Open Systems for the Working Mathematician

Owen Lynch

April 29, 2020

Contents

1 Introduction

1.1 The Purpose of this Review

In this review, we attempt to collect and summarize recent work that has been happening across the field of applied category theory around open systems. There is a good deal of work put into making open systems accessible to the scientist who was previously unfamiliar with categories. Therefore, in order to make an original contribution, in this review we attempt to make open systems accessible to the category theorist who was previously unfamiliar with science.

This review is organized in a grid fashion. In the second section, we will introduce three application domains for the later study of theory. In each subsequent section, when we introduce a new concept we will talk about how that subject interfaces with all three application domains. Our hope is that this will congeal the concepts presented, and make it more clear how they tie together.

Before we dive into the math, however, we need to talk about what open systems mean philosophically, and before we do that, we need to talk about systems.

1.2 What is a System?

This has an easy answer: a system is an object whose class of behaviors we want to study. This seems unsatisfying, but from a mathematical perspective what this is saying is that "system" is a term left undefined, like "point" or "line". We will develop a theory around "systems" that gives them meaning from context (though this theory will not be nearly as rigorous as Euclidean geometry).

We can apply various adjectives to "system" that narrow the type of behavior associated with a system. For instance, one large class of systems is "dynamical systems". Dynamical systems are characterized by a focus on evolving a system over time. Typically, this is done by means of differential equations.

Example 1. Consider the cylinder $S^1 \times \mathbb{R}$, which we use to represent the possible position, velocity pairs of a pendulum. Let $(\theta, \dot{\theta})$ be coordinates for the cylinder. Then the evolution of $(\theta, \dot{\theta})$ is governed by the differential equations

$$\frac{\partial}{\partial t}\,\dot{\theta} = \cos(\theta)$$
$$\frac{\partial}{\partial t}\,\theta = \dot{\theta}$$

These differential equations are a presentation of a section of the tangent bundle of $S^1 \times \mathbb{R}$. We use this section to generate a 1-parameter diffeomorphism group acting on $S^1 \times \mathbb{R}$.

There are more adjectives that we can use to describe the system above: it is *continuous time*, and it is *deterministic*. Its *state space* is $S^1 \times \mathbb{R}$, and its *time axis* is $\mathbb{R}$. *Mechanics* is the study of continuous time, deterministic dynamical systems.

In general, a dynamical system will have a 1-parameter *monoid* acting on it. This monoid will typically be $\mathbb{R}_{\geq 0}$ or $\mathbb{N}$, but it could be $\mathbb{R}$ or $\mathbb{Z}$. That is, if $\mathrm{B}M$ is a category with one object and morphisms given by elements of M, and C is some category like the category of manifolds, or the category of metric spaces, or the category of vector spaces, then the formal object corresponding to an *evolution* of $c \in C$ is a functor from $\mathrm{B}M$ to C that sends the one object of $\mathrm{B}T$ to c.

Example 2. Consider a matrix $T \in \mathbb{R}^{\mathbb{Z}\times\mathbb{Z}}$ defined by $T_{i,i+1} = 0.5$, $T_{i,i-1} = 0.5$, $T_{i,j} = 0$ otherwise. Then we define an action of $\mathbb{N}$ on $\Delta^{\mathbb{Z}}$, the space of probability distributions on the integers, by $k \mapsto T^k$. This represents a random walk on the set of integers. If $p \in \Delta^{\mathbb{Z}}$ is a probability distribution over $\mathbb{Z}$, then $T^k p$ is the probability distribution of the state of the random walker after k steps, if the state of the random walker was originally distributed as p.

This is a *discrete time*, *stochastic* dynamical system. Its *state space* is $\mathbb{Z}$, and its *time axis* is $\mathbb{N}$. Note that one could also look at this as a *deterministic* system with state space $\Delta^{\mathbb{Z}}$ (the set of probability distributions over $\mathbb{Z}$); in general a stochastic system can also be viewed as a deterministic system where the state space is the set of probability distributions over the original state space. The difference is the *approach*: the study of stochastic systems uses methods of probability theory because we typically care about questions about our systems that have probabilisitc answers.

There are also *static* systems, which can be viewed as special cases of dynamical systems that don't evolve over time. A dynamical system with

arbitrary time axis A and state space E can be viewed as a static system with state space E^A.

Example 3. Consider a diagram in Set of the form

$$\mathbb{R}_{\geq 0} \xleftarrow[\rho]{} E \underset{t}{\overset{s}{\rightrightarrows}} V$$

This is a *decorated graph*: a graph where each edge $e \in E$ has an associated real number $\rho(e)$. E and V are sets of edges and vertices, respectively, and s and t are the source and target maps. We interpret this system as an electronic circuit with resistances $\rho(e)$, and a behavior in this system as an assignment of a voltage to each vertex that satisfy certain physical laws which we will discuss later. However, it could also model biological diffusion across membranes, if we replace electric potential with chemical potential [Sch81].

Static systems also come up as so-called *steady state* solutions for dynamical systems, which are fixed points of the evolution. The previous example is actually the steady state solutions for a dynamical system which would evolve the currents and voltages through time.

What the reader should keep in mind, therefore, is that these adjectives (dynamical, static, stochastic, deterministic) and nouns (state space, time axis, system, behavior) are *not* mathematical terms. Rather, they are useful for communicating an approach to mathematically modeling some phenomena.

1.3 What is an Open System?

Similarly, "open system" is not a mathematical term. An open system is a system that has a "boundary" along which it can be influenced by an un-modeled environment (i.e., we know how the system responds to the environment, but we don't know what the environment is doing).

Example 4. As an extension of Example 3, consider a diagram of the form

$$\begin{array}{ccccc} & & & & B \\ & & & & \downarrow{\scriptstyle b} \\ \mathbb{R}_{\geq 0} & \xleftarrow[\rho]{} & E & \underset{t}{\overset{s}{\rightrightarrows}} & V \end{array}$$

The way we talk about the behaviors associated to this system is different than in the previous systems; we say that *given* a behavior of the boundary (i.e., the system restricted to B), there are a collection of behaviors *compatible* with that behavior. At the boundary, the physical laws that held earlier no longer need to hold, because energy can flow in or out to maintain equilibrium.

There also may be *no* system behavior compatible with a boundary behavior, and we interpret this as the system enforcing a constraint on its boundary.

Example 5. Consider the same pendulum system, but now with an additional term added to the differential equation

$$\frac{\partial}{\partial t}\,\dot{\theta} = \cos(\theta) + F(t)$$
$$\frac{\partial}{\partial t}\,\theta = \dot{\theta}$$

This additional term represents an external force. We still end up with a evolution through time, but now it takes a slightly different form. That is, we care not just about the total time that the system has evolved for, but also the starting and ending time. Therefore, our "action" is a functor from $\mathbb{R}$ viewed as a poset: any morphism $t \to t'$ gives a diffeomorphism from the cylinder to itself.

Example 6. Any closed system is an open system with empty boundary.

Just as a drunk searches for his keys under a lamp post because that's where he can see, scientists have concentrated on systems whose behavior can be studied in isolation, i.e. systems without boundary. This is because it is easier to make and test hypotheses when one does not have to worry about unmodeled interactions with the environment.

Classically, when systems that we want to model are open systems in the real world, we get around their dependence on the environment by either neglecting it, or adding more parts of the environment to the model. And in fact, when we are only concerned about one system, this is desirable: we want to ask questions about a single system, so we add everything we know about that system in order to get the best picture possible.

However, when we explicitly model the environment to create a closed system, we lose the ability to compose the model that we have made with *other* models of the environment. This is the *key feature* of open systems: open systems are most naturally discussed in a *compositional* framework where one can "glue" two open systems together along a part of their boundary.

In short, just as dynamical systems are distinguished by an approach that cares about time evolution, and stochastic systems are distinguished by an approach that cares about nondeterminism, open systems are distinguished by an approach that cares about compositionality. And that is where the category theory comes in.

2 Material

Here we introduce three examples that will run through this paper and illustrate each of the tools that we build. We will not be able to explain these examples thoroughly now, because we don't have the right tools with which to do so. However, keeping these examples in mind as we run through the different tools for open systems will be useful.

2.1 Decorated Graphs

Fix a set $\mathcal{D}$, which we will call the set of decorations. Then a decorated graph with decorations $\mathcal{D}$ is a diagram in Set of the form

$$\mathcal{D} \xleftarrow{d} E \underset{t}{\overset{s}{\rightrightarrows}} V$$

The elements of E are called "edges", and the elements of V are called "vertices". Note that this definition of graph allows for self-edges and multiple edges, and edges are directed with a "source" $s(e)$ and "target" $t(e)$.

Each edge has an associated "decoration" $d(e)$, which is an element of $\mathcal{D}$.

Example 7. A social network is a decorated graph. The set of decorations is the set of labels for relationships between two people, for instance "parent", "friend", "boss", "priest", "doctor", etc. Edges from a vertex to itself make sense in this context because someone could be their own boss, or their own doctor. Additionally, two people could have several relationships with each other, so there could be several edges between two vertices.

To build the category of decorated graphs over the decoration set $\mathcal{D}$, let C be the following category:

$$d \xrightarrow{\delta} e \underset{\tau}{\overset{\sigma}{\leftleftarrows}} n$$

Then take the subcategory of $\mathsf{Set}^{C^{\mathrm{op}}}$ where the objects are F such that $F(d) = \mathcal{D}$ and the arrows are α such that $\alpha_\delta = 1_{\mathcal{D}}$.

2.2 Chemical Reactions

Fix a finite set S, which we call the set of *species*. Then we call $C = \mathbb{N}^S$ the set of *complexes* on S: a complex is an assignment of a natural number to each species. A *reaction* is an element r of $C \times C$: we call $\pi_1(c)$ the input and $\pi_2(c)$ the output.

A *Petri net* is nothing more than a set of reactions.

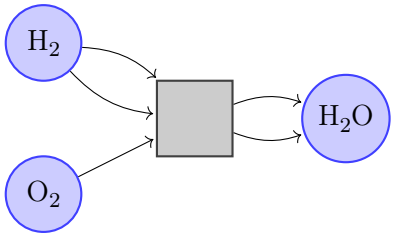


Figure 1: A Petri net for making water.

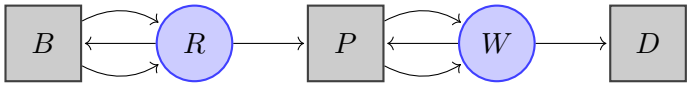


Figure 2: A Petri net describing predator-prey interaction.

Example 8. Let $S = \{\mathrm{H}_2, \mathrm{O}_2, \mathrm{H}_2\mathrm{O}\}$. Then

$$2\,\mathrm{H}_2 + \mathrm{O}_2 \to 2\,\mathrm{H}_2\mathrm{O}$$

is a reaction written down in the language of chemistry. We invite the reader to take a moment and think about how to parse this formula into the fomalism above.

Example 9. Let $S = \{\mathrm{W}, \mathrm{R}\}$. In this case, W and R do not stand in for molecules; they stand in for wolves and rabbits. Then a standard predator prey model can be expressed as

$$\begin{aligned} \mathrm{R} + \mathrm{W} &\to 2\,\mathrm{W} \\ \mathrm{R} &\to 2\,\mathrm{R} \\ \mathrm{W} &\to \end{aligned}$$

The first "reaction" represents predation and wolf reproduction. The assumption is that one rabbit is converted into one wolf, catalyzed by a wolf which "eats" the rabbit and "gives birth" to the new wolf. The second reaction is rabbit reproduction, in this case asexual: one rabbit splits into two rabbits. Finally, wolves occasionally die spontaneously. This is obviously a simplification, but even so this model can capture real population dynamics.

There is a neat diagram that one can associate to a Petri net, which one can see in Figure 1 and Figure 2. The blue circles correspond to species, and the grey squares correspond to reactions. For each input to a reaction, there is an arrow coming from the corresponding species circle to the reaction

square, and for each output from a reaction, there is an arrow going from the reaction to the corresponding species.

A morphism of Petri nets is exactly like a graph morphism. That is, if $(S_k, R_k, s_k \colon R_k \to \mathbb{N}^{S_k}, t_k \colon R_k \to \mathbb{N}^{S_k})$, $k \in \{1, 2\}$ are two Petri nets, then a morphism between them is a function $f_S \colon S_1 \to S_2$ and a function $f_R \colon R_1 \to R_2$ that commutes with source and target. If the Petri net is decorated with rates, then we require that f_R also preserves rates.

2.3 Linear Algebra

The reader should be familiar with Vec_k, the category of vector spaces over a field k. What might be less familiar is the the *category of linear relations* over a field k, which we call LinRel_k. The objects in this category are vector spaces over k, and a morphism between U and V is a linear subspace $R \subseteq U \times V$, which is interpreted as a relation in a standard way. Then to compose $R \subseteq U \times V$ and $S \subseteq V \times W$, we define $S \circ R$ by

$$\{(u, w) \in U \times W \mid \exists v \in V \text{ such that } (u, v) \in R \text{ and } (v, w) \in S\}.$$

We care about LinRel_k because it is a convenient and simple category for the behavior of open systems. An open system doesn't have inputs and outputs in a traditional sense: rather it enforces constraints between different parts of its boundary.

For instance, an electronic circuit enforces a relationship between the voltages of different nodes in the circuit. A Petri net enforces a relationship between the quantities of different substances.

From a mathematical perspective as well, LinRel_k has an interesting algebraic presentation, which we will discuss later on.

3 Monoidal Categories

3.1 Overview

We will eventually compose systems by gluing them along part of their boundary. However, before we do this, we first study composing systems by just juxtaposing them and not worrying about the glue yet. The correct structure for this sort of composition is a *monoidal category*.

Recall that a *monoid* can be viewed as a category with only one object. Classically, we don't think about the single object, and we just think of the monoid as a set with an associative binary operation that has a unit. However, this definition as a one object 1-category is useful, because it makes monoidal

categories a natural generalization: a monoidal category is a 2-category with only one object. Classically, we think of a monoidal category as a 1-category that has a binary operation on it satisfying certain laws about its interaction with the arrows of the category. These laws are much easier to remember, and are more naturally derived, however, if one simply thinks of the monoidal category as a 2-category with only one object.

For the reader who has only passing familiarity with 2-categories, we will do a brief review. A 2-category is composed of objects A, B, 1-morphisms $f, g\colon A \to B$, and 2-morphisms $\alpha\colon f \Rightarrow g$, that satisfy the following.

1. $\mathrm{Hom}(A, B)$ is a 1-category for any objects A, B, with objects the 1-morphisms and arrows the 2-morphisms.

2. For all objects A, B, C, we have a functor $\circ\colon \mathrm{Hom}(A, B) \times \mathrm{Hom}(B, C) \to \mathrm{Hom}(A, C)$ such that:

   (a) $(- \circ -) \circ -$ and $- \circ (- \circ -)$ are naturally isomorphic, and this natural isomorphism respects the Maclane pentagon identity.

   (b) For every A, there exists $1_A \in \mathrm{Hom}(A, A)$ such that $1_A \circ -$ is naturally isomorphic to $1_{\mathrm{Hom}(B,A)}$, and $- \circ 1_A$ is naturally isomorphic to $1_{\mathrm{Hom}(A,B)}$ for all B.

We call a 2-category *strict* if the natural isomorphisms in 2a and 2b are equalities, and *weak* otherwise. It is an important theorem that any 2-category is equivalent to a strict 2-category if we take the right notion of equivalence of 2-categories. This means that we don't really have to worry about the difference between strict 2-categories and weak 2-categories. However, it is important to get in the habit of worrying about strict vs. weak, because it will be important for some structures later on.

Example 10. The category of categories and functors, Cat, has a natural 2-category structure given by adding natural transformations as 2-morphisms. This is a *strict* 2-category.

A 2-category with one object, $*$, is comprised of a 1-category $\mathrm{Hom}(*, *)$ along with an associative and unital binary operation, $\circ\colon \mathrm{Hom}(*, *) \times \mathrm{Hom}(*, *) \to \mathrm{Hom}(*, *)$. Normally in this case, we call this operation $\otimes$ instead of $\circ$, and reserve $\circ$ for composition of morphisms in $\mathrm{Hom}(*, *)$. We also typically call the identity element I.

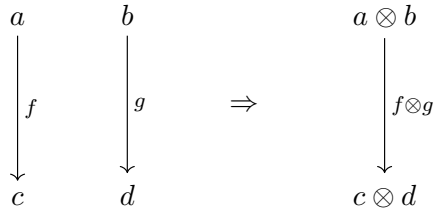
Example 11. In the above 2-category, take the subcategory with single object

FinSet, 1-morphisms generated by the basepoint functor,

$$F\colon \mathsf{FinSet} \to \mathsf{FinSet}$$
$$A \mapsto A \sqcup \{*\}$$

and 2-morphisms being all natural transformations $\alpha\colon F^n \Rightarrow F^m$. If we think about this as a 1-category with a binary operation, this is a skeleton of FinSet with the binary operation of disjoint union, i.e. the set of objects is isomorphic to $\mathbb{N}$, and the binary operation is $+$ with unit 0. This is a very important monoidal category; we will call it $(\mathscr{F}, +, 0)$.

One part of the definition of a monoidal category that is important to emphasize is the ability to compose morphisms *sideways* as well as vertically. That is, if $(C, \otimes, I)$ is a monoidal category, then we can compose morphisms as illustrated in the following diagram.



Holding with the general idea in category theory that the morphisms are more interesting than the objects, it is this structure on the arrows in a monoidal category that makes monoidal categories interesting.

Example 12. Let C be a category with products and terminal objects. Then if we let $\otimes$ be the product and I be the terminal object, it can be shown that $(C, \otimes, I)$ is a monoidal category. Similarly, we can do this with sums and the initial object.

Example 13. If R is a commutative ring, then $(\mathsf{Mod}_R, \otimes_R, R)$ is a monoidal category, where $\otimes$ is the normal tensor product.

Example 14. The category of decorated graphs has coproducts (disjoint union of nodes and edges), and initial object (the empty graph), and we customarily use the monoidal category structure given by this.

Example 15. Vec_k can be viewed as a monoidal category $(\mathsf{Vec}_k, \otimes, k)$ using the tensor product of vector spaces, or it can be viewed as a monoidal categegory $(\mathsf{Vec}_k, \oplus, 0)$ using direct product. This second monoidal category structure also works for LinRel_k, as given $R_1 \subseteq V_1 \times W_1$ and $R_2 \subseteq V_2 \times W_2$, we can construct

$$R_1 \times R_2 \subseteq (V_1 \times W_1) \times (V_2 \times W_2) \cong (V_1 \times V_2) \times (W_1 \times W_2)$$

3.2 Monoidal Functors

The naive way to define a monoidal functor between $(C, \otimes, I)$ and $(D, \boxtimes, J)$ is to say that a monoidal functor is an ordinary functor F where

$$F(c_1 \otimes c_2) = F(c_1) \boxtimes F(c_2)$$
$$F(I) = J$$

However, this is too restrictive (this is what we call a *strict* monoidal functor). A more permissive way to define a monoidal functor is to require that $F(c_1 \otimes c_2) \cong F(c_1) \boxtimes F(c_2)$ and that $F(I) \cong J$, and that these isomorphisms satisfy some coherence conditions. This is the restriction of the notion of a *pseudofunctor* between 2-categories, and is called a *weak* monoidal functor; to learn the details of the coherence conditions for pseudofunctors, we refer the reader to [Lei03].

It is important to note that while all 2-categories are equivalent to strict 2-categories, not all monoidal functors are equivalent to strict monoidal functors.

3.3 Symmetric Monoidal Categories

Another example of when weakness is or isn't important is when we try to make an analogue of commutative monoids. If we require that $a \otimes b = b \otimes a$, not many categories satisfy this condition. Categories which do, we call "strict symmetric monoidal categories".

However, the requirement that $(a, b) \mapsto a \otimes b$ and $(a, b) \mapsto b \otimes a$ are naturally isomorphic functors is much less restrictive, and is satisfied by many important monoidal categories. In fact, this is the case for all of the examples we have given up to now. When this is true, we call the category a *braided monoidal category*, and we call the natural isomorphism the *braiding*. We will see why this is a natural term and give an example later.

But that condition by itself is too loose. If $\alpha_{a,b} \colon a \otimes b \to b \otimes a$ are the components of the natural isomorphism, then we want it to be the case that $\alpha_{a,b}$ is just "swapping" a and b, and if we swap them back with $\alpha_{b,a}$ we should end up in the same place we started with. That is, $\alpha \cdot \alpha$, where $\cdot$ is composition of natural transformations, should be the identity. This is not always true, but when it is true, we call the category a *symmetric* monoidal category.

Most of the categories that we expect the reader to be familiar with fall into this designation. For instance, all of the categories with a monoidal operation defined by product or coproduct are symmetric monoidal, and

the categories with a tensor product that is adjoint to the Hom functor are symmetric monoidal. We will use the symmetric monoidal structure of coproduct on the category of Petri nets and the category of electronic circuits.

3.4 Petri Nets as Presentations of SSMCs

Not only does that category of Petri nets have a symmetric monoidal structure, but also an individual Petri net can be viewed as generating a (strict) symmetric monoidal category.

Recall that for any graph, we can make the "free category" on that graph. This construction is the left adjoint to the "forgetful functor" that takes a category to its underlying graph. Similarly, there is a functor from the category of strict symmetric monoidal categories to the category of *monoidal* graphs (the category of graphs with a monoid structure on the nodes). This functor also has a left adjoint, which sends a monoidal graph to the free strict symmetric monoidal category generated by edges in that graph.

Just like in the construction of a free category, the objects of the free strict symmetric monoidal category are the same as the vertices of the monoidal graph. However, a morphism in this category is more complicated. This is because there are two ways of composing morphism in a monoidal category: by tensor product, and by normal composition. The morphisms in the free strict symmetric monoidal category are freely generated by *both* of these operations.

A Petri net can naturally be seen as a monoidal graph: the objects are the set of complexes, $\mathbb{N}^S$, and the arrows are the reactions. Even better: there is a functor from the category of Petri nets to the category of monoidal graphs. We can compose this functor with the functor from monoidal graphs to strict symmetric monoidal categories, and we end up with a functor assigining a strict symmetric monoidal category to each Petri net. In this symmetric monoidal category, we interpret a morphism from one complex to another complex as a sequence of reactions. Each reaction applies to a subcomplex, and replaces that subcomplex with the result of that reaction. In this category, we say that the reaction *generate* the morphisms.

Note that this is *not* a full embedding. There are more morphisms in the category of strict symmetric monoidal categories. For instance, a morphism could collapse a series of reactions to a single reaction, or expand a single reaction to several reactions, neither of which are possible in the category of Petri networks. This allows one to potentially simplify a Petri net while maintaining essential properties. For more information, see [BC18].

We would give an example, however we do not yet have the right notation

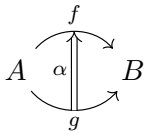
to make this intuitive. Monoidal categories are confusing because there are two dimensions of composition: there is ordinary morpism composition $f \circ g$ and there is "sideways" morphism composition $f \otimes g$. In order to fully make sense of this, we need new notation that captures these two dimensions, and that brings us to our next section.

4 String Diagrams

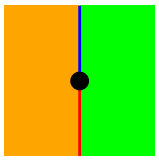
In this section, we will present a new notation for working with monoidal categories (and more generally, 2-categories), and we will give several interpretations of this notation in order help the reader develop an intuition for it and a facility for its use. The emphasis will be on notation rather than category theory. We will finish with a presentation of the category of linear relations using string diagrams.

4.1 String Diagrams as Dual to Commutative Diagrams

In typical commutative diagrams, we draw objects as dots, 1-morphisms as lines, and 2-morphisms as regions. In the language of cell complexes, objects are 0-cells, 1-morphisms are 1-cells, and 2-morphisms are 2-cells. Classically, this looks something like:



where we imagine that α is a region filling in the space between f and g. This same diagram in string diagram notation would look like:



Often string diagrams are labelled with colors rather than symbols, but this is merely an aesthetic change. The main difference is that what was a k-cell in the previous diagram is now a $(2-k)$-cell. A student of topology will recognize this as Poincaré duality.

The commutative diagram had 0-cells A and B; the string diagram has corresponding orange and green 2-cells. The commutative diagram had 1-cells f and g, and the string diagram has corresponding blue and red 1-cells which

are perpendicular to the f and g in the commutative diagram. Finally, α, the one 2-cell in the original diagram, has become a 0-cell.

In the case of a monoidal category, there is only one object, so we may leave the regions blank, and this is how we will write string diagrams for the rest of this survey. We have introduced string diagrams in the more general context of 2-categories in order to firmly disassociate 0-cells and objects in the reader's mind; we have found that this is the hardest thing to explain about string diagrams.

One last thing we will note: we are not at all consistent in how we draw string diagrams; each section will have a slightly different style. This is for two reasons. The first is that "in the wild" there are many styles of string diagram, and it is important to recognize the commonalities and basic structure between them. The second reason is that there are aspects of string diagrams that are better captured by one style than others.

4.2 String Diagrams as Histories

In the context of Petri nets there is another way of interpreting a string diagram. Recall that a Petri net is a presentation for a strict, symmetric monoidal category. Each object in this monoidal category (i.e., 1-morphism in a 2-category with one object) is a complex, and each arrow (i.e. 2-morphism) is a sequence of reactions. A string diagram is a way of displaying a composition of 2-morphisms, so in this context a string diagram is a way of displaying several reactions in sequence.

Before we go any farther, the reader is invited to look at Figure 3 and imagine how it might represent a sequence of reactions, if the green is H_2, the red is O_2, the blue is H_2O, the black dot is the reaction

$$2\,H_2 + O_2 \longrightarrow 2\,H_2O$$

and the white dot is the reverse reaction.

The way we advise the reader to interpret such a diagram is as a time history, flowing from bottom to top. Each horizontal slice of the diagram corresponds to the state of the system at a point in time. This view is pictured in Figure 4.

For instance, slicing the diagram at the very bottom gives four green threads and 2 red threads, which corresponds to the complex $4\,H_2 + 2\,O_2$. Then, as we move through time, reactions happen at distinct points, and

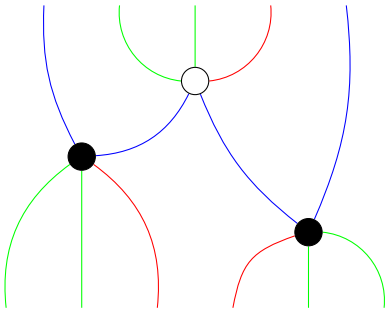


Figure 3: Water Formation

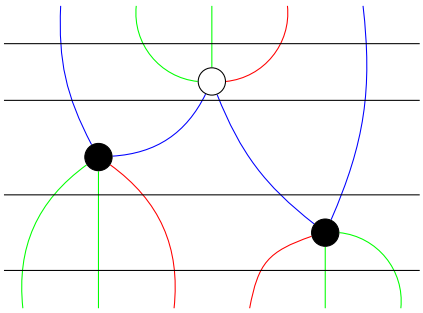


Figure 4: Sliced Water Formation

change the complex.[1] The sequence of reactions is

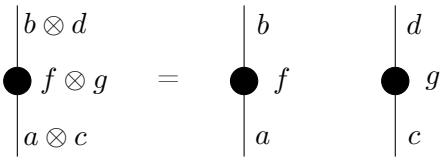
$$4\,H_2 + 2\,O_2 \longrightarrow 2\,H_2 + O_2 + 2\,H_2O \longrightarrow 4\,H_2O \longrightarrow 2\,H_2 + O_2 + 2\,H_2O$$

The reader may object that we are not worrying about the order in which the reactants are listed. This is because earlier we defined the monoidal category associated with a Petri net to be *strictly* symmetric, so that $H_2 + O_2 = O_2 + H_2$. In the following section, this will not be true.

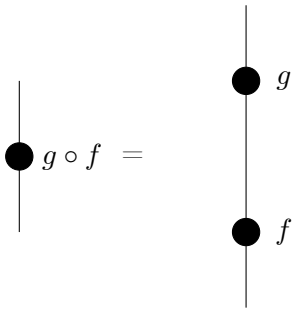
Now, the most important part of category theory is *composition*, and it is in composition where string diagrams start to really shine. We noted earlier that it is hard to keep straight the two different types of composition in a monoidal category, that is $f \otimes g$ and $f \circ g$. String diagrams solve this problem very neatly. To tensor together two morphisms f and g, we simply put the two diagrams that represent them side by side.

In the "time-history" interpretation, horizontal composition signifies two time histories happening in parallel, not affecting each other. Normal com-

[1] For the reader who has experience with Morse theory, the idea that the complex stays the same except for at "critical points" should be familiar.

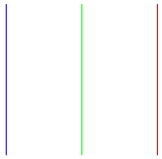


position is achieved by vertical stacking; the interpretation here is that two processes happen in sequence.



There is one more ingredient that is necessary to interpret these diagrams. Namely, what is a plain wire? A fan of 0 might have have noticed that we gave diagrammatic interpretations for two types of composition, but we did not give a diagrammatic interpretation for the other key part of a category: the identities. The identity for the monoidal composition is very simple, it looks like this:

That is, the identity for the monoidal/horizontal composition is an empty diagram.[2] However, the identity for regular/vertical composition is not just a blank diagram. It is a straight wire with no decorations, or several straight wires. For instance, the following figure is the identity on $H_2O + H_2 + O_2$.



It is the identity which allows us to apply a reaction to a subcomplex; if f is a reaction that has domain A, then we can apply f to the complex

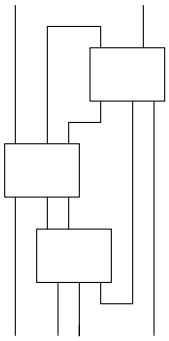
[2]Actually, in the 2-category setting where we color the background, the monoidal identity would be a monochromatic square.

$A \otimes B$ by tensoring f with 1_B.

There are more to string diagrams than particle-histories, however, so the reader is advised to not get too attached to this interpretation. The main take-away from this section should be the correspondence between the graphical notation and the categorical operations: this is what will be important for the next section.

4.3 String Diagrams As Circuits

We start by presenting a new style for string diagrams. The reader should look at the following picture and try and work out for themselves how it works as a string diagram.[3]

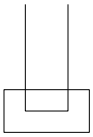


We hope that the reader should have no trouble exchanging dots for boxes. The real challenge is that there are lines which loop back around. If we interpret the diagram as an electronic circuit, where the lines are wires and the boxes are circuit elements, then it is obvious how to interpret the bends in the wires; one simply bends the wires. On the other hand, from the perspective of Petri nets, the bends seem to break the laws that we have established. How can a particle "loop around"—it seems like it is going back in time!

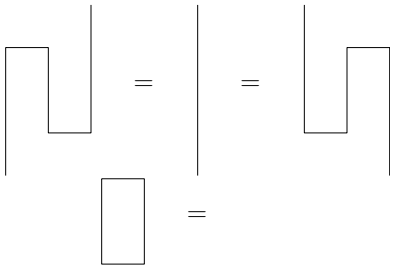
However, there is a simple trick that allows us to interpret such bends. Recall that the monoidal identity is an empty diagram. With that in mind, the following diagram should give a clue on how we interpret bends.

Namely, we define a bend to be a morphism with input I (the identity) and output $A \otimes A$ (and cobend is the other way around). Bend and cobend should also satisfy some common-sense identies pictured below, which essentially just

[3]Our hope is that by forcing the reader to interpret new styles as string diagrams, we encourage the reader to be on the alert for unintentional string diagrams in the notation of other fields.



say that there are no "side-effects" to a bend; it is just a way of redirecting wires.



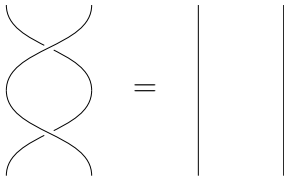
Categorically speaking, bend and cobend are *natural transformations* between the functors $A \mapsto A \otimes A$ and $A \mapsto I$, that satisfy the above laws.

We will refrain from a rigorous treatment of electrical circuits as string diagrams until after we have studied cospans.

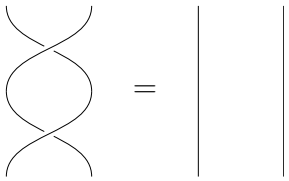
4.4 The Braid Category

We promised earlier that we would discuss braided monoidal categories. We now have the notation to make the word "braided" intuitive. In a braided monoidal category, we write the braiding $A \otimes B \cong B \otimes A$ as one strand crossing *over* another.

The inverse of the braiding is depicted by reversing the directions that the strands cross, and it should make intuitive sense why it is the inverse: one can imagine deforming the first diagram into the second without moving the ends, if the strings are embedded in 3-space.

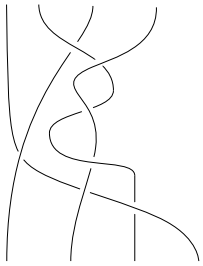


On the other hand, in a general braided monoidal category the below equation does *not* hold.



If the string diagram is embedded in 3-space, then one cannot untangle the two strands without moving the ends.[4]

The canonical example of a braided category is the *braid category*, which is the *free* braided symmetric monoidal category on one object T. Note that because it is a monoidal category, there are actually $\mathbb{N}$ objects in this category, all tensor powers of the one generating objects. The morphisms in this category are generated by the identity and the braiding: a typical morphism in the braid category is pictured below.



If the reader has studied knot theory, the braid group should be familiar. It might be surprising to learn that the braid group on n strands has an *algebraic* presentation: it is the full subcategory of the braid category spanned by the object $T^{\otimes n}$.

[4] On the other hand, if the string diagram is embedded in 4-space, then one *can* untangle the two strands. Somehow, this has to do with the distinction between braided monoidal and symmetric monoidal: the interested reader should consult [BS09] for an investigation of interesting alignments along these lines.

4.5 Linear Algebra in String Diagrams

The previous two examples have both been *schematic*. That is, we used string diagrams essentially as just fancy graphs. In this section, we will emphasize the *algebraic* aspects of string diagrams; i.e. the ability for string diagrams to be used as a tool for computation.

Consider the category LinRel_k. This has a monoidal structure given by direct product, as we talked about before. We also claimed that LinRel_k had an interesting algebraic structure. In this section we will discuss this algebraic structure.

By algebraic structure, we mean that in the subcategory monoidally generated on each object (i.e. maps $V^{\otimes n}$ to $V^{\otimes m}$), there are distinguished morphisms that satisfy some interesting algebraic structure. We call these morphisms *operations*, in analogy to the operations of a ring. However, there are much more of them.

Additionally, it turns out that if we restrict our attention to the subcategory monoidally generated by k, these operations end up generating all of the morphisms, and we will give a sketch of the proof for this.

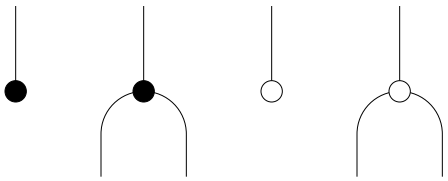
We start out with the bend and cobend. This will save time, because we can use the bend and cobend to dualize everything else, and thus avoid having to present things twice. In the style for this section, bend and cobend look like this (bend on the right):

We define the bend morphism by

$$\{((v, v), 0) \in V^{\oplus 2} \times V^{\oplus 0} | v \in V\}$$

and cobend is defined similarly.

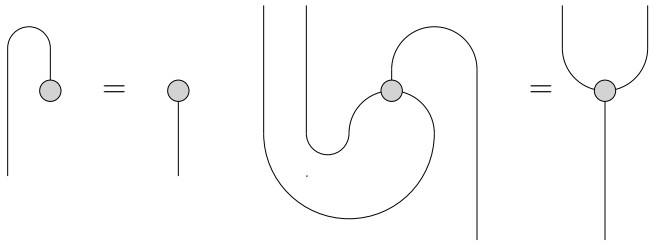
There are then two pairs of operations which share a great deal of similarities, so we will treat them at the same time.



We will defer the definition of these operations for a few moments, because we want to confront the reader with the abstract syntax. If the reader

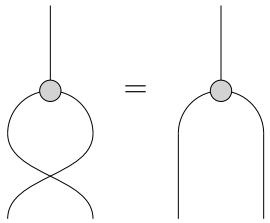
prefers concrete definitions, they are advised to skip the next page, read the definitions, and come back.

Each of the operations can be dualized, and we write the dualized form in the following manner. If the reader imagines that the strings can be pulled and bent, then the dualization should seem (pictorially) very natural: we are just spinning around the operations.

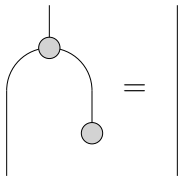


This dualization *should be* defined by $R \subseteq V^{\oplus n} \oplus V^{\oplus m}$ mapsto $R^{\mathrm{op}} \subseteq V^{\oplus m} \oplus V^{\oplus n}$ by just applying $(v, w) \mapsto (w, v)$. However, it is not immediately clear how this dualization corresponds to the dualization via bend and cobend. The reader should attempt to work out for themselves this correspondence before moving on.

Each of the "forks" are commutative, that is



and each pair satisfies an "identity" law:



More generally, each of the pairs satisfy the laws for a *Frobenius algebra*, which are given in [FS19, §6.3]. However, rather than getting too bogged down in all of the algebraic details, we now give concrete definitions in the

category LinRel_k and encourage the reader to come up with his or her own laws that the operations satisfy.

The black dot operations are named *zero*:

$$\{(0,0) \in V^{\oplus 0} \oplus V^{\oplus 1}\}$$

and *sum*

$$\{((x,y),z) \in V^{\oplus 2} \oplus V^{\oplus 1} \mid x+y=z\}$$

The motivated reader will check that each of the properties that we have claimed to hold do in fact hold for this definition.

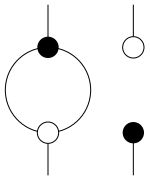
The white dot operations are named *free*

$$\{(0,x) \in V^{\oplus 0} \oplus V^{\oplus 1}\}$$

and *coclone*

$$\{((x,x),x) \in V^{\oplus 2} \oplus V^{\oplus 1}\}$$

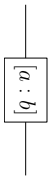
Again, the reader is invited to verify the properties that we have stated. In addition, the reader should work out the meaning of the following diagram before moving forward.



The last operation we need is called *ratio*, and is defined for $[a:b] \in kP^1$ (1-dimensional projective space over k). The definition is

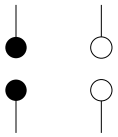
$$\{(x,y) \in V^{\oplus 1} \oplus V^{\oplus 1} \mid ay = bx\}$$

and the operation looks like.



Note that we can now generate all relations between k and k. All of the one-dimensional relations are handled by *ratio*, and the zero and

two-dimensionsional relations are handled by the following two diagrams, respectively.



We will now sketch the proof that all linear relations between $k^n = k^{\oplus n}$ and $k^m = k^{\oplus m}$ are generated by these operations. We prove this with a sequence of lemmas, working backwards from the conclusion. One thing that we should note before starting, however, is that the key thing to realize is that what we will be doing is making a *presentation* of a relation. Therefore, it will not be unique, and will involve *matrices*.

Lemma 1. *Any linear relation $R \subseteq k^n \oplus k^m$ can be written as*

$$x \; R \; y \Leftrightarrow \textit{there exists } w \in k^\ell \textit{ such that } Aw = x \textit{ and } Bw = y$$

for some fixed k^ℓ, and some fixed matrices A and B. More strongly: if $\{(w, Aw) \mid w \in k^\ell\}$ and $\{(w, Bw) \mid w \in k^\ell\}$ are relations written in terms of the basic operations, then we can write R in terms of the basic relations.

Proof. Fix a basis for R, and let $k^\ell \cong R$ be the identification corresponding to that basis. Then let A be a matrix representing π_1 and let B be a matrix representing π_2. To see the stronger claim, observe Figure 6. □

The problem now reduces to showing that we can generate the relationship $\{(v, Av) | v \in k^n\} \subseteq k^n \oplus k^m$ for any $n \times m$ matrix A.

Lemma 2. *Suppose that for $i = 1, \ldots, n$, we have written the relation $R_i \subseteq k \oplus k^m$ using our collection of basic operators. Then we can write the relation $R \subseteq k^n \oplus k^m$, which is defined by*

$$x \; R \; y \Leftrightarrow x_i \; R_i \; y \textit{ for all } i$$

using our collection of basic operators.

Proof. The basic idea is to generalize Figure 7. Essentially, we use the /clone/ operation to split the input up into n different copies. We then apply R_i to each of those copies. □

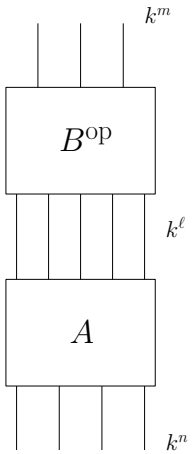


Figure 6: Relations from Matrices

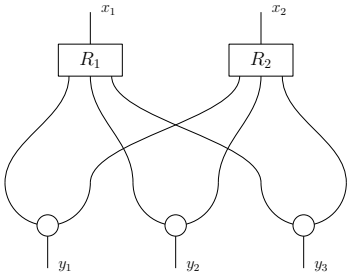


Figure 7: Composition of row vectors to produce a matrix

Lemma 3. *Any linear relation of the form* $\{(y, \sum_j a_j y_j) \mid y \in k^n\}$ *can be written using the basic operations above.*

Proof. Again, the basic idea is to generalize Figure 8. We multiply each input by the corresponding element of the row vector, and then add all of them together. □

We are now ready for the main proof.

Proof. With these lemmas, the proof of the main theorem becomes almost trivial. To construct a diagram that presents a relation $R \subseteq k^n \oplus k^m$, let $w_1, \ldots, w_\ell$ be a basis for R. Then let A be the matrix that represents π_1 in this basis, and let B be the matrix that represents π_2. By Lemma 1, it suffices to find relationships $\{(Aw, w) \mid w \in k^\ell\}$ and $\{(w, Bw) \mid w \in k^\ell\}$. By Lemma 2, it suffices to find relationships $\{(A_i w, w) \mid w \in k^\ell\}$ for all rows A_i

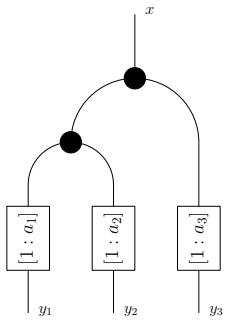


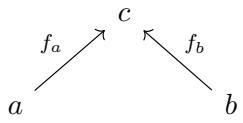
Figure 8: Creation of a row vector

of A, and $\{(w, B_i w) \mid w \in k^\ell\}$ for all rows B_i of B. Finally, we can do this by Lemma 3. □

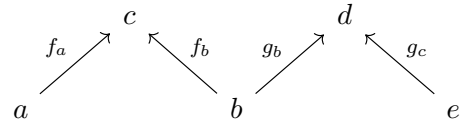
5 Cospans

5.1 Basic Cospans

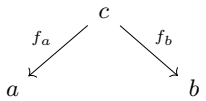
Let C be a category with pushouts. Then we define a category $\mathrm{Csp}(C)$ in the following way. The objects of $\mathrm{Csp}(C)$ are just the objects of C. However, an arrow between $a, b \in \mathrm{Csp}(C)$ is a diagram in C of the form



This is called a *cospan*.[5] We also write a cospan inline like this: $a \xrightarrow{f_a} c \xleftarrow{f_b} b$. Now, if we have two cospans that share a common "foot", i.e.

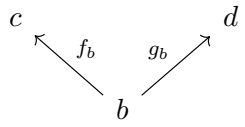


[5] A diagram of the form

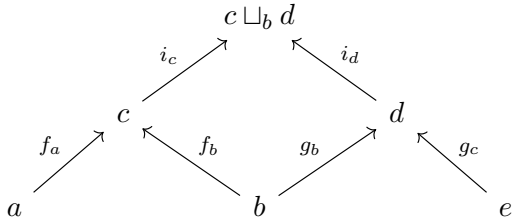


is called a *span*.

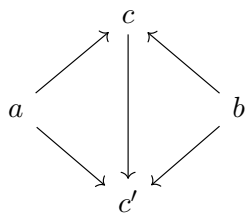
we can compose them into a cospan from a to e by taking the pushout of



in order to get



However, the categorically fastidious reader may have noticed a problem. Pushouts are only defined up to isomorphism, while composition of morphisms in a category is supposed to be a function. The solution is to just take the morphisms in $\mathrm{Csp}(C)$ to be isomorphism classes of cospans, where a morphism between two cospans is a vertical arrow in the following diagram that makes it commute:



As one might expect, the dual of a cospan is a span.

Example 16. As we saw earlier in Lemma 1, any linear relation $R \subseteq V \oplus W$ can be described by a *span*: $V \xleftarrow{\pi_1} R \xrightarrow{\pi_2} W$. However, it is also possible to describe a linear relation by a *cospan*. Namely, $x\ R\ y$ if and only if $f(x) = g(y)$, where $V \xrightarrow{f} U \xleftarrow{g} W$ is a cospan. It is left to the reader to discover whether all relations can be written in this form.

If the underlying category is monoidal, then the (co)span category will be monoidal as well. Composing (co)spans is done in a natural way:

$$
\begin{array}{ccccc}
A & \longrightarrow & C & \longleftarrow & B \\
\otimes & & = & & A \otimes X \longrightarrow C \otimes Z \longleftarrow B \otimes Y \\
X & \longrightarrow & Z & \longleftarrow & Y
\end{array}
$$

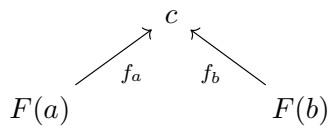
Example 17. The category of (possibly decorated) graphs has pushouts, so we can form the category of cospans over this category. This category allows us to glue together graphs along common subgraphs. Moreover, it inherits a monoidal structure from the monoidal structure on the category of decorated graphs.

From a purely theoretical standpoint, there is nothing wrong with the previous example. However, if we want to model electronic circuits, in physical practice we only glue (or rather, solder) together *terminals*. It does not make sense to identify two resistors. In order to solve this issue, we need a more advanced framework.

5.2 Structured Cospans

One problem with cospans is that we often want to restrict the type of feet. For instance, in the case of electronic circuits, the apex of the cospan can be very complicated: it can be a highly decorated graph. However, we want the feet to act like "plugs", so the feet should be discrete graphs.

The naive way to achieve this is to say that only a subset of the objects of a category can be feet. However, this is not very categorical. A better way is to have a "category of feet" D, along with a functor $F\colon D \to C$. Then a "structured cospan" is a diagram of the form



Just as we can do with regular cospans, we can construct a category ${}_F\operatorname{Csp}(C)$, where $\operatorname{ob}({}_F\operatorname{Csp}(C)) = \operatorname{ob}(D)$ and morphisms are isomorphism classes of decorated cospans. This construction is due to [BC20]. Previously, there was another way of doing a similar thing called *decorated* cospans; it is our opinion that structured cospans are more straightforward and elegant

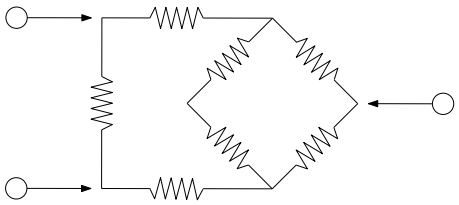


Figure 9: An Open Circuit

way of presenting many of the categories that decorated cospans were used for.

If D and C are monoidal categories and F is a monoidal functor, then ${}_F\operatorname{Csp}(C)$ inherits a monoidal structure in the same way that $\operatorname{Csp}(C)$ does,

$$\begin{array}{ccccc} F(a) & \longrightarrow & c & \longleftarrow & F(b) \\ & & \otimes & & \\ F(x) & \longrightarrow & z & \longleftarrow & F(y) \end{array} \quad = \quad F(a \otimes x) \longrightarrow c \otimes z \longleftarrow F(b \otimes y)$$

5.3 Categories of Open Systems

Using the technology of structured cospans, we can efficiently describe categories of open systems.

Example 18. In the section on electronic circuits, we deferred the discussion of how to construct the underlying category. We are now ready to give a more thorough treatment of this. Let F be the functor from FinSet to the category of $\mathbb{R}_{\geq 0}$-decorated graphs, Grph.

Then ${}_F\operatorname{Csp}(\mathsf{Grph})$ is a monoidal category. A morphism in this category is a circuit comprised of resistors along with an assignment of "input" and "output" ports. This is pictured in Figure 9. Note that unlike our previous convention, we draw this horizontally. This is for two reasons. One is that it fits better on the page. The other is that it is often drawn horizontally in the literature, and we would like to accustom the reader to reading string diagrams horizontally as well as vertically.

Example 19. We can also construct a category of open Petri nets. There is a functor from FinSet to Petri which sends a finite set to the Petri net with that

set of species and no reactions. Taking the structured cospan category on this functor leads to a category where we can compose (isomorphism classes of) Petri nets. [BC20].

5.4 Corelations

In the next big section, we will take these structured cospan categories and develop *semantics* for them. However, this review would be incomplete if it did not mention the theory of corelations. The essential idea of corelations is that if all we care about in a (structured) cospan is the relationship that it induces on the feet (as we saw before, a cospan of vector spaces induces a linear relation on the feet), then often there are "inessential" parts of the apex of the cospan that we can discard. For instance, if we have an open circuit, then connected components that are not connected to the input or output we should be able to do without. Corelations give a way of finding a "minimal" cospan that represents the same relationship as a given cospan.

We will not develop the theory here, because we do not need it for later sections, but the interested reader should refer to [Fon16].

6 Semantic Functors

One of the reasons that we are interested in open processes is so that large unwieldy closed models can be broken up into smaller closed pieces and studied. However, we need some sort of guarantee that after we break up the large unwieldy model, study the pieces separately, and put the pieces back together, we will have the same picture.

This guarantee comes in the form of *functorial semantics*. The idea is that there are two categories, S and B. S is the category of *schematics*: the arrows and morphisms are *descriptions* for models. Then B is the category of *behaviors*: the arrows and morphisms are the actual fleshed-out model.

Then the functor assigns a semantic to each model, and the functor laws ensure that this assigning of semantics respects composition of models.

From a mathematical standpoint, there is nothing special about semantic functors. The interest in semantic functors comes from taking a semantic that already exists for a certain type of model, and realizing that it is in fact a functor; i.e. that the semantic respects composition.

In this section, we will discuss several semantics for the models that we have been talking about, and show that these semantics are functorial.

6.1 Steady State Solutions for Resistor Networks

Suppose that

$$\mathbb{R}_{\geq 0} \xleftarrow[\rho]{} W \underset{t}{\overset{s}{\rightrightarrows}} J$$

is a decorated graph, which we will call C. The intent is for C to represent a simple electronic circuit. W is the set of "wires", J is the set of "junctions", and $\rho(w)$ is the "inverse resistance" associated to each wire.

We are interested in *steady-state* (i.e., not time-varying) assignments of voltages to each junction that satisfy some laws from physics.

Definition 1. If $\{V_i\}_{i \in J} \in \mathbb{R}^J$ is an assignment of a voltage to each circuit junction in C, then the *current* flowing through a wire $w \colon i \to j$ is defined to be $\rho(w)(V_j - V_i)$. In the real world, this is called Ohm's law (typically formulated as $V = IR$); here it is just the definition of current.

In the real world, current is the rate at which charge flows. This is important, because it means that in order for charge to be conserved, the current flowing into any junction must be equal to the current flowing out of that junction.

Definition 2. Wires in the real world are undirected. Therefore, our decorated graph has extraneous information. To simplify matters, we will say that $w \colon i \leftrightarrow j$ if $w \colon i \to j$ or $w \colon i \leftarrow j$.

Definition 3. We say that the current incident on a junction i is

$$I_i = \sum_{j \in J} \sum_{w \colon i \leftrightarrow j} \rho(w)(V_j - V_i)$$

If I_i is positive, then current is coming into i and leaking out of the circuit somehow. We say that current is *conserved* at i if $I_i = 0$.

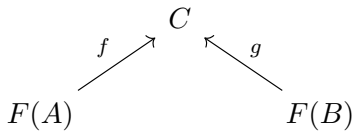
Notice that this equation is linear in $\{V_i\}_{i \in J}$. The set of solutions to this equation is therefore a real vector space.

Now, in a closed system, we would require $I_i = 0$ for all $i \in J$. However, this does not lead to anything interesting, because it turns out that each connected component of the graph would have to have the same voltage in all of its junctions in order to satisfy the current conservation laws. Thus, the solution set is a vector space with dimension equal to the number of connected components.

In order to get interesting behavior out of a circuit, we have to add nodes where current can flow in and out, that is nodes where $I_i \neq 0$. This is always

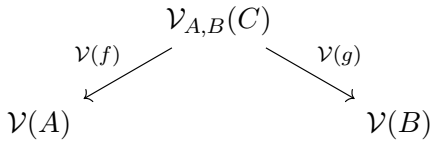
the case in the real world: a circuit has (at least) a power supply, which is held at a constant high voltage, and a ground wire, which is held at a constant low voltage. Conservation of current does not apply at the power supply junction or at the grounding junction.

To model this, we return to the structured cospan category for decorated graphs over FinSet. Consider a structured cospan



Then we say that a node is an *external* node if it lies in the image of f or g, and an internal node otherwise.

We assign to this structured cospan a *span* in $\mathsf{Vec}_{\mathbb{R}}$, which looks like



$\mathcal{V}_{A,B}(C)$ is the vector space corresponding to all assignments of voltages to the junctions in C that satisfy charge conservation at *internal* nodes. That is, if $\{V_i\}_{i\in J} \in \mathcal{V}_{A,B}(C)$, then $I_i = 0$ for all internal $i \in J$.

$\mathcal{V}(A)$ is $\mathbb{R}^A \times \mathbb{R}^A$, and $\mathcal{V}(B)$ is $\mathbb{R}^B \times \mathbb{R}^B$. $\mathcal{V}(f)$ and $\mathcal{V}(g)$ are defined by

$$\mathcal{V}(f)(\{V_i\}_{i\in J}) = \{V_{f(a)}\}_{a\in A} \times \{-I_{f(a)}\}_{a\in A}$$
$$\mathcal{V}(g)(\{V_i\}_{i\in J}) = \{V_{g(a)}\}_{a\in A} \times \{I_{g(a)}\}_{a\in A}$$

Intuitively, $\mathcal{V}(f)$ sends a voltage assignment to the voltages and "input" currents of the "input" $(F(A) \to C)$ and $\mathcal{V}(g)$ sends a voltage assignment to the voltages and "output" currents of the "output" $(C \leftarrow F(B))$.

So that we don't "double-count" current outputs, we require that f and g are injective and have disjoint images. It can be verified that this requirement leads to a subcategory of the cospan category.

The upshot of this is that when we compose spans using pullback, matching up currents and voltages ensures that current is conserved in what is now a new internal node.

6.2 Open Dynamical Systems

In order to apply a similar semantic functor to the category of Petri nets, we will take a brief diversion to discuss what the codomain looks like.

What is a dynamical system? For the sake of this thesis, we will think about a dynamical system as a functor E from the category BM (the category representing a linearly ordered commutative monoid M) into some other category. For instance, in physics, M is $\mathbb{R}_{\geq 0}$ and the "other category" is often the category of manifolds and smooth functions. This gives a smooth "evolution" operator $E_t \colon X \to X$ for each t such that $E_t \circ E_{t'} = E_{t'+t}$.

One way of deriving such a functor is by defining a vector field (i.e. a section of the tangent bundle) on X and letting E_t be the 1-parameter diffeomorphism group associated with that vector field. This will not always work, for instance if the vector field is not well-behaved. However, for the purposes of this exposition, we will not worry about such details. We can get away with this by only considering *algebraic* vector fields on $\mathbb{R}^n$ (i.e., a polynomial map $\mathbb{R}^n \to \mathbb{R}^n$); these are known to be sufficiently well-behaved for our purposes. Therefore, it suffices to define a dynamical system as an algebraic vector field on $\mathbb{R}^n$.

But what is an open dynamical system? In the introduction, we defined an open system as one which receives influence through its boundary. So we need to designate a boundary, and we need to define how the open system receives influence through this boundary.

Designating a boundary at this point is routine: if the system is S (i.e. we will put a differential equation on $\mathbb{R}^S$), then a boundary is a map of sets $B \to S$. Before we define how the system is affected through B, we will allows ourselves some philosophy.

Intuitively, differential equations in dynamical systems enforce conservation laws. Famously, Newton's laws conserve mass, energy, and momentum; Maxwell's laws conserve charge and energy. When we allow these differential equations not to hold, we are allowing conserved quantities to flow in and out of our system. Therefore, material flowing through the boundary is represented by a deviation from the differential equation. We express this deviation by defining a function $D \colon \mathbb{R}_{\geq 0} \to \mathbb{R}^B$, and making a new differential equation

$$\frac{\partial}{\partial t}\, y(t) = A(y(t)) + d_*(D(t))$$

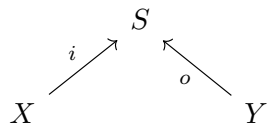
where $d_*(D(t))$ is the *pushforward* of v along d, defined by $d_*(v)_j = \sum_{d(i)=j} v_j \in \mathbb{R}^S$. D represents flows of material or energy through the boundary B.

Example 20. In an electrodynamical system, $D(t)$ could represent the current flowing through the boundary at time t.

Example 21. In a thermodynamical system where y_0, y_1 are the temperature and pressure of the environment, then we might assert through fiat that

$y_0(t) = C_0$ and $y_1(t) = C_1$ for all time, and exempt them from being affected by the system. This would result in heat transfers and volume or molecule transfers through the boundary of the system, and we could capture this by making $D_i(t) = -A_i(y(t))$, so that $\frac{\partial}{\partial t} y_i(t) = 0$.

We want to make a category of open dynamical systems. In order to do this, we need to split the boundary into input and output components, so that we have a cospan.



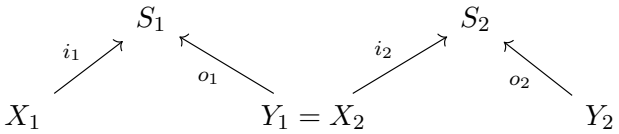
Then, given $I\colon \mathbb{R}_{\geq 0} \to \mathbb{R}^X$ and $O\colon \mathbb{R}_{\geq 0} \to \mathbb{R}^Y$, we define the equation associated to an open dynamical system to be:

$$\frac{\partial}{\partial t} x(t) = A(x(t)) + i_*(I(t)) - o_*(O(t))$$

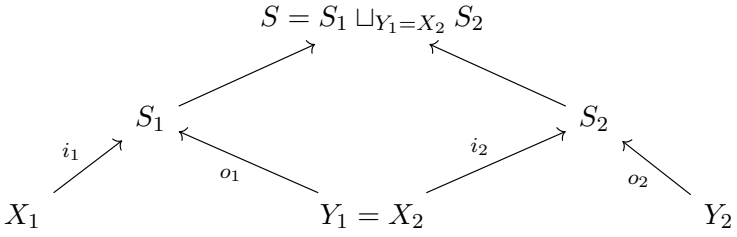
i_* and o_* are the *pushforward* maps, which are defined by:

$$i_*(v)_k = \sum_{i(j)=k} v_j$$
$$o_*(v)_k = \sum_{o(j)=k} v_k$$

Finally, to compose two open dynamical systems



with vector fields $A_1\colon \mathbb{R}^{S_1} \to \mathbb{R}^{S_1}$ and $A_2\colon \mathbb{R}^{S_2} \to \mathbb{R}^{S_2}$, we take the pushout



and associate with it a vector field $A\colon \mathbb{R}^S \to \mathbb{R}^S$ defined by $A(v) = A_1(v|_{S_1}) + A_2(v|_{S_2})$.

That this construction ends up making a well-defined category is proved in [Pol17].

The way that we suggest thinking about the category of open dynamical systems is that when we fix flows on the boundary, we end up being able to evolve the state of the system through time. As we suggested in the introduction, a good way to do this is by defining a functor from the poset $\mathbb{R}$ to the category of manifolds, so that we have a smooth map from $\mathbb{R}^n$ to $\mathbb{R}^n$ for every poset morphism $t \to t'$. The mechanism of the algebraic vector field is a somewhat arbitrary implementation of this, which we put in place because it guarantees good behavior.

6.3 Rate Equation for Petri Nets

Our motivation for introducing the category of open dynamical systems was to give a semantic for Petri nets. By what we said above, it suffices to associate a differential equation to each Petri net (or more formally, an algebraic vector field). Once we have done this, a structured cospan $F(a) \to c \leftarrow F(b)$ with feet finite sets, and with apex a Petri net, will map to an open dynamical system in a natural way.

We derive the form of this differential equation through three principles [BB12]. But before we get into the derivation, we make one clarifying remark. Previously, we associated to each reaction a *rate*. However, we would now like to use "rate" to refer to the speed at which the reaction is taking place at an instant in time. Therefore, we will use the term "rate multiplier" to refer to the earlier "rate", and reserve the term "rate" for *instantaneous* rate.

With that cleared up, here are the three principles.

1. The rate of a reaction only depends on the current concentrations of species, not on the rate of any other reaction.

2. The rate of a reaction is proportional to the product of the current quantities of the inputs, multiplied by the base rate.

3. The derivative of species concentration in a single-reaction Petri net is proportional to the difference between the number of that species produced and consumed by that reaction, multiplied by the rate of that reaction.

These principles have the following consequences.

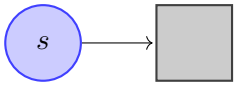


Figure 10: Exponential Decay

1. $A = A_1 + \ldots + A_k$, where A_i is the operator associated to a Petri net with only reaction i. Therefore, we assume that $k = 1$, and we let r be the rate associated to the single reaction, n_i be the number of species i output by the reaction and m_i be the number of species i consumed by the reaction.
2. If $y_i = E_t(x_0)_i$, then the rate of the one reaction is $r \prod_i y_i^{m_i}$.
3. The jth component of $A(y)$ is $(n_j - m_j) r \prod_i y_i^{m_i}$.

The differential equation that we end up with after applying these principles is called the *rate equation*. It is possible to write down the rate equation in a closed form, but it is our opinion that no clarity is gained from that. Instead, we will do examples.

Example 22. Suppose that we are tasked to model the behavior of a lump of uranium. Famously, uranium randomly decays into other particles (which we don't care about). We model this with a Petri net with one species, uranium, and one reaction, decay, which has input a single molecule of uranium, and output nothing. The schematic for this is in Figure 10. We only have one species and one reaction, and $m_0 = 1$ while $n_0 = 0$. Therefore, $A(y) = (0 - 1) r y^1 = -ry$. This is the well-known equation for exponential decay, with solution $y_t = y_0 e^{-rt}$. Therefore, $E_t = e^{-rt}$. It is rare that there is a closed form solution for E_t, except for in simple cases like this one.

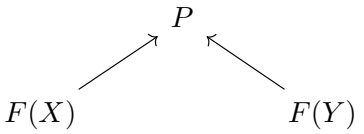
Example 23. Recall the Petri net presented in Figure 2. Let r_B, r_P and r_D be the rate multipliers associated to "birth", "predation", and "death", and let y_R be the concentration of rabbits while y_W is the concentration of wolves. Then the rate of birth is $r_B y_R$, the rate of predation is $r_P y_R y_W$, and the rate of death is $r_D y_W$. The resulting system of differential equations is

$$\frac{\partial}{\partial t} y_R = r_B y_R - r_P y_R y_W$$
$$\frac{\partial}{\partial t} y_W = r_P y_R y_W - r_D y_W$$

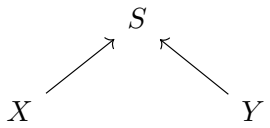
There is not a corresponding closed form solution for the evolution operator, however this can be evaluated numerically.

The author has created a tool for numerical solutions of the rate equation, which is available online at `https://owenlynch.org/static/ez-petri`. No download is required: it runs in the browser. The source code is also freely available at `https://github.com/olynch/ez-petri`.

Now that we have defined the rate equation, we can define a functor from the category of open Petri nets to the category of open dynamical systems. This functor is more or less straightforward: we send an open Petri net



to a open dynamical system with underlying cospan



where S is the set of species in P, along with the rate equation on S.

References

[BB12] John C. Baez and Jacob Biamonte. *Quantum Techniques for Stochastic Mechanics*. 2012. arXiv: 1209.3632v5 [quant-ph].

[BC18] John C. Baez and Kenny Courser. "Coarse-Graining Open Markov Processes." In: (Nov. 21, 2018). arXiv: 1710.11343 [math-ph]. URL: http://arxiv.org/abs/1710.11343 (visited on 03/05/2020).

[BC20] John C. Baez and Kenny Courser. "Structured Cospans." In: (Jan. 3, 2020). arXiv: 1911.04630 [math]. URL: http://arxiv.org/abs/1911.04630 (visited on 04/13/2020).

[BS09] John C. Baez and Mike Stay. "Physics, Topology, Logic and Computation: A Rosetta Stone." In: (2009). arXiv: 0903.0340v3 [quant-ph].

[Fon16] Brendan Fong. "The Algebra of Open and Interconnected Systems." In: (Sept. 17, 2016). arXiv: 1609.05382 [math]. URL: http://arxiv.org/abs/1609.05382 (visited on 04/13/2020).

[FS19] Brendan Fong and David I. Spivak. *An Invitation to Applied Category Theory: Seven Sketches in Compositionality*. 1st ed. Cambridge University Press, July 18, 2019. ISBN: 978-1-108-66880-4 978-1-108-48229-5 978-1-108-71182-1. DOI: 10.1017/9781108668804. URL: https://www.cambridge.org/core/product/identifier/9781108668804/type/book (visited on 04/13/2020).

[Lei03] Tom Leinster. "Higher Operads, Higher Categories." In: (May 2, 2003). arXiv: math/0305049. URL: http://arxiv.org/abs/math/0305049 (visited on 04/23/2020).

[Pol17] Blake S. Pollard. "Open Markov Processes and Reaction Networks." In: (Sept. 29, 2017). arXiv: 1709.09743 [cond-mat, physics:math-ph]. URL: http://arxiv.org/abs/1709.09743 (visited on 04/13/2020).

[Sch81] J. Schnakenberg. *Thermodynamic Network Analysis of Biological Systems*. Universitext. Berlin, Heidelberg: Springer Berlin Heidelberg, 1981. ISBN: 978-3-540-10612-8 978-3-642-67971-1. DOI: 10.1007/978-3-642-67971-1. URL: http://link.springer.com/10.1007/978-3-642-67971-1 (visited on 04/13/2020).