
NewRecoveryIA

Functorial Semantics

There is a clear distinction between the model and a specific way of extracting information from it. For example, one can take a stock and flow model and run it either deterministically or stochastically, or extract from it a system structure diagram.

To do this we treat different ways of running a model as different "functors" from a category whose morphisms are models of some specific type to various other categories. This is called **functorial semantics**.



A **dynamical system** on some finite set of variables X is a vector field v on $\mathbb{R}^X$. This lets us write down a system of first-order ordinary differential equations.

For example, if $X = \{S, I, D, R\}$ and v is the vector field (v_S, v_I, v_D, v_R) on $\mathbb{R}^X \cong \mathbb{R}^4$, we get

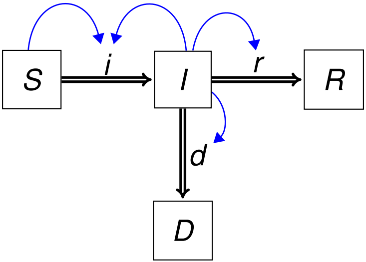
$$\begin{aligned}
\frac{d}{dt} S(t) &= v_S(S(t), I(t), D(t), R(t)) \\
\frac{d}{dt} I(t) &= v_I(S(t), I(t), D(t), R(t)) \\
\frac{d}{dt} D(t) &= v_D(S(t), I(t), D(t), R(t)) \\
\frac{d}{dt} R(t) &= v_R(S(t), I(t), D(t), R(t))
\end{aligned}$$

A **dynamical system** on some finite set of variables X is a vector field v on $\mathbb{R}^X$. This lets us write down a system of first-order ordinary differential equations.

For example, if $X = \{S, I, D, R\}$ and v is the vector field (v_S, v_I, v_D, v_R) on $\mathbb{R}^X \cong \mathbb{R}^4$, we get

$$\begin{aligned}
\frac{dS}{dt} &= v_S \\
\frac{dI}{dt} &= v_I \\
\frac{dD}{dt} &= v_D \\
\frac{dR}{dt} &= v_R
\end{aligned}$$

Any stock-flow diagram gives a dynamical system:



$\phi_i : \mathbb{R}^2 \to \mathbb{R}, \; \phi_r : \mathbb{R} \to \mathbb{R}, \; \phi_d : \mathbb{R} \to \mathbb{R}$

$$\frac{dS}{dt} = -\phi_i(S, I)$$

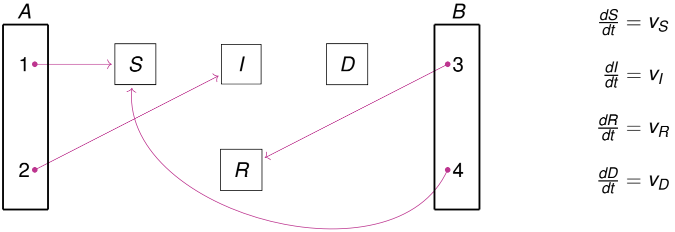
$$\frac{dI}{dt} = \phi_i(S, I) - \phi_r(I) - \phi_d(I)$$

$$\frac{dR}{dt} = \phi_r(I)$$

$$\frac{dD}{dt} = \phi_d(I)$$

An **open dynamical system** $A \xrightarrow{v} B$ is a dynamical system v on some finite set X equipped with maps $L \colon A \to X$, $R \colon B \to X$ for some finite sets A, B.

For example:



Here $X = \{S, I, D, R\}$.

Just as we constructed the category StockFlow, we can construct a category Dynam with:

- finite sets as objects,
- open dynamical systems as morphisms.

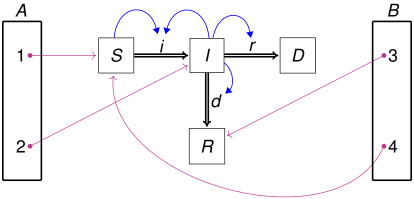
Just as we constructed the category StockFlow, we can construct a category Dynam with:

- finite sets as objects,
- open dynamical systems as morphisms.

The process we've already seen for converting stock flow diagrams into dynamical systems also lets us convert *open* stock flow diagrams into *open* dynamical systems. We call this conversion process

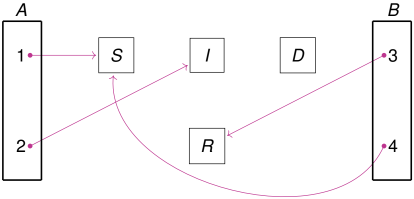
$$\Phi \colon \mathrm{StockFlow} \to \mathrm{Dynam}$$

For example, Φ maps this open stock flow diagram:



$$\phi_i : \mathbb{R}^2 \to \mathbb{R}, \quad \phi_r : \mathbb{R} \to \mathbb{R}, \quad \phi_d : \mathbb{R} \to \mathbb{R}$$

to this open dynamical system:



$$\frac{dS}{dt} = -\phi_i(S, I)$$

$$\frac{dI}{dt} = \phi_i(S, I) - \phi_r(I) - \phi_d(I)$$

$$\frac{dR}{dt} = \phi_r(I)$$

$$\frac{dD}{dt} = \phi_d(I)$$

We say Φ is a **functor** because

$$\Phi(GF) = \Phi(G)\,\Phi(F)$$

This equation says we can

- compose two open stock-flow models and then convert the result into an open dynamical system

or

- convert two open stock-flow models into open dynamical systems and then compose those

and *the results are the same!*

This is the key feature of “functorial semantics”.

Interoperability

Models expressed in different frameworks can be coupled or interoperated. This is a consequence of compositionality and functorial semantics.

The reason is that you can build models out of smaller pieces living in different categories, as long as they all come with functors to some common category. For example:

$$\mathrm{StockFlow} \xrightarrow{\Phi} \mathrm{Dynam} \xleftarrow{\Gamma} \mathrm{Petri}$$

There are many more ways category theory is helping the creation of new modeling techniques... and surely many more waiting to be discovered!

Good luck!