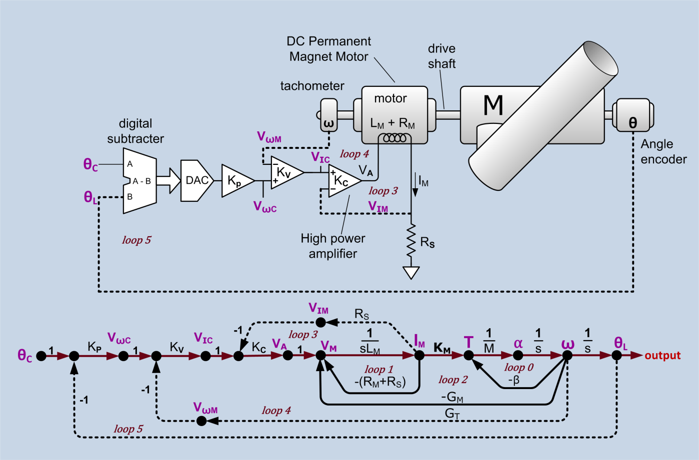
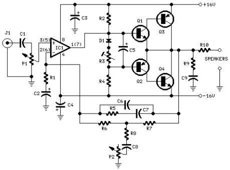
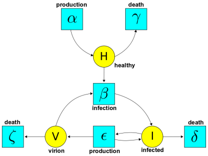
Applied Category Theory

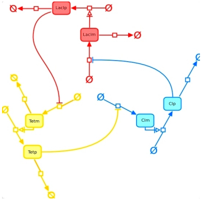


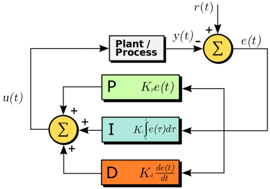
John Baez

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

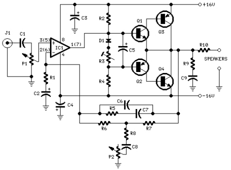


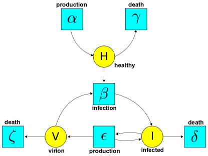


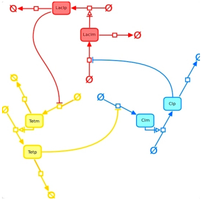


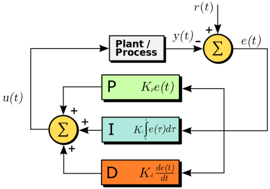


In many areas of science and engineering, people use *diagrams of networks*, with boxes connected by wires:





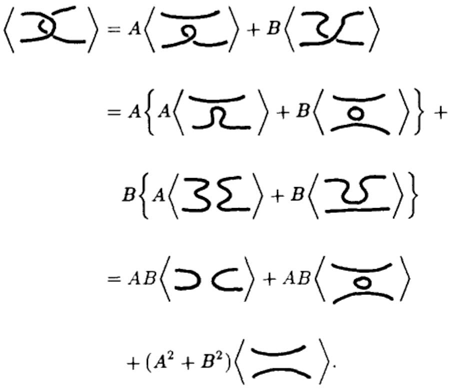




We need a good mathematical theory of these.

Categories must be part of the solution. This became clear in the 1980s, at the interface of knot theory and quantum physics:

Proof. (a)

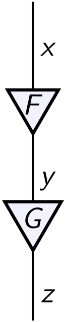


Part (b) is left for the reader.

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

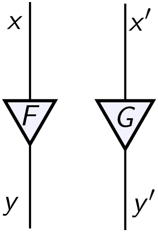
The input and output are called **objects**.

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



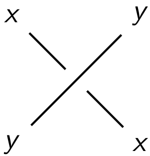
Here we are **composing** morphisms $F: x \to y$ and $G: y \to z$ to get a morphism $GF: x \to z$.

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

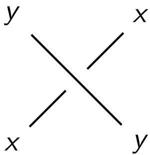


Here we are **tensoring** $F\colon x \to y$ and $G\colon x' \to y'$ to get a morphism $F \otimes G\colon x \otimes x' \to y \otimes y'$.

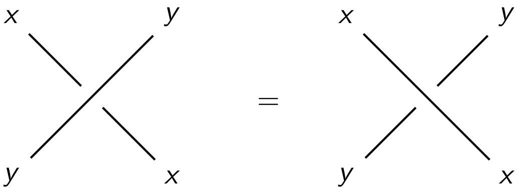
In a **braided** monoidal category, we have a process of switching:



This is called the **braiding** $B_{x,y} \colon x \otimes y \to y \otimes x$. It has an inverse:



In a **symmetric** monoidal category it doesn't matter which wire goes over which:

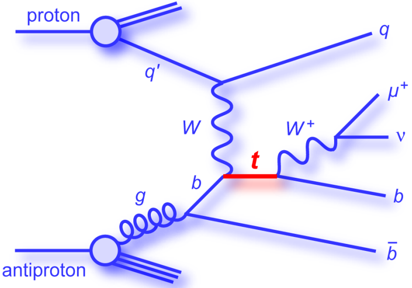


All these kinds of categories obey some axioms, which are easy to find.

The category with vector spaces as objects and linear maps between these as morphisms becomes a symmetric monoidal category with the usual $\otimes$.

The category with vector spaces as objects and linear maps between these as morphisms becomes a symmetric monoidal category with the usual $\otimes$.

In particle physics, 'Feynman diagrams' are pictures of morphisms in this category:

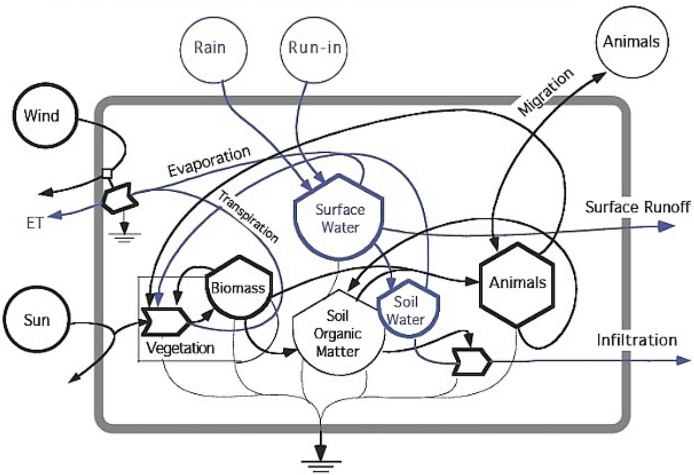


But why should particle physicists have all the fun? This is the century of biology.

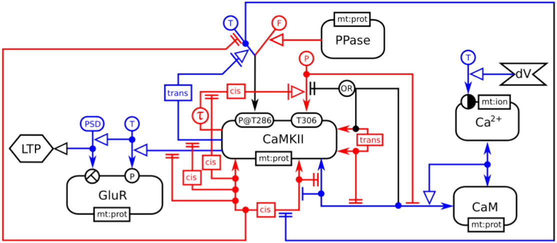
But why should particle physicists have all the fun? This is the century of biology.

Now is our chance to understand the biosphere, and stop destroying it! We should use everything we can — even mathematics — to do this.

Back in the 1950's, Howard Odum introduced an Energy Systems Language for ecology:

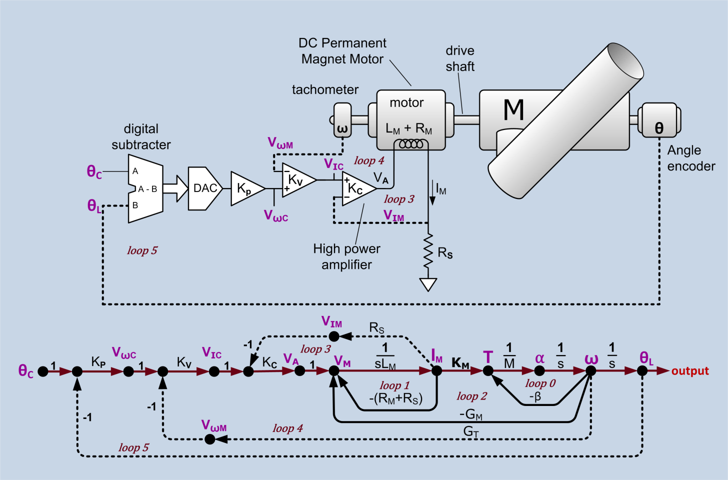


Biologists use diagrams to describe the complex processes they find in life. They use at least three different diagram languages, as formalized in Systems Biology Graphical Notation.



We should try to understand these diagrams using all the tools of modern mathematics!

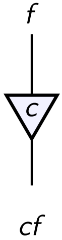
But let's start with something easier: engineering. Engineers use 'signal-flow graphs' to describe processes where signals flow through a system and interact:



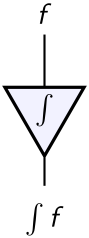
Think of a signal as a smooth real-valued function of time:

$$f : \mathbb{R} \to \mathbb{R}$$

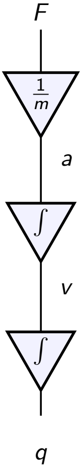
We can multiply a signal by a constant and get a new signal:



We can also integrate a signal:



Here is what happens when you push on a mass m with a time-dependent force F:



Integration introduces an ambiguity: the constant of integration. But electrical engineers often use Laplace transforms to write signals as linear combinations of exponentials

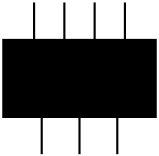
$$f(t) = e^{-st} \quad \text{for some } s > 0$$

Then they define

$$(\textstyle\int f)(t) = \frac{e^{-st}}{s}$$

This lets us think of integration as a special case of scalar multiplication! We extend our field of scalars from $\mathbb{R}$ to $\mathbb{R}(s)$, the field of rational real functions in one variable s.

Let us be general and work with an arbitrary field k. For us, any signal-flow graph with m input edges and n output edges



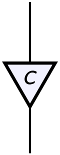
will stand for a linear map

$$F\colon k^m \to k^n$$

In other words: signal-flow graphs are pictures of morphisms in $\mathrm{FinVect}_k$, the category of finite-dimensional vector spaces over k...
where we make this into a monoidal category using $\oplus$, not $\otimes$.

We build these pictures from a few simple 'generators'.

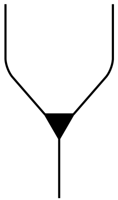
First, we have scalar multiplication:



This is a notation for the linear map

$$\begin{array}{ccc} k & \to & k \\ f & \mapsto & cf \end{array}$$

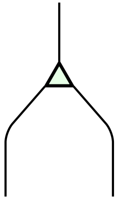
Second, we can add two signals:



This is a notation for

$$+ \colon k \oplus k \to k$$

Third, we can 'duplicate' a signal:



This is a notation for the diagonal map

$$\begin{array}{rccc} \Delta\colon & k & \to & k \oplus k \\ & f & \mapsto & (f, f) \end{array}$$

Fourth, we can ‘delete’ a signal:

This is a notation for the linear map

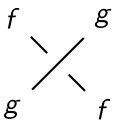
$$\begin{array}{rcl} k & \to & \{0\} \\ f & \mapsto & 0 \end{array}$$

Fifth, we have the zero signal:

This is a notation for the linear map

$$\begin{array}{ccc} \{0\} & \to & k \\ 0 & \mapsto & 0 \end{array}$$

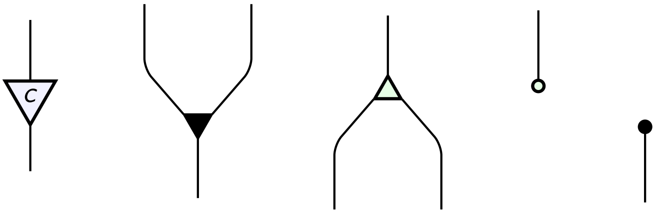
Furthermore, $(\mathrm{FinVect}_k, \oplus)$ is a *symmetric* monoidal category. This means we have a 'braiding': a way to switch two signals:



This is a notation for the linear map

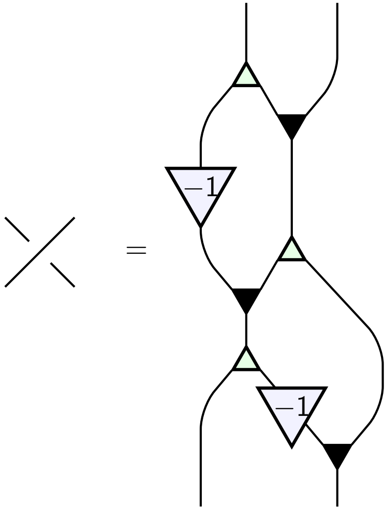
$$\begin{array}{ccc} k \oplus k & \to & k \oplus k \\ (f, g) & \mapsto & (g, f) \end{array}$$

From these ‘generators’:



together with the braiding, we can build complicated signal flow diagrams. In fact, we can describe *any* linear map $F\colon k^m \to k^n$ this way!

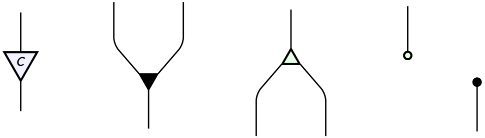
But these generators obey some unexpected relations:



Luckily, we can derive *all* the relations from some very nice ones!

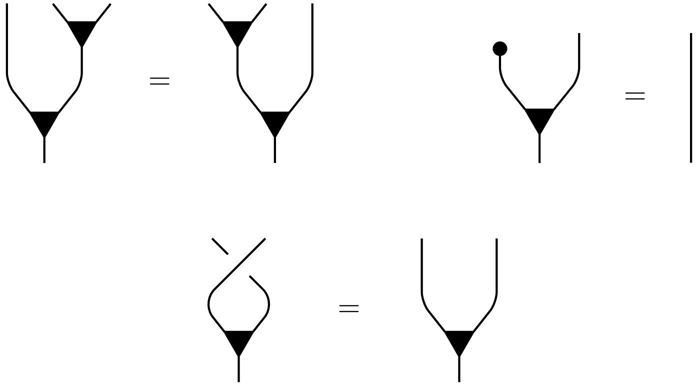
Theorem (Jason Erbele)

$FinVect_k$ is equivalent to the symmetric monoidal category generated by the object k and these morphisms:

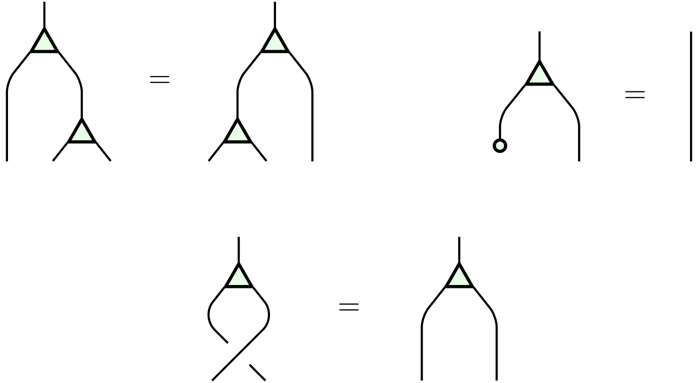


where $c \in k$, with the following relations.

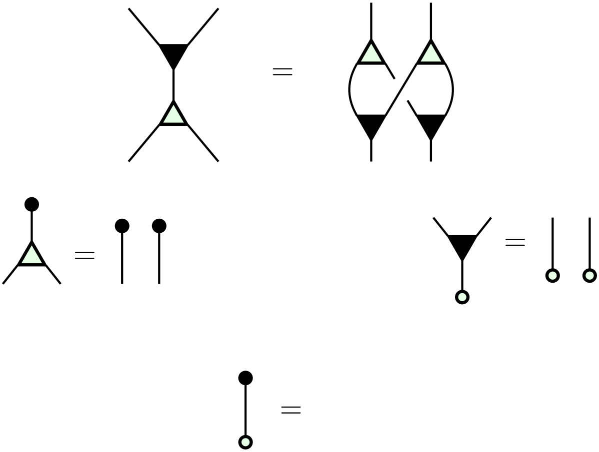
Addition and zero make k into a commutative monoid:



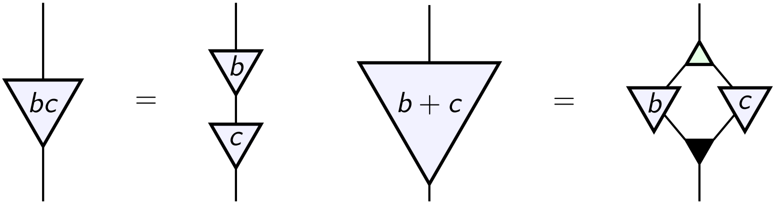
Duplication and deletion make k into a cocommutative comonoid:

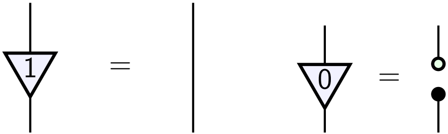


The monoid and comonoid operations are compatible, giving a **bimonoid**:

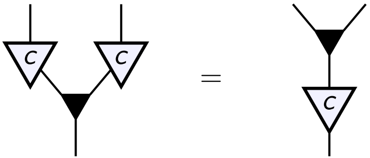


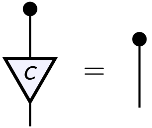
The ring structure of k can be recovered from the generators:



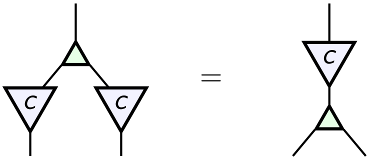


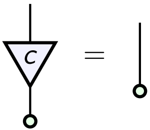
Scalar multiplication is linear (compatible with addition and zero):





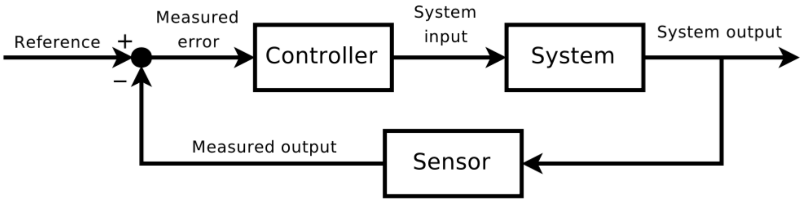
Scalar multiplication is 'colinear' (compatible with duplication and deletion):





Those are all the relations we need!

However, control theory also needs more general signal-flow graphs, which have ‘feedback loops’:



This is the most important concept in control theory: letting the output of a system affect its input.

To allow feedback loops we need morphisms more general than linear maps. We need linear relations!

A **linear relation** $F \colon U \rightsquigarrow V$ from a vector space U to a vector space V is a linear subspace $F \subseteq U \oplus V$.

We can compose linear relations $F \colon U \rightsquigarrow V$ and $G \colon V \rightsquigarrow W$ and get a linear relation $G \circ F \colon U \rightsquigarrow W$:

$$G \circ F = \{(u, w) : \exists v \in V \;\; (u, v) \in F \text{ and } (v, w) \in G\}.$$

A linear map $\phi\colon U \to V$ gives a linear relation $F\colon U \rightsquigarrow V$, namely the graph of that map:

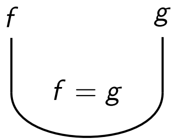
$$F = \{(u, \phi(u)) : u \in U\}$$

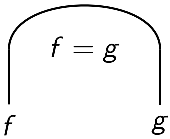
Composing linear maps becomes a special case of composing linear relations.

There is a symmetric monoidal category FinRel_k with finite-dimensional vector spaces over the field k as objects and linear relations as morphisms. This has $\mathrm{FinVect}_k$ as a subcategory.

Fully general signal-flow diagrams are pictures of morphisms in FinRel_k, typically with $k = \mathbb{R}(s)$.

Jason Erbele showed that besides the previous generators of $\mathrm{FinVect}_k$, we only need two more morphisms to generate all the morphisms in FinRel_k: the 'cup' and 'cap'.





These linear relations say that when a signal goes around a bend in a wire, the signal coming out equals the signal going in!

More formally, the **cup** is the linear relation

$$\cup \colon k \oplus k \rightsquigarrow \{0\}$$

that is, the subspace

$$\cup \ \subseteq \ k \oplus k \oplus \{0\}$$

given by:

$$\cup = \{(f, f, 0) : f \in k\}$$

More formally, the **cup** is the linear relation

$$\cup\colon k \oplus k \rightsquigarrow \{0\}$$

that is, the subspace

$$\cup \ \subseteq \ k \oplus k \oplus \{0\}$$

given by:

$$\cup = \{(f, f, 0) : f \in k\}$$

Similarly, the **cap**

$$\cap\colon \{0\} \rightsquigarrow k \oplus k$$

is the subspace

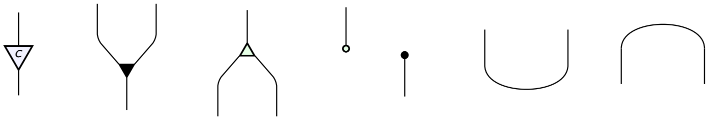
$$\cap \ \subseteq \ \{0\} \oplus k \oplus k$$

given by:

$$\cap = \{(0, f, f) : f \in k\}$$

Theorem (Jason Erbele)

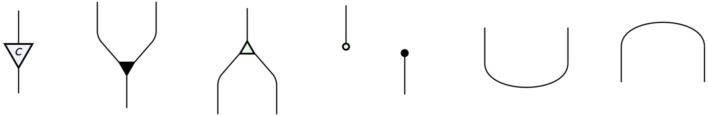
FinRel_k is equivalent to the symmetric monoidal category generated by the object k and these morphisms:



where $c \in k$, and an explicit list of relations.

Theorem (Jason Erbele)

FinRel_k is equivalent to the symmetric monoidal category generated by the object k and these morphisms:



where $c \in k$, and an explicit list of relations.

For details, see:

- J. Baez and Jason Erbele, Categories in control.
- Filippo Bonchi, Pawel Sobocinski and Fabio Zanasi, Interacting Hopf algebras.

Besides signal-flow diagrams, we have also analyzed the categories where morphisms are:

- electrical circuits

Besides signal-flow diagrams, we have also analyzed the categories where morphisms are:

- electrical circuits
- open Markov processes

Besides signal-flow diagrams, we have also analyzed the categories where morphisms are:

- electrical circuits
- open Markov processes
- open Petri nets

Besides signal-flow diagrams, we have also analyzed the categories where morphisms are:

- electrical circuits
- open Markov processes
- open Petri nets
- open chemical reaction networks

Besides signal-flow diagrams, we have also analyzed the categories where morphisms are:

- electrical circuits
- open Markov processes
- open Petri nets
- open chemical reaction networks

So, we are working toward a unified theory of networks — but there's a lot more to do!

For more, see:

http://math.ucr.edu/home/baez/networks/