Network Models from Petri Nets with Catalysts

John C. Baez
University of California, Riverside
California, USA
Centre for Quantum Technologies,
National University of Singapore,
117543, Singapore
baez@math.ucr.edu

John Foley
Metron Scientific Solutions, Inc.,
1818 Library St., Suite 600,
Reston, VA 20190, USA
foley@metsci.com

Joe Moeller
University of California, Riverside
California, USA
moeller@math.ucr.edu

Petri nets are a widely studied formalism for describing collections of entities of different types, and how they turn into other entities [3, 6]. *Network models* are a formalism for designing and tasking networks of agents [1, 4]. Here we combine the two. This is worthwhile because while both formalisms involve networks, they serve a different function, and are in some sense complementary.

For a set X, let $\mathbb{N}[X]$ denote the set of finite formal sums of elements of X. A **Petri net** is given by the following data: a set S of **places**, a set T of **transitions**, a **source** function $s\colon T \to \mathbb{N}[S]$, a **target** function $t\colon T \to \mathbb{N}[S]$. In applications to chemistry, places are also called 'species'. 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. 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, and Cat is considered with its cartesian monoidal structure. 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 [1]. Subsequently we introduced a more general noncommutative overlay operation [4]. 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.

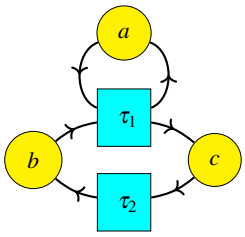
This idea meshes well with Petri net theory, because any Petri net P determines a symmetric monoidal category FP whose morphisms are processes that can be carried out using this Petri net [2]. More precisely, the objects in FP are markings of P, and the morphisms are sequences of ways to change these markings using transitions.

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

Submitted to:
QPL 2019

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.

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 full subcategory of FP where the objects are markings that have this specified amount of each catalyst.

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

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

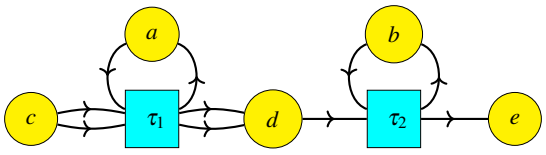
given by the tensor product in FP, and the map

$$\phi\colon 1 \to G(0)$$

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 $0 \in G(0)$

We use the following simple example to demonstrate the ideas. More elaborate examples are not hard to imagine.

Example. 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.

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 [5]. 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. We make this precise by more concretely describing $\int G$ as a symmetric monoidal category, and clarifying its relation to FP.

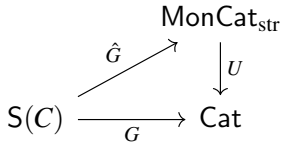
Theorem. *Let $G\colon \mathsf{S}(C) \to \mathsf{Cat}$ be the Petri network model associated to the Petri net with catalysts (P,C). The symmetric monoidal category $\int G$ is monoidally equivalent to the full subcategory of $\mathsf{S}(C) \times FP$ whose objects are those of the form (x,a) with $x \in \mathsf{S}(C)$ and $a \in G(x)$.*

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 $G(x)$ where $x \in \mathsf{S}(C)$ specifies the amount of each catalyst present. The tensor product on FP has the property that tensoring an object in $G(x)$ with one in $G(y)$ gives an object in $G(x \otimes y)$, and similarly for morphisms. However, we show that each subcategory $G(x)$ also has its own tensor product, denoted $\otimes_x$, which describes doing one process *after* another while reusing catalysts. Briefly, to do $f \otimes_x f'$, first do f' and then f by reusing catalysts from the collection x.

Theorem. *The tensor product $\otimes_x$ makes $G(x)$ into a strict monoidal category.*

Finally, 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.

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



where $\hat{G}(x)$ is the category $G(x)$ with the $\otimes_x$ monoidal structure, where U is the forgetful functor.

A preprint can be found on the first author's website at http://math.ucr.edu/home/baez/catalysts.pdf.

References

[1] John C. Baez, John Foley, Joseph Moeller & Blake Pollard (2017): *Network Models*. ArXiv:1711.00037 [math.CT].

[2] John C. Baez & Jade Master (2018): *Open Petri Construction*. ArXiv:1808.05415 [math.CT].

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

[4] Joe Moeller (2018): *Noncommutative Network Models*. ArXiv:1804.07402 [math.CT].

[5] Joe Moeller & Christina Vasilakopoulou (2019): *Monoidal Grothendieck Construction*. ArXiv:1809.00727 [math.CT].

[6] James Peterson (1981): *Petri Net Theory and the Modeling of Systems*. Prentice–Hall, New Jersey.