

# STRUCTURED COSPANS

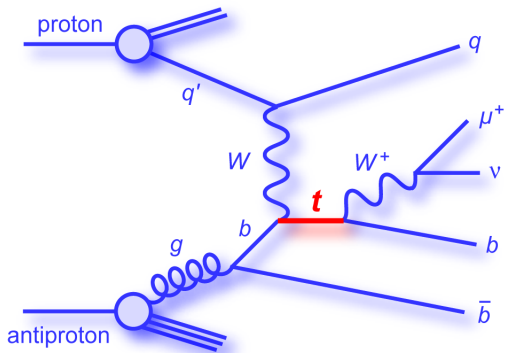


**John Baez and Kenny Courser**

QPL 2019

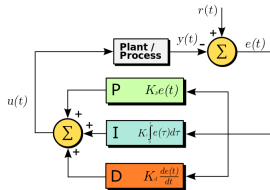
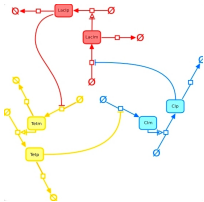
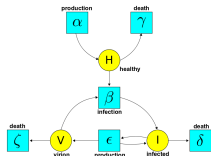
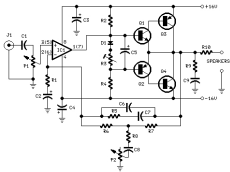
June 12, 2019

In the 1990s we learned something amazing! Feynman diagrams in particle physics are string diagrams in symmetric monoidal categories...



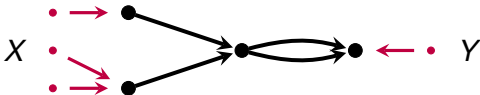
... so the world is made of categories!

Throughout science and engineering, people use *networks*, drawn as boxes connected by wires:



So, they're using categories! Which categories are these?

Networks of some particular kind, with specified inputs and outputs, can be seen as morphisms in some symmetric monoidal category:



Such networks let us describe *open* systems, meaning systems where:

- ▶ stuff can flow in or out;
- ▶ we can combine systems to form larger systems by *composition* and *tensoring*.

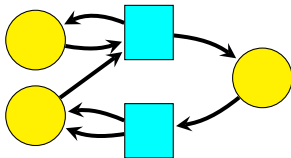
How can we construct symmetric monoidal categories? There are ‘algebraic’ and ‘geometrical’ ways to do this.


The ‘algebraic’ way is to present a symmetric monoidal category by specifying:

- ▶ generating objects,
- ▶ generating morphisms, and
- ▶ relations between morphisms.

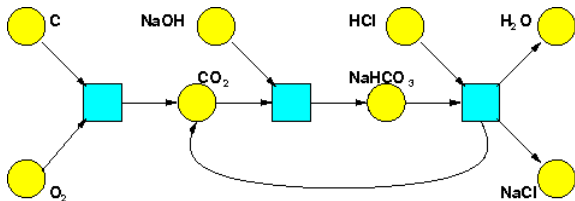
If we don’t need any relations, we can do this using a ‘Petri net’.

A **Petri net** is a bipartite directed multigraph:

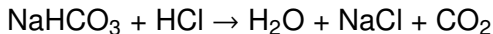
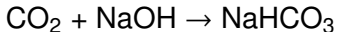
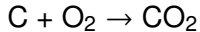


We call the two kinds of vertices **places**  and **transitions** .

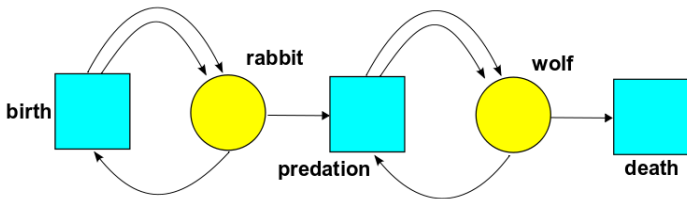
Petri nets became popular in computer science starting in the 1970s. But they were invented for chemistry in 1939:



as an alternative to the more familiar 'reaction networks':



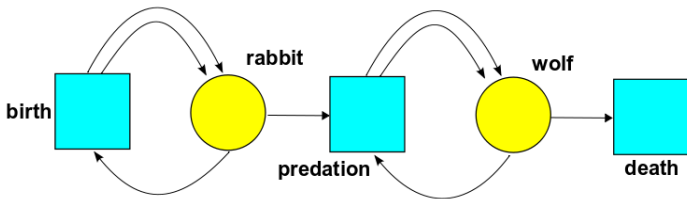
Petri nets are also used in models of population biology and epidemiology:





Any Petri net gives a 'freely generated' symmetric monoidal category where:

- ▶ the objects are tensor products of places
- ▶ the morphisms are tensor products and composites of transitions

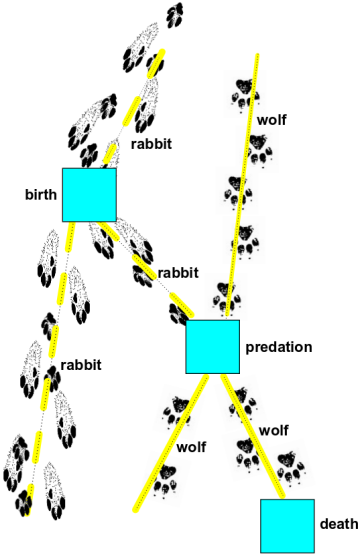


$$\text{birth} : R \rightarrow R \otimes R$$

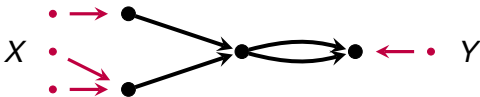
$$\text{predation} : R \otimes W \rightarrow W \otimes W$$

$$\text{death} : W \rightarrow I$$

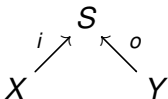
Morphisms in this category can be drawn as string diagrams:



The 'geometrical' way to describe symmetric monoidal categories is to use cospans with extra structure. For example, this:



is really a cospan of finite sets:



where  $S$  is decorated with extra structure: edges making  $S$  into the vertices of a graph.

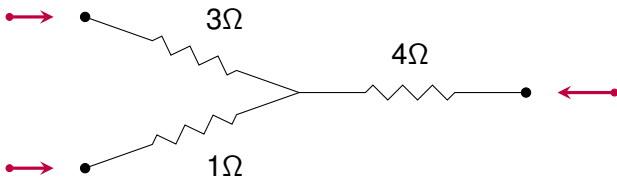
Fong invented 'decorated cospans' to make this precise:

- ▶ Brendan Fong, [Decorated cospans](https://arxiv.org/abs/1502.00872), arXiv:1502.00872.

Decorated cospans are good for studying categories where the morphisms are networks.

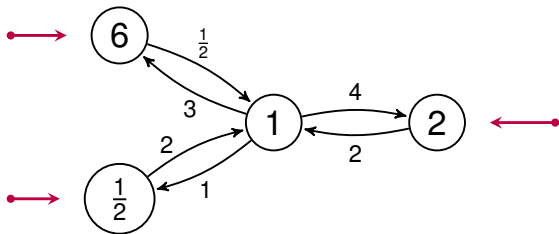
Electrical circuits:

- ▶ JB and Brendan Fong, [A compositional framework for passive linear networks](#), arXiv:1504.05625.



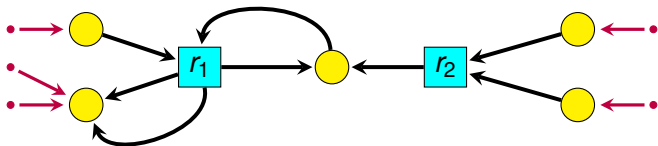
Markov processes:

- ▶ JB, Brendan Fong and Blake Pollard, [A compositional framework for Markov processes](#), arXiv:1508.06448.



Petri nets with rates:

- ▶ JB and Blake Pollard, [A compositional framework for reaction networks](#), arXiv:1704.02051.



Now Kenny Courser has developed a simpler formalism — ‘structured cospans’ — that avoids certain problems with decorated cospans.

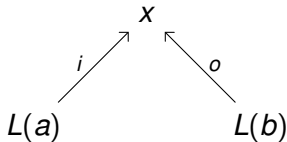
Kenny has redone most of the previous work using structured cospans:

- ▶ Kenny Courser, *Open Systems: A Double Categorical Perspective*, <https://tinyurl.com/courser-thesis>.

Given a functor

$$L: A \rightarrow X$$

a **structured cospan** is a diagram



Think of  $A$  as a category of objects with 'less structure', and  $X$  as a category of objects with 'more structure'.



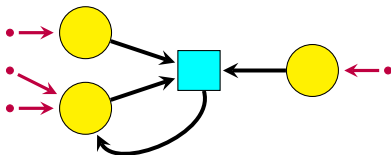
For example, there's a category of Petri nets, and a functor

$$L: \text{Set} \rightarrow \text{Petri}$$

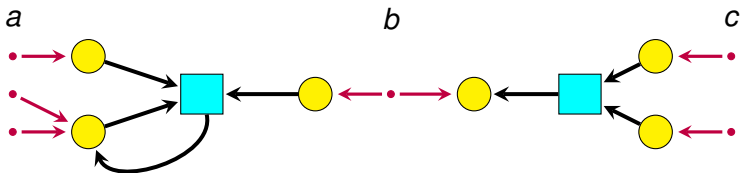
sending any set to the Petri net with that set of places, and no transitions. In this case, a structured cospan

$$\begin{array}{ccc} & X & \\ i \nearrow & & \nwarrow o \\ L(a) & & L(b) \end{array}$$

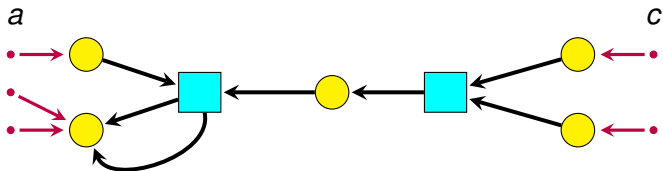
is called an **open Petri net**:



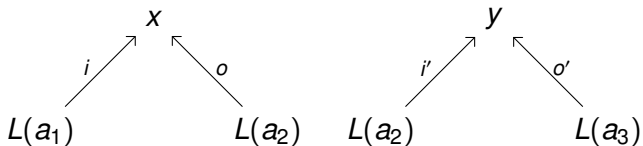
To compose open Petri nets  $f: a \rightarrow b$  and  $g: b \rightarrow c$ :



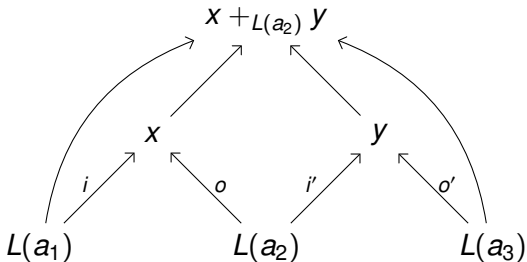
we identify the outputs of  $f$  with the inputs of  $g$ :



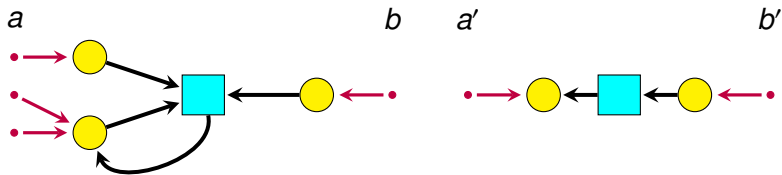
In other words, given open Petri nets:



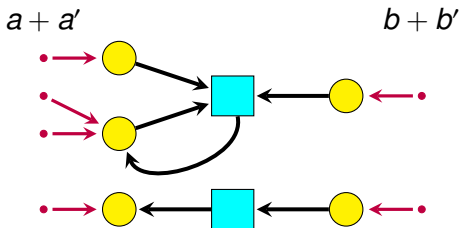
we compose them by taking a pushout in the category Petri:



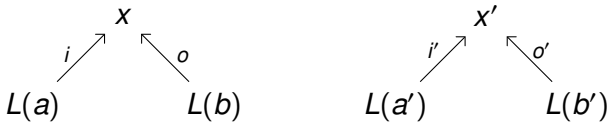
To tensor open Petri nets  $f: a \rightarrow b$  and  $f': a' \rightarrow b'$ :



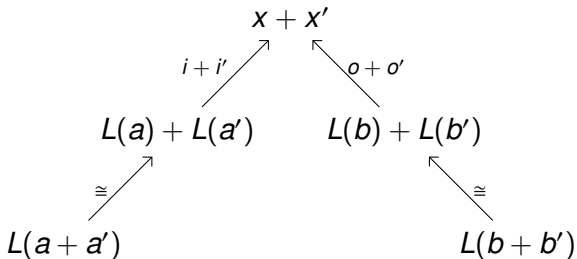
we set them side by side:



In other words, to tensor open Petri nets:



we use coproducts in Set and Petri:



and the fact that  $L: \text{Set} \rightarrow \text{Petri}$  preserves coproducts.

In general:

### Theorem (JB, Kenny Courser)

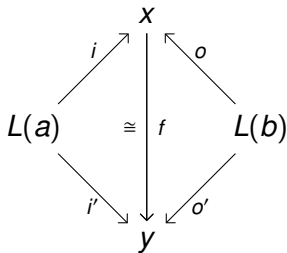
*Let  $A$  be a category with finite coproducts,  $X$  a category with finite coproducts and pushouts, and  $L: A \rightarrow X$  a functor preserving finite coproducts.*

*Then there is a symmetric monoidal category  ${}_L\text{Csp}(X)$  where:*

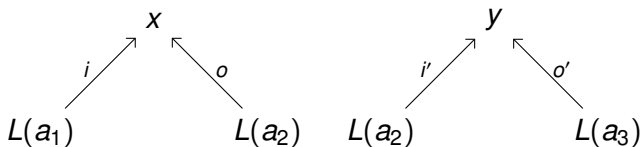
- ▶ an object is an object of  $A$ ;*
- ▶ a morphism is an isomorphism class of structured cospans — that is, diagrams in  $X$  of the form:*

$$\begin{array}{ccc} & X & \\ i \nearrow & & \nwarrow o \\ L(a) & & L(b) \end{array}$$

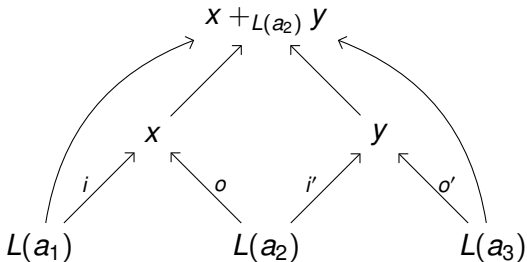
Here two structured cospans are **isomorphic** if there is a commuting diagram of this form:



We compose structured cospans

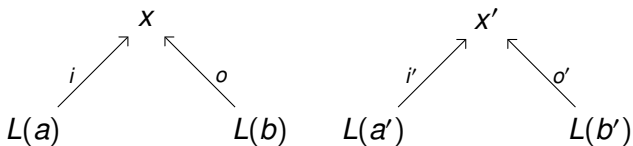


using a pushout in  $\mathbf{X}$ :

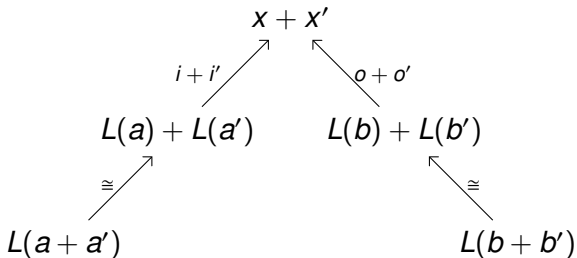




and we tensor structured cospans:



using coproducts in A and X:



Moreover:

### Theorem (JB, Kenny Courser)

*Under the same conditions,  ${}_L\text{Csp}(X)$  is a hypergraph category.*

It follows that:

- ▶  ${}_L\text{Csp}(X)$  is a dagger-category, so every morphism  $f: a \rightarrow b$  can be turned around to give a morphism  $f^\dagger: b \rightarrow a$ .
- ▶ Every object  $a \in {}_L\text{Csp}(X)$  is a commutative Frobenius monoid and thus its own dual.
- ▶ Duals and daggers interact well:  ${}_L\text{Csp}(X)$  is dagger-compact.

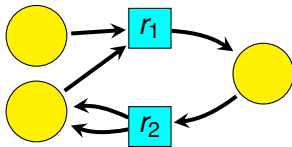
These theorems apply to many examples, giving structured cospan categories whose morphisms are:

- ▶ open electrical circuits
- ▶ open Markov processes
- ▶ open Petri nets
- ▶ open Petri nets with rates

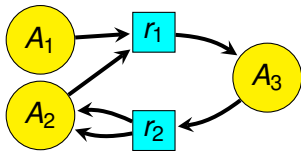
etcetera. In all these examples  $A$  and  $X$  have finite colimits and  $L: A \rightarrow X$  is a left adjoint, so all the conditions of the theorems hold!

Let's look at one example: open Petri nets with rates.

In a Petri net **with rates**, each transition is assigned a **rate constant**: a positive real number. We can then write down a 'rate equation' describing dynamics. For example, this Petri net with rates:



In a Petri net **with rates**, each transition is assigned a **rate constant**: a positive real number. We can then write down a 'rate equation' describing dynamics. For example, this Petri net with rates:



gives this rate equation:

$$\frac{dA_1}{dt} = -r_1 A_1 A_2$$

$$\frac{dA_2}{dt} = -r_1 A_1 A_2 + 2r_2 A_3$$

$$\frac{dA_3}{dt} = r_1 A_1 A_2 - r_2 A_3$$

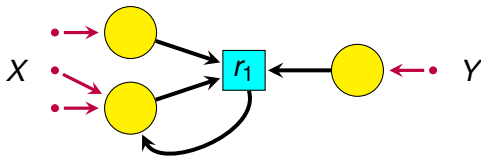
There's a category  $\text{Petri}_r$  of Petri nets with rates, and a functor

$$L: \text{Set} \rightarrow \text{Petri}_r$$

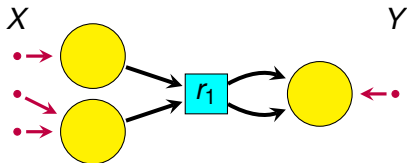
sending any set to the Petri net with rates having that set of places, and no transitions.

Our theorems apply, so we get a symmetric monoidal category  $\text{Open}(\text{Petri}_r)$  where:

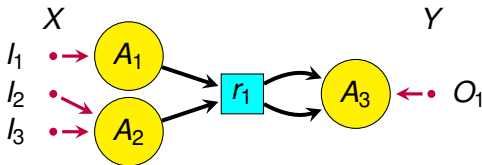
- ▶ an object is a finite set;
- ▶ a morphism  $f: X \rightarrow Y$  is a Petri net with rates together with functions from  $X$  and  $Y$  to its set of places:



An open Petri net with rates  $f: X \rightarrow Y$  gives an **open rate equation** involving flows in and out, which can be arbitrary smooth functions of time. For example this:



An open Petri net with rates  $f: X \rightarrow Y$  gives an **open rate equation** involving flows in and out, which can be arbitrary smooth functions of time. For example this:



gives:

$$\frac{dA_1}{dt} = -r_1 A_1 A_2 + l_1(t)$$

$$\frac{dA_2}{dt} = -r_1 A_1 A_2 + l_2(t) + l_3(t)$$

$$\frac{dA_3}{dt} = 2r_1 A_1 A_2 - O_1(t)$$



In fact, the map sending open Petri nets to their open rate equations is a symmetric monoidal functor

$$\square: \text{Open}(\text{Petri}_r) \rightarrow \text{Dynam}$$

where  $\text{Dynam}$  is a category of ‘open dynamical systems’.

So, we can describe dynamical systems *compositionally*, a piece at a time, using open Petri nets with rates.

That's what structured cospans are good for:

- ▶ describing networks and how to build bigger networks from smaller ones, using categories;
- ▶ describing the *behavior* of networks in a compositional way, using functors.