Network Models from Petri Nets with Catalysts

John C. Baez[1,2], John Foley[3], and Joe Moeller[1]

[1]Department of Mathematics, University of California, Riverside CA, 92521, USA

[2]Centre for Quantum Technologies, National University of Singapore, 117543, Singapore

[3]Metron, Inc., 1818 Library St., Suite 600, Reston, VA 20190, USA

Petri networks and network models are two frameworks for the compositional design of systems of interacting entities. Here we show how to combine them using the concept of a 'catalyst': an entity that is neither destroyed nor created by any process it engages in. In a Petri net, a place is a catalyst if its in-degree equals its out-degree for every transition. We show how a Petri net with a chosen set of catalysts gives a network model. This network model maps any list of catalysts from the chosen set to the category whose morphisms are all the processes enabled by this list of catalysts. Applying the Grothendieck construction, we obtain a category fibered over the category whose objects are lists of catalysts. This category has as morphisms all processes enabled by *some* list of catalysts. While this category has a symmetric monoidal structure that describes doing processes in parallel, its fibers also have premonoidal structures that describe doing one process and then another while reusing the catalysts.

1 Introduction

Petri nets are a widely studied formalism for describing collections of entities of different types, and how they turn into other entities [9, 18]. Network models are a formalism for designing and tasking networks of agents [2, 16]. Here we combine the two. This is worthwhile because while both formalisms involve networks, they serve different functions, and are in some sense complementary.

A Petri net can be drawn as a bipartite directed graph with vertices of two kinds: 'places', drawn as circles below, and 'transitions' drawn as squares:
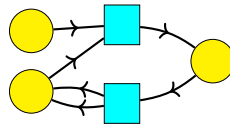
In applications to chemistry, places are also called 'species'. When we run a Petri net, we start by placing a finite number of 'tokens' in each place:
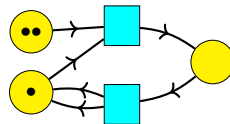
This is called a 'marking'. Then we repeatedly change the marking using the transitions. For example, the above marking can change to this:
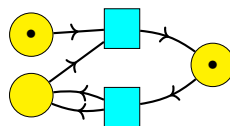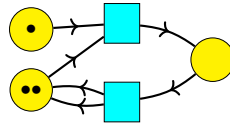
John C. Baez: baez@math.ucr.edu
John Foley: foley@metsci.com
Joe Moeller: moeller@math.ucr.edu

and then this:

Thus, the places represent different *types* of entity, and the transitions describe ways that one collection of entities of specified types can turn into another such collection.

Network models serve a different function than Petri nets: they are a general tool for working with networks of many kinds. Mathematically a network model is a lax symmetric monoidal functor $G\colon \mathsf{S}(C) \to \mathsf{Cat}$, where $\mathsf{S}(C)$ is the free strict symmetric monoidal category on a set C. Elements of C represent different kinds of 'agents'. Unlike in a Petri net, we do not usually consider processes where these agents turn into other agents. Instead, we wish to study everything that can be done with a fixed collection of agents. Any object $x \in \mathsf{S}(C)$ is of the form $c_1 \otimes \cdots \otimes c_n$ for some $c_i \in C$; thus, it describes a collection of agents of various kinds. The functor G maps this object to a category $G(x)$ that describes everything that can be done with this collection of agents.

In many examples considered so far, $G(x)$ is a category whose morphisms are graphs whose nodes are agents of types $c_1, \ldots, c_n$. Composing these morphisms corresponds to 'overlaying' graphs. Network models of this sort let us design networks where the nodes are agents and the edges are communication channels or shared commitments. In our first paper the operation of overlaying graphs was always commutative [2]. Subsequently we introduced a more general noncommutative overlay operation [16]. This lets us design networks where each agent has a limit on how many communication channels or commitments it can handle; the noncommutativity allows us to take a 'first come, first served' approach to resolving conflicting commitments.

Here we take a different tack: we instead take $G(x)$ to be a category whose morphisms are *processes that the given collection of agents, x, can carry out*. Composition of morphisms corresponds to carrying out first one process and then another.

This idea meshes well with Petri net theory, because any Petri net P determines a symmetric monoidal category FP whose morphisms are processes that can be carried out using this Petri net. More precisely, the objects in FP are markings of P, and the morphisms are sequences of ways to change these markings using transitions, e.g.:
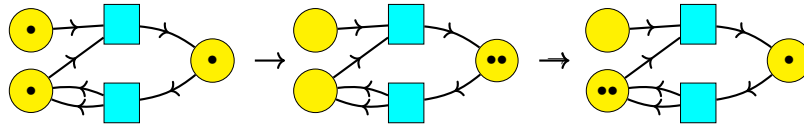
Given a Petri net, then, how do we construct a network model $G\colon \mathsf{S}(C) \to \mathsf{Cat}$, and in particular, what is the set C? In a network model the elements of C represent different kinds of agents. In the simplest scenario, these agents persist in time. Thus, it is natural to take C to be some set of 'catalysts'. In chemistry, a reaction may require a catalyst to proceed, but it neither increases nor decrease the amount of this catalyst present. For a Petri net, 'catalysts' are species that are neither increased nor decreased in number by any transition. For example, species a is a catalyst in the following Petri net, so we outline it in red:
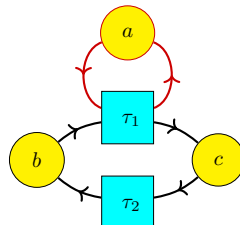
but neither b nor c is a catalyst. The transition τ_1 requires one token of type a as input to proceed, but it also outputs one token of this type, so the total number of such tokens is unchanged. Similarly, the transition τ_2 requires no tokens of type a as input to proceed, and it also outputs no tokens of this type, so the total number of such tokens is unchanged.

In Theorem 9 we prove that given any Petri net P, and any subset C of the catalysts of P, there is a network model $G\colon \mathsf{S}(C) \to \mathsf{Cat}$. An object $x \in \mathsf{S}(C)$ says how many tokens of each catalyst are present; $G(x)$ is then the subcategory of FP where the objects are markings that have this specified amount of each catalyst, and morphisms are processes going between these.

From the functor $G\colon \mathsf{S}(C) \to \mathsf{Cat}$ we can construct a category $\int G$ by 'gluing together' all the categories $G(x)$ using the Grothendieck construction. Because G is symmetric monoidal we can use an enhanced version of this construction to make $\int G$ into a symmetric monoidal category [15]. The tensor product in $\int G$ describes doing processes 'in parallel'. The category $\int G$ is similar to FP, but it is better suited to applications where agents each have their own 'individuality', because FP is actually a *commutative* monoidal category, where permuting agents has no effect at all, while $\int G$ is not so degenerate. In Theorem 12 we make this precise by more concretely describing $\int G$ as a symmetric monoidal category, and clarifying its relation to FP.

There are no morphisms between an object of $G(x)$ and an object of $G(x')$ unless $x \cong x'$, since no transitions can change the amount of catalysts present. The category FP is thus a 'disjoint union', or more precisely a coproduct, of subcategories FP_i where i, an element of free commutative monoid on C, specifies the amount of each catalyst present. The tensor product on FP has the property that tensoring an object in FP_i with one in FP_j gives an object in FP_{i+j}, and similarly for morphisms.

However, in Prop. 15 we show that each subcategory FP_i also has its own tensor product, which describes doing one process and then another while reusing catalyst tokens. This tensor product makes FP_i into a 'premonoidal category'—an interesting generalization of a monoidal category which we recall. Finally, in Theorem 17 we show that these monoidal structures define a lift of the functor $G\colon \mathsf{S}(C) \to \mathsf{Cat}$ to a functor $\hat{G}\colon \mathsf{S}(C) \to \mathsf{PreMonCat}$, where $\mathsf{PreMonCat}$ is the category of strict premonoidal categories.

2 Petri Nets

A Petri net generates a symmetric monoidal category whose objects are tensor products of species and whose morphisms are built from the transitions by repeatedly taking composites and tensor products. There is a long line of work on this topic starting with the papers of Meseguer–Montanari [14] and Engberg–Winskel [7], both dating to roughly 1990. It continues to this day, because the issues involved are surprisingly subtle [6, 13, 20–23]. In particular, there are various kinds of symmetric monoidal categories to choose from. Following our work with Master [3] we use 'commutative' monoidal categories. These are just commutative monoid objects in Cat, so their associator:

$$\alpha_{a,b,c}\colon (a \otimes b) \otimes c \xrightarrow{\sim} a \otimes (b \otimes c),$$

their left and right unitor:

$$\lambda_a\colon I \otimes a \xrightarrow{\sim} a, \qquad \rho_a\colon a \otimes I \xrightarrow{\sim} a,$$

and even their braiding:

$$\sigma_{a,b}\colon a \otimes b \xrightarrow{\sim} b \otimes a$$

are all identity morphisms. While every symmetric monoidal category is equivalent to one with trivial associator and unitors, this ceases to be true if we also require the braiding to be trivial. However, it seems that Petri nets most naturally serve to present symmetric monoidal categories of this very strict sort. Thus, we shall describe a functor from the category of Petri nets to the category of commutative monoidal categories, which we call CMC:

$$F\colon \mathsf{Petri} \to \mathsf{CMC}.$$

To begin, let CMon be the category of commutative monoids and monoid homomorphisms. There is a forgetful functor from CMon to Set that sends commutative monoids to their underlying sets and monoid homomorphisms to their underlying functions. It has a left adjoint $\mathbb{N}\colon \mathsf{Set} \to \mathsf{CMon}$ sending any set X to the free commutative monoid on X. An element $a \in \mathbb{N}[X]$ is formal linear combination of elements of X:

$$a = \sum_{x \in X} a_x\, x,$$

where the coefficients a_x are natural numbers and all but finitely many are zero. The set X naturally includes in $\mathbb{N}[X]$, and for any function $f\colon X \to Y$, $\mathbb{N}[f]\colon \mathbb{N}[X] \to \mathbb{N}[Y]$ is the unique monoid homomorphism that extends f. We often abuse language and use $\mathbb{N}[X]$ to mean the underlying set of the free commutative monoid on X.

Definition 1. *A* **Petri net** *is a pair of functions of the following form:*

$$T \underset{t}{\overset{s}{\rightrightarrows}} \mathbb{N}[S].$$

We call T the set of **transitions**, *S the set of* **places** *or* **species**, *s the* **source** *function, and t the* **target** *function. We call an element of $\mathbb{N}[S]$ a* **marking** *of the Petri net.*

For example, in this Petri net:
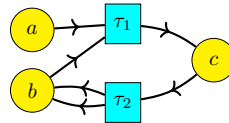
we have $S = \{a, b, c\}$, $T = \{\tau_1, \tau_2\}$, and

$$\begin{aligned}
s(\tau_1) &= a + b & t(\tau_1) &= c \\
s(\tau_2) &= c & t(\tau_2) &= 2b.
\end{aligned}$$

The term 'species' is used in applications of Petri nets to chemistry. Since the concept of 'catalyst' also arose in chemistry, we henceforth use the term 'species' rather than 'places'.

Definition 2. *A* **Petri net morphism** *from the Petri net P to the Petri net P' is a pair of functions $(f\colon T \to T',\ g\colon S \to S')$ such that the following diagrams commute:*

$$
\begin{array}{ccc}
T & \overset{s}{\longrightarrow} & \mathbb{N}[S] \\
{\scriptstyle f}\downarrow & & \downarrow{\scriptstyle \mathbb{N}[g]} \\
T' & \underset{s'}{\longrightarrow} & \mathbb{N}[S']
\end{array}
\qquad
\begin{array}{ccc}
T & \overset{t}{\longrightarrow} & \mathbb{N}[S] \\
{\scriptstyle f}\downarrow & & \downarrow{\scriptstyle \mathbb{N}[g]} \\
T' & \underset{t'}{\longrightarrow} & \mathbb{N}[S']
\end{array}
$$

Let Petri *denote the category of Petri nets and Petri net morphisms with composition defined by*

$$(f, g) \circ (f', g') = (f \circ f', g \circ g').$$

Definition 3. *A* **commutative monoidal category** *is a commutative monoid object in* $(\mathsf{Cat}, \times)$. *Let* CMC *denote the category of commutative monoid objects in* $(\mathsf{Cat}, \times)$.

More concretely, a commutative monoidal category is a strict monoidal category for which $a \otimes b = b \otimes a$ for all pairs of objects and all pairs of morphisms, and the braid isomorphism $a \otimes b \to b \otimes a$ is the identity map.

Every Petri net $P = (s, t\colon T \to \mathbb{N}[S])$ gives rise to a commutative monoidal category FP as follows. We take the commutative monoid of objects $\mathrm{Ob}(FP)$ to be the free commutative monoid on S. We construct the commutative monoid of morphisms $\mathrm{Mor}(FP)$ as follows. First we generate morphisms recursively:

- for every transition $\tau \in T$ we include a morphism $\tau\colon s(\tau) \to t(\tau)$;

- for any object a we include a morphism $1_a\colon a \to a$;

- for any morphisms $f\colon a \to b$ and $g\colon a' \to b'$ we include a morphism denoted $f + g\colon a + a' \to b + b'$ to serve as their tensor product;

- for any morphisms $f\colon a \to b$ and $g\colon b \to c$ we include a morphism $g \circ f\colon a \to c$ to serve as their composite.

Then we quotient by an equivalence relation on morphisms that imposes the laws of a commutative monoidal category, obtaining the commutative monoid $\mathrm{Mor}(FP)$.

Similarly, morphisms between Petri nets give morphisms between their commutative monoidal categories. Given a Petri net morphism

$$
\begin{array}{ccc}
T & \Longrightarrow & \mathbb{N}[S] \\
f \downarrow & & \downarrow \mathbb{N}[g] \\
T' & \Longrightarrow & \mathbb{N}[S']
\end{array}
$$

we define the functor $F(f,g)\colon FP \to FP'$ to be $\mathbb{N}[g]$ on objects, and on morphisms to be the unique map extending f that preserves identities, composition, and the tensor product. This functor is strict symmetric monoidal.

Proposition 4. *There is a functor* $F\colon \mathsf{Petri} \to \mathsf{CMC}$ *defined as above.*

Proof. This is straightforward; the proof that F is a left adjoint is harder [13], but we do not need this here. ∎

3 Catalysts

One thinks of a transition τ of a Petri net as a process that consumes the source species $s(\tau)$ and produces the target species $t(\tau)$. An example of something that can be represented by a Petri net is a chemical reaction network [1, 4]. Indeed, this is why Carl Petri originally invented them. A 'catalyst' in a chemical reaction is a species that is necessary for the reaction to occur, or helps lower the activation energy for reaction, but is neither increased nor depleted by the reaction. We use a modest generalization of this notion, defining a *catalyst* in a Petri net to be a species that is neither increased nor depleted by *any* transition in the Petri net.

Given a Petri net $s,t\colon T \to \mathbb{N}[S]$, recall that for any marking $a \in \mathbb{N}[S]$ we have

$$
a = \sum_{x \in S} a_x x
$$

for certain coefficients $a_x \in \mathbb{N}$. Thus, for any transition τ of a Petri net, $s(\tau)_x$ is the coefficient of the place x in the source of τ, while $t(\tau)_x$ is its coefficient in the target of τ.

Definition 5. *A species* $x \in S$ *in a Petri net* $P = (s,t\colon T \to \mathbb{N}[S])$ *is called a* **catalyst** *if* $s(\tau)_x = t(\tau)_x$ *for every transition* $\tau \in T$. *Let* $S_{\mathrm{cat}} \subseteq S$ *denote the set of catalysts in* P.

Definition 6. *A* **Petri net with catalysts** *is a Petri net* $P = (s,t\colon T \to \mathbb{N}[S])$ *with a chosen subset* $C \subseteq S_{\mathrm{cat}}$. *We denote a Petri net* P *with catalysts* C *as* (P,C).

Suppose we have a Petri net with catalysts (P,C). Recall that the set of objects of FP is the free commutative monoid $\mathbb{N}[C]$. We have a natural isomorphism

$$
\mathbb{N}[S] \cong \mathbb{N}[C] \times \mathbb{N}[S \setminus C].
$$

We write

$$
\pi_C \colon \mathbb{N}[S] \to \mathbb{N}[C]
$$

for the projection. Given any object $a \in FP$, $\pi_C(a)$ says how many catalysts of each species in C occur in a.

Definition 7. *Given a Petri net with catalysts* (P,C) *and any* $i \in \mathbb{N}[C]$, *let* FP_i *be the full subcategory of* FP *whose objects are objects* $a \in FP$ *with* $\pi_C(a) = i$.

Morphisms in FP_i describe processes that the Petri net can carry out with a specific fixed amount of every catalyst. Since no transition in P creates or destroys any catalyst, if $f\colon a \to b$ is a morphism in FP then

$$
\pi_C(a) = \pi_C(b).
$$

Thus, FP is the coproduct of all the subcategories FP_i:

$$FP \cong \coprod_{i \in \mathbb{N}[C]} FP_i$$

as categories. The subcategories FP_i are not generally monoidal subcategories because if $a, b \in FP$ and $a + b$ is their tensor product then

$$\pi_C(a + b) = \pi_C(a) + \pi_C(b)$$

so for any $i, j \in \mathbb{N}[C]$ we have

$$a \in FP_i, \ b \in FP_j \Rightarrow a + b \in FP_{i+j}$$

and similarly for morphisms. Thus, we can think of FP as a commutative monoidal category 'graded' by $\mathbb{N}[C]$. But note we are free to reinterpret any process as using a *greater* amount of various catalysts, by tensoring it with identity morphism on this *additional* amount of catalysts. That is, given any morphism in FP_i, we can always tensor it with the identity on j to get a morphism in FP_{i+j}.

Since $\mathbb{N}[C]$ is a commutative monoid we can think of it as a commutative monoidal category with only identity morphisms, and we freely do this in what follows. Network models rely on a similar but less trivial way of constructing a symmetric monoidal category from a set C. Namely, for any set C there is a category $\mathsf{S}(C)$ for which:

- Objects are formal expressions of the form

$$c_1 \otimes \cdots \otimes c_n$$

  for $n \in \mathbb{N}$ and $c_1, \ldots, c_n \in C$. When $n = 0$ we write this expression as I.

- There exist morphisms

$$f \colon c_1 \otimes \cdots \otimes c_m \to c_1' \otimes \cdots \otimes c_n'$$

  only if $m = n$, and in that case a morphism is a permutation $\sigma \in S_n$ such that $c_{\sigma(i)}' = c_i$ for all $i = 1, \ldots, n$.

- Composition is the usual composition of permutations.

In short, an object of $\mathsf{S}(C)$ is a list of catalysts, possibly empty, and allowing repetitions. A morphism is a permutation that maps one list to another list.

As shown in [2, Prop. 17], $\mathsf{S}(C)$ is the free strict symmetric monoidal category on the set C. There is thus a strict symmetric monoidal functor

$$p \colon \mathsf{S}(C) \to \mathbb{N}[C]$$

sending each object $c_1 \otimes \cdots \otimes c_n$ to the object $c_1 + \cdots + c_n$, and sending every morphism to an identity morphism. This can also be seen directly. In what follows, we use this functor p to construct a lax symmetric monoidal functor $G \colon \mathsf{S}(C) \to \mathsf{Cat}$, where Cat is made symmetric monoidal using its cartesian product.

Proposition 8. *Given a Petri net with catalysts (P, C), there exists a unique functor $G \colon \mathsf{S}(C) \to \mathsf{Cat}$ sending each object $x \in \mathsf{S}(C)$ to the category $FP_{p(x)}$ and each morphism in $\mathsf{S}(C)$ to an identity functor.*

Proof. The uniqueness is clear. For existence, note that since $\mathbb{N}[C]$ has only identity morphisms there is a functor $H \colon \mathbb{N}[C] \to \mathsf{Cat}$ sending each object $x \in \mathbb{N}[C]$ to the category $FP_{p(x)}$. If we compose H with the functor $p \colon \mathsf{S}(C) \to \mathbb{N}[C]$ described above we obtain the functor G. ∎

Theorem 9. *The functor $G \colon \mathsf{S}(C) \to \mathsf{Cat}$ becomes lax symmetric monoidal with the lax structure map*

$$\Phi_{x,y} \colon FP_{p(x)} \times FP_{p(y)} \to FP_{p(x \otimes y)}$$

given by the tensor product in FP, and the map

$$\phi\colon 1 \to FP_0$$

sending the unique object of the terminal category $1 \in \mathsf{Cat}$ to the unit for the tensor product in FP, which is the object $0 \in FP_0$.

Proof. Recall that G is the composite of $p\colon \mathsf{S}(C) \to \mathbb{N}[C]$ and $H\colon \mathbb{N}[C] \to \mathsf{Cat}$. The functor p is strict symmetric monoidal. The functor p is strict symmetric monoidal. One can check that the functor H becomes lax symmetric monoidal if we equip it with the lax structure map

$$FP_i \times FP_j \to FP_{i+j}$$

given by the tensor product in FP, and the map

$$1 \to FP_0$$

sending the unique object of $1 \in \mathsf{Cat}$ to the unit for the tensor product in FP, namely $0 \in \mathbb{N}[S] = \mathrm{Ob}(FP)$. Composing the lax symmetric monoidal functor H and with the strict symmetric monoidal functor p, we obtain the lax symmetric monoidal functor G described in the theorem statement. ∎

In our previous paper [2], a **C-colored network model** was defined to be a lax symmetric monoidal functor from $\mathsf{S}(C)$ to Cat.

Definition 10. *We call the C-colored network model $G\colon \mathsf{S}(C) \to \mathsf{Cat}$ of Theorem 9 the **Petri network model** associated to the Petri net with catalysts (P, C).*

Example 11. *The following Petri net P has species $S = \{a, b, c, d, e\}$ and transitions $T = \{\tau_1, \tau_2\}$:*
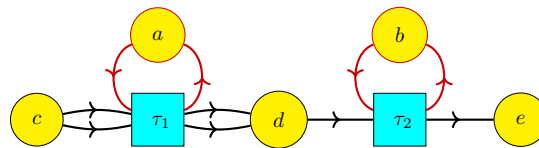

Species a and b are catalysts, and the rest are not. We thus can take $C = \{a, b\}$ and obtain a Petri net with catalysts (P, C), which in turn gives a Petri network model $G\colon \mathsf{S}(C) \to \mathsf{Cat}$. We outline catalyst species in red, and also draw the edges connecting them to transitions in red.

Here is one possible interpretation of this Petri net. Tokens in c represent people at a base on land, tokens in d are people at the shore, and tokens in e are people on a nearby island. Tokens in a represent jeeps, each of which can carry two people at a time from the base to the shore and then return to the base. Tokens in b represent boats that carry one person at a time from the shore to the island and then return.

Let us examine the effect of the functor $G\colon \mathsf{S}(C) \to \mathsf{Cat}$ on various objects of $\mathsf{S}(C)$. The object $a \in \mathsf{S}(C)$ describes a situation where there is one jeep present but no boats. The category $G(a)$ is isomorphic to FX, where X is this Petri net:
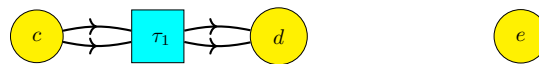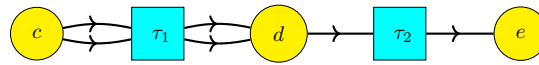

That is, people can go from the base to the shore in pairs, but they cannot go to the island. Similarly, the object b describes a situation with one boat present but no jeeps, and the category $G(b)$ is isomorphic to FY, where Y is this Petri net:
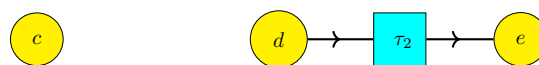
Now people can only go from the shore to the island, one at a time.

The object $a \otimes b \in \mathsf{S}(C)$ describes a situation with one jeep and one boat. The category $G(a \otimes b)$ is isomorphic to FZ for this Petri net Z:

Now people can go from the base to the shore in pairs and also go from the shore to the island one at a time.

Surprisingly, an object $x \in \mathsf{S}(C)$ with additional jeeps and/or boats always produces a category $G(x)$ that is isomorphic to one of the three just shown: $G(a), G(b)$ and $G(a \otimes b)$. For example, consider the object $b \otimes b \in \mathsf{S}(C)$, where there are two boats present but no jeeps. There is an isomorphism of categories

$$- + b \colon G(b) \to G(b \otimes b)$$

defined as follows. Recall that $G(b) = FP_b$ and $G(b \otimes b) = FP_{b+b}$, where FP_b and FP_{b+b} are subcategories of FP. The functor

$$- + b \colon FP_b \to FP_{b+b}$$

sends each object $x \in FP_b$ to the object $x + b$, and sends each morphism $f \colon x \to y$ in FP_b to the morphism $1_b + f \colon b + x \to b + y$. That this defines a functor is clear; the surprising part is that it is an isomorphism. One might have thought that the presence of a second boat would enable one to carry out a given task in more different ways.

Indeed, while this is true in real life, the category FP is commutative monoidal, so tokens of the same species have no 'individuality': permuting them has no effect. There is thus, for example, no difference between the following two morphisms in FP_{b+b}:

- *using one boat to transport one person from the base to shore and another boat to transport another person, and*

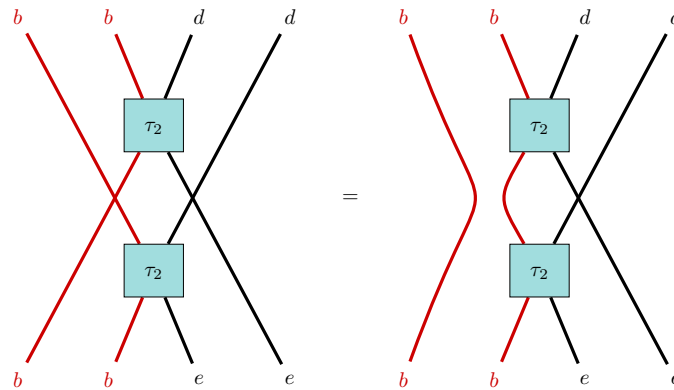- *using one boat to transport first one person and then another.*

It is useful to draw morphisms in FP as string diagrams, since such diagrams serve as a general notation for morphisms in monoidal categories [12]. For expository treatments, see [5, 24]. The rough idea is that objects of a monoidal category are drawn as labelled wires, and a morphism $f \colon x_1 \otimes \cdots \otimes x_m \to y_1 \otimes \cdots \otimes y_n$ is drawn as a box with m wires coming in on top and n wires coming out at the bottom. Composites of morphisms are drawn by attaching output wires of one morphism to input wires of another, while tensor products of morphisms are drawn by setting pictures side by side. In symmetric monoidal categories, the braiding is drawn as a crossing of wires. The rules governing string diagrams let us manipulate them while not changing the morphisms they denote. In the case of symmetric monoidal categories, these rules are well known [12, 24]. For commutative monoidal categories there is one additional rule:

This says both that $x \otimes y = y \otimes x$ and that the braiding $\sigma_{x,y} \colon x \otimes y \to y \otimes x$ is the identity.

Here is the string diagram notation for the equation we mentioned between two morphisms in

FP:

We draw the object b (standing for a boat) in red to emphasize that it serves as a catalyst. At left we are first using one boat to transport one person from the base to shore, and then using another boat to transport another person. At right we are using the same boat to transport first one person and then another, while another boat stands by and does nothing. These morphisms are equal because they differ only by the presence of the braiding $\sigma_{b,b}: b + b \to b + b$ in the left hand side, and this is an identity morphism.

The above example illustrates an important point: in the commutative monoidal category FP, permuting catalyst tokens has no effect. Next we construct a symmetric monoidal category $\int G$ in which permuting such tokens has a nontrivial effect. One reason for wanting this is that in applications, the catalyst tokens may represent agents with their own individuality. For example, when directing a boat to transport a person from base to shore, we need to say *which* boat should do this. For this we need a symmetric monoidal category that gives the catalyst tokens a nontrivial braiding.

To create this category, we use the symmetric monoidal Grothendieck construction [15]. Given any symmetric monoidal category X and any lax symmetric monoidal functor $F: X \to \mathsf{Cat}$, this construction gives a symmetric monoidal category $\int F$ equipped with a functor (indeed an opfibration) $\int F \to X$. In our previous work [2] we used this construction to build an operad from any network model, whose operations are ways to assemble larger networks from smaller ones. Now this construction has a new significance.

Starting from a Petri network model $G: \mathsf{S}(C) \to \mathsf{Cat}$, the symmetric monoidal Grothendieck construction gives a symmetric monoidal category $\int G$ in which:

- an object is a pair (x, a) where $x \in \mathsf{S}(C)$ and $a \in FP_{p(x)}$.

- a morphism from (x, a) to (x', a') is a pair (σ, f) where $\sigma: x \to x'$ is a morphism in $\mathsf{S}(C)$ and $f: a \to a'$ is a morphism in FP.

- morphisms are composed componentwise.

- the tensor product is computed componentwise: in particular, the tensor product of objects (x, a) and (x', a') is $(x \otimes x', a + a')$.

- the associators, unitors and braiding are also computed componentwise (and hence are trivial in the second component, since FP is a commutative monoidal category).

The functor $\int G \to \mathsf{S}(C)$ simply sends each pair to its first component.

This is simpler than one typically expects from the Grothendieck construction. There are two main reasons: first, G maps every morphism in $\mathsf{S}(C)$ to an identity morphism in Cat, and second, the lax structure map for G is simply the tensor product in FP. However, this construction still has an important effect: it makes the process of switching two tokens of the same catalyst species into a nontrivial morphism in $\int G$. More formally, we have:

Theorem 12. *If $G: \mathsf{S}(C) \to \mathsf{Cat}$ is the Petri network model associated to the Petri net with catalysts (P, C), then $\int G$ is equivalent, as a symmetric monoidal category, to the full subcategory of $\mathsf{S}(C) \times FP$ whose objects are those of the form (x, a) with $x \in \mathsf{S}(C)$ and $a \in FP_{p(x)}$.*

Proof. One can read this off from the description of $\int G$ given above. ∎

The difference between $\int G$ and FP is that the former category keeps track of processes where catalyst tokens are permuted, while the latter category treats them as identity morphisms. In the terminology of Glabbeek and Plotkin, $\int G$ implements the 'individual token philosophy' on catalysts, in which permuting tokens of the same catalyst is regarded as having a nontrivial effect [11]. By contrast, FP implements the 'collective token philosophy', where all that matters is the *number* of tokens of each catalyst, and permuting them has no effect.

There is a map from $\int G$ to FP that forgets the individuality of the catalyst tokens. A morphism in $\int G$ is a pair (σ, f) where $\sigma \colon x \to x'$ is a morphism in $\mathsf{S}(C)$ and $f \colon a \to a'$ is a morphism in FP with $a \in G(x), a' \in G(x')$. There is a symmetric monoidal functor

$$\int G \to FP$$

that discards this extra information, mapping (σ, f) to f. The symmetric monoidal Grothendieck construction also gives a symmetric monoidal functor

$$\int G \to \mathsf{S}(C)$$

and this maps (σ, f) to σ. This functor is an opfibration on general grounds [15].

Example 13. *Let (P, C) be the Petri net with catalysts in Ex. 11, and $G \colon \mathsf{S}(C) \to \mathsf{Cat}$ the resulting Petri network model. In $\int G$ the following two morphisms are not equal:*
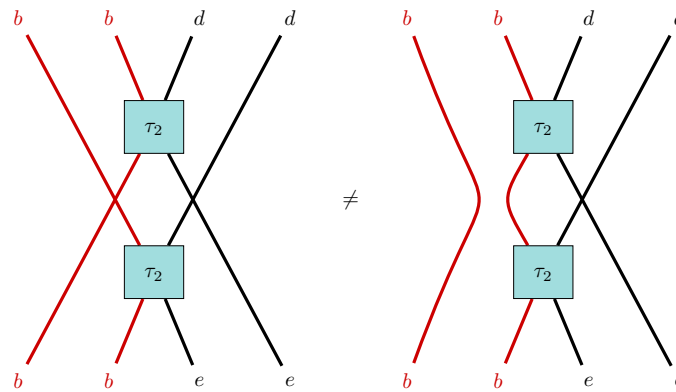
because the braiding of catalyst species in $\int G$ is nontrivial. This says that in $\int G$ we consider these two processes as different:

- *using one boat to transport one person from the base to shore and another boat to transport another person, and*

- *using one boat to transport first one person and then another.*

On the other hand, in $\int G$ we have

Accepted in Compositionality on 2019-11-26. Click on the title to verify.

10
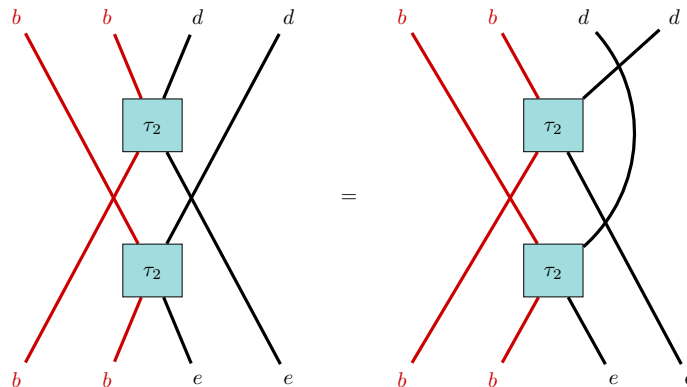
because these morphisms differ only by two people on the shore switching place before they board the boats, and the braiding of non-catalyst species is the identity. In short, the $\int G$ construction implements the individual token philosophy only for catalyst tokens; tokens of other species are governed by the collective token philosophy.

4 Premonoidal Categories

We have seen that for a Petri net P, a choice of catalysts C lets us write the category FP as a coproduct of subcategories FP_i, one for each possible amount $i \in \mathbb{N}[C]$ of the catalysts. The subcategory FP_i is only a *monoidal* subcategory when $i = 0$. Indeed, only FP_0 contains the monoidal unit of FP. However, we shall see that each subcategory FP_i can be given the structure of a *premonoidal* category, as defined by Power and Robinson [19]. We motivate our use of this structure by describing two failed attempts to make FP_i into a monoidal category.

Given two morphisms in FP_i we typically cannot carry out these two processes simultaneously, because of the limited availability of catalysts. But we can do first one and then the other. For example, imagine that two people are trying to walk through a doorway, but the door is only wide enough for one person to walk through. The door is a resource that is not depleted by its use, and thus a catalyst. Both people can use the door, but not at the same time: they must make an arbitrary choice of who goes first.

We can attempt to define a tensor product on FP_i using this idea. Fix some amount of catalysts $i \in \mathbb{N}[C]$. Objects of FP_i are of the form $i + a$ with $a \in \mathbb{N}[S - C]$. On objects we define
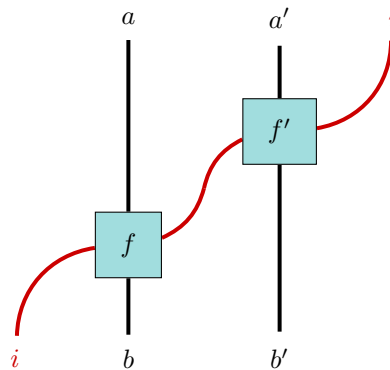
$$(i + a) \otimes_i (i + a') = i + a + a'.$$

The unit object for $\otimes_i$ is therefore $i + 0$, or simply i. For morphisms

$$f : i + a \to i + b$$
$$f' : i + a' \to i + b'$$

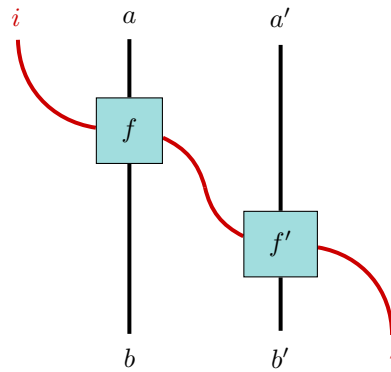we define

$$f \otimes_i f' = (f + 1_{b'}) \circ (1_a + f').$$

The tensor product $f \otimes_i f' = (f + 1_{b'}) \circ (1_a + f')$ of morphisms in FP_i involves an arbitrary choice: namely, the choice to do f' first. This is perhaps clearer if we draw this morphism as a string diagram in FP.

If instead we choose to do f first, we can define a tensor product $_i\otimes$ which is the same on objects but given on morphisms by

$$f \,_i\otimes f' = (1_b + f') \circ (f + 1_{a'}).$$

It looks like this:

Unfortunately, neither of these tensor products makes FP_i into a monoidal category! Each makes the set of objects $\mathrm{Ob}(FP_i)$ and the set of morphisms $\mathrm{Mor}(FP_i)$ into a monoid in such a way that the source and target maps $s, t \colon \mathrm{Mor}(FP_i) \to \mathrm{Ob}(FP_i)$, as well as the identity-assigning map $i \colon \mathrm{Ob}(FP_i) \to \mathrm{Mor}(FP_i)$, are monoid homomorphisms. The problem is that neither obeys the interchange law, so neither of these tensor products defines a functor from $FP_i \times FP_i$ to FP_i. For example,

$$(1 \otimes_i f') \circ (f \otimes_i 1) \neq (f \otimes_i 1) \circ (1 \otimes_i f').$$

The other tensor product suffers from the same problem.

What is going on here? It turns out that FP_i is a 'strict premonoidal category'. While these structures first arose in computer science [19], they are also mathematically natural, for the following reason. There are only two symmetric monoidal closed structures on Cat, up to isomorphism [8]. One is the the cartesian product. The other is the 'funny tensor product' [25]. A monoid in Cat with its cartesian product is a strict monoidal category, but a monoid in Cat with its funny tensor product is a strict premonoidal category. The **funny tensor product** $\mathsf{C}\square\mathsf{D}$ of categories C and D is defined as the following pushout in Cat:

$$
\begin{array}{ccc}
\mathsf{C}_0 \times \mathsf{D}_0 & \xrightarrow{1 \times j} & \mathsf{C}_0 \times \mathsf{D} \\
{\scriptstyle i \times 1}\downarrow & & \downarrow \\
\mathsf{C} \times \mathsf{D}_0 & \longrightarrow & \mathsf{C}\square\mathsf{D}
\end{array}
$$

Here C_0 is the subcategory of C consisting of all the objects and only identity morphisms, $i \colon \mathsf{C}_0 \to \mathsf{C}$ is the inclusion, and similarly for $j \colon \mathsf{D}_0 \to \mathsf{D}$. Thus, given morphisms $f \colon x \to y$ in C and $f' \colon x' \to y'$ in C, the category $C\square D$ in contains a square of the form

$$
\begin{array}{ccc}
x\square x' & \xrightarrow{1\square f'} & x\square y' \\
{\scriptstyle f\square 1}\downarrow & & \downarrow{\scriptstyle f\square 1} \\
x'\square y & \xrightarrow[1\square f']{} & x'\square y',
\end{array}
$$

but in general this square does not commute, unlike the corresponding square in $\mathsf{C} \times \mathsf{D}$.

Definition 14. *A* **strict premonoidal category** *is a category* C *equipped with a functor* $\boxtimes \colon \mathsf{C}\square\mathsf{C} \to \mathsf{C}$ *that obeys the associative law and an object* $I \in \mathsf{C}$ *that serves as a left and right unit for* $\boxtimes$.

Given two morphisms $f \colon x \to y$, $f' \colon x' \to y'$ in a strict premonoidal category C we obtain a square

$$
\begin{array}{ccc}
x \boxtimes x' & \xrightarrow{1\boxtimes f'} & x \boxtimes y' \\
{\scriptstyle f\boxtimes 1}\downarrow & & \downarrow{\scriptstyle f\boxtimes 1} \\
x' \boxtimes y & \xrightarrow[1\boxtimes f']{} & x' \boxtimes y',
\end{array}
$$

but this square may not commute. There are thus two candidates for a morphism from $x \boxtimes x'$ to $y \boxtimes y'$. When these always agree, we can make C monoidal by setting $f \boxtimes f'$ equal to either (and thus both) of these candidates. We shall give FP_i a strict premonoidal structure where these two candidates do not agree: one is $f \otimes_i f'$ while the other is $f \,_i\otimes f'$. This explains the meaning of these two failed attempts to give FP_i a monoidal structure.

Thanks to the description of $\mathsf{C}\square\mathsf{C}$ as a pushout, to know the tensor product $\boxtimes$ in a strict premonoidal category C it suffices to know $x \boxtimes y$, $x \boxtimes f$ and $f \boxtimes y$ for all objects x, y and morphisms f of C. (Here we find it useful to write $x \boxtimes f$ for $1_x \boxtimes f$ and $f \boxtimes y$ for $f \boxtimes 1_y$.) In the case at hand, we define

$$\boxtimes_i \colon FP_i \square FP_i \to FP_i$$

on objects by setting

$$(i + a) \boxtimes_i (i + a') = i + a + a'$$

for all $a, a' \in \mathbb{N}[S - C]$, while for morphisms

$$f \colon i + a \to i + b$$
$$f' \colon i + a' \to i + b'$$

we set

$$a \boxtimes f' = f' + 1_a, \qquad f \boxtimes a' = f + 1_{a'}.$$

Proposition 15. *The tensor product $\boxtimes_i$ makes FP_i into a strict premonoidal category.*

Proof. This can be checked directly, but this is also a special case of a construction in Power and Robinson's paper on premonoidal categories [19, Ex. 3.4]. They describe a construction, sometimes called 'linear state passing' [17], that takes any object i in any symmetric monoidal category C and yields a premonoidal category C_i where objects are of the form $i \otimes c$ for $c \in C$ and morphisms are morphisms in C of the form $f \colon i \otimes c \to i \otimes c'$. We are considering the special case where $C = FP$, and because FP is commutative monoidal the resulting premonoidal category is strict: all the coherence isomorphisms are identities. ∎

Finally, we show that the tensor products $\boxtimes_i$ on the categories FP_i let us lift our network model G from Cat to the category of strict premonoidal categories.

Definition 16. *Let $\mathsf{PreMonCat}$ be the category of strict premonoidal categories and* **strict premonoidal functors***, meaning functors between strict premonoidal categories that strictly preserve the tensor product. Let $U \colon \mathsf{PreMonCat} \to \mathsf{Cat}$ denote the forgetful functor which sends a strict premonoidal category to its underlying category.*

Theorem 17. *The network model $G \colon \mathsf{S}(C) \to \mathsf{Cat}$ lifts to a functor $\hat{G} \colon \mathsf{S}(C) \to \mathsf{PreMonCat}$:*

$$
\begin{array}{ccc}
 & \mathsf{PreMonCat} & \\
 \scriptstyle\hat{G}\; \nearrow & & \downarrow \scriptstyle U \\
 \mathsf{S}(C) \xrightarrow[\;\;\;G\;\;\;]{} & \mathsf{Cat} &
\end{array}
$$

where $\hat{G}(x) = FP_{p(x)}$ with the strict premonoidal structure described in Prop. 15.

Proof. Since G sends each morphism in $\mathsf{S}(C)$ to an identity functor, so must $\hat{G}$. ∎

5 Conclusions

A couple of mathematical questions arise naturally from this work. First, is there a string diagram calculus for premonoidal categories, like that for monoidal categories but omitting the interchange law? This is hinted at in the work of Jeffrey [10], but ideally there would be a theorem justifying the use of such string diagrams just as Joyal and Street [12] justified the use of planar progressive string diagrams for monoidal categories. We could then omit the red lines in our string diagrams and treat the resulting diagrams as describing morphisms in premonoidal categories.

Accepted in Compositionality on 2019-11-26. Click on the title to verify.

13

Second, is there a monoidal functor from $(\mathsf{Cat}, \square)$ to $(\mathsf{Cat}, \times)$? If so, we could turn a strict premonoidal category into a strict monoidal category just by applying this functor. Ideally this would impose the interchange law on the tensor product, forcing all squares of the form

$$
\begin{array}{ccc}
x \boxtimes x' & \xrightarrow{1 \boxtimes f'} & x \boxtimes y' \\
{\scriptstyle f \boxtimes 1} \downarrow & & \downarrow {\scriptstyle f \boxtimes 1} \\
x' \boxtimes y & \xrightarrow[1 \boxtimes f']{} & x' \boxtimes y'
\end{array}
$$

to commute. This could be useful in applications where we do not care which of two processes uses a catalyst first.

Acknowledgments

References

[1] J. C. Baez and J. Biamonte, *Quantum Techniques in Stochastic Mechanics*, World Scientific, Singapore, 2018. Available as arXiv:1209.3632, DOI: https://doi.org/10.1142/10623.

[2] J. C. Baez, J. Foley, J. Moeller and B. Pollard, Network models. Available as arXiv:1711.00037.

[3] J. C. Baez and J. Master, Open Petri nets. Available as arXiv:1808.05415.

[4] J. C. Baez and B. S. Pollard, A compositional framework for reaction networks, *Rev. Math. Phys.* **29** (2017), 1750028. Available as arXiv:1704.02051, DOI: https://doi.org/10.1142/S0129055X17500283.

[5] J. C. Baez and M. Stay, in *New Structures for Physics*, ed. Bob Coecke, Lecture Notes in Physics vol. 813, Springer, Berlin, 2011, pp. 95–172. Available as arXiv:0903.0340.

[6] P. Degano, J. Meseguer and U. Montanari, Axiomatizing net computations and processes, in *Logic in Computer Science, 1989*, IEEE, New Jersey, pp. 175–185. Available at https://www.computer.org/csdl/proceedings/lics/1989/1954/00/00039172.pdf.

[7] U. Engberg and G. Winskel, Petri nets as models of linear logic, in *Colloquium on Trees in Algebra and Programming*, Springer, Berlin, 1990, pp. 147–161.

[8] F. Foltz, G. M. Kelly, and C. Lair, Algebraic categories with few monoidal biclosed structures or none, *JPAA* **17** (1980), 171–177, DOI: https://doi.org/10.1016/0022-4049(80)90082-1.

[9] C. Girault and R. Valk, *Petri Nets for Systems Engineering: a Guide to Modeling, Verification, and Applications*, Springer, Berlin, 2013.

[10] A. Jeffrey, Premonoidal categories and a graphical view of programs, preprint, 1997. Available as http://fpl.cs.depaul.edu/ajeffrey/papers/premonA.pdf, DOI: https://doi.org/10.1016/S1571-0661(05)80688-7.

[11] R. J. van Glabbeek and G. D. Plotkin, Configuration structures, event structures and Petri nets, *Theoretical Computer Science* **410** (2009), 4111–4159. Available as arXiv:0912.4023, DOI: https://doi.org/10.1016/j.tcs.2009.06.014.

[12] A. Joyal and R. Street, The geometry of tensor calculus, I, *Adv. Math.* **88** (1991), 55–112, DOI: https://doi.org/10.1016/0001-8708(91)90003-P.

[13] J. Master, Generalized Petri nets. Available as arXiv:1904.09091

[14] J. Meseguer and U. Montanari, Petri nets are monoids, *Information and Computation* **88** (1990), 105–155, DOI: https://doi.org/10.1016/0890-5401(90)90013-8.

[15] J. Moeller and C. Vasilakopoulou, Monoidal Grothendieck construction. Available as arXiv:1809.00727.

[16] J. Moeller, Noncommutative network models. Available as arXiv:1804.07402.

[17] R. E. Møgelberg and S. Staton, Linear usage of state, *Logical Methods in Computer Science* **10** (2014), lmcs:743. Also available as arXiv:1403.1477.

[18] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice–Hall, New Jersey, 1981.

[19] J. Power and E. Robinson, Premonoidal categories and notions of computation, *Math. Str. Comp. Sci.*, **7** (1997), 453–468, DOI: https://doi.org/10.1017/S0960129597002375.

[20] V. Sassone, Strong concatenable processes: an approach to the category of Petri net computations, *BRICS Report Series*, Dept. of Computer Science, U. Aarhus, 1994. Available at https://tidsskrift.dk/brics/article/view/21610/19059, DOI: https://doi.org/10.7146/brics.v1i33.21610.

[21] V. Sassone, On the category of Petri net computations, in *Colloquium on Trees in Algebra and Programming*, Springer, Berlin, 1995. Available at https://eprints.soton.ac.uk/261951/1/strong-conf.pdf, DOI: https://doi.org/10.7146/brics.v1i33.21610.

[22] V. Sassone, An axiomatization of the algebra of Petri net concatenable processes, in *Theoretical Computer Science* **170** (1996), 277–296. Available at https://eprints.soton.ac.uk/261820/1/P-of-N-Off.pdf, DOI: https://doi.org/10.1016/S0304-3975(96)80709-2.

[23] V. Sassone and P. Sobociński, A congruence for Petri nets, *Electronic Notes in Theoretical Computer Science* **127** (2005), 107–120. Available at https://eprints.soton.ac.uk/262302/1/petriCongPNGToff.pdf, DOI: https://doi.org/10.1016/j.entcs.2005.02.008.

[24] P. Selinger, A survey of graphical languages for monoidal categories, in *New Structures for Physics*, ed. B. Coecke, Lecture Notes in Physics vol. 813, Springer, Berlin, 2011, pp. 289–355. Available as arXiv:0908.3347, DOI: https://doi.org/10.1007/978-3-642-12821-9_4.

[25] M. Weber, Free products of higher operad algebras. Available as arXiv:0909.4722.

Accepted in Compositionality on 2019-11-26. Click on the title to verify.

15