NETWORK MODELS FROM PETRI NETS WITH CATALYSTS

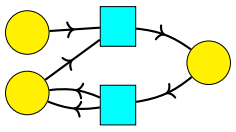
JOHN C. BAEZ[1,2], JOHN FOLEY[3], AND JOE MOELLER[1]

Abstract. 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 non-symmetric monoidal structures that describe doing one process and then another while reusing the catalysts.

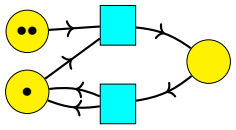
1. Introduction

Petri nets are a widely studied formalism for describing collections of entities of different types, and how they turn into other entities [6, 12]. Network models are a formalism for designing and tasking networks of agents [2, 11]. 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:

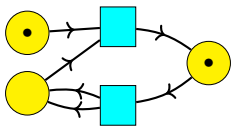


In applications to chemistry, places are also called 'species'. When we run a Petri net, we start by placing a finite number of 'tokens' in each place:

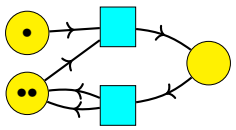


[1]Department of Mathematics, University of California, Riverside CA, 92521, USA
[2]Centre for Quantum Technologies, National University of Singapore, 117543, Singapore
[3]Metron, Inc., 1818 Library St., Suite 600, Reston, VA 20190, USA
E-mail addresses: baez@math.ucr.edu, foley@metsci.com, moeller@math.ucr.edu.

This is called a 'marking'. Then we repeatedly change the marking using the transitions. For example, the above marking can change to this:



and then this:



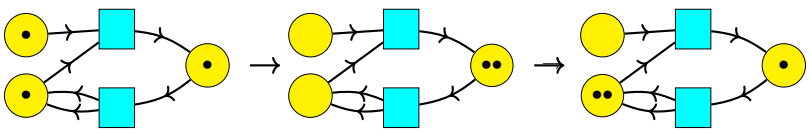
Thus, the places represent different *types* of entity, and the transitions describe ways that one collection of entities of specified types can turn into another such collection.

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.

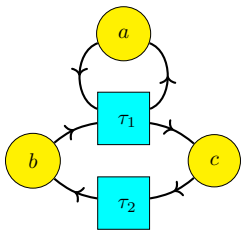
In many examples considered so far, $G(x)$ is a category whose morphisms are graphs whose nodes are agents of types $c_1, \ldots, c_n$. Composing these morphisms corresponds to 'overlaying' graphs. Network models of this sort let us design networks where the nodes are agents and the edges are communication 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 [11]. 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, in the following Petri net, species a is a catalyst:



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 [10]. 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 Theorem 14 we show that each subcategory FP_i also has its own tensor product, which describes doing one process *after* another while reusing catalysts. Finally, in Theorem 16 we show that these monoidal structures define a lift of the functor $G\colon \mathsf{S}(C) \to \mathsf{Cat}$ to a functor $\hat{G}\colon \mathsf{S}(C) \to \mathsf{MonCat}_{\mathrm{str}}$, where $\mathsf{MonCat}_{\mathrm{str}}$ is the category of strict monoidal 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 Meseguer and Montanari's 1990 paper [9]. It continues to this day, because the issues involved are surprisingly subtle [5, 13, 14, 15, 16]. 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 symmetry:

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

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 symmetry 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.*

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:*

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

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 internal to* Cat. *Let* CMC *denote the category of commutative monoid objects internal to* Cat.

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 mod out 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

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

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 $F\colon$ Petri $\to$ CMC defined as above.*

Proof. This is Prop. 10 of [3], where this functor is called $\hat{F}$. ∎

To dispel any possible misimpression, this functor F is *not* a left adjoint of the forgetful functor $U\colon$ CMC $\to$ Petri for which the species of UC are the objects of $C \in$ CMC and the transitions of UC are the morphisms of C. However, we can make F into a left adjoint by replacing CMC with a slightly more complicated category. There are at least two ways to do this [3, 8].

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.

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. Thus, $\mathsf{S}(C)$ implements the 'individual token philosophy' on catalysts, in which permuting tokens of the same catalyst is regarded as having a nontrivial effect [7]. By contrast, $\mathbb{N}[C]$ implements the 'collective token philosophy', where all that matters is the *number* of tokens of each catalyst, and permuting them has no effect. The interplay between these two philosophies is crucial in what follows.

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

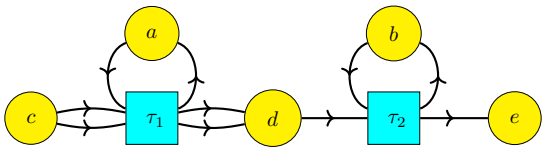
$$1 \to FP_0$$

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).*

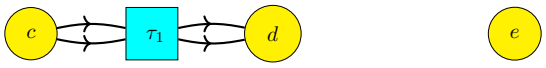
Example 11. The following Petri net P has species $S = \{a, b, c, d, e\}$ and transitions $T = \{\tau_1, \tau_2\}$:



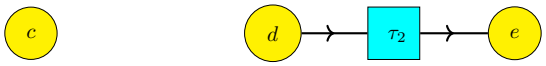
Species a and b are catalysts and the rest are not. We thus can take $C = \{a, b\}$ and obtain a Petri net with catalysts (P, C), which in turn gives a Petri network model $G\colon \mathsf{S}(C) \to \mathsf{Cat}$.

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:

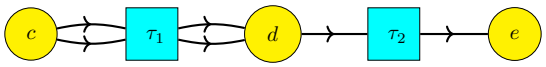


That is, people can go from the base to the shore in pairs, but they cannot go to the island. Similarly, the object b describes a situation with one boat present but no jeeps, and the category $G(b)$ is isomorphic to FY, where Y is this Petri net:



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

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

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

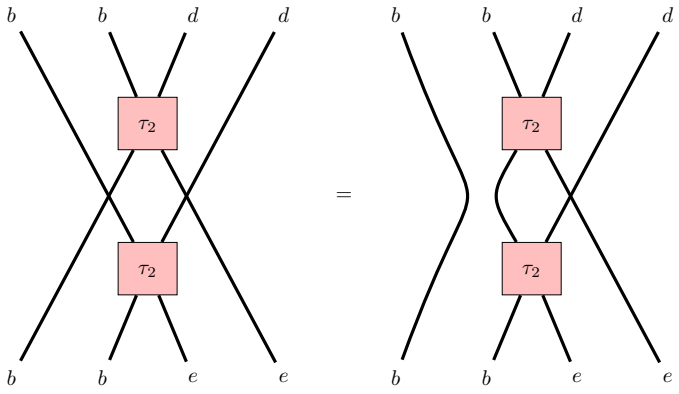
$$- + b \colon FP_b \to FP_{b+b}$$

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.

These morphisms can be drawn as string diagrams:



These morphisms are equal because they differ only by the presence of an extra braiding $b + b \to b + b$ in the left hand side, and this braiding is the identity.

The above example illustrates an important point: the commutative monoidal category FP implements the collective token philosophy. Next we construct a symmetric monoidal category $\int G$ that implements the individual token philosophy

for catalyst tokens: in $\int G$, permuting such tokens has a nontrivial effect. One reason for wanting this is that in applications the catalyst tokens 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.

To create a category that gives the catalyst tokens a nontrivial braiding, we use the symmetric monoidal Grothendieck construction [10]. Given any symmetric monoidal category X and any lax symmetric monoidal functor $F\colon X \to \mathsf{Cat}$, 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 $G\colon \mathsf{S}(C) \to \mathsf{Cat}$, 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 $\sigma\colon x \to x'$ is a morphism in $\mathsf{S}(C)$ and $f\colon a \to a'$ 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 $G\colon \mathsf{S}(C) \to \mathsf{Cat}$ 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 precisely that the former category keeps track of processes where catalyst tokens are permuted, while the latter category treats them as identity morphisms. 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

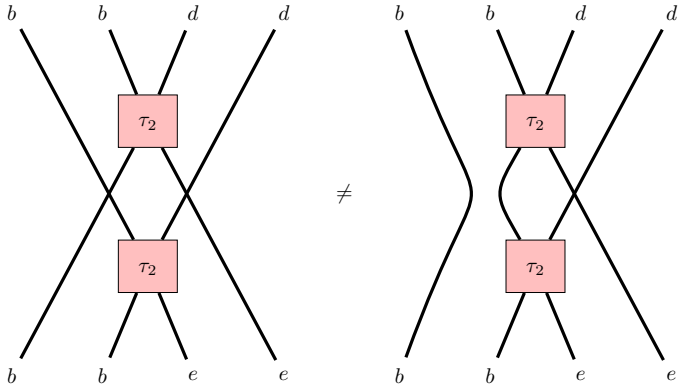
$$\int G \to FP$$

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 [10].

Example 13. Let (P, C) be the Petri net with catalysts in Example 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. As we saw in Example 11, the images of these morphisms under the above functor $\int G \to FP$ are equal.

4. Monoidal Structure

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. Despite this, a monoidal structure is definable on each subcategory FP_i using the structures present!

Given two morphisms in FP_i, say f and f', 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, and use this to define their tensor product, which we call $f \otimes_i f'$.

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. This is essentially how we shall define a monoidal structure on each category FP_i.

Fix some amount of catalysts $i \in \mathbb{N}[C]$. Objects of FP_i are of the form $i + a$ with $a \in \mathbb{N}[S - C]$. We define a tensor product of objects in FP_i by the following recipe:

$$(i + a) \otimes_i (i + a') = i + a + a'.$$

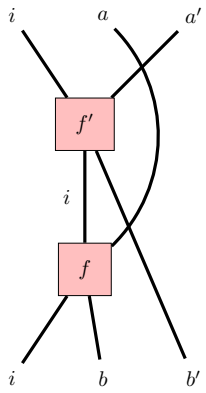
The unit object for $\otimes$ is therefore $i + 0$, or simply i. For morphisms

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

we define

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

There is an arbitrary choice in this definition: namely, the choice to do f' first. This is perhaps clearer if we draw the morphism $f \otimes f'$ as a string diagram in FP.



Here the crossings are identity morphisms, because FP is a commutative monoidal category.

In short: to do $f \otimes_i f'$, we do first f' and then f by reusing i. In FP, the monoidal structure $+$ captures the idea that if you have enough resources to do two processes, you can do them in parallel. In FP_i, the monoidal structure $\otimes_i$ captures the idea that if you have enough catalysts to do either of two processes, and enough non-catalysts to do both, you can do first one and then the other by reusing the catalysts.

Theorem 14. *The above tensor product makes FP_i into a strict monoidal category.*

Proof. On objects $i+a, i+b, i+c \in FP_i$, the associativity of the tensor product is evident:

$$((i+a) \otimes_i (i+b)) \otimes_i (i+c) \;=\; i+a+b+c \;=\; (i+a) \otimes_i ((i+b) \otimes_i (i+c)).$$

For morphisms

$$\begin{aligned} f &\colon i+a \to i+b \\ f' &\colon i+a' \to i+b' \\ f'' &\colon i+a'' \to i+b'' \end{aligned}$$

we check associativity as follows:

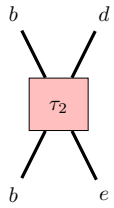
$$\begin{aligned} (f \otimes_i f') \otimes_i f'' &= ((f + 1_{b'}) \circ (1_a + f')) \otimes_i f'' \\ &= (((f + 1_{b'}) \circ (1_a + f')) + 1_{b''}) \circ (1_{a+a'} + f'') \\ &= (f + 1_{b'+b''}) \circ (f' + 1_{a+b''}) \circ (f'' + 1_{a+a'}) \\ &= (f + 1_{b'+b''}) \circ (1_a + ((f' + 1_{b''}) \circ (1_{a'} + f''))) \\ &= f \otimes_i ((f' + 1_{b''}) \circ (1_{a'} + f'')) \\ &= f \otimes_i (f' \otimes_i f''). \end{aligned}$$

The unit law for the tensor product in FP_i also holds strictly, since for any object $i+a \in FP_i$ we have

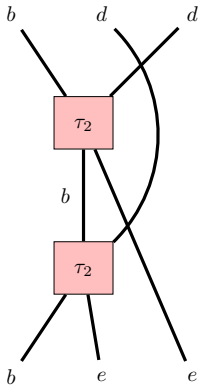
$$(i+0) \otimes_i (i+a) = i+a = (i+a) \otimes_i (i+0).$$

■

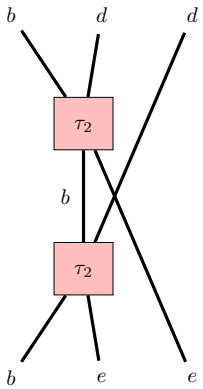
Example 15. Returning to Example 11, let FP_b be the category where morphisms are processes that can be carried out with just one boat. An example is the morphism τ_2, which describes a boat carrying one person from the shore to the island. We can draw this morphism using a string diagram in the commutative monoidal category FP, of which FP_b is a subcategory:



The morphism $\tau_2 \otimes_b \tau_2$ describes the boat carrying first one person and then another from the shore to the island. As a string diagram in FP, this morphism looks like this:



But because the braiding in FP_b is trivial, $\tau_2 \otimes_b \tau_2$ is also equal to this:



In short, the morphism $\tau_2 \otimes_b \tau_2$ does not depend on which person the boat transports first.

A curious feature of this monoidal category FP_i is that there is in general no way to extend it to a symmetric, or even braided, monoidal category. To see this, note that the only isomorphisms in FP are identities: this follows from how the

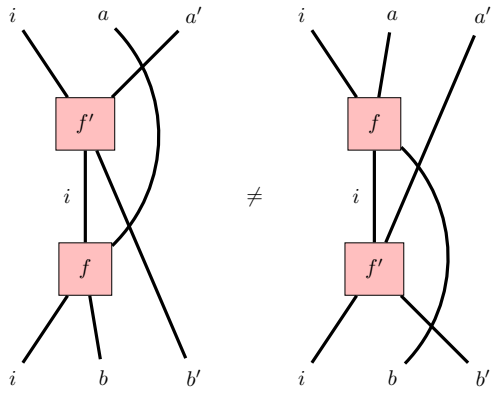
morphisms in FP are recursively generated, with the only equations imposed being the laws of a commutative monoidal category. Thus, the only option for a braiding on FP_i, say

$$B_{i+a,i+b} \colon (i+a) \otimes_i (i+b) \to (i+b) \otimes_i (i+a),$$

is the identity. Since the tensor product of objects in FP_i is commutative, this is a plausible candidate. However, given morphisms

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

we in general do not have $f \otimes f' = f' \otimes f$, as would be required by naturality of the braiding if the braiding were the identity. This can be checked in examples with the help of string diagrams:



Finally, we show that these monoidal structures on the categories FP_i let us lift our network model G so that it takes values in $\mathsf{MonCat}_{\mathrm{str}}$, the category of strict monoidal categories and strict monoidal functors. Let $U \colon \mathsf{MonCat}_{\mathrm{str}} \to \mathsf{Cat}$ denote the forgetful functor which sends a strict monoidal category to its underlying category.

Theorem 16. *The network model* $G \colon \mathsf{S}(C) \to \mathsf{Cat}$ *lifts to a functor* $\hat{G} \colon \mathsf{S}(C) \to \mathsf{MonCat}_{\mathrm{str}}$*:*

$$\begin{array}{ccc} & & \mathsf{MonCat}_{\mathrm{str}} \\ & \overset{\hat{G}}{\nearrow} & \big\downarrow U \\ \mathsf{S}(C) & \xrightarrow[G]{} & \mathsf{Cat} \end{array}$$

where $\hat{G}(x) = FP_{p(x)}$ *with the monoidal structure described in Theorem 14.*

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

Acknowledgements. This work was supported by the DARPA Complex Adaptive System Composition and Design Environment (CASCADE) project. We thank Chris Boner, Tony Falcone, Marisa Hughes, Joel Kurucar, Jade Master, Tom Mifflin, John Paschkewitz, Thy Tran and Didier Vergamini for helpful discussions. JB also thanks the Centre for Quantum Technologies, where some of this work was done.

References

[1] J. C. Baez and J. Biamonte, *Quantum Techniques in Stochastic Mechanics*, World Scientific, Singapore, 2018. Available as arXiv:1209.3632. (Referred to on page 6.)

[2] J. C. Baez, J. Foley, J. Moeller and B. Pollard, Network models. Available as arXiv:1711.00037. (Referred to on page 1, 2, 7, 8, 10.)

[3] J. C. Baez and J. Master, Open Petri nets. Available as arXiv:1808.05415. (Referred to on page 4, 5.)

[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. (Referred to on page 6.)

[5] 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. (Referred to on page 4.)

[6] C. Girault and R. Valk, *Petri Nets for Systems Engineering: a Guide to Modeling, Verification, and Applications*, Springer, Berlin, 2013. (Referred to on page 1.)

[7] 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. (Referred to on page 7.)

[8] J. Master, Generalized Petri nets. In preparation. (Referred to on page 5.)

[9] J. Meseguer and U. Montanari, Petri nets are monoids, *Information and Computation* **88** (1990), 105–155. (Referred to on page 4.)

[10] J. Moeller and C. Vasilakopoulou, Monoidal Grothendieck construction. Available as arXiv:1809.00727. (Referred to on page 3, 10.)

[11] J. Moeller, Noncommutative network models. Available as arXiv:1804.07402. (Referred to on page 1, 2.)

[12] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice–Hall, New Jersey, 1981. (Referred to on page 1.)

[13] 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. (Referred to on page 4.)

[14] 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. (Referred to on page 4.)

[15] 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. (Referred to on page 4.)

[16] 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. (Referred to on page 4.)