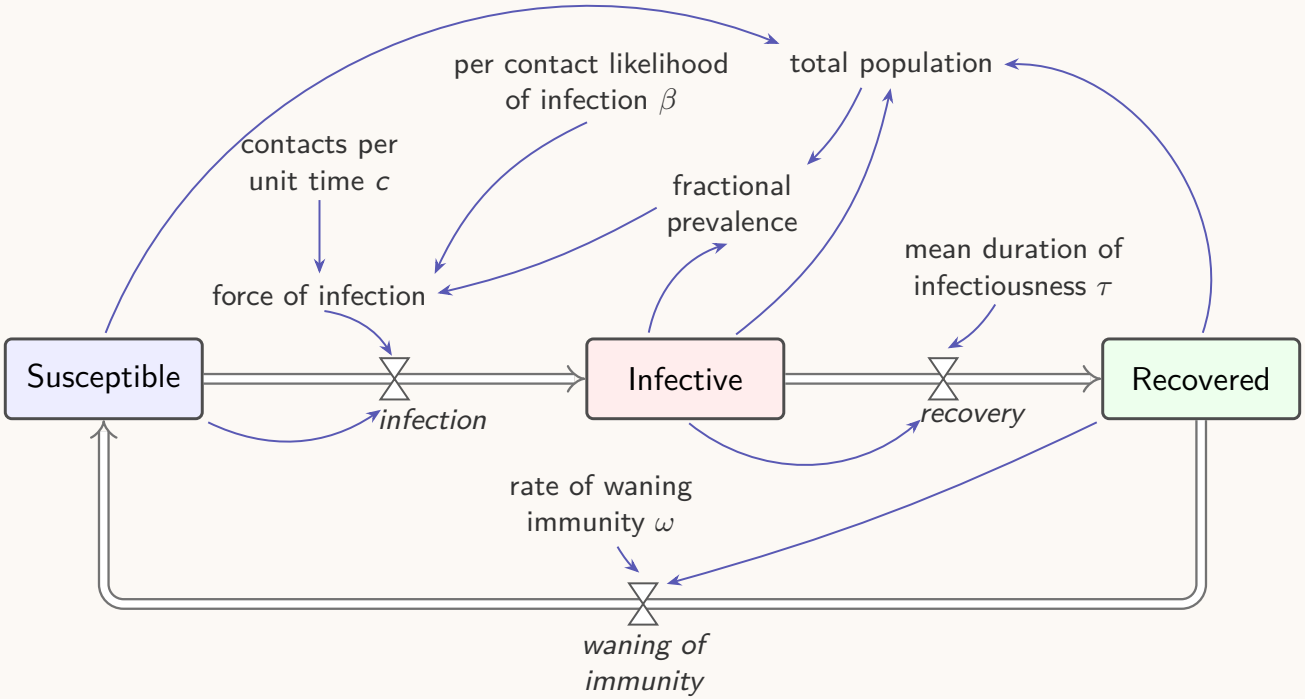
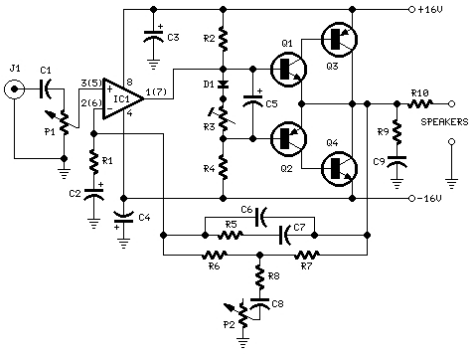
Applied Category Theory and The Mathematics of Networks – I

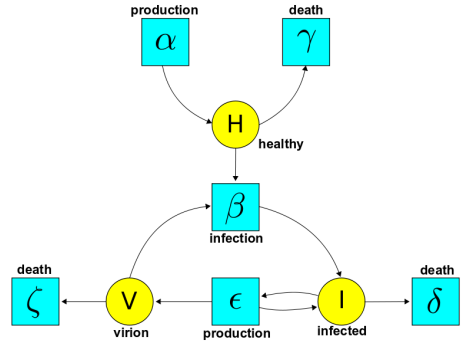


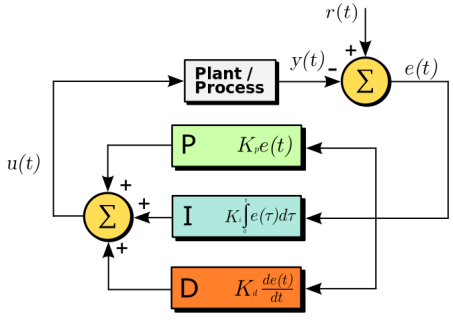
John Baez, University of Edinburgh

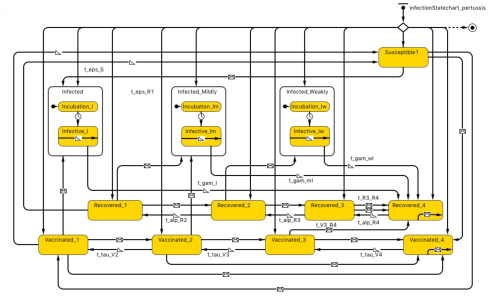
GeoCOW 2026

In many areas of science and engineering, people use *networks*, drawn as boxes connected by wires:

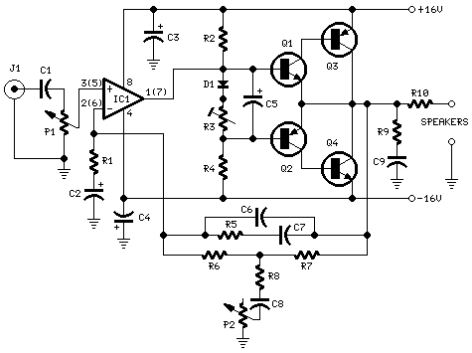


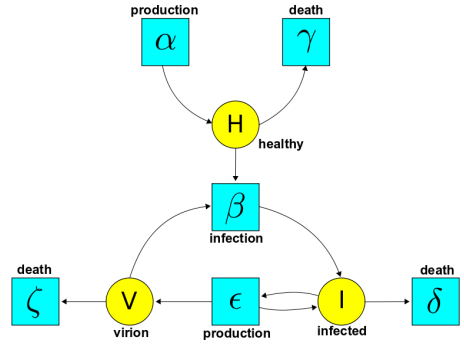


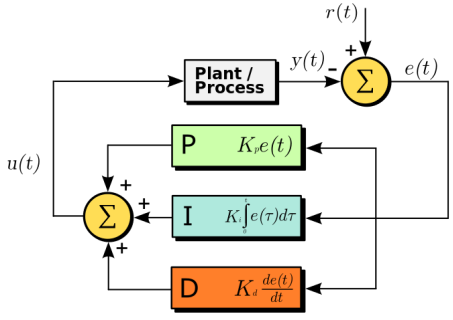


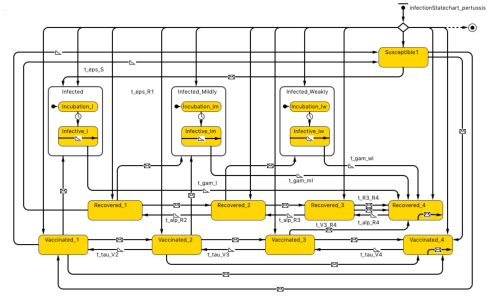


In many areas of science and engineering, people use *networks*, drawn as boxes connected by wires:





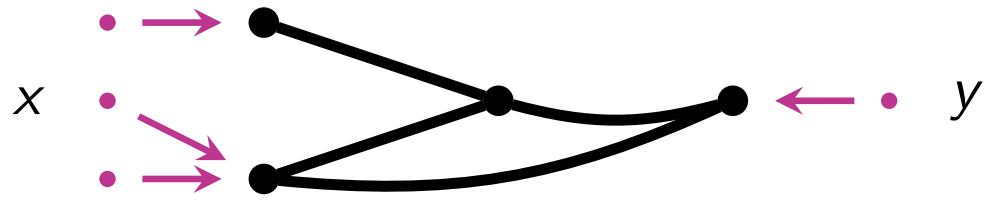




Applied category theory provides a unified way to work formally with these.

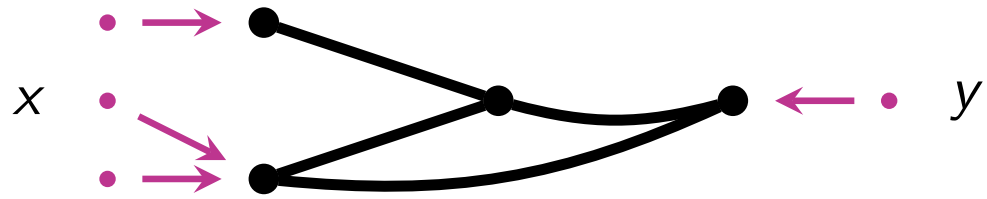
It is crucial to focus on "open" networks, which are networks with specified "boundaries".

By "network" we might mean a simple graph

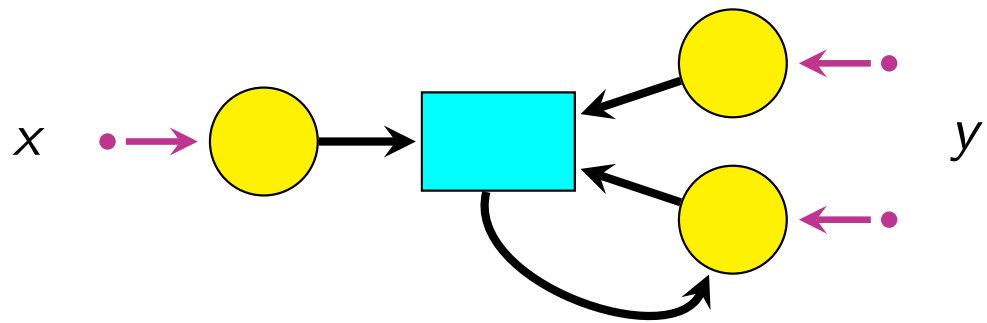


It is crucial to focus on "open" networks, which are networks with specified "boundaries".

By "network" we might mean a simple graph



or something more interesting, like a Petri net:



or dozens of other things: electrical circuits, signal flow diagrams, state charts, stock-flow diagrams, etc.

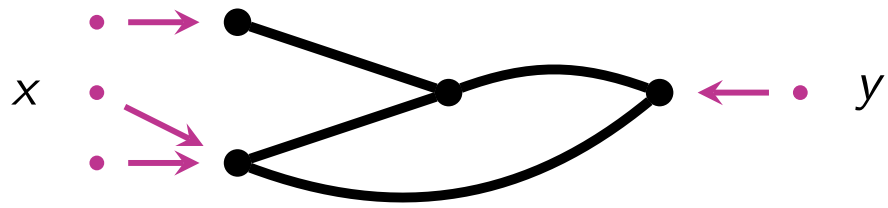
If our network has no boundaries, we call it a **closed** network—just an ordinary network. These are commonly studied.

But there are more operations on open networks. The two most fundamental are "composition" and "tensoring".

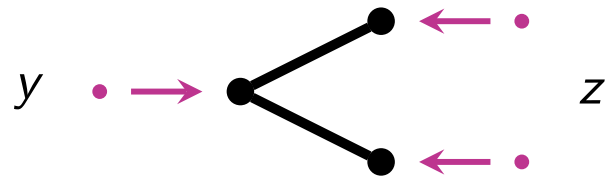
Using these operations it is enough to consider networks with *two* boundaries x and y.

We can "compose" open networks of a fixed kind, gluing them together end to end in series.

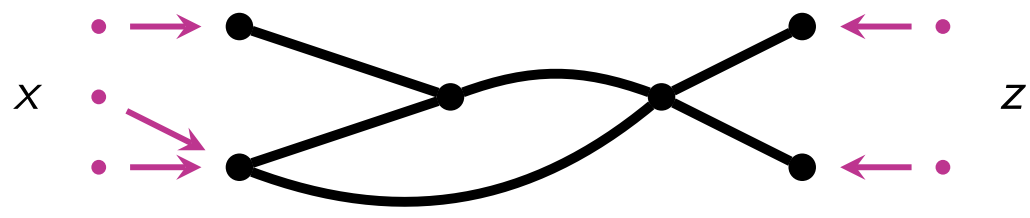
Composing $g : x \to y$



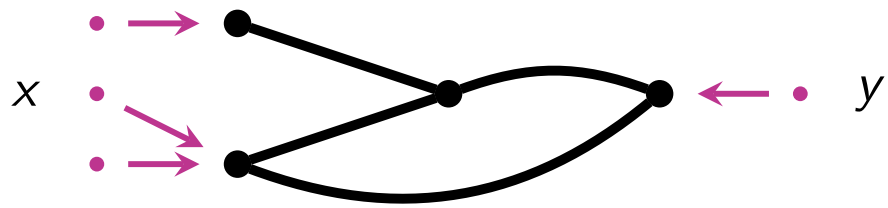
and $f : y \to z$



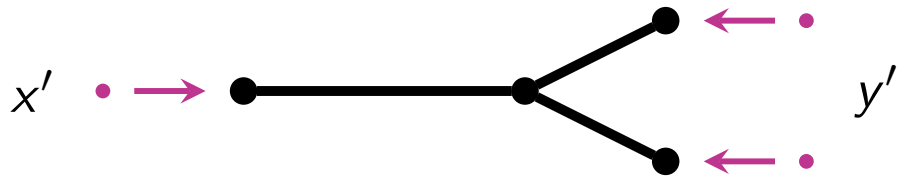
we get $f \circ g : x \to z$:



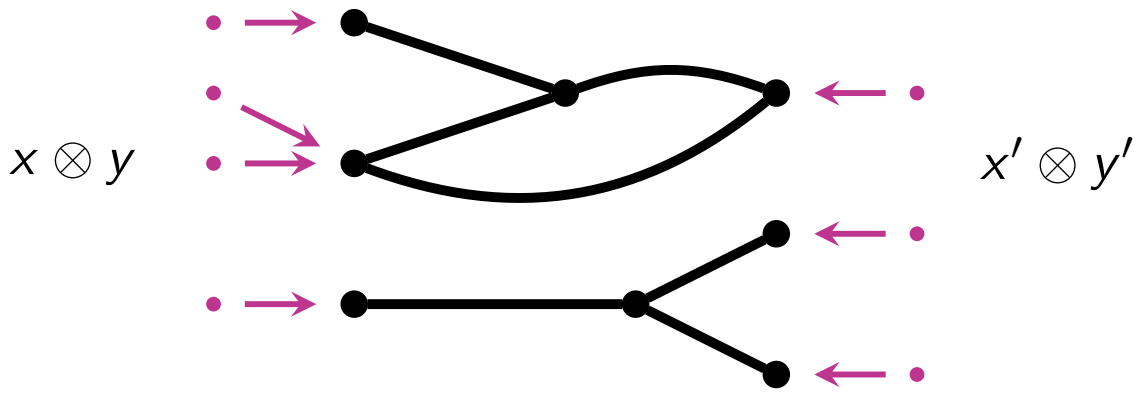
We can also "tensor" open networks of a fixed kind, putting them side by side in parallel. Tensoring $f : x \to y$



and $g : x' \to y'$



we get $f \otimes g : x \otimes y \to x' \otimes y'$

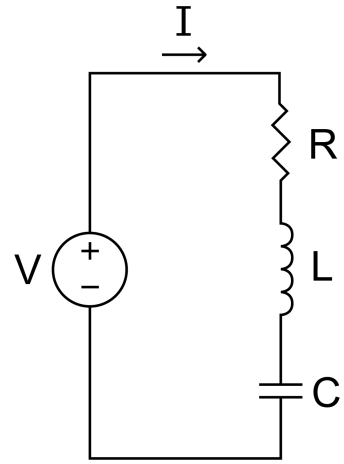


A particular kind of open network, with composition and tensoring that obey some rules, defines a "monoidal category" $\mathbf{C}$. I'll define this concept next time.

But there's more:

In modeling and engineering we use networks to describe systems.

Thus, we want to say what networks of a particular kind *do*, or *mean*. E.g., we may want to map any network of this kind to a system of equations. For example this electrical circuit diagram:



gives this system of differential equations:

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2} I(t) + \frac{R}{L}\frac{\mathrm{d}}{\mathrm{d}t} I(t) + \frac{1}{L\,C} I(t) = 0$$

If we have open systems described by some monoidal category $\mathbf{C}$, we can say what they *do*, or *mean*, using a "monoidal functor". This a map of monoidal categories $F : \mathbf{C} \to \mathbf{D}$.

I'll also define this concept next time, but the key rules are

$$F(f \circ g) = F(f) \circ F(g)$$

$$F(f \otimes g) = F(f) \otimes F(g)$$

In words:

The dynamics of an open system built from parts is determined by the dynamics of the parts, and how those parts are assembled.

If we have open systems described by some monoidal category $\mathbf{C}$, we can say what they *do*, or *mean*, using a "monoidal functor". This a map of monoidal categories $F \colon \mathbf{C} \to \mathbf{D}$.

I'll also define this concept next time, but the key rules are

$$F(f \circ g) = F(f) \circ F(g)$$

$$F(f \otimes g) = F(f) \otimes F(g)$$

In words:

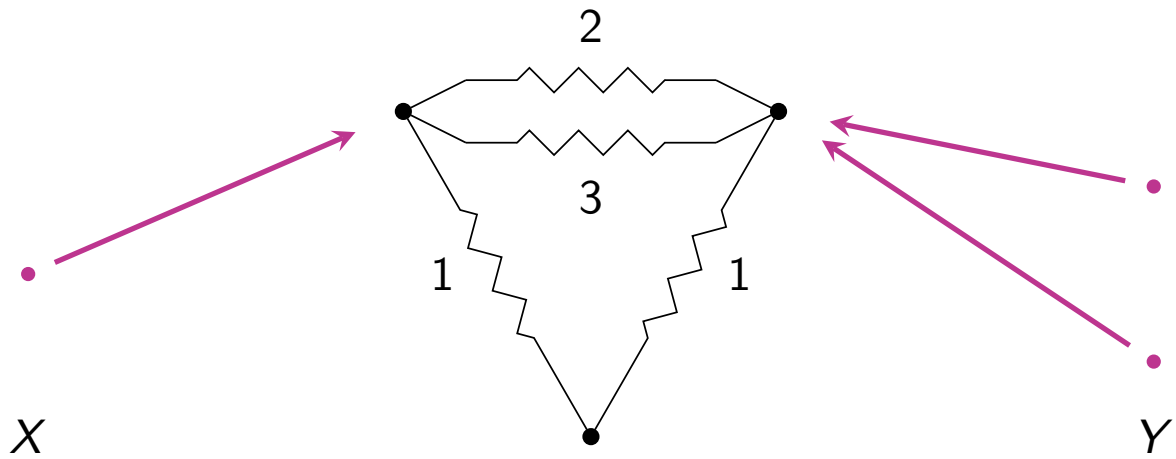
The dynamics of an open system built from parts is determined by the dynamics of the parts, and how those parts are assembled.

For example, there's a monoidal category $\mathbf{C}$ of open electrical circuits and a monoidal functor $F \colon \mathbf{C} \to \mathbf{D}$ sending open electrical circuits to open systems of differential equations.

There are many mathematically beautiful monoidal categories at work in science and engineering. For example:

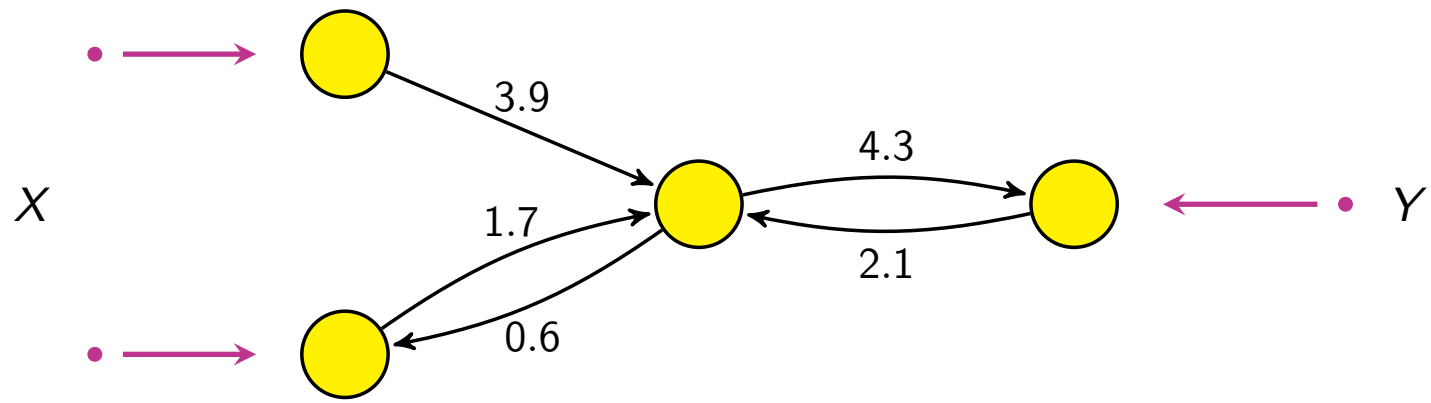
There are many mathematically beautiful monoidal categories at work in science and engineering. For example:

- **Electrical circuits**: JB and B. Fong, A compositional framework for passive linear networks.



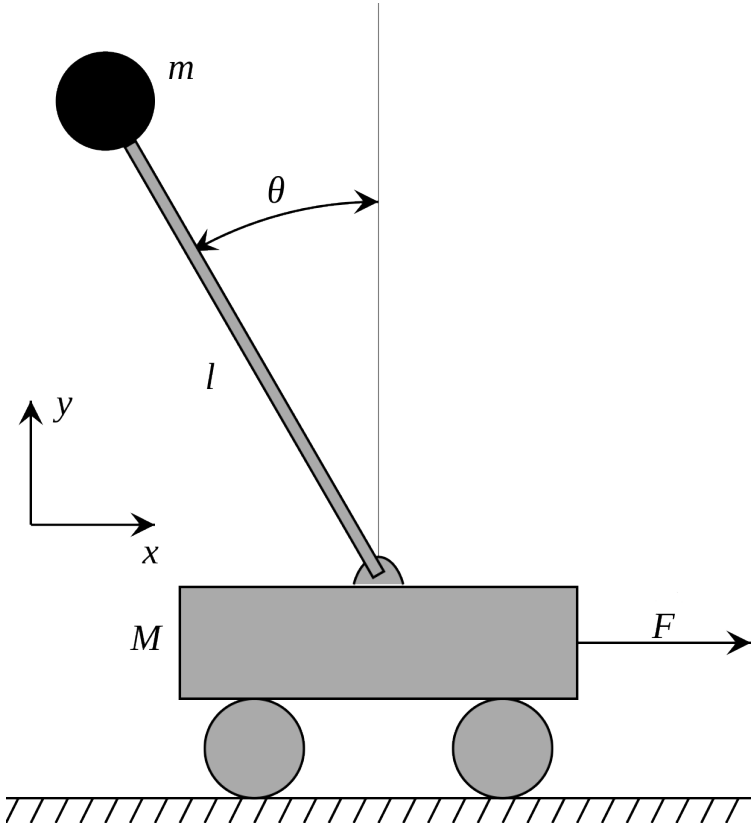
- **Markov processes**: JB, Brendan Fong and Blake Pollard, A compositional framework for Markov processes.

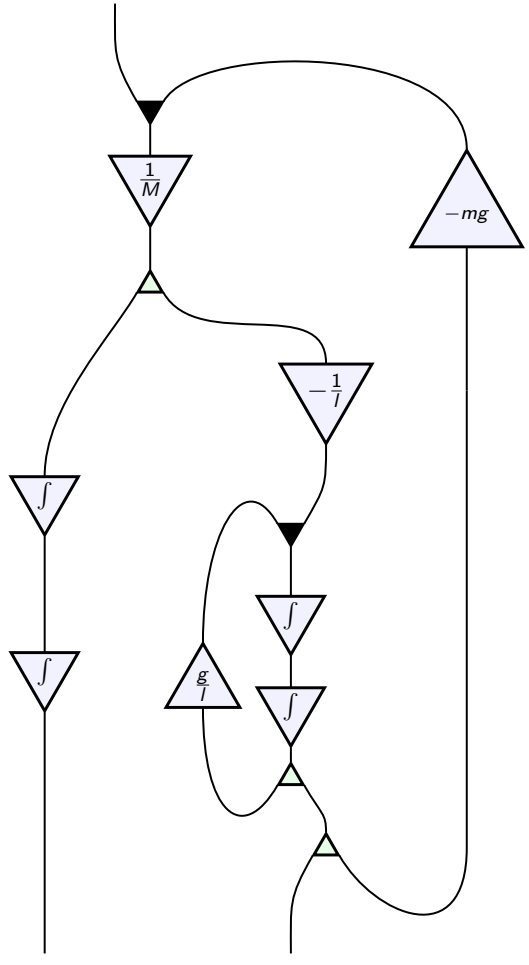
  JB and Kenny Courser Coarse-graining open Markov processes.



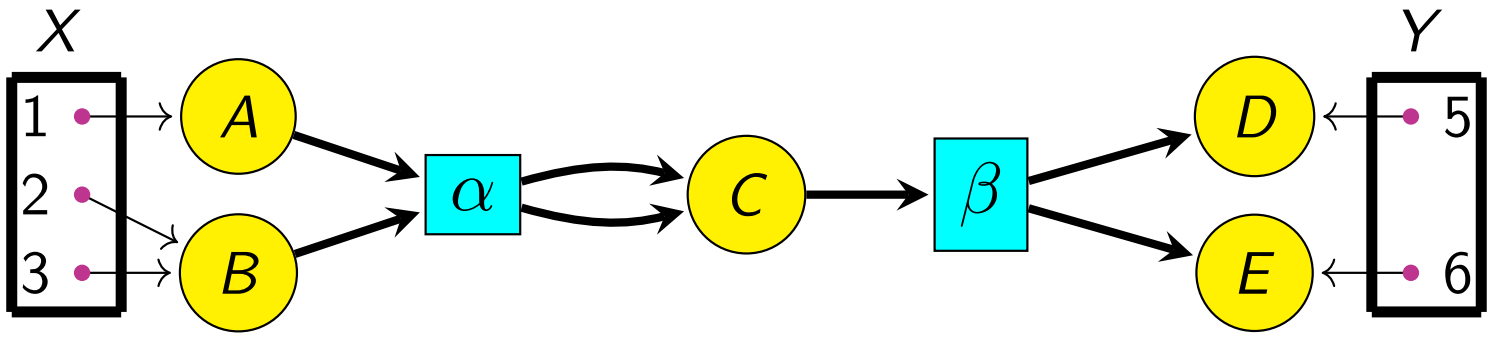
- **Signal-flow graphs in control theory**:
  Jason Erbele, *Categories in Control: Applied PROPs*.

  Filippo Bonchi, Paweł Sobociński and Fabio Zanasi, A categorical semantics of signal flow graphs.

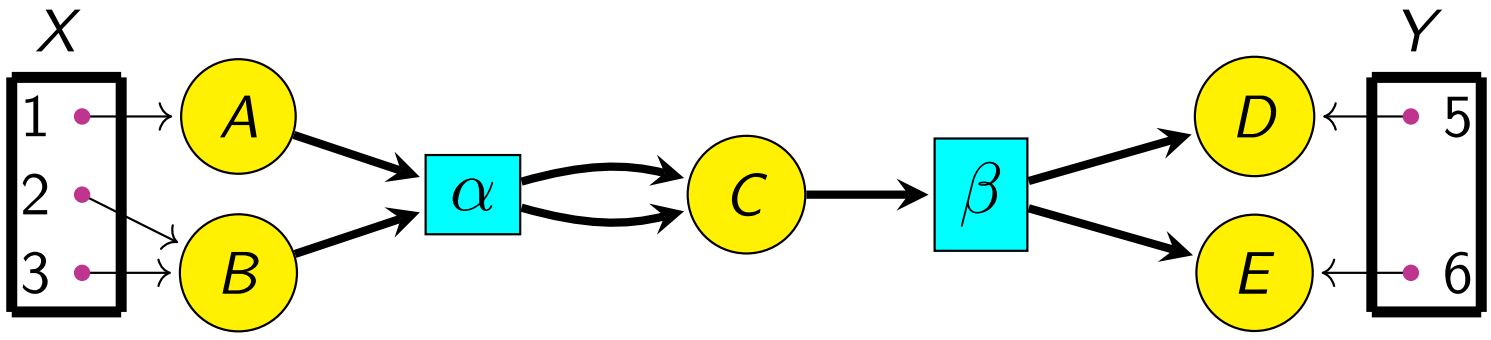




- **Petri nets with rates:** JB and Blake Pollard, A compositional framework for reaction networks.

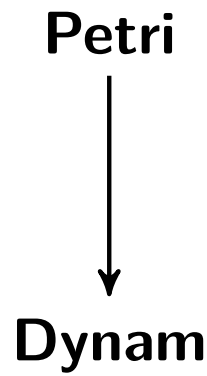


- **Petri nets with rates:** JB and Blake Pollard, A compositional framework for reaction networks.

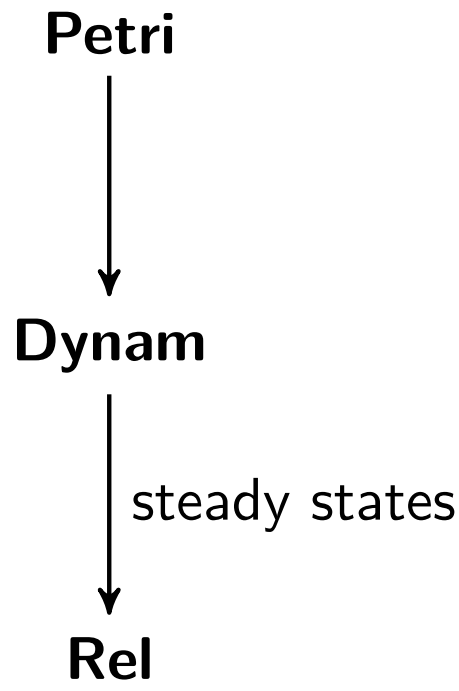


$$
\begin{aligned}
dA(t)/dt &= -\alpha\, A(t)\, B(t) + i_1(t) \\
dB(t)/dt &= -\alpha\, A(t)\, B(t) + i_2(t) + i_3(t) \\
dC(t)/dt &= 2\alpha\, A(t)\, B(t) - \beta C(t) \\
dD(t)/dt &= \beta\, C(t) - o_5(t) \\
dE(t)/dt &= \beta\, C(t) - o_6(t)
\end{aligned}
$$

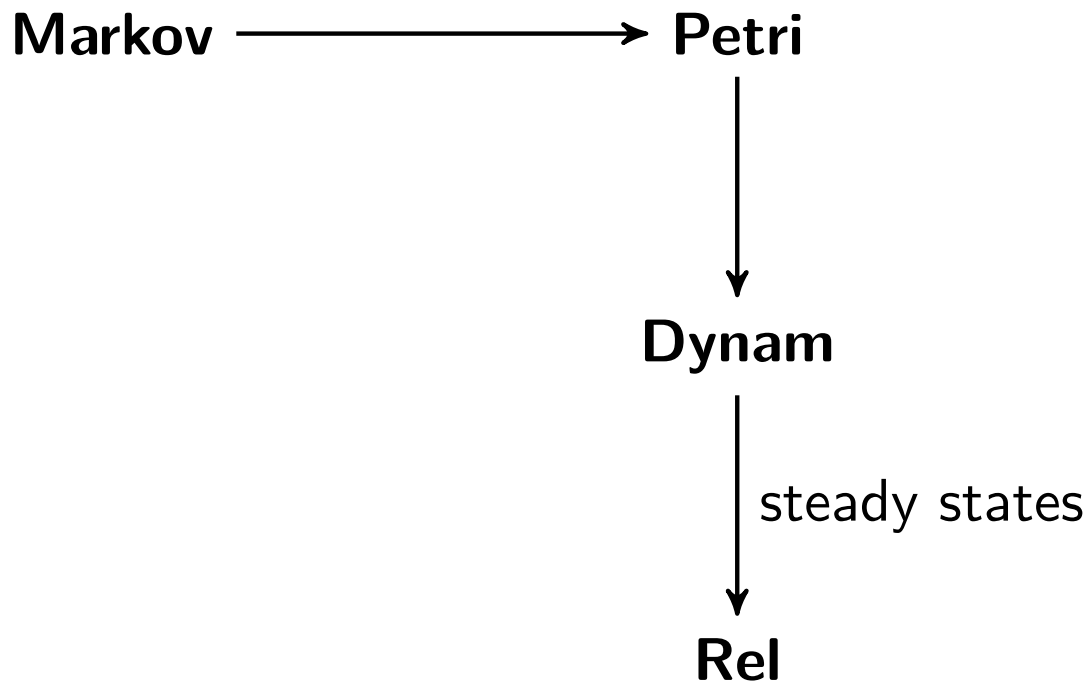
There is a 'network of network languages': interesting monoidal categories connected by monoidal functors:



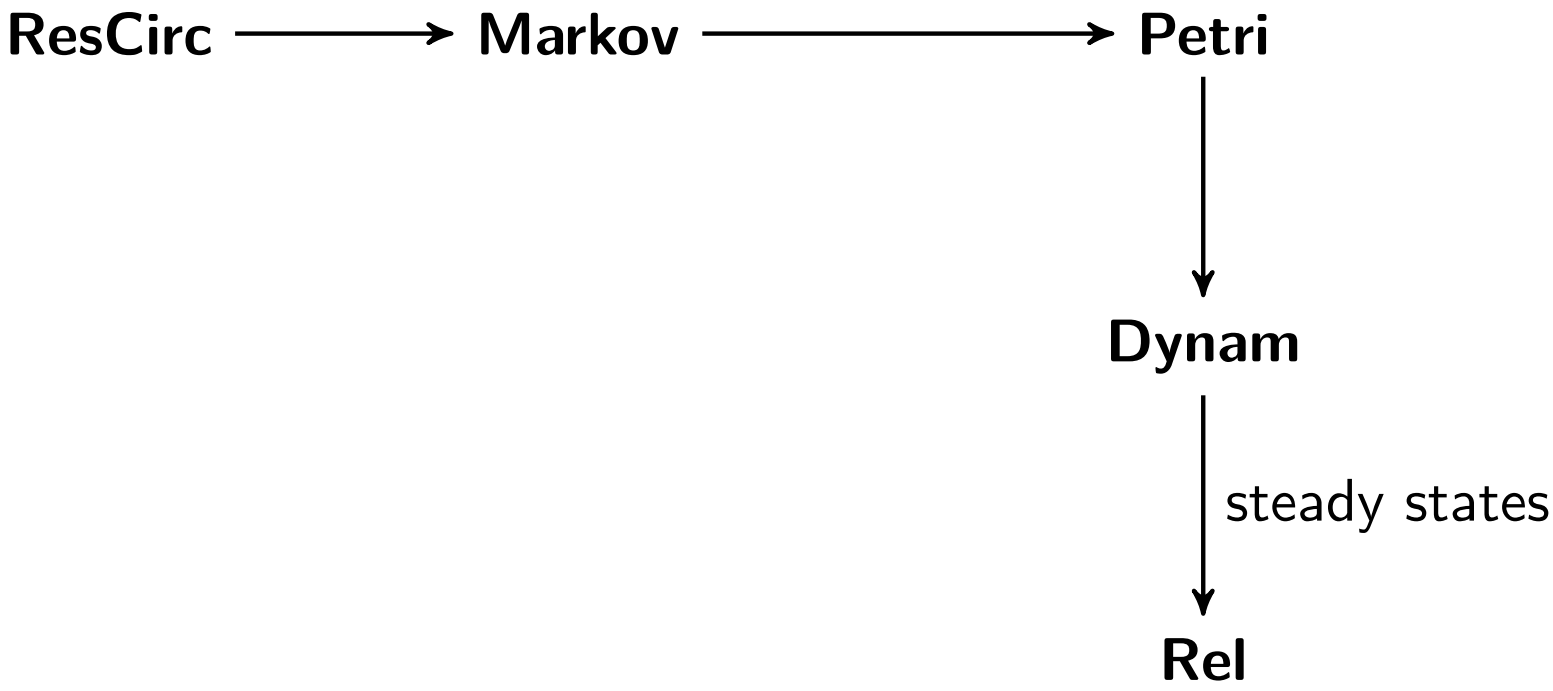
There is a 'network of network languages': interesting monoidal categories connected by monoidal functors:



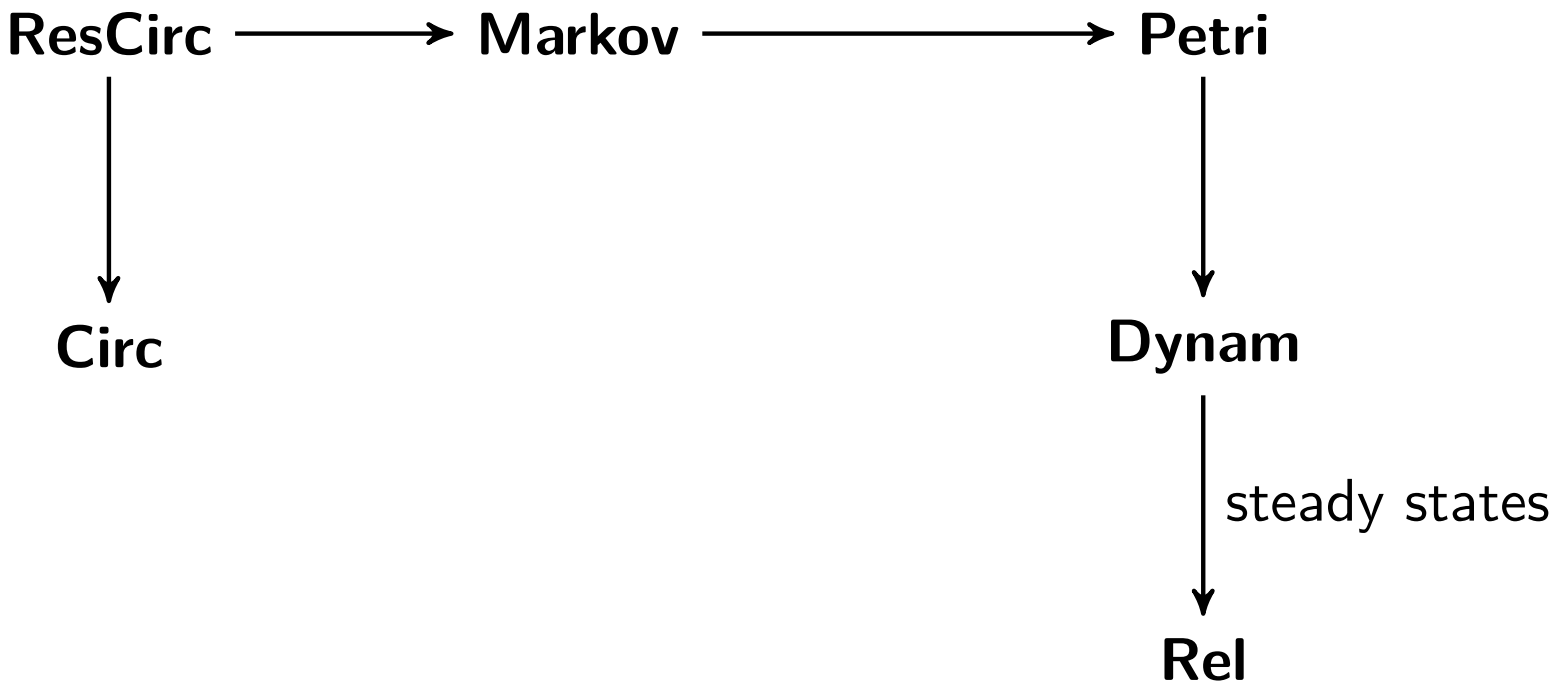
There is a 'network of network languages': interesting monoidal categories connected by monoidal functors:



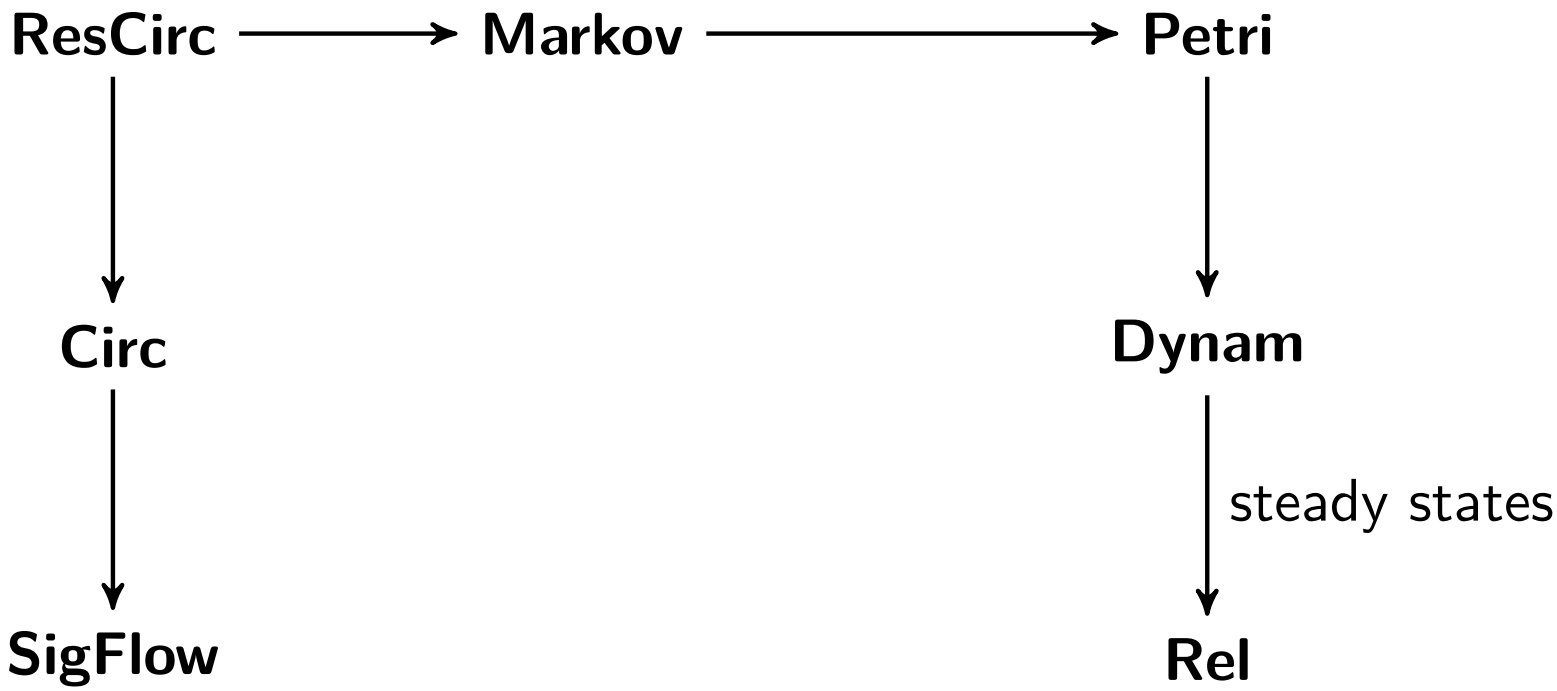
There is a 'network of network languages': interesting monoidal categories connected by monoidal functors:



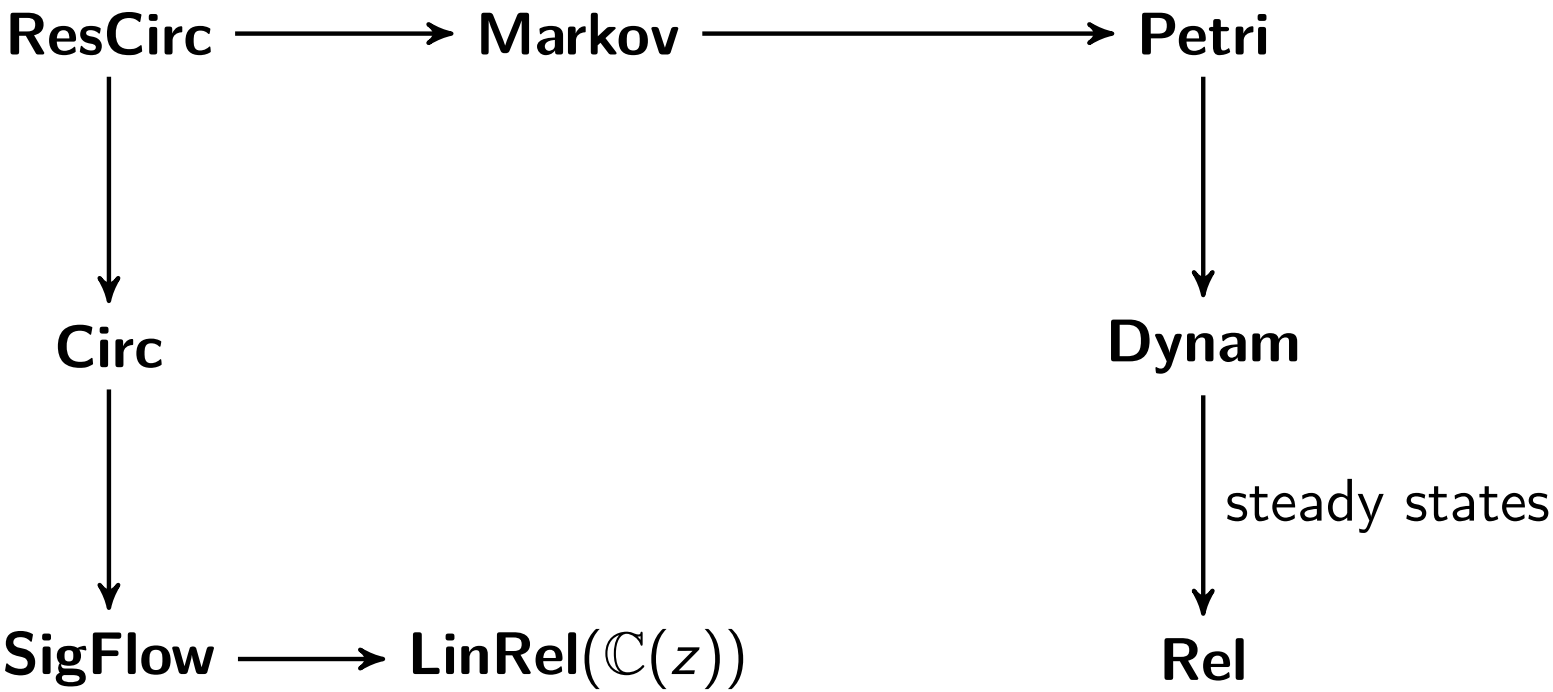
There is a 'network of network languages': interesting monoidal categories connected by monoidal functors:



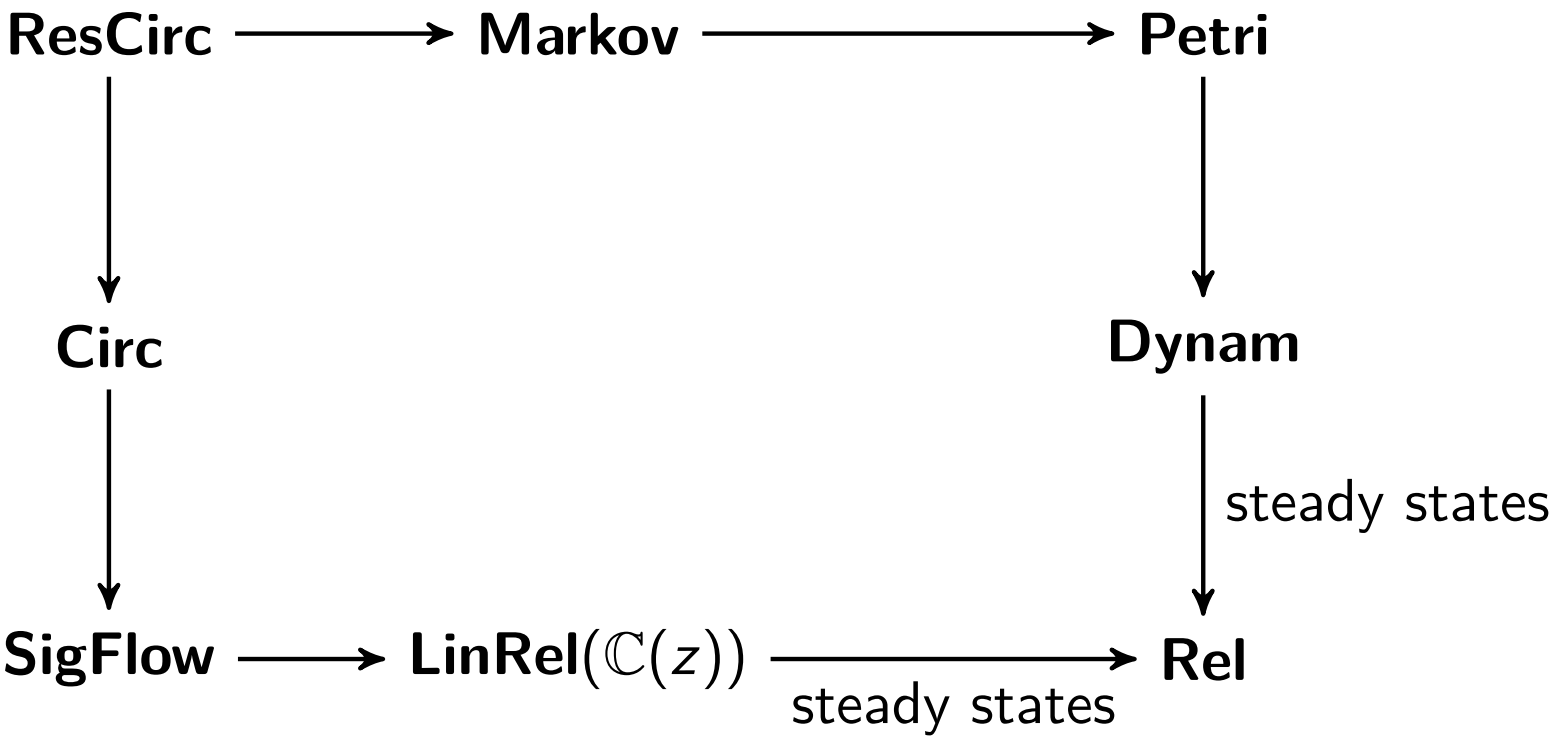
There is a 'network of network languages': interesting monoidal categories connected by monoidal functors:



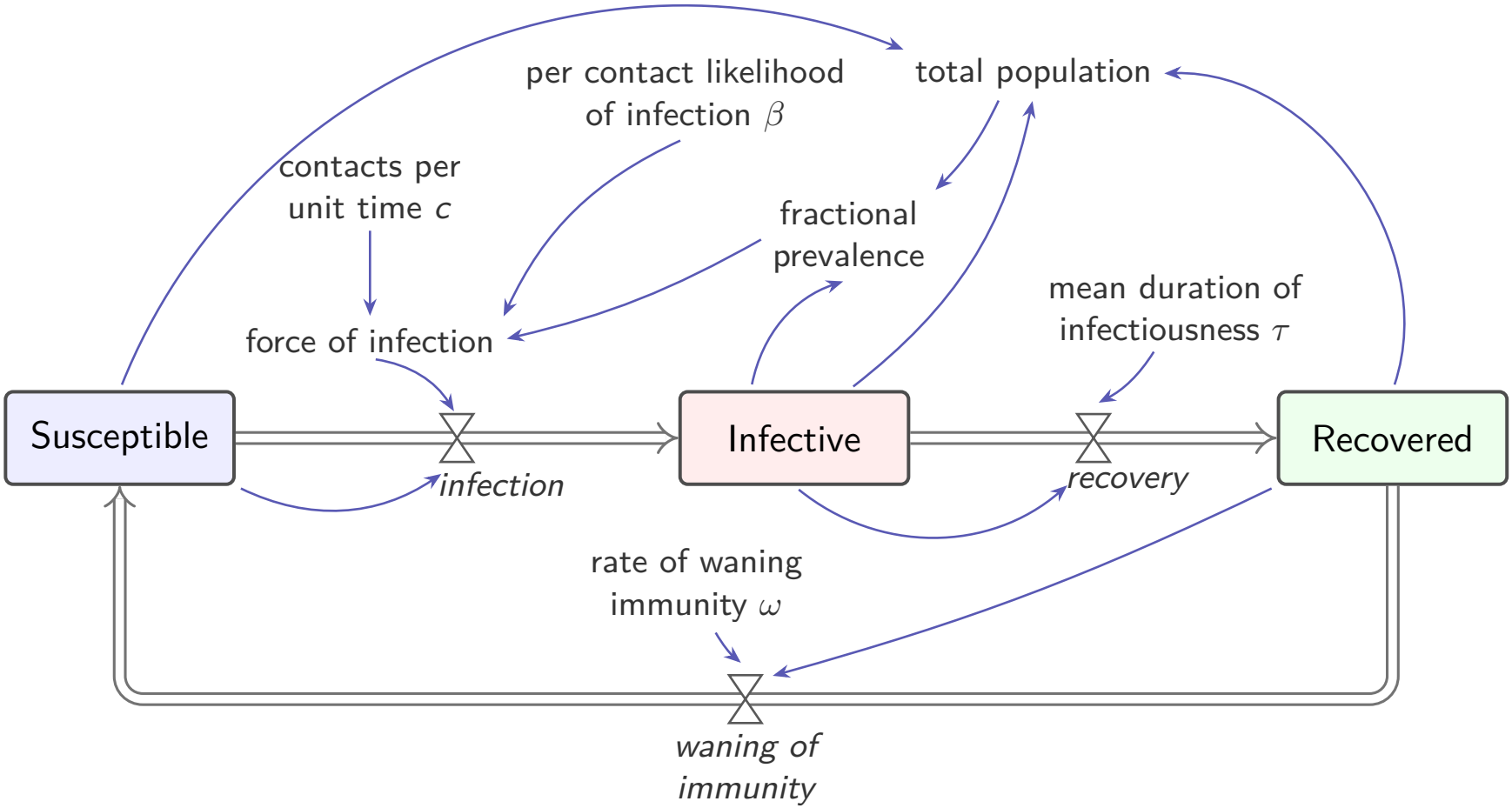
There is a 'network of network languages': interesting monoidal categories connected by monoidal functors:



There is a 'network of network languages': interesting monoidal categories connected by monoidal functors:

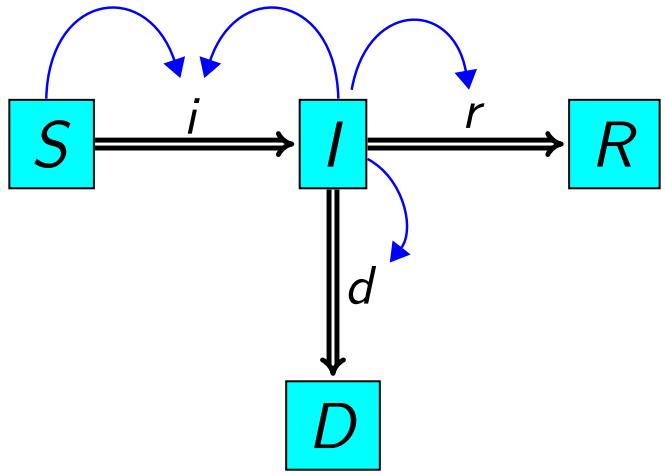


Next time I'll focus on "stock-flow diagrams":



These are widely used in economics, population biology, epidemiology, etc. They are part of a modeling discipline called System Dynamics.

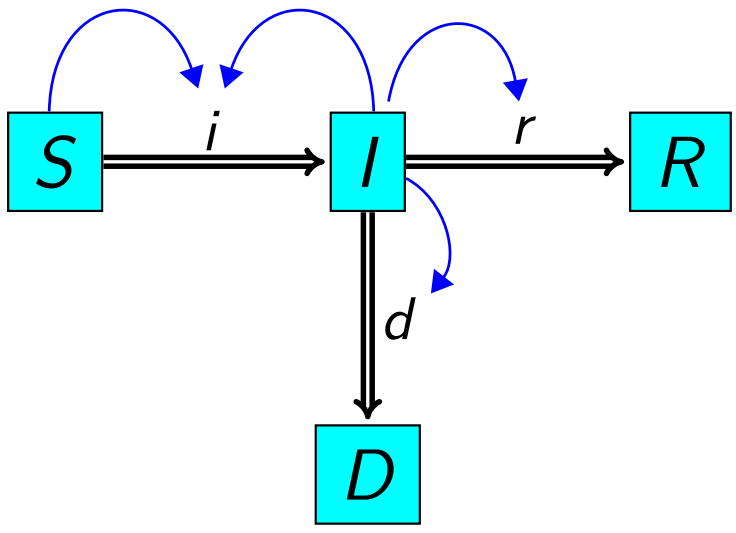
Here is a simple example of a stock-flow diagram:



The boxes are **stocks**, the double arrows are **flows**, and the blue arrows are **links** from stocks to flows.

Each flow is also equipped with a **flow function**, not drawn here, which says how the rate of that flow depends on the levels of the stocks with links pointing to it.

Any stock-flow diagram gives a dynamical system (ordinary differential equations) describing how the levels of the stocks change with time.



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

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

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

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

Later we'll see a monoidal functor sending "open" stock-flow diagrams to "open" dynamical systems, but this example is "closed".

Why don't we just write down the differential equations?

Why don't we just write down the differential equations?

One reason is that the diagram shows you the *structure* of the system.

Why don't we just write down the differential equations?

One reason is that the diagram shows you the *structure* of the system.

Another is that — shockingly — most people don't understand differential equations!

In "community-based modeling", diagrams help experts work with community members to build models of the problems they face.

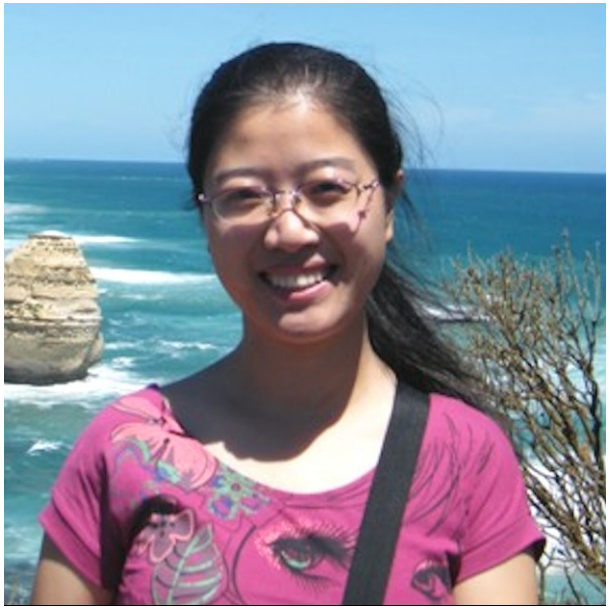


Sociologists would say the power of diagrams is that they're "boundary objects":

> *A* **boundary object** *is any object that is part of multiple social worlds and facilitates communication between them; it has a different identity in each social world that it inhabits.*

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





Most stock-flow modeling is done using software called AnyLogic. It's powerful, but it has several big problems:

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

Most stock-flow modeling is done using software called AnyLogic. It's powerful, but it has several big problems:

- It has no support for *"composing"* models: taking several smaller models and putting them together to form a larger model.
- It has no support for *"functorial semantics"*: the ability to take the same model and convert it into dynamics in several different ways.

Most stock-flow modeling is done using software called AnyLogic. It's powerful, but it has several big problems:

- It has no support for *"composing"* models: taking several smaller models and putting them together to form a larger model.
- It has no support for *"functorial semantics"*: the ability to take the same model and convert it into dynamics in several different ways.
- It has no support for *"stratifying"* models: taking a model and splitting one stock into several stocks (e.g. age groups). (In category theory this is called "taking a pullback".)

Most stock-flow modeling is done using software called AnyLogic. It's powerful, but it has several big problems:

- It has no support for *"composing"* models: taking several smaller models and putting them together to form a larger model.
- It has no support for *"functorial semantics"*: the ability to take the same model and convert it into dynamics in several different ways.
- It has no support for *"stratifying"* models: taking a model and splitting one stock into several stocks (e.g. age groups). (In category theory this is called "taking a pullback".)
- It has no support for *collaboratively* building models.

Most stock-flow modeling is done using software called AnyLogic. It's powerful, but it has several big problems:

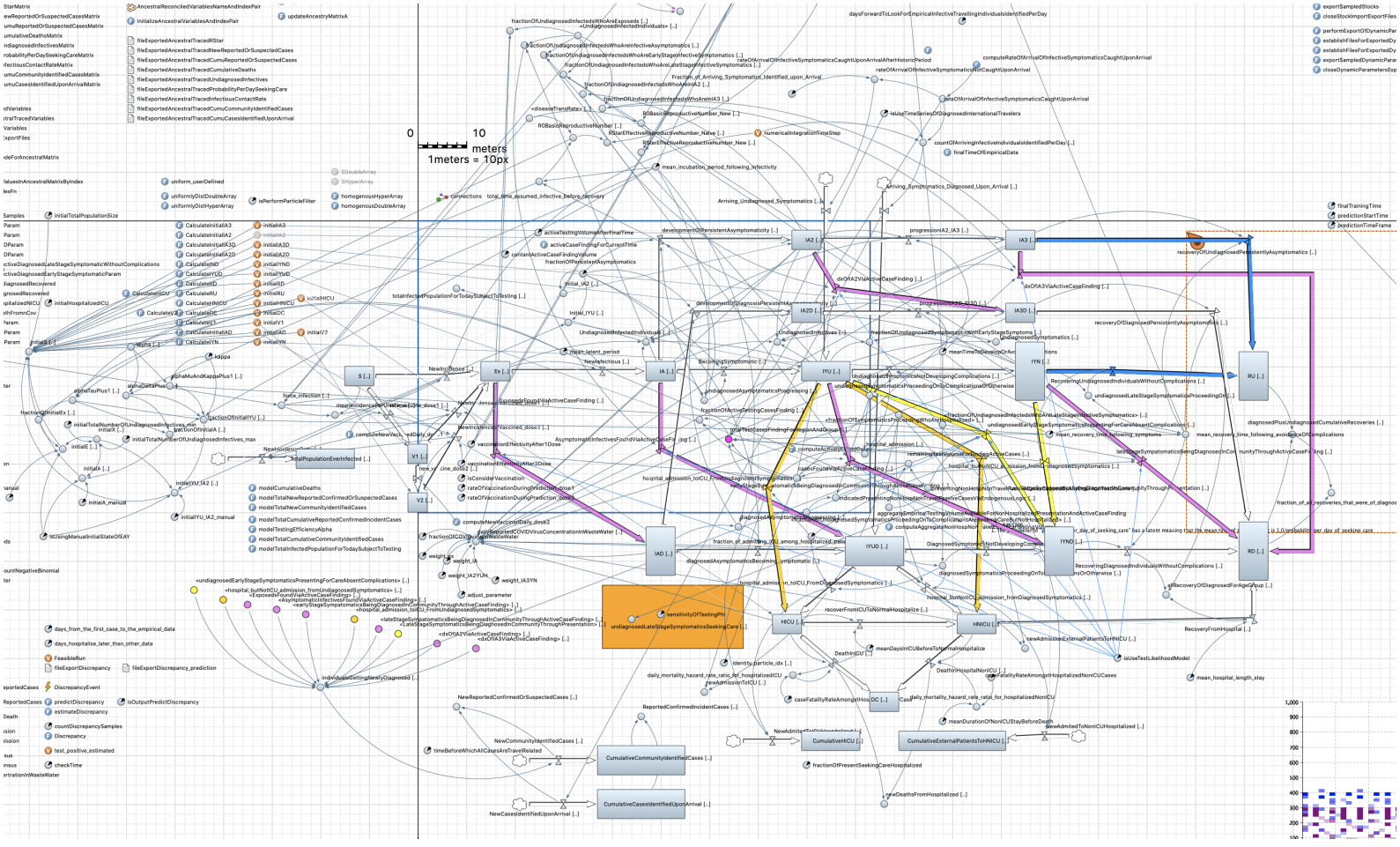
- It has no support for *"composing"* models: taking several smaller models and putting them together to form a larger model.
- It has no support for *"functorial semantics"*: the ability to take the same model and convert it into dynamics in several different ways.
- It has no support for *"stratifying"* models: taking a model and splitting one stock into several stocks (e.g. age groups). (In category theory this is called "taking a pullback".)
- It has no support for *collaboratively* building models.
- It is not *free* and not *open-source!*

Most stock-flow modeling is done using software called AnyLogic. It's powerful, but it has several big problems:

- It has no support for *"composing"* models: taking several smaller models and putting them together to form a larger model.
- It has no support for *"functorial semantics"*: the ability to take the same model and convert it into dynamics in several different ways.
- It has no support for *"stratifying"* models: taking a model and splitting one stock into several stocks (e.g. age groups). (In category theory this is called "taking a pullback".)
- It has no support for *collaboratively* building models.
- It is not *free* and not *open-source!*

Our work aims to fix all these problems.

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:



In the next parts I'll explain

- monoidal categories and functors,
- the monoidal category of stock-flow models,
- the monoidal functor mapping them to systems of ordinary differential equations,
- some category-based software for creating, composing and running stock-flow models.