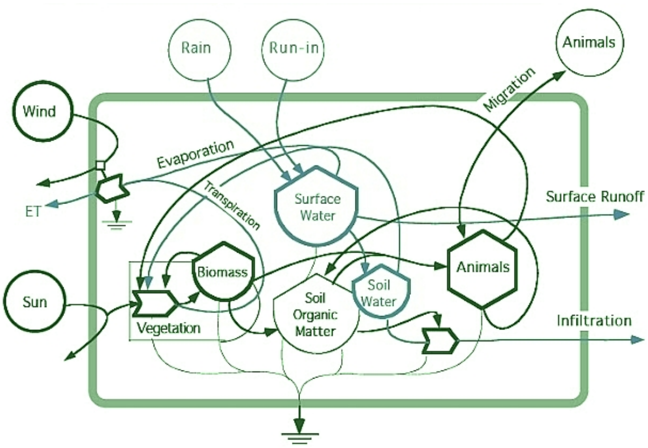
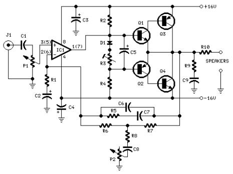
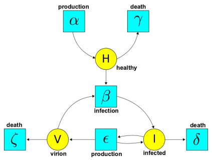
Symmetric Monoidal Categories: a Rosetta Stone

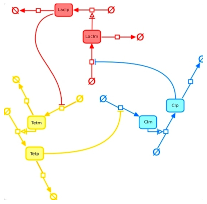


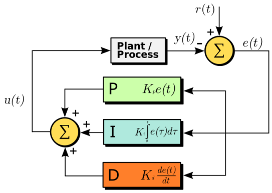
John Baez

In many branches of science and engineering, people use *diagrams of networks*, with boxes connected by wires:

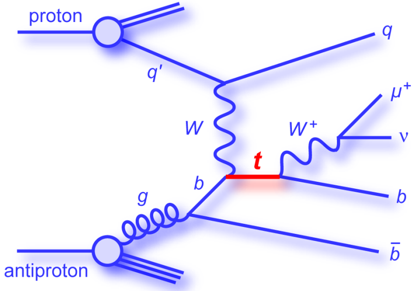






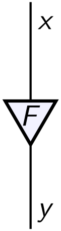


In particle physics, 'Feynman diagrams' describe processes involving elementary particles:



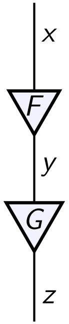
In the 1990s it became clear that mathematically Feynman diagrams depict morphisms in monoidal categories.

Categories are great for describing processes. A process with input x and output y is a **morphism** $F\colon x \to y$, and we can draw it like this:



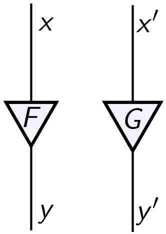
We call x and y **objects**.

We can do one process after another if the output of the first equals the input of the second:



Here we are composing morphisms $F\colon x \to y$ and $G\colon y \to z$ to get a morphism $G \circ F\colon x \to z$.

In a **monoidal** category, we can also do processes ‘in parallel’:

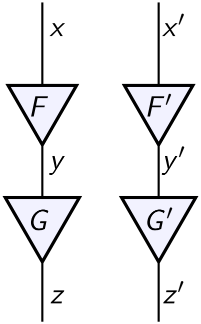


Here we are **tensoring** $F\colon x \to y$ and $G\colon x' \to y'$ to get a morphism $F \otimes G\colon x \otimes x' \to y \otimes y'$.

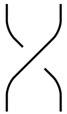
In a monoidal category, composition and tensoring must obey some laws, which all look obvious when drawn as diagrams. For example

$$(G \circ F) \otimes (G' \circ F') = (G \otimes G') \circ (F \otimes F')$$

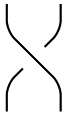
says two ways of reading this diagram agree:



In a **braided monoidal category** we also have morphisms $B_{x,y} \colon x \otimes y \to y \otimes x$ called **braidings**:

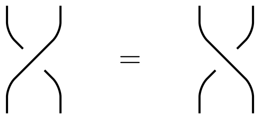


These have inverses $B^{-1}_{x,y} \colon y \otimes x \to x \otimes y$, drawn like this:



These let us draw diagrams where wires cross. Again, some obvious-looking laws must hold.

A **symmetric monoidal category** is a braided monoidal category obeying an extra law that says it doesn't matter which wire crosses which:



The most important symmetric monoidal category for traditional math is $(\mathbf{Set}, \times)$. Here the objects are *sets* and the morphisms are *functions*, and the tensor product is

$$S \times T = \{(s,t) : \; s \in S, t \in T\}.$$

The most important for quantum physics is $(\mathbf{Hilb}, \otimes)$. Here the objects are *Hilbert spaces*, the morphisms are *linear operators*, and the tensor product describes how we combine quantum systems.

The dramatic differences between $(\mathbf{Set}, \times)$ and $(\mathbf{Hilb}, \otimes)$ explain why quantum mechanics seems weird.

Logic gives us symmetric monoidal categories where objects are *statements* and a morphism $F\colon x \to y$ is a *proof* that x implies y.

Given proofs $F\colon x \to y$ and $G\colon y \to z$ we can compose them to get a proof

$$G \circ F\colon x \to z$$

Given proofs $F\colon x \to y$ and $G\colon x' \to y'$ we can tensor them to get a proof

$$F \wedge G\colon x \wedge x' \to y \wedge y'$$

Here $x \wedge y$ means "x and y".

Computer science gives us symmetric monoidal categories where objects are *data types* and a morphism $F\colon x \to y$ is a *program* that takes data of type x as input and gives data of type y as output.

Given programs $F\colon x \to y$ and $G\colon y \to z$ we can compose them to get a program

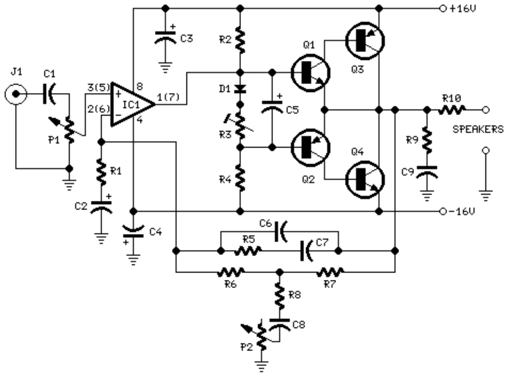
$$G \circ F\colon x \to z$$

Given programs $F\colon x \to y$ and $G\colon x' \to y'$ we can "tensor" them to get a program

$$F \times G\colon x \times x' \to y \times y'$$

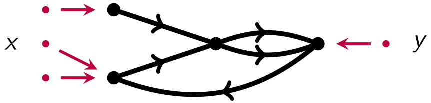
Here $x \times x'$ is a "product type".

Based on all these analogies, Brendan Fong and I started studying symmetric monoidal categories where the morphisms are *electrical circuits*:

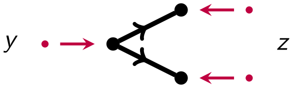


or other kinds of networks used in science and engineering.

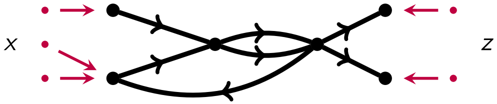
We can **compose** networks $F\colon x \to y$



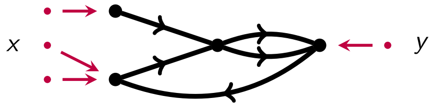
and $G\colon y \to z$



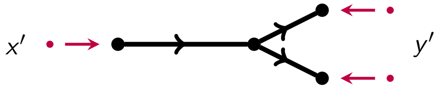
by gluing them together, obtaining $G \circ F\colon x \to z$



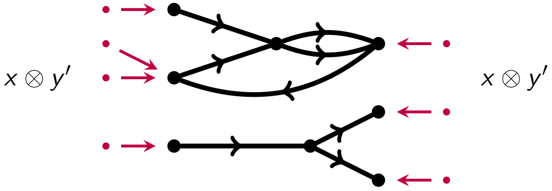
We can **tensor** networks $F\colon x \to y$



and $G\colon x' \to y'$



by setting them side by side, obtaining $F \otimes G\colon x \otimes y' \to x \otimes y'$



The math of electrical circuits also describes many other branches of engineering, thanks to these analogies:

	displacement q	flow $\dot{q}$	momentum p	effort $\dot{p}$
Electronics	charge	current	flux linkage	voltage
Mechanics (translation)	position	velocity	momentum	force
Mechanics (rotation)	angle	angular velocity	angular momentum	torque
Hydraulics	volume	flow	pressure momentum	pressure
Thermodynamics	entropy	entropy flow	temperature momentum	temperature
Chemistry	moles	molar flow	chemical momentum	chemical potential

My students have also studied symmetric monoidal categories where the morphisms are open Petri nets, reaction networks, signal flow diagrams, etc.

Quite generally, these morphisms describe “open systems”.

An open system is a system that interacts with its environment. Typically stuff — matter, energy, information, etc. — flows in and out of an open system.

By treating open systems as morphisms in symmetric monoidal categories, we formalize our ability to build bigger open systems out of smaller parts.

Physics often focuses on “closed systems”, but in engineering closed systems are useless: we need to interact with a system to use it!

Example lessons from open systems theory

1. The development of life on Earth does not violate the Second Law of Thermodynamics.

This law says the entropy of any *closed* system must increase over time.

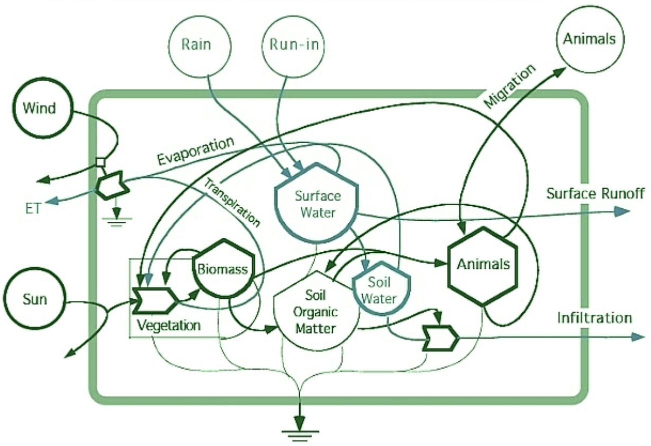
2. The so-called "collapse of the wavefunction" does not violate the unitary time evolution of quantum mechanics.

Unitary time evolution applies to *closed* systems; we need its generalization to open systems to understand measurement processes.

3. A cell phone is not a Turing machine.

A Turing machine is a *closed* system that evolves deterministically given its initial state; a cell phone is an open system.

Lastly: any sort of agent or intelligence or organism or ecosystem or manufacturing process is an open system.



Neglecting this fact is a serious mistake. We need abstractions that handle this adroitly.