Theory and Applications of Categories, Vol. 35, No. 20, 2020, pp. 700–744.

NETWORK MODELS

JOHN C. BAEZ, JOHN FOLEY, JOE MOELLER, AND BLAKE S. POLLARD

Abstract. Networks can be combined in various ways, such as overlaying one on top of another or setting two side by side. We introduce 'network models' to encode these ways of combining networks. Different network models describe different kinds of networks. We show that each network model gives rise to an operad, whose operations are ways of assembling a network of the given kind from smaller parts. Such operads, and their algebras, can serve as tools for designing networks. Technically, a network model is a lax symmetric monoidal functor from the free symmetric monoidal category on some set to Cat, and the construction of the corresponding operad proceeds via a symmetric monoidal version of the Grothendieck construction.

1. Introduction

In this paper we study operads suited for designing networks. These could be networks where the vertices represent fixed or moving agents and the edges represent communication channels. More generally, they could be networks where the vertices represent entities of various types, and the edges represent relationships between these entities, e.g. that one agent is committed to take some action involving the other. This paper arose from an example where the vertices represent planes, boats and drones involved in a search and rescue mission in the Caribbean [2, 3]. However, even for this one example, we want a flexible formalism that can handle networks of many kinds, described at a level of detail that the user is free to adjust.

To achieve this flexibility, we introduce a general concept of 'network model'. Simply put, a network model is a *kind* of network. Any network model gives an operad whose operations are ways to build larger networks of this kind by gluing smaller ones. This operad has a 'canonical' algebra where the operations act to assemble networks of the given kind. But it also has other algebras, where it acts to assemble networks of this kind *equipped with extra structure and properties*. This flexibility is important in applications.

What exactly is a 'kind of network'? At the crudest level, we can model networks as simple graphs. If the vertices are agents of some sort and the edges represent communication channels, this means we allow at most one channel between any pair of agents. However, simple graphs are too restrictive for many applications. If we allow multiple communication channels between a pair of agents, we should replace simple graphs with

Received by the editors 2018-01-12 and, in final form, 2020-05-24.
Transmitted by Giuseppe Rosolini. Published on 2020-05-27.
2020 Mathematics Subject Classification: 18D30, 18M05, 18M35, 18M60, 18M80, 68M10, 90B18.
Key words and phrases: Grothendieck construction, graph, monoidal category, network, operad.

'multigraphs'. Alternatively, we may wish to allow directed channels, where the sender and receiver have different capabilities: for example, signals may only be able to flow in one direction. This requires replacing simple graphs with 'directed graphs'. To combine these features we could use 'directed multigraphs'. It is also important to consider graphs with colored vertices, to specify different types of agents, and colored edges, to specify different types of channels. This leads us to 'colored directed multigraphs'. All these are examples of what we mean by a 'kind of network'. Even more complicated kinds, such as hypergraphs or Petri nets, are likely to become important as we proceed. Thus, instead of separately studying all these kinds of networks, we introduce a unified notion that subsumes all these variants: a 'network model'. Namely, given a set C of 'vertex colors', a **network model** F is a lax symmetric monoidal functor $F\colon \mathsf{S}(C) \to \mathsf{Cat}$, where $\mathsf{S}(C)$ is the free strict symmetric monoidal category on C and Cat is the category of small categories, considered with its cartesian monoidal structure. Unpacking this definition takes a little work. It simplifies in the special case where F takes values in Mon, the category of monoids. It simplifies further when C is a singleton, since then $\mathsf{S}(C)$ is the groupoid S, where objects are natural numbers and morphisms from m to n are bijections $\sigma\colon \{1, \ldots, m\} \to \{1, \ldots, n\}$. If we impose both these simplifying assumptions, we have what we call a **one-colored network model**: a lax symmetric monoidal functor $F\colon \mathsf{S} \to \mathsf{Mon}$. As we shall see, the network model of simple graphs is a one-colored network model, and so are many other motivating examples.

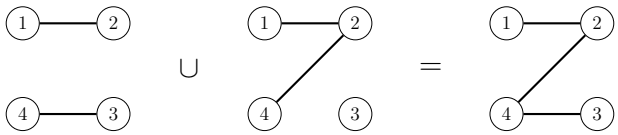
Joyal began an extensive study of functors $F\colon \mathsf{S} \to \mathsf{Set}$, which are now commonly called 'species' [5, 14, 15]. Any type of extra structure that can be placed on finite sets and transported along bijections defines a species if we take $F(n)$ to be the set of structures that can be placed on the set $\{1, \ldots, n\}$. From this perspective, a one-colored network model is a species with some extra operations.

This perspective is helpful for understanding what a one-colored network model $F\colon \mathsf{S} \to \mathsf{Mon}$ is actually like. If we call elements of $F(n)$ 'networks with n vertices', then:

1. Since $F(n)$ is a monoid, we can **overlay** two networks with the same number of vertices and get a new one. We denote this operation by

$$\cup\colon F(n) \times F(n) \to F(n).$$

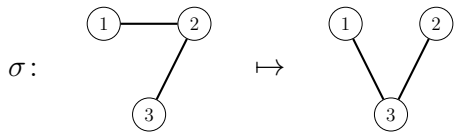
For example:



2. Since F is a functor, the group S_n acts on the monoid $F(n)$. Thus, for each $\sigma \in S_n$, we have a monoid automorphism that we call

$$\sigma\colon F(n) \to F(n).$$

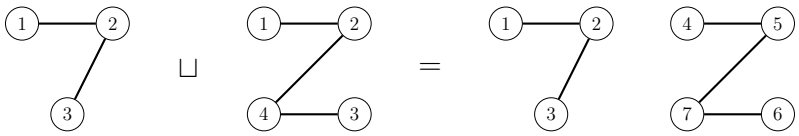
For example, if $\sigma = (2\,3) \in S_3$, then



3. Since F is lax monoidal, we have an operation

$$\sqcup \colon F(m) \times F(n) \to F(m+n)$$

We call this operation the **disjoint union** of networks. For example:



The first two operations are present whenever we have a functor $F \colon \mathsf{S} \to \mathsf{Mon}$. The last two are present whenever we have a lax symmetric monoidal functor $F \colon \mathsf{S} \to \mathsf{Set}$. When F is a one-colored network model we have all three—and unpacking the definition further, we see that they obey some equations, which we list in Theorem 2.3. For example, the 'interchange law'

$$(g \cup g') \sqcup (h \cup h') = (g \sqcup h) \cup (g' \sqcup h')$$

holds whenever $g, g' \in F(m)$ and $h, h' \in F(n)$.

In Section 2 we study one-colored network models more formally, and give many examples. In Section 3 we describe a systematic procedure for getting one-colored network models from monoids. In Section 4 we study general network models and give examples of these. In Section 5 we describe a category NetMod of network models, and show that the procedure for getting network models from monoids is functorial. We also make NetMod into a symmetric monoidal category, and give examples of how to build new networks models by tensoring old ones.

Our main result is that any network model gives a typed operad, also known as a 'colored operad' or 'symmetric multicategory' [23]. A typed operad describes ways of sticking together things of various types to get new things of various types. An algebra of the operad gives a particular specification of these things and the results of sticking them together. A bit more precisely, a typed operad O has:

- a set T of **types**,
- sets of **operations** $O(t_1, ..., t_n; t)$ where $t_i, t \in T$,
- ways to **compose** any operation

$$f \in O(t_1, \ldots, t_n; t)$$

with operations

$$g_i \in O(t_{i1}, \ldots, t_{ik_i}; t_i) \qquad (1 \le i \le n)$$

to obtain an operation

$$f \circ (g_1, \ldots, g_n) \in O(t_{1i}, \ldots, t_{1k_1}, \ldots, t_{n1}, \ldots t_{nk_n}; t),$$

- and ways to permute the arguments of operations,

which obey some rules [23]. An algebra A of O specifies a set $A(t)$ for each type $t \in T$ such that the operations of O act on these sets. Thus, it has:

- for each type $t \in T$, a set $A(t)$ of **things** of type t,
- ways to **apply** any operation

$$f \in O(t_1, \ldots, t_n; t)$$

to things

$$a_i \in A(t_i) \qquad (1 \le i \le n)$$

to obtain a thing

$$\alpha(f)(a_1, \ldots, a_n) \in A(t).$$

Again, we demand that some rules hold [23].

In Thm. 7.4 we describe the typed operad O_F arising from a one-colored network model F. The set of types is $\mathbb{N}$, since we can think of 'network with n vertices' as a type. The sets of operations are given as follows:

$$\mathrm{O}_F(n_1, \ldots, n_k; n) = \begin{cases} S_n \times F(n) & \text{if } n_1 + \cdots + n_k = n \\ \emptyset & \text{otherwise.} \end{cases}$$

The key idea here is that we can overlay a network in $F(n)$ on the disjoint union of networks with $n_1, \ldots, n_k$ vertices and get a new network with n vertices as long as $n_1 + \cdots n_k = n$. We can also permute the vertices; this accounts for the group S_n. But the most important fact is that *networks serve as operations to assemble networks*, thanks to our ability to overlay them.

Using this fact, we show in Ex. 8.1 that the operad O_F has a canonical algebra A_F whose elements are simply networks of the kind described by F:

$$A_F(n) = F(n).$$

In this algebra any operation

$$(\sigma, g) \in \mathrm{O}_F(n_1, \ldots, n_k; n) = S_n \times F(n)$$

acts on a k-tuple of networks

$$h_i \in A_F(n_i) = F(n_i) \qquad (1 \leq i \leq k)$$

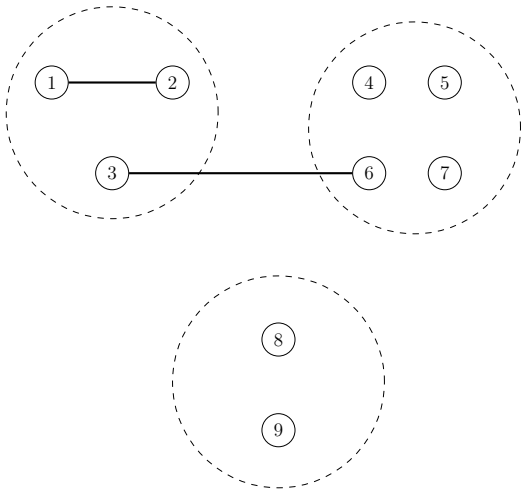
to give the network

$$\alpha(\sigma, g)(h_1, \ldots, h_k) = g \cup \sigma(h_1 \sqcup \cdots \sqcup h_k) \in A_F(n).$$

In other words, we first take the disjoint union of the networks h_i, then permute their vertices with σ, and then overlay the network g.

An example is in order, since the generality of the formalism may hide the simplicity of the idea. The easiest example of our theory is the network model for simple graphs. In Ex. 2.4 we describe a one-colored network model $\mathrm{SG}\colon \mathsf{S} \to \mathsf{Mon}$ such that $\mathrm{SG}(n)$ is the collection of simple graphs with vertex set $\mathbf{n} = \{1, \ldots, n\}$. Such a simple graph is really a collection of 2-element subsets of $\mathbf{n}$, called 'edges'. Thus, we may overlay simple graphs $g, g' \in \mathrm{SG}(n)$ by taking their union $g \cup g'$. This operation makes $\mathrm{SG}(n)$ into a monoid.

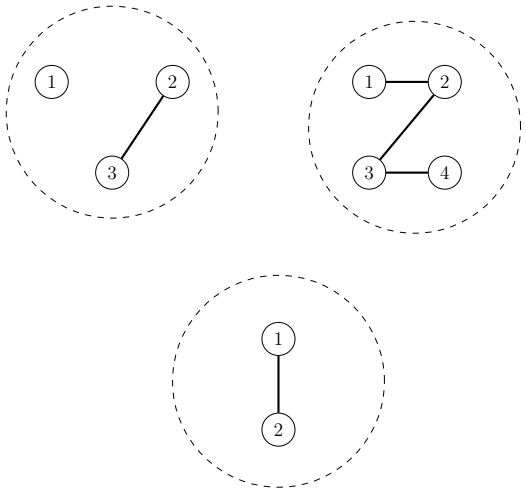
Now consider an operation $f \in \mathrm{O}_{\mathrm{SG}}(3, 4, 2; 9)$. This is an element of $S_9 \times \mathrm{SG}(9)$: a permutation of the set $\{1, \ldots, 9\}$ together with a simple graph having this set of vertices. If we take the permutation to be the identity for simplicity, this operation is just a simple graph $g \in \mathrm{SG}(9)$. We can draw an example as follows:



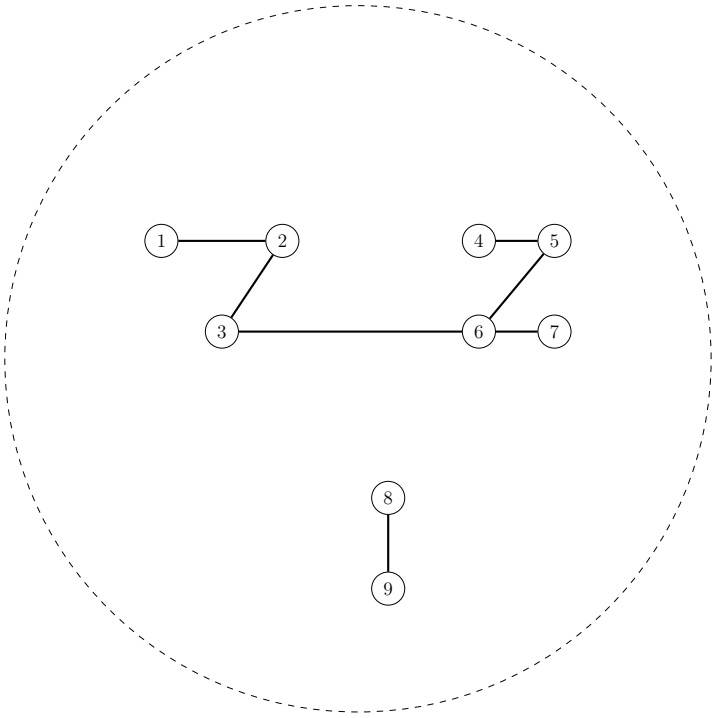
The dashed circles indicate that we are thinking of this simple graph as an element of $\mathrm{O}(3, 4, 2; 9)$: an operation that can be used to assemble simple graphs with 3, 4, and 2 vertices, respectively, to produce one with 9 vertices.

Next let us see how this operation acts on the canonical algebra A_{SG}, whose elements

are simple graphs. Suppose we have elements $a_1 \in A_{\mathrm{SG}}(3)$, $a_2 \in A_{\mathrm{SG}}(4)$ and $a_3 \in A_{\mathrm{SG}}(2)$:



We can act on these by the operation f to obtain $\alpha(f)(a_1, a_2, a_3) \in A_{\mathrm{SG}}(9)$. It looks like this:



We have simply taken the disjoint union of a_1, a_2, and a_3 and then overlaid g, obtaining a simple graph with 9 vertices.

The canonical algebra is one of the simplest algebras of the operad O_{SG}. We can define many more interesting algebras for this operad. For example, we might wish to use this operad to describe communication networks where the communicating entities have locations and the communication channels have limits on their range. To include location

data, we can choose $A(n)$ for $n \in \mathbb{N}$ to be the set of all graphs with n vertices where each vertex is an actual point in the plane $\mathbb{R}^2$. To handle range-limited communications, we could instead choose $A(n)$ to be the set of all graphs with n vertices in $\mathbb{R}^2$ where an edge is permitted between two vertices only if their Euclidean distance is less than some specified value. This still gives a well-defined algebra: when we apply an operation, we simply omit those edges from the resulting graph that would violate this restriction.

Besides the plethora of interesting algebras for the operad O_{SG}, and useful homomorphisms between these, one can also modify the operad by choosing another network model. This provides additional flexibility in the formalism. Different network models give different operads, and the construction of operads from network models is functorial, so morphisms of network models give morphisms of operads.

The technical heart of our paper is Section 6, which provides the machinery to construct operads from network models in a functorial way. This section is of independent interest, because it describes enhancements of the well-known 'Grothendieck construction' of the category of elements $\int F$ of a functor $F\colon C \to \mathsf{Cat}$, where C is any small category. For example, suppose C is symmetric monoidal and $F\colon C \to \mathsf{Cat}$ is lax symmetric monoidal, where we give Cat its cartesian symmetric monoidal structure. Then we show $\int F$ is symmetric monoidal. Moreover, we show that the construction sending the lax symmetric monoidal functor F to the symmetric monoidal category $\int F$ is functorial.

In Section 7 we apply this machinery to build operads from network models. In Section 8 we describe some algebras of these operads, and in Ex. 8.4 we discuss an algebra whose elements are networks of range-limited communication channels. In future work we plan to give many more detailed examples, and to explain how these algebras, and the homomorphisms between them, can be used to design and optimize networks.

2. One-colored network models

We begin with a special class of network models: those where the vertices of the network have just one color. To define these, we use S to stand for a skeleton of the groupoid of finite sets and bijections:

2.1. Definition. *Let* S*, the* **symmetric groupoid***, be the groupoid for which:*

- *objects are natural numbers* $n \in \mathbb{N}$*,*
- *a morphism from* m *to* n *is a bijection* $\sigma\colon \{1, \ldots, m\} \to \{1, \ldots, n\}$

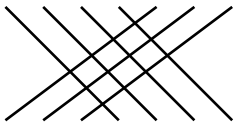
and bijections are composed in the usual way.

There are no morphisms in S from m to n unless $m = n$. For each $n \in \mathbb{N}$, the morphisms $\sigma\colon n \to n$ form the symmetric group S_n. It is convenient to write $\mathbf{n}$ for the set $\{1, \ldots, n\}$, so that a morphism $\sigma\colon n \to n$ in S is the same as a bijection $\sigma\colon \mathbf{n} \to \mathbf{n}$.

There is a functor $+\colon \mathsf{S} \times \mathsf{S} \to \mathsf{S}$ defined as follows. Given $m, n \in \mathbb{N}$ we let $m + n$ be the usual sum, and given $\sigma \in S_m$ and $\tau \in S_n$, let $\sigma + \tau \in S_{m+n}$ be as follows:

$$(\sigma + \tau)(j) = \begin{cases} \sigma(j) & \text{if } j \leq m \\ \tau(j - m) + m & \text{otherwise.} \end{cases} \tag{1}$$

For objects $m, n \in \mathsf{S}$, let $B_{m,n}$ be the block permutation of $m + n$ which swaps the first m with the last n. For example $B_{4,3}\colon 7 \to 7$ is the permutation (1473625):



The tensor product $+$ and braiding B give S the structure of a strict symmetric monoidal category. This follows as a special case of Prop. 4.1.

2.2. Definition. *A* **one-colored network model** *is a lax symmetric monoidal functor*

$$F\colon \mathsf{S} \to \mathsf{Mon}.$$

Here Mon *is the category with monoids as objects and monoid homomorphisms as morphisms, considered with its cartesian monoidal structure.*

For lax symmetric monoidal functors, see Mac Lane [17], who however calls them just symmetric monoidal functors.

Many examples of network models are given below. A pedestrian way to verify that these examples really are network models is to use the following result:

2.3. Theorem. *A one-colored network model* $F\colon \mathsf{S} \to \mathsf{Mon}$ *is the same as:*

- *a family of sets* $\{F(n)\}_{n \in \mathbb{N}}$
- *distinguished* **identity** *elements* $e_n \in F(n)$
- *a family of* **overlaying** *functions* $\cup\colon F(n) \times F(n) \to F(n)$
- *a bijection* $\sigma\colon F(n) \to F(n)$ *for each* $\sigma \in S_n$
- *a family of* **disjoint union** *functions* $\sqcup\colon F(m) \times F(n) \to F(m + n)$

satisfying the following equations:

1. $e_n \cup g = g \cup e_n = g$
2. $g_1 \cup (g_2 \cup g_3) = (g_1 \cup g_2) \cup g_3$
3. $\sigma(g_1 \cup g_2) = \sigma g_1 \cup \sigma g_2$

4. $\sigma e_n = e_n$
5. $(\sigma_2\sigma_1)g = \sigma_2(\sigma_1 g)$
6. $1(g) = g$
7. $(g_1 \cup g_2) \sqcup (h_1 \cup h_2) = (g_1 \sqcup h_1) \cup (g_2 \sqcup h_2)$
8. $e_m \sqcup e_n = e_{m+n}$
9. $\sigma g \sqcup \tau h = (\sigma + \tau)(g \sqcup h)$
10. $g_1 \sqcup (g_2 \sqcup g_3) = (g_1 \sqcup g_2) \sqcup g_3$
11. $e_0 \sqcup g = g \sqcup e_0 = g$
12. $B_{m,n}(h \sqcup g) = g \sqcup h$

for $g, g_i \in F(n)$, $h, h_i \in F(m)$, $\sigma, \sigma_i \in S_n$, $\tau \in S_m$, *and* 1 *the identity of* S_n.

PROOF. Having a functor $F\colon \mathsf{S} \to \mathsf{Mon}$ is equivalent to having the first four items satisfying equations 1–6. The binary operation $\cup$ gives the set $F(n)$ the structure of a monoid, with e_n acting as the identity. Equation 1 tells us e_n acts as an identity, and Equation 2 gives the associativity of $\cup$. Equations 3 and 4 tell us that σ is a monoid homomorphism. Equations 5 and 6 say that the map $(\sigma, g) \mapsto \sigma g$ defines an action of S_n on $F(n)$ for each n. All of these actions together give us the functor $F\colon \mathsf{S} \to \mathsf{Mon}$.

That the functor is lax monoidal is equivalent to having item 5 satisfying Equations 7–11. Equations 7 and 8 tell us that $\sqcup$ is a family of monoid homomorphisms. Equation 9 tells us that it is a natural transformation. Equation 10 tells us that the associativity hexagon diagram for lax monoidal functors commutes for F. Equation 11 implies the commutativity of the left and right unitor square diagrams. That the lax monoidal functor is symmetric is equivalent to Equation 12. It tells us that the square diagram for symmetric monoidal functors commutes for F. ∎

This is one of the simplest examples of a network model:

2.4. EXAMPLE. [**Simple graphs**] Let a **simple graph** on a set V be a set of 2-element subsets of V, called **edges**. There is a one-colored network model $\mathrm{SG}\colon \mathsf{S} \to \mathsf{Mon}$ such that $\mathrm{SG}(n)$ is the set of simple graphs on $\mathbf{n}$.

To construct this network model, we make $\mathrm{SG}(n)$ into a monoid where the product of simple graphs $g_1, g_2 \in \mathrm{SG}(n)$ is their union $g_1 \cup g_2$. Intuitively speaking, to form their union, we 'overlay' these graphs by taking the union of their sets of edges. The simple graph on $\mathbf{n}$ with no edges acts as the unit for this operation. The groups S_n acts on the monoids $\mathrm{SG}(n)$ by permuting vertices, and these actions define a functor $\mathrm{SG}\colon \mathsf{S} \to \mathsf{Mon}$.

Given simple graphs $g \in \mathrm{SG}(m)$ and $h \in \mathrm{SG}(n)$ we define $g \sqcup h \in \mathrm{SG}(m+n)$ to be their disjoint union. This gives a monoid homomorphism $\sqcup\colon \mathrm{SG}(m) \times \mathrm{SG}(n) \to \mathrm{SG}(m+n)$ because

$$(g_1 \cup g_2) \sqcup (h_1 \cup h_2) = (g_1 \sqcup h_1) \cup (g_2 \sqcup h_2).$$

This in turn gives a natural transformation with components

$$\sqcup_{m,n} \colon \mathrm{SG}(m) \times \mathrm{SG}(n) \to \mathrm{SG}(m+n),$$

which makes SG into lax symmetric monoidal functor.

One can prove this construction really gives a network model using either Thm. 2.3, which requires verifying a list of equations, or Thm. 3.1, which gives a general procedure for getting a network model from a monoid M by letting elements of $\Gamma_M(n)$ be maps from the complete graph on $\mathbf{n}$ to M. If we take $M = \mathbb{B} = \{F, T\}$ with 'or' as the monoid operation, this procedure gives the network model $\mathrm{SG} = \Gamma_{\mathbb{B}}$. We explain this in Ex. 3.2.

There are many other kinds of graph, and many of them give network models:

2.5. Example. [**Directed graphs**] Let a **directed graph** on a set V be a collection of ordered pairs $(i, j) \in V^2$ such that $i \neq j$. These pairs are called **directed edges**. There is a network model $\mathrm{DG} \colon \mathsf{S} \to \mathsf{Mon}$ such that $\mathrm{DG}(n)$ is the set of directed graphs on $\mathbf{n}$. As in Ex. 2.4, the monoid operation on $\mathrm{DG}(n)$ is union.

2.6. Example. [**Multigraphs**] Let a **multigraph** on a set V be a multiset of 2-element subsets of V. If we define $\mathrm{MG}(n)$ to be the set of multigraphs on $\mathbf{n}$, then there are at least two natural choices for the monoid operation on $\mathrm{MG}(n)$. The most direct generalization of SG of Ex. 2.4 is the network model $\mathrm{MG} \colon \mathsf{S} \to \mathsf{Mon}$ with values $(\mathrm{MG}(n), \cup)$ where $\cup$ is now union of edge multisets. That is, the multiplicity of $\{i, j\}$ in $g \cup h$ is maximum of the multiplicity of $\{i, j\}$ in g and the multiplicity of $\{i, j\}$ in h. Alternatively, there is another network model $\mathrm{MG}^+ \colon \mathsf{S} \to \mathsf{Mon}$ with values $(\mathrm{MG}(n), +)$ where $+$ is multiset sum. That is, $g + h$ obtained by adding multiplicities of corresponding edges.

2.7. Example. [**Directed multigraphs**] Let a **directed multigraph** on a set V be a multiset of ordered pairs $(i, j) \in V^2$ such that $i \neq j$. There is a network model $\mathrm{DMG} \colon \mathsf{S} \to \mathsf{Mon}$ such that $\mathrm{DMG}(n)$ is the set of directed multigraphs on $\mathbf{n}$ with monoid operation the union of multisets. Alternatively, there is a network model with values $(\mathrm{DMG}(n), +)$ where $+$ is multiset sum.

2.8. Example. [**Hypergraphs**] Let a **hypergraph** on a set V be a set of nonempty subsets of V, called **hyperedges**. There is a network model $\mathrm{HG} \colon \mathsf{S} \to \mathsf{Mon}$ such that $\mathrm{HG}(n)$ is the set of hypergraphs on $\mathbf{n}$. The monoid operation $\mathrm{HG}(n)$ is union.

2.9. Example. [**Graphs with colored edges**] Fix a set B of **edge colors** and let $\mathrm{SG} \colon \mathsf{S} \to \mathsf{Mon}$ be the network model of simple graphs as in Ex. 2.4. Then there is a network model $H \colon \mathsf{S} \to \mathsf{Mon}$ with

$$H(n) = \mathrm{SG}(n)^B$$

making the product of B copies of the monoid $\mathrm{SG}(n)$ into a monoid in the usual way. In this model, a network is a B-tuple of simple graphs, which we may view as a graph with at most one edge of each color between any pair of distinct vertices. We describe this construction in more detail in Ex. 5.5.

There are also examples of network models not involving graphs:

2.10. Example. [**Partitions**] A poset is a lattice if every finite subset has both an infimum and a supremum. If L is a lattice, then $(L, \vee)$ and $(L, \wedge)$ are both monoids, where $x \vee y$ is the supremum of $\{x, y\} \subseteq L$ and $x \wedge y$ is the infimum.

Let $P(n)$ be the set of partitions of the set $\mathbf{n}$. This is a lattice where $\pi \leq \pi'$ if the partition π is finer than π'. Thus, $P(n)$ can be made a monoid in either of the two ways mentioned above. Denote these monoids as $P^\vee(n)$ and $P^\wedge(n)$. These monoids extend to give two network models $P^\vee, P^\wedge \colon \mathsf{S} \to \mathsf{Mon}$.

3. One-colored network models from monoids

There is a systematic procedure that gives many of the network models we have seen so far. To do this, we take networks to be ways of labelling the edges of a complete graph by elements of some monoid M. The operation of overlaying two of these networks is then described using the monoid operation.

For example, consider the Boolean monoid $\mathbb{B}$: that is, the set $\{F, T\}$ with 'inclusive or' as its monoid operation. A complete graph with edges labelled by elements of $\mathbb{B}$ can be seen as a simple graph if we let T indicate the presence of an edge between two vertices and F the absence of an edge. To overlay two simple graphs g_1, g_2 with the same set of vertices we simply take the 'or' of their edge labels. This gives our first example of a network model, Ex. 2.4.

To formalize this we need some definitions. Given $n \in \mathbb{N}$, let $\mathcal{E}(n)$ be the set of 2-element subsets of $\mathbf{n} = \{1, \ldots, n\}$. We call the members of $\mathcal{E}(n)$ **edges**, since they correspond to edges of the complete graph on the set $\mathbf{n}$. We call the elements of an edge $e \in \mathcal{E}(n)$ its **vertices**.

Let M be a monoid. For $n \in \mathbb{N}$, let $\Gamma_M(n)$ be the set of functions $g \colon \mathcal{E}(n) \to M$. Define the operation $\cup \colon \Gamma_M(n) \times \Gamma_M(n) \to \Gamma_M(n)$ by $(g_1 \cup g_2)(e) = g_1(e) g_2(e)$ for $e \in \mathcal{E}(n)$. Define the map $\sqcup \colon \Gamma_M(m) \times \Gamma_M(n) \to \Gamma_M(m+n)$ by

$$(g_1 \sqcup g_2)(e) = \begin{cases} g_1(e) & \text{if both vertices of } e \text{ are } \leq m \\ g_2(e) & \text{if both vertices of } e \text{ are } > m \\ \text{the identity of } M & \text{otherwise} \end{cases}$$

The symmetric group S_n acts on $\Gamma_M(n)$ by $\sigma(g)(e) = g(\sigma^{-1}(e))$.

3.1. Theorem. *For each monoid M the data above gives a one-colored network model $\Gamma_M \colon \mathsf{S} \to \mathsf{Mon}$.*

Proof. We can define Γ_M as the composite of two functors, $\mathcal{E} \colon \mathsf{S} \to \mathsf{Inj}$ and $M^- \colon \mathsf{Inj} \to \mathsf{Mon}$, where Inj is the category of sets and injections.

The functor $\mathcal{E} \colon \mathsf{S} \to \mathsf{Inj}$ sends each object $n \in \mathsf{S}$ to $\mathcal{E}(n)$, and it sends each morphism $\sigma \colon n \to n$ to the permutation of $\mathcal{E}(n)$ that maps any edge $e = \{x, y\} \in \mathcal{E}(n)$ to $\sigma(e) = \{\sigma(x), \sigma(y)\}$. The category Inj does not have coproducts, but it is closed under coproducts in Set. It thus becomes symmetric monoidal with $+$ as its tensor product and the empty

set as the unit object. For any $m, n \in \mathsf{S}$ there is an injection

$$\mu_{m,n} \colon \mathcal{E}(m) + \mathcal{E}(n) \to \mathcal{E}(m+n)$$

expressing the fact that a 2-element subset of either m or $\mathbf{n}$ gives a 2-element subset of $\mathsf{m} + \mathbf{n}$. The functor $\mathcal{E} \colon \mathsf{S} \to \mathsf{Inj}$ becomes lax symmetric monoidal with these maps $\mu_{m,n}$ giving the lax preservation of the tensor product.

The functor $M^- \colon \mathsf{Inj} \to \mathsf{Mon}$ sends each set X to the set M^X made into a monoid with pointwise operations, and it sends each function $f \colon X \to Y$ to the monoid homomorphism $M^f \colon M^X \to M^Y$ given by

$$(M^f g)(y) = \begin{cases} g(f^{-1}(y)) & \text{if } y \in \mathrm{im}(f) \\ 1 & \text{otherwise} \end{cases}$$

for any $g \in M^X$. Using the natural isomorphisms $M^{X+Y} \cong M^X \times M^Y$ and $M^\emptyset \cong 1$ this functor can be made symmetric monoidal.

As the composite of the lax symmetric monoidal functor $\mathcal{E} \colon \mathsf{S} \to \mathsf{Inj}$ and the symmetric monoidal functor $M^- \colon \mathsf{Inj} \to \mathsf{Mon}$, the functor $\Gamma_M \colon \mathsf{S} \to \mathsf{Mon}$ is lax symmetric monoidal, and thus a network model. With the help of Thm. 2.3, it is easy to check that this description of Γ_M is equivalent to that in the theorem statement. ∎

3.2. Example. [**Simple graphs, revisited**] Let $\mathbb{B} = \{F, T\}$ be the Boolean monoid. If we interpret T and F as 'edge' and 'no edge' respectively, then $\Gamma_{\mathbb{B}}$ is just SG, the network model of simple graphs discussed in Example 2.4.

Recall from Ex. 2.6 that a multigraph on the set $\mathbf{n}$ is a multisubset of $\mathcal{E}(n)$, or in other words, a function $g \colon \mathcal{E}(n) \to \mathbb{N}$. There are many ways to create a network model $F \colon \mathsf{S} \to \mathsf{Mon}$ for which $F(n)$ is the set of multigraphs on the set $\mathbf{n}$, since $\mathbb{N}$ has many monoid structures. Two of the most important are these:

3.3. Example. [**Multigraphs with addition for overlaying**] Let $(\mathbb{N}, +)$ be $\mathbb{N}$ made into a monoid with the usual notion of addition as $+$. In this network model, overlaying two multigraphs $g_1, g_2 \colon \mathcal{E}(n) \to \mathbb{N}$ gives a multigraph $g \colon \mathcal{E}(n) \to \mathbb{N}$ with $g(e) = g_1(e) + g_2(e)$. In fact, this notion of overlay corresponds to forming the multiset sum of edge multisets and $\Gamma_{(\mathbb{N},+)}$ is the network model of multigraphs called MG^+ in Ex. 2.6.

3.4. Example. [**Multigraphs with maximum for overlaying**] Let $(\mathbb{N}, \max)$ be $\mathbb{N}$ made into a monoid with max as the monoid operation. Then $\Gamma_{(\mathbb{N},\max)}$ is a network model where overlaying two multigraphs $g_1, g_2 \colon \mathcal{E}(n) \to \mathbb{N}$ gives a multigraph $g \colon \mathcal{E}(n) \to \mathbb{N}$ with $g(e) = g_1(e) \max g_2(e)$. For this monoid structure overlaying two copies of the same multigraph gives the same multigraph. In other words, every element in each monoid $\Gamma_{(\mathbb{N},\max)}(n)$ is idempotent and $\Gamma_{(\mathbb{N},\max)}$ is the network model of multigraphs called MG in Ex. 2.6.

3.5. Example. [**Multigraphs with at most k edges between vertices**] For any $k \in \mathbb{N}$, let $\mathbb{B}_k$ be the set $\{0, \ldots, k\}$ made into a monoid with the monoid operation $\oplus$ given by

$$x \oplus y = (x + y) \min k$$

and 0 as its unit element. For example, $\mathbb{B}_0$ is the trivial monoid and $\mathbb{B}_1$ is isomorphic to the Boolean monoid. There is a network model $\Gamma_{\mathbb{B}_k}$ such that $\Gamma_{\mathbb{B}_k}(n)$ is the set of multigraphs on $\mathbf{n}$ with at most k edges between any two distinct vertices.

4. Network models

The network models described so far allow us to handle graphs with colored edges, but not with colored vertices. Colored vertices are extremely important for applications in which we have a network of agents of different types. Thus, network models will involve a set C of vertex colors in general. This requires that we replace S by the free strict symmetric monoidal category generated by the color set C. Thus, we begin by recalling this category.

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 $c_1, \ldots, c_n \in C$. We denote the unique object with $n = 0$ as I.

- There exist morphisms from $c_1 \otimes \cdots \otimes c_m$ 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.

- Composition is the usual composition of permutations.

Note that elements of C can be identified with certain objects of $\mathsf{S}(C)$, namely the one-fold tensor products. We do this in what follows.

4.1. Proposition. *$\mathsf{S}(C)$ can be given the structure of a strict symmetric monoidal category making it into the free strict symmetric monoidal category on the set C. Thus, if A is any strict symmetric monoidal category and $f \colon C \to \mathrm{Ob}(\mathsf{A})$ is any function from C to objects of the A, there exists a unique strict symmetric monoidal functor $F \colon \mathsf{S}(C) \to \mathsf{A}$ with $F(c) = f(c)$ for all $c \in C$.*

Proof. This is well-known; see for example Sassone [21, Sec. 3] or Gambino and Joyal [8, Sec. 3.1]. The tensor product of objects is $\otimes$, the unit for the tensor product is I, and the braiding

$$(c_1 \otimes \cdots \otimes c_m) \otimes (c'_1 \otimes \cdots \otimes c'_n) \to (c'_1 \otimes \cdots \otimes c'_n) \otimes (c_1 \otimes \cdots \otimes c_m)$$

is the block permutation $B_{m,n}$. Given $f\colon C \to \mathrm{Ob}(\mathsf{A})$, we define $F\colon \mathsf{S}(C) \to \mathsf{A}$ on objects by

$$F(c_1 \otimes \cdots \otimes c_n) = f(c_1) \otimes \cdots \otimes f(c_n),$$

and it is easy to check that F is strict symmmetric monoidal, and the unique functor with the required properties. ∎

4.2. Definition. *Let C be a set, called the set of* **vertex colors**. *A* **C-colored network model** *is a lax symmetric monoidal functor*

$$F\colon \mathsf{S}(C) \to \mathsf{Cat}.$$

A **network model** *is a C-colored network model for some set C.*

If C has just one element, $\mathsf{S}(C) \cong \mathsf{S}$ and a C-colored network model is a one-colored network model in the sense of Def. 2.2. Here are some more interesting examples:

4.3. Example. [**Simple graphs with colored vertices**] There is a network model of simple graphs with C-colored vertices. To construct this, we start with the network model of simple graphs $\mathrm{SG}\colon \mathsf{S} \to \mathsf{Mon}$ given in Ex. 2.4. There is a unique function from C to the one-element set. By Prop. 4.1, this function extends uniquely to a strict symmetric monoidal functor

$$F\colon \mathsf{S}(C) \to \mathsf{S}.$$

An object in $\mathsf{S}(C)$ is formal tensor product of n colors in C; applying F to this object we forget the colors and obtain the object $n \in \mathsf{S}$. Composing F and SG, we obtain a lax symmetric monoidal functor

$$\mathsf{S}(C) \xrightarrow{F} \mathsf{S} \xrightarrow{\mathrm{SG}} \mathsf{Mon}$$

which is the desired network model. We can use the same idea to 'color' any of the network models in Section 2.

Alternatively, suppose we want a network model of simple graphs with C-colored vertices where an edge can only connect two vertices of the same color. For this we take a cartesian product of C copies of the functor SG, obtaining a lax symmetric monoidal functor

$$\mathrm{SG}^C\colon \mathsf{S}^C \to \mathsf{Mon}^C.$$

There is a function $h\colon C \to \mathrm{Ob}(\mathsf{S}^C)$ sending each $c \in C$ to the object of S^{C} that equals $1 \in \mathsf{S}$ in the cth place and $0 \in \mathsf{S}$ elsewhere. Thus, by Prop. 4.1, h extends uniquely to a strict symmetric monoidal functor

$$H_C\colon \mathsf{S}(C) \to \mathsf{S}^C.$$

Furthermore, the product in Mon gives a symmetric monoidal functor

$$\Pi\colon \mathsf{Mon}^C \to \mathsf{Mon}.$$

Composing all these, we obtain a lax symmetric monoidal functor

$$\mathsf{S}(C) \xrightarrow{H_C} \mathsf{S}^C \xrightarrow{\mathrm{SG}^C} \mathsf{Mon}^C \xrightarrow{\Pi} \mathsf{Mon}$$

which is the desired network model.

More generally, if we have a network model $F_c \colon \mathsf{S} \to \mathsf{Mon}$ for each color $c \in C$, we can use the same idea to create a network model:

$$\mathsf{S}(C) \xrightarrow{H_C} \mathsf{S}^C \xrightarrow{\prod_{c \in C} F_c} \mathsf{Mon}^C \xrightarrow{\Pi} \mathsf{Mon}$$

in which the vertices of color $c \in C$ partake in a network of type F_c.

4.4. Example. [**Petri nets**] Petri nets are a kind of network widely used in computer science, chemistry and other disciplines [1]. A **Petri net** (S, T, i, o) is a pair of finite sets and a pair of functions $i, o \colon S \times T \to \mathbb{N}$. Let $P(m, n)$ be the set of Petri nets $(\mathsf{m}, \mathbf{n}, i, o)$. This becomes a monoid with product

$$(\mathsf{m}, \mathbf{n}, i, o) \cup (\mathsf{m}, \mathbf{n}, i', o') = (\mathsf{m}, \mathbf{n}, i + i', o + o')$$

The groups $S_m \times S_n$ naturally act on these monoids, so we have a functor

$$P \colon \mathsf{S}^2 \to \mathsf{Mon}.$$

There are also 'disjoint union' operations

$$\sqcup \colon P(m, n) \times P(m', n') \to P(m + m', n + n')$$

making P into a lax symmetric monoidal functor. In Ex. 4.3 we described a strict symmetric monoidal functor $H_C \colon \mathsf{S}(C) \to \mathsf{S}^C$ for any set C. In the case of the 2-element set this gives

$$H_2 \colon \mathsf{S}(2) \to \mathsf{S}^2.$$

We define the network model of Petri nets to be the composite

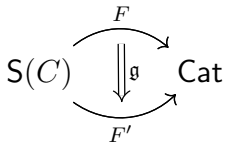
$$\mathsf{S}(2) \xrightarrow{H_2} \mathsf{S}^2 \xrightarrow{P} \mathsf{Mon}.$$

5. Categories of network models

For each choice of the set C of vertex colors, we can define a category NetMod_C of C-colored network models. However, it is useful to create a larger category NetMod containing all these as subcategories, since there are important maps between network models that involve changing the vertex colors.

5.1. Definition. *For any set C, let NetMod_C be the category for which:*

- *an object is a C-colored network model, that is, a lax symmetric monoidal functor $F\colon \mathsf{S}(C) \to \mathsf{Cat}$,*
- *a morphism is a monoidal natural transformation between such functors:*



and composition is the usual composition of monoidal natural transformations.

In particular, NetMod_1 is the category of one-colored network models. For an example involving this category, consider the network models built from monoids in Sec. 3. Any monoid M gives a one-colored network model Γ_M for which an element of $\Gamma_M(n)$ is a way of labelling the edges of the complete graph on $\mathbf{n}$ by elements of M. Thus, we should expect any homomorphism of monoids $f\colon M \to M'$ to give a morphism of network models $\Gamma_f\colon \Gamma_M \to \Gamma_{M'}$ for which

$$\Gamma_f(n)\colon \Gamma_M(n) \to \Gamma_{M'}(n)$$

applies f to each edge label.

Indeed, this is the case. As explained in the proof of Thm. 3.1, the network model Γ_M is the composite

$$\mathsf{S} \xrightarrow{\mathcal{E}} \mathsf{Inj} \xrightarrow{M^-} \mathsf{Mon}.$$

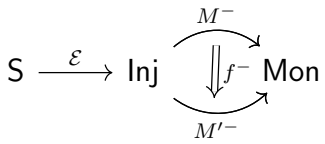
The homomorphism f gives a natural transformation

$$f^-\colon M^- \Rightarrow M'^-$$

that assigns to any finite set X the monoid homomorphism

$$\begin{array}{rccc} f^X\colon & M^X & \to & M'^X \\ & g & \mapsto & f \circ g. \end{array}$$

It is easy to check that this natural transformation is monoidal. Thus, we can whisker it with the lax symmetric monoidal functor $\mathcal{E}$ to get a morphism of network models:



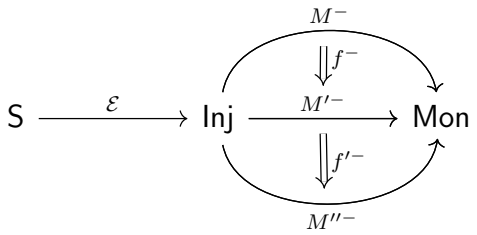
and we call this $\Gamma_f\colon \Gamma_M \to \Gamma_{M'}$.

5.2. Theorem. *There is a functor*

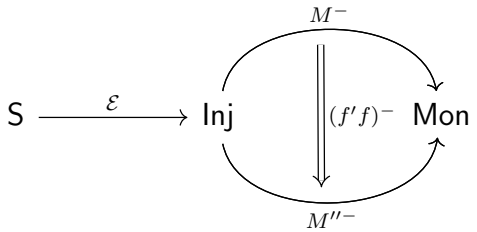
$$\Gamma\colon \mathsf{Mon} \to \mathsf{NetMod}_1$$

sending any monoid M to the network model Γ_M and any homomorphism of monoids $f\colon M \to M'$ to the morphism of network models $\Gamma_f\colon \Gamma_M \to \Gamma_{M'}$.

Proof. To check that Γ preserves composition, note that



equals



since $f'^- f^- = (f'f)^-$. Similarly Γ preserves identities. ∎

It has been said that category theory is the subject in which even the examples need examples. So, we give an example of the above result:

5.3. Example. [**Imposing a cutoff on the number of edges**] In Ex. 3.3 we described the network model of multigraphs MG^+ as $\Gamma_{(\mathbb{N},+)}$. In Ex. 3.5 we described a network model $\Gamma_{\mathbb{B}_k}$ of multigraphs with at most k edges between any two distinct vertices. There is a homomorphism of monoids

$$\begin{array}{rccc} f\colon & (\mathbb{N},+) & \to & \mathbb{B}_k \\ & n & \mapsto & n \min k \end{array}$$

and this induces a morphism of network models

$$\Gamma_f\colon \Gamma_{(\mathbb{N},+)} \to \Gamma_{\mathbb{B}_k}.$$

This morphism imposes a cutoff on the number of edges between any two distinct vertices: if there are more than k, this morphism keeps only k of them. In particular, if $k = 1$, $\mathbb{B}_k$ is the Boolean monoid, and

$$\Gamma_f\colon \mathrm{MG}^+ \to \mathrm{SG}$$

sends any multigraph to the corresponding simple graph.

One useful way to combine C-colored networks is by 'tensoring' them. This makes NetMod_C into a symmetric monoidal category:

5.4. Theorem. *For any set C, the category NetMod_C can be made into a symmetric monoidal category with the tensor product defined pointwise, so that for objects $F, F' \in \mathsf{NetMod}_C$ we have*

$$(F \otimes F')(x) = F(x) \times F'(x)$$

for any object or morphism x in $\mathsf{S}(C)$, and for morphisms ϕ, ϕ' in NetMod_C we have

$$(\phi \otimes \phi')_x = \phi_x \times \phi'_x$$

for any object $x \in \mathsf{S}(C)$.

Proof. More generally, for any symmetric monoidal categories A and B, there is a symmetric monoidal category $\hom_{\mathsf{smCat}}(\mathsf{A}, \mathsf{B})$ whose objects are lax symmetric monoidal functors from A to B and whose morphisms are monoidal natural transformations, with the tensor product defined pointwise. The proof in the 'weak' case was given by Hyland and Power [11], and the lax case works the same way. ∎

If $F, F' \colon \mathsf{S}(C) \to \mathsf{Mon}$ then their tensor product again takes values in Mon. There are many interesting examples of this kind:

5.5. Example. [**Graphs with colored edges, revisited**] In Ex. 2.9 we described network models of simple graphs with colored edges. The above result lets us build these network models starting from more basic data. To do this we start with the network model for simple graphs, $\mathrm{SG} \colon \mathsf{S} \to \mathsf{Mon}$, discussed in Ex. 2.4. Fixing a set B of 'edge colors', we then take a tensor product of copies of SG, one for each $b \in B$. The result is a network model $\mathrm{SG}^{\otimes B} \colon \mathsf{S} \to \mathsf{Mon}$ with

$$\mathrm{SG}^{\otimes B}(n) = \mathrm{SG}(n)^B$$

for each $n \in \mathbb{N}$.

5.6. Example. [**Combined networks**] We can also combine networks of different kinds. For example, if $\mathrm{DG} \colon \mathsf{S} \to \mathsf{Mon}$ is the network model of directed graphs given in Ex. 2.5 and $\mathrm{MG} \colon \mathsf{S} \to \mathsf{Mon}$ is the network model of multigraphs given in Ex. 2.6, then

$$\mathrm{DG} \otimes \mathrm{MG} \colon \mathsf{S} \to \mathsf{Mon}$$

is another network model, and we can think of an element of $(\mathrm{DG} \otimes \mathrm{MG})(n)$ as a directed graph with red edges together with a multigraph with blue edges on the set $\mathbf{n}$.

Next we describe a category NetMod of network models with arbitrary color sets, which includes all the categories NetMod_C as subcategories. To do this, first we introduce 'color-changing' functors. Recall that elements of C can be seen as certain objects of $\mathsf{S}(C)$, namely the 1-fold tensor products. If $f \colon C \to C'$ is a function, there exists a unique strict symmetric monoidal functor $f_* \colon \mathsf{S}(C) \to \mathsf{S}(C')$ that equals f on objects of the form $c \in C$. This follows from Prop. 4.1.

Next, we define an indexed category $\mathsf{NetMod}_- \colon \mathsf{Set}^{\mathrm{op}} \to \mathsf{CAT}$ that sends any set C to NetMod_C and any function $f \colon C \to D$ to the functor that sends any D-colored network model $F \colon \mathsf{S}(D) \to \mathsf{Cat}$ to the C-colored network model given by the composite

$$\mathsf{S}(C) \xrightarrow{f_*} \mathsf{S}(D) \xrightarrow{F} \mathsf{Cat}.$$

Applying the Grothendieck construction to this indexed category, we define the category of network models to be

$$\mathsf{NetMod} = \int \mathsf{NetMod}_-.$$

In elementary terms, NetMod has:

- pairs (C, F) for objects, where C is a set and $F \colon \mathsf{S}(C) \to \mathsf{Cat}$ is a C-colored network model.
- pairs $(f, g) \colon (C, F) \to (D, G)$ for morphisms, where $f \colon C \to D$ is a function and $g \colon F \Rightarrow G \circ f_*$ is a morphism of network models.

5.7. Example. [**Simple graphs with colored vertices, revisited**] In Ex. 4.3 we constructed the network model of simple graphs with colored vertices. We started with the network model for simple graphs, which is a one-colored network model $\mathrm{SG} \colon \mathsf{S} \to \mathsf{Mon}$. The unique function $! \colon C \to 1$ gives a strict symmetric monoidal functor $!_* \colon \mathsf{S}(C) \to \mathsf{S}(1) \cong \mathsf{S}$. The network model of simple graphs with C-colored vertices is the composite

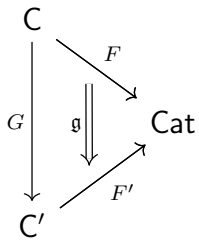
$$\mathsf{S}(C) \xrightarrow{!_*} \mathsf{S} \xrightarrow{\mathrm{SG}} \mathsf{Mon}$$

and there is a morphism from this to the network model of simple graphs, which has the effect of forgetting the vertex colors.

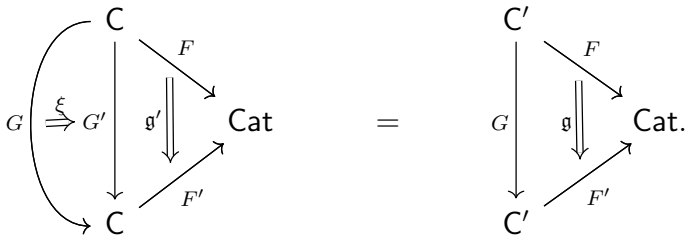
In fact, NetMod can be understood as a subcategory of the following category:

5.8. Definition. *Let* smlCat *be the category where:*

- *objects are pairs* (C, F) *where* C *is a small symmetric monoidal category and* $F \colon \mathsf{C} \to \mathsf{Cat}$ *is a lax symmetric monoidal functor, where* Cat *is considered with its cartesian monoidal structure.*
- *morphisms from* (C, F) *to* (C', F') *are pairs* $(G, \mathfrak{g})$ *where* $G \colon \mathsf{C} \to \mathsf{C}'$ *is a lax symmetric monoidal functor and* $\mathfrak{g} \colon F \Rightarrow F' \circ G$ *is a symmetric monoidal natural transformation:*



We shall use this way of thinking in the next two sections to build operads from network models. For experts, it is worth admitting that smlCat is part of a 2-category where a 2-morphism $\xi\colon (G,\mathfrak{g}) \Rightarrow (G',\mathfrak{g}')$ is a natural transformation $\xi\colon G \to G'$ such that



This lets us define 2-morphisms between network models, extending NetMod to a 2-category. We do not seem to need these 2-morphisms in our applications, so we suppress 2-categorical considerations in most of what follows. However, we would not be surprised if the 2-categorical aspects of network models turn out to be important, and we touch on them in Sec. 6.22.

6. The Grothendieck construction

In this section we describe how to build symmetric monoidal categories using the Grothendieck construction. In the next section we use this to construct operads from network models, but the material here is self-contained and of independent interest.

In what follows we always give Cat its cartesian symmetric monoidal structure. Given a small category C and a functor $F\colon \mathsf{C} \to \mathsf{Cat}$, the **Grothendieck construction** gives a category $\int F$ where:

- the objects are pairs (c,x), where $c \in \mathsf{C}$ and x is an object in Fc;
- the morphisms are $(f,g)\colon (c,x) \to (d,y)$ where $f\colon c \to d$ is a morphism in C and $g\colon Ff(x) \to y$ is a morphism in Fd;
- the composite of
$$(f',g')\colon (c,x) \to (d,y)$$
and
$$(f,g)\colon (d,y) \to (e,z)$$
is given by
$$(f,g) \circ (f',g') = (f \circ f', g \circ F(f)(g')). \tag{2}$$

In what follows we prove:

- **Thm. 6.7**: if C is monoidal and F is lax monoidal, then $\int F$ is monoidal category.
- **Thm. 6.14**: If C is braided monoidal and F is lax braided monoidal, then $\int F$ is braided monoidal.

- **Thm. 6.18**: If C is symmetric monoidal and F is lax symmetric monoidal, then $\int F$ is symmetric monoidal.

Moreover, in each case the Grothendieck construction is functorial. For example, in Def. 5.8 we described a category smICat where an object is a symmetric monoidal category C equipped with a lax symmetric monoidal functor $F\colon \mathsf{C} \to \mathsf{Cat}$. In Thm. 6.18 we show that the Grothendieck construction gives a functor

$$\textstyle\int \colon \mathsf{smICat} \to \mathsf{smCat}.$$

In proving these results we take a self-contained and elementary approach which provides the equations that we need later. We sketch a more high-powered 2-categorical approach using fibrations and indexed categories in Sec. 6.22.

We use the following lemma implicitly whenever we need to construct an isomorphism in $\int F$:

6.1. Lemma. *If $f\colon c \to d$ and $g\colon F(f)(x) \to y$ are isomorphisms, then (f, g) is an isomorphism in $\int F$.*

Proof. Naïvely one might think that (f^{-1}, g^{-1}) should be the inverse of (f, g). However, (f^{-1}, g^{-1}) is not even a morphism from (d, y) to (c, x) since g^{-1} does not go from $Ff^{-1}(y)$ to x. The inverse of (f, g) is $(f^{-1}, Ff^{-1}g^{-1})$:

$$\begin{aligned}(f, g) \circ (f^{-1}, Ff^{-1}g^{-1}) &= (f \circ f^{-1}, g \circ (F(f))(F(f^{-1})g^{-1})) \\ &= (1_c, g \circ (F(f) \circ F(f^{-1}))(g^{-1})) \\ &= (1_c, g \circ g^{-1}) \\ &= (1_c, 1_x)\end{aligned}$$

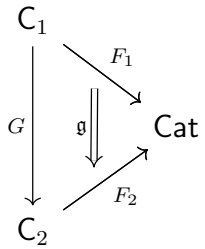
$$\begin{aligned}(f^{-1}, F(f^{-1})g^{-1}) \circ (f, g) &= (f^{-1} \circ f, Ff^{-1}(g^{-1}) \circ Ff^{-1}(g)) \\ &= (1_d, Ff^{-1}(g^{-1} \circ g)) \\ &= (1_d, 1_y)\end{aligned}$$

∎

Next we discuss the functoriality of the Grothendieck construction.

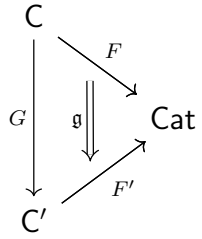
6.2. Definition. *Let ICat denote the category where*

- *an object is a pair (C, F) where C is a small category and $F\colon \mathsf{C} \to \mathsf{Cat}$ is a functor*
- *a morphism $(\mathsf{C}_1, F_1) \to (\mathsf{C}_2, F_2)$ is a pair $(G, \mathfrak{g})$ where $G\colon \mathsf{C}_1 \to \mathsf{C}_2$ is a functor and $\mathfrak{g}\colon F_1 \Rightarrow F_2 \circ G$ is a natural transformation:*



For brevity, we denote the object (C, F) *as simply* F*, and the morphism* $(G, \mathfrak{g})$ *as simply* G*.*

Having defined the Grothendieck construction on objects of ICat, we proceed to define it on morphisms. Suppose we have a morphism in ICat:



Then we can define a functor $\widehat{G}\colon \int F \to \int F'$ as follows.

$$\widehat{G}\colon \quad \begin{array}{c} (c,x) \\ \downarrow{\scriptstyle (f,g)} \\ (d,y) \end{array} \quad \mapsto \quad \begin{array}{c} (Gc, \mathfrak{g}_c x) \\ \downarrow{\scriptstyle (Gf, \mathfrak{g}_d g)} \\ (Gd, \mathfrak{g}_d y) \end{array}$$

The following result is well-known [6]:

6.3. Theorem. *There exists a unique functor, the* **Grothendieck construction**

$$\textstyle\int \colon \mathsf{ICat} \;\to\; \mathsf{Cat}$$

sending any object F *to the category* $\int F$ *and sending any morphism* $G\colon F \to F'$ *to the functor* $\widehat{G}\colon \int F \to \int F'$*.*

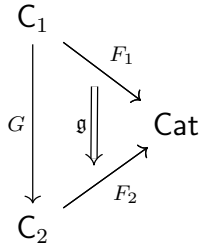
6.4. The monoidal Grothendieck construction. Next we explain how to use the Grothendieck construction to build monoidal categories.

6.5. Definition. *Let* mCat *be the category with small monoidal categories as objects and lax monoidal functors as morphisms.*

6.6. Definition. *Let* mICat *be the category of* lax monoidal functors into Cat*, where:*

- *objects are pairs* (C, F) *where* C *is a small monoidal category and* $F\colon \mathsf{C} \to \mathsf{Cat}$ *is a lax monoidal functor.*

- *morphisms from* (C_1, F_1) *to* (C_2, G_2) *are pairs* $(G, \mathfrak{g})$ *where* $G\colon \mathsf{C}_1 \to \mathsf{C}_2$ *is a lax monoidal functor and* $\mathfrak{g}\colon F \Rightarrow F' \circ G$ *is a monoidal natural transformation:*



Our goal in this subsection is to refine the Grothendieck construction to a functor

$$\textstyle\int : \mathsf{mlCat} \to \mathsf{mCat}.$$

Given C a monoidal category, and $F\colon \mathsf{C} \to \mathsf{Cat}$ a lax monoidal functor, we define a monoidal structure on $\int F$. This construction makes use of every aspect of the monoidal structure on C: the functor $\otimes\colon \mathsf{C} \times C \to \mathsf{C}$, the unit object $I \in \mathsf{C}$, and the natural isomorphisms $\alpha_{c,d,e}\colon (c \otimes d) \otimes e \to c \otimes (d \otimes e)$, $\lambda_c\colon I \otimes c \to c$, and $\rho_c\colon c \otimes I \to c$. It also uses the lax monoidal structure of F: the natural transformation $\Phi_{c,c'}\colon Fc \times Fc' \to F(c \otimes c')$ and the morphism $\phi\colon I_{\mathsf{Cat}} \to I$.

First, given two objects (c, x) and (c', x') of $\int F$, we define their tensor product by

$$(c, x) \otimes_F (c', x') = (c \otimes c', \Phi_{c,c'}(x, x')).$$

Next, consider two morphisms

$$(f, g)\colon (c, x) \to (d, y)$$
$$(f', g')\colon (c', x') \to (d', y')$$

in $\int F$. We take the first component of $(f, g) \otimes (f', g')\colon (c \otimes c', \Phi_{c,c'}(x, x')) \to (d \otimes d', \Phi_{d,d'}(y, y'))$ to be $f \otimes f'$. The second component must then be a morphism from $F(f \otimes f')(\Phi_{c,c'}(x, x'))$ to $\Phi_{d,d'}(y, y')$. To meet this condition we define the tensor product of morphisms in $\int F$ by

$$(f, g) \otimes_F (f', g') = (f \otimes f', \Phi_{d,d'}(g, g')). \tag{3}$$

Since Φ is a natural transformation, the diagram

$$\begin{array}{ccc} Fc \times Fc' & \xrightarrow{\Phi_{c,c'}} & F(c \otimes c') \\ {\scriptstyle Ff \times Ff'}\downarrow & & \downarrow{\scriptstyle F(f \otimes f')} \\ Fd \times Fd' & \xrightarrow[\Phi_{d,d'}]{} & F(d \otimes d') \end{array}$$

commutes, so $\Phi_{d,d'}(g,g')$ is a morphism from $\Phi_{d,d'}(Ff \times Ff')(x,x') = F(f \otimes f')(\Phi_{c,c'}(x,x'))$ to $\Phi_{d,d'}(y,y')$ as required.

We define the unit object of $\int F$ to be $I_F = (I,\phi)$. Then $I_F \otimes_F (c,x) = (I,\phi) \otimes_F (c,x) = (I \otimes c, \mu_{I,c}(\phi,x))$. We take the first component of the left unitor $\lambda^F_{(c,x)} \colon (I \otimes c, \mu_{I,c}(\phi,x)) \to (c,x)$ to be the map $\lambda_c \colon I \otimes c \to c$. The second component must then be a morphism from $F\lambda_c \Phi_{I,c}(\phi,x)$ to x. To meet this condition we define the left unitor for $\int F$ to be

$$\lambda^F_{(c,x)} = (\lambda_c, 1_x) \tag{4}$$

Since F is a lax monoidal functor, the diagram

$$\begin{array}{ccc} I_{\mathsf{Cat}} \times Fc & \xrightarrow{\phi \times Fc} & F(I) \times Fc \\ {\scriptstyle \lambda^{\mathsf{Cat}}_{I_{\mathsf{Cat}},Fc}}\downarrow & & \downarrow{\scriptstyle \Phi_{I,c}} \\ Fc & \xleftarrow[F\lambda_c]{} & F(I \otimes c) \end{array}$$

commutes, giving the equation

$$F\lambda_c \Phi_{I,c}(\phi,x) = x$$

as required. Similarly, we define the right unitor to be

$$\rho^F_{(c,x)} = (\rho_c, 1_x). \tag{5}$$

We take the first component of the associator $\alpha^F_{(c,x),(d,y),(e,z)}$ to be $\alpha_{c,d,e}$. The second component must then be a morphism from $F\alpha_{c,d,e}\Phi_{c\otimes d,e}(\Phi_{c,d}(x,y),z)$ to $\Phi_{c,d\otimes e}(x,\Phi_{d,e}(y,z))$. However, these two objects are equal, since the diagram

$$\begin{array}{ccc} (Fc \times Fd) \times Fe & \xrightarrow{\Phi_{c,d} \times Fe} & F(c \otimes d) \times Fe \\ {\scriptstyle \alpha^{\mathsf{Cat}}_{Fc,Fd,Fe}}\downarrow & & \downarrow{\scriptstyle \Phi_{c\otimes d,e}} \\ Fc \times (Fd \times Fe) & & F((c \otimes d) \otimes e) \\ {\scriptstyle Fc \times \Phi_{d,e}}\downarrow & & \downarrow{\scriptstyle F\alpha_{c,d,e}} \\ Fc \times F(d \otimes e) & \xrightarrow[\Phi_{c,d\otimes e}]{} & F(c \otimes (d \otimes e)) \end{array}$$

commutes. Thus, we can meet this condition by defining the associator for $\int F$ to be

$$\alpha^F_{(c,x),(d,y),(e,z)} = (\alpha_{c,d,e}, 1_{\Phi_{c,d\otimes e}(x,\Phi_{d,e}(y,z))}). \tag{6}$$

6.7. THEOREM. *If $F \colon \mathsf{C} \to \mathsf{Cat}$ is a lax monoidal functor then $\int F$ becomes a monoidal category when equipped with the above tensor product, unit object, unitors and associator.*

PROOF. Since C is a monoidal category, the following diagrams commute.

$$\begin{array}{ccc} (c \otimes I) \otimes d & & \\ {\scriptstyle \alpha_{c,I,d}}\downarrow & \searrow^{\rho_c \otimes d} & \\ c \otimes (I \otimes d) & \xrightarrow[c \otimes \lambda_d]{} & c \otimes d \end{array}$$

$$\begin{array}{ccc} ((b \otimes c) \otimes d) \otimes e & \xrightarrow{\alpha_{b \otimes c, d, e}} & (b \otimes c) \otimes (d \otimes e) \\ {\scriptstyle \alpha_{b,c,d} \otimes e}\downarrow & & \\ (b \otimes (c \otimes d)) \otimes e & & \downarrow{\scriptstyle \alpha_{b,c,d \otimes e}} \\ {\scriptstyle \alpha_{b,c \otimes d,e}}\downarrow & & \\ b \otimes ((c \otimes d) \otimes e) & \xrightarrow[b \otimes \alpha_{c,d,e}]{} & b \otimes (c \otimes (d \otimes e)) \end{array}$$

It then follows that the corresponding diagrams also commute for $\int F$, α^F, λ^F, and ρ^F.

$$\begin{array}{ccc} ((c,x) \otimes_F I_F) \otimes_F (d,y) & & \\ {\scriptstyle \alpha^F_{(c,x),I_F,(d,y)}}\downarrow & \searrow^{\rho^F_{(c,x)} \otimes_F (d,y)} & \\ (c,x) \otimes_F (I_F \otimes_F (d,y)) & \xrightarrow[(c,x) \otimes_F \lambda^F_{(d,y)}]{} & (c,x) \otimes_F (d,y) \end{array}$$

$$\begin{array}{ccc} (((b,w) \otimes_F (c,x)) \otimes_F (d,y)) \otimes_F (e,z) & \xrightarrow{\alpha^F_{(b,w) \otimes_F (c,x),(d,y),(e,z)}} & ((b,w) \otimes_F (c,x)) \otimes_F ((d,y) \otimes_F (e,z)) \\ {\scriptstyle \alpha^F_{(b,w),(c,x),(d,y)} \otimes_F (e,z)}\downarrow & & \\ ((b,w) \otimes_F ((c,x) \otimes_F (d,y))) \otimes_F (e,z) & & \downarrow{\scriptstyle \alpha^F_{(b,w),(c,x),(d,y) \otimes_F (e,z)}} \\ {\scriptstyle \alpha^F_{(b,w),(c,x) \otimes_F (d,y),(e,z)}}\downarrow & & \\ (b,w) \otimes_F (((c,x) \otimes_F (d,y)) \otimes_F (e,z)) & \xrightarrow[(b,w) \otimes_F \alpha^F_{(c,x),(d,y),(e,z)}]{} & (b,w) \otimes_F ((c,x) \otimes_F ((d,y) \otimes_F (e,z))) \end{array}$$

■

Next we show that a morphism in mlCat gives a lax monoidal functor. Recall that such a morphism is a quadruple $(G, \Gamma, \gamma, \mathfrak{g})\colon (F, \Phi, \phi) \to (F', \Phi', \phi')$ where

$$\begin{aligned} G &\colon C \to C' \\ \Gamma_{c,d} &\colon Gc \otimes' Gd \to G(c \otimes d) \\ \gamma &\colon I_{\mathsf{C}'} \to G(I) \\ \mathfrak{g} &\colon F \Rightarrow F'G. \end{aligned}$$

We already know how to get a functor $\widehat{G}\colon \int F \to \int F'$ from this data. We next define $\widehat{\Gamma}$ and $\widehat{\gamma}$ to make $\widehat{G}$ into a lax monoidal functor:

$$\widehat{\Gamma}_{(c,x),(d,x')} = (\Gamma_{c,d}, 1), \tag{7}$$

$$\widehat{\gamma} = (\gamma, 1). \tag{8}$$

One can check that these have the required source and target.

6.8. Theorem. *There exists a unique functor, the* **monoidal Grothendieck construction**

$$\textstyle\int \colon \mathsf{mlCat} \;\to\; \mathsf{mCat},$$

that sends any object F to the monoidal category $\int F$ given in Thm. 6.7 and sends any morphism $G\colon F \to F'$ to the lax monoidal functor $\widehat{G}\colon \int F \to \int F'$ defined above.

Proof. Uniqueness follows because the theorem specifies $\int$ on objects and morphisms. For existence, we need to check that $\widehat{G}$ is a lax monoidal functor and that $\int$ preserves composition and identities.

We already know that $\widehat{G}$ is a functor. We start by checking that $\widehat{\Gamma}$ is a natural transformation. Let $f\colon c \to d$, $f'\colon c' \to d'$ be morphisms in C. Since Γ is a natural transformation, the following diagram commutes

$$\begin{array}{ccc} Gc \otimes' Gc' & \xrightarrow{\Gamma_{c,c'}} & G(c \otimes c') \\ {\scriptstyle Gf \otimes' Gf'}\downarrow & & \downarrow{\scriptstyle G(f \otimes f')} \\ Gd \otimes' Gd' & \xrightarrow[\Gamma_{d,d'}]{} & G(d \otimes d') \end{array}$$

giving the equation $G(f \otimes f') \circ \Gamma_{c,c'} = \Gamma_{d,d'} \circ (Gf \otimes' Gf')$. Since $\mathfrak{g}$ is a monoidal natural transformation, the following diagram commutes

$$\begin{array}{ccc} Fd \times Fd' & \xrightarrow{\mathfrak{g}_d \times \mathfrak{g}_{d'}} & F'Gd \times F'Gd' \\ {\scriptstyle \Phi_{d,d'}}\downarrow & & \downarrow{\scriptstyle F'\Gamma_{d,d'}\Phi'_{Gd,Gd'}} \\ F(d \otimes d') & \xrightarrow[\mathfrak{g}_{d\otimes d'}]{} & F'G(d \otimes d') \end{array}$$

giving the equation $\mathfrak{g}_{d\otimes d'}\Phi_{d,d'} = F'\Gamma_{d,d'}\Phi'_{Gd,Gd'}(\mathfrak{g}_d \times \mathfrak{g}_{d'})$. Then

$$\begin{aligned} \widehat{G}((f,g) \otimes_F (f',g')) \circ \widehat{\Gamma}_{(c,x),(c',x')} &= \widehat{G}(f \otimes f', \Phi_{d,d'}(g,g')) \circ (\Gamma_{c,c'}, 1) \\ &= (G(f \otimes f'), \mathfrak{g}_{d\otimes d'}\Phi_{d,d'}(g,g')) \circ (\Gamma_{c,c'}, 1) \\ &= (G(f \otimes f') \circ \Gamma_{c,c'}, \mathfrak{g}_{d\otimes d'}\Phi_{d,d'}(g,g')) \\ &= (\Gamma_{d,d'} \circ (Gf \otimes' Gf'), F'\Gamma_{d,d'}\Phi'_{Gd,Gd'}(\mathfrak{g}_d g, \mathfrak{g}_{d'} g')) \\ &= (\Gamma_{d,d'}, 1) \circ (Gf \otimes Gf', \Phi'_{Gd,Gd'}(\mathfrak{g}_d g, \mathfrak{g}_{d'} g') \\ &= (\Gamma_{d,d'}, 1) \circ (Gf, \mathfrak{g}_d g) \otimes_{F'} (Gf', \mathfrak{g}_{d'} g') \\ &= \widehat{\Gamma}_{(d,y),(d',y')} \circ \widehat{G}(f,g) \otimes_{F'} \widehat{G}(f',g') \end{aligned}$$

which tells us that the following diagram commutes.

$$\begin{array}{ccc}
\widehat{G}(c,x) \otimes_{F'} \widehat{G}(c',x') & \xrightarrow{\widehat{\Gamma}_{(c,x),(c',x')}} & \widehat{G}((c,x) \otimes_F (c',x')) \\
{\scriptstyle \widehat{G}(f,g)\otimes_{F'}\widehat{G}(f',g')}\downarrow & & \downarrow{\scriptstyle \widehat{G}((f,g)\otimes_F(f',g'))} \\
\widehat{G}(d,y) \otimes_{F'} \widehat{G}(d',y') & \xrightarrow[\widehat{\Gamma}_{(d,y),(d',y')}]{} & \widehat{G}((d,y) \otimes_F (d',y'))
\end{array}$$

Next, we check that $\widehat{\Gamma}$ satisfies the necessary conditions to be a lax structure map. Since (G, Γ) is a lax monoidal functor, the following diagrams commute.

$$\begin{array}{ccc}
(Gc \otimes' Gd) \otimes' Ge & \xrightarrow{\alpha'_{Gc,Gd,Ge}} & Gc \otimes' (Gd \otimes' Ge) \\
{\scriptstyle \Gamma_{c,d}\otimes' Ge}\downarrow & & \downarrow{\scriptstyle Gc\otimes'\Gamma_{d,e}} \\
G(c \otimes d) \otimes' Ge & & Gc \otimes' G(d \otimes e) \\
{\scriptstyle \Gamma_{c\otimes d,e}}\downarrow & & \downarrow{\scriptstyle \Gamma_{c,d\otimes e}} \\
G((c \otimes d) \otimes e) & \xrightarrow[G\alpha_{c,d,e}]{} & G(c \otimes (d \otimes e))
\end{array}$$

$$\begin{array}{ccc}
I' \otimes' Gc & \xrightarrow{\gamma\otimes' Gc} & GI \otimes' Gc \\
{\scriptstyle \lambda'_{Gc}}\downarrow & & \downarrow{\scriptstyle \Gamma_{I,c}} \\
Gc & \xleftarrow[G\lambda_c]{} & G(I \otimes c)
\end{array}$$

$$\begin{array}{ccc}
Gc \otimes' I' & \xrightarrow{Gc\otimes'\gamma} & Gc \otimes' GI \\
{\scriptstyle \rho'_{Gc}}\downarrow & & \downarrow{\scriptstyle \Gamma_{c,I}} \\
Gc & \xleftarrow[G\rho_c]{} & G(c \otimes I)
\end{array}$$

It then follows that the corresponding diagrams also commute for $\widehat{G}$, $\widehat{\Gamma}$, and $\widehat{\gamma}$.

$$\begin{array}{ccc}
(\widehat{G}(c,x) \otimes_{F'} \widehat{G}(d,y)) \otimes_{F'} \widehat{G}(e,z) & \xrightarrow{\alpha^{F'}_{\widehat{G}(c,x),\widehat{G}(d,y),\widehat{G}(e,z)}} & \widehat{G}(c,x) \otimes_{F'} (\widehat{G}(d,y) \otimes_{F'} \widehat{G}(e,z)) \\
{\scriptstyle \widehat{\Gamma}_{(c,x),(d,y)}\otimes_{F'}\widehat{G}(e,z)}\downarrow & & \downarrow{\scriptstyle \widehat{G}(c,x)\otimes_{F'}\widehat{\Gamma}_{(d,y),(e,z)}} \\
\widehat{G}((c,x) \otimes (d,y)) \otimes_{F'} \widehat{G}(e,z) & & \widehat{G}(c,x) \otimes_{F'} \widehat{G}((d,y) \otimes_F (e,z)) \\
{\scriptstyle \widehat{\Gamma}_{(c,x)\otimes F(d,y),(e,z)}}\downarrow & & \downarrow{\scriptstyle \widehat{\Gamma}_{(c,x),(d,y)\otimes F(e,z)}} \\
\widehat{G}(((c,x) \otimes_F (d,y)) \otimes_F (e,z)) & \xrightarrow[\widehat{G}\alpha^F_{(c,x),(d,y),(e,z)}]{} & \widehat{G}((c,x) \otimes_F ((d,y) \otimes_F (e,z)))
\end{array}$$

$$\begin{array}{ccc} I_{F'} \otimes_{F'} \widehat{G}(c,x) & \xrightarrow{\widehat{\gamma} \otimes_{F'} \widehat{G}(c,x)} & \widehat{G} I_F \otimes_{F'} \widehat{G}(c,x) \\ {\scriptstyle \lambda^{F'}_{\widehat{G}(c,x)}}\downarrow & & \downarrow{\scriptstyle \widehat{\Gamma}_{I,(c,x)}} \\ \widehat{G}(c,x) & \xleftarrow[\widehat{G}\lambda^F_{(c,x)}]{} & \widehat{G}(I_F \otimes_F (c,x)) \end{array}$$

$$\begin{array}{ccc} \widehat{G}(c,x) \otimes_{F'} I_{F'} & \xrightarrow{\widehat{G}(c,x) \otimes_{F'} \widehat{\gamma}} & \widehat{G}(c,x) \otimes_{F'} \widehat{G} I \\ {\scriptstyle \rho^{F'}_{\widehat{G}(c,x)}}\downarrow & & \downarrow{\scriptstyle \widehat{\Gamma}_{(c,x),I}} \\ \widehat{G}(c,x) & \xleftarrow[\widehat{G}\rho^F_{(c,x)}]{} & \widehat{G}((c,x) \otimes_F I) \end{array}$$

Finally, we check that composition is preserved.

$$\begin{aligned} ((G,\Gamma,\gamma,\mathfrak{g}) \circ (G',\Gamma',\gamma',\mathfrak{g}'))\widehat{\ } &= (G \circ G', G\Gamma' \circ \Gamma_{G'}, G\gamma' \circ \gamma, \mathfrak{g}_G \circ \mathfrak{g}')\widehat{\ } \\ &= (\widehat{G \circ G'}, (G\Gamma' \circ \Gamma_{G'}, 1), (G\gamma' \circ \gamma, 1)) \\ &= (\widehat{G} \circ \widehat{G'}, (G\Gamma', 1) \circ (\Gamma_{G'}, 1), (G\gamma', 1) \circ (\gamma, 1)) \\ &= (\widehat{G} \circ \widehat{G'}, \widehat{G}(\Gamma', 1) \circ (\Gamma_{G'}, 1), \widehat{G}(\gamma', 1) \circ (\gamma, 1)) \\ &= (\widehat{G} \circ \widehat{G'}, \widehat{G}\widehat{\Gamma'} \circ \widehat{\Gamma}_{\widehat{G'}}, \widehat{G}\widehat{\gamma'} \circ \widehat{\gamma}) \\ &= (\widehat{G}, \widehat{\Gamma}, \widehat{\gamma}) \circ (\widehat{G'}, \widehat{\Gamma'}, \widehat{\gamma'}) \\ &= (G,\Gamma,\gamma,\mathfrak{g})\widehat{\ } \circ (G',\Gamma',\gamma',\mathfrak{g}')\widehat{\ } \end{aligned}$$

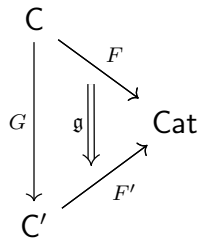
■

Under some conditions the Grothendieck construction gives *strict* monoidal categories and functors:

6.9. Proposition. *If C is a strict monoidal category and $F\colon \mathsf{C} \to \mathsf{Cat}$ is a lax monoidal functor, then $\int F$ as defined in Thm. 6.7 is a strict monoidal category.*

Proof. This follows from Eq. 4 for the left unitor, Eq. 5 for the right unitor, and Eq. 6 for the associator in $\int F$. These isomorphisms are all built from the corresponding isomorphisms in C in such a way that if C is strict monoidal, so is $\int F$. ■

6.10. Proposition. *If*



is a morphism in mlCat *such that G is a strict monoidal functor, then $\widehat{G}\colon \int F \to \int F'$ as defined in Thm. 6.8 is a strict monoidal functor.*

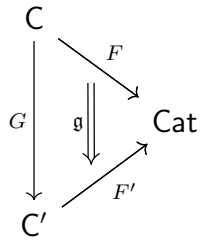
PROOF. This follows from Eq. 7 and Eq. 8, which give the morphisms describing how $\widehat{G}$ laxly preserves of the tensor product and unit for the tensor product. These morphisms are built from the corresponding morphisms for G in such a way that if G is strict monoidal, so is $\widehat{G}$. ∎

6.11. THE BRAIDED GROTHENDIECK CONSTRUCTION. Next we consider the braided case.

6.12. DEFINITION. *Let* bmCat *be the category with small braided monoidal categories as objects and lax braided monoidal functors as morphisms.*

6.13. DEFINITION. *Let* bmlCat *be the category where:*

- *objects are pairs* (C, G) *where* C *is a small braided monoidal category and* $G\colon \mathsf{C} \to \mathsf{Cat}$ *is a lax braided monoidal functor.*
- *morphisms from* (C_1, G_1) *to* (C_2, G_2) *are pairs* $(G, \mathfrak{g})$ *where* $G\colon \mathsf{C}_1 \to \mathsf{C}_2$ *is a lax braided monoidal functor and* $\mathfrak{g}\colon F \Rightarrow F' \circ G$ *is a braided monoidal natural transformation:*



Let C be a braided monoidal category with braiding $B_{c,d}\colon c \otimes d \to d \otimes c$. Let $F\colon \mathsf{C} \to \mathsf{Cat}$ a lax braided monoidal functor with lax structure map Φ, so that the following diagram commutes:

$$\begin{array}{ccc} Fc \times Fd & \xrightarrow{\Phi_{c,d}} & F(c \otimes d) \\ {\scriptstyle B_{c,d}}\downarrow & & \downarrow{\scriptstyle FB_{c,d}} \\ Fd \times Fc & \xrightarrow[\Phi_{d,c}]{} & F(d \otimes c). \end{array}$$

We claim that in this situation we can make $\int F$ into a braided monoidal category, giving it a braiding

$$B^F_{(c,x),(d,y)}\colon (c \otimes d, \Phi_{c,d}(x,y)) \to (d \otimes c, \Phi_{d,c}(y,x)).$$

We take the first component of this morphism to be $B_{c,d}$. The second component must then be a morphism from $FB_{c,d}(\Phi_{c,d}(x,y))$ to $\Phi_{d,c}(y,x)$, but

$$\begin{aligned} FB_{c,d}(\Phi_{c,d}(x,y)) &= \Phi_{d,c}(B_{c,d}(x,y)) \\ &= \Phi_{d,c}(y,x). \end{aligned}$$

so if we define the braiding in $\int F$ by

$$B^F_{(c,x),(d,y)} = (B_{c,d}, 1)$$

then this condition is met.

6.14. THEOREM. *If $F\colon \mathsf{C} \to \mathsf{Cat}$ is a lax braided monoidal functor, then $\int F$ made monoidal as in Thm. 6.7 and given the braiding B^F is a braided monoidal category.*

PROOF. We need to show that B^F is a natural transformation and that it obeys the hexagon identities. Let $(f,g)\colon (c,x) \to (d,y)$ and $(f',g')\colon (c',x') \to (d',y')$ be morphisms in C. Since C is braided monoidal with the braiding B, the following diagram commutes:

$$\begin{array}{ccc} c \otimes c' & \xrightarrow{B_{c,c'}} & c' \otimes c \\ {\scriptstyle f\otimes f'}\downarrow & & \downarrow{\scriptstyle f'\otimes f} \\ d \otimes d' & \xrightarrow[B_{d,d'}]{} & d' \otimes d \end{array}$$

giving the equation $(f' \otimes f) \circ B_{c,c'} = B_{d,d'} \circ (f \otimes f')$. Since F is lax braided monoidal, the following diagram commutes:

$$\begin{array}{ccc} Fd \times Fd' & \xrightarrow{B_{Fd,Fd'}} & Fd' \times Fd \\ {\scriptstyle \Phi_{d,d'}}\downarrow & & \downarrow{\scriptstyle \Phi_{d',d}} \\ F(d \otimes d') & \xrightarrow[FB_{d,d'}]{} & F(d' \otimes d') \end{array}$$

giving the equation $FB_{d,d'}(\Phi_{d,d'}) = \Phi_{d',d}(B_{Fd,Fd'})$. Thus

$$\begin{aligned} B^F_{(d,y),(d',y')} \circ ((f,g) \otimes_F (f',g')) &= (B_{d,d'}, 1) \circ (f \otimes f', \Phi_{d,d'}(g,g')) \\ &= (B_{d,d'} \circ f \otimes f', FB_{d,d'}(\Phi_{d,d'}(g,g'))) \\ &= (f' \otimes f \circ B_{c,c'}, \Phi_{d',d}(B_{Fd,Fd'}(g,g')) \\ &= (f' \otimes f \circ B_{c,c'}, \Phi_{d',d}(g',g)) \\ &= (f' \otimes f, \Phi_{d',d}(g',g)) \circ (B_{c,c'}, 1) \\ &= ((f',g') \otimes_F (f,g)) \circ B^F_{(c,x),(c',x')}. \end{aligned}$$

This tells us that the following diagram commutes, and thus B^F is natural.

$$\begin{array}{ccc} (c,x) \otimes_F (c',x') & \xrightarrow{B^F_{(c,x),(c',x')}} & (c',x') \otimes_F (c,x) \\ {\scriptstyle (f,g)\otimes_F(f',g')}\downarrow & & \downarrow{\scriptstyle (f',g')\otimes_F(f,g)} \\ (d,y) \otimes_F (d',y') & \xrightarrow[B^F_{(d,y),(d',y')}]{} & (d',y') \otimes_F (d,y) \end{array}$$

Next we show that the necessary diagrams commute to make B^F a braiding for $\int F$. Notice that the diagrams

$$\begin{array}{ccc}
c \otimes (d \otimes e) & \xrightarrow{\alpha^{-1}_{c,d,e}} & (c \otimes d) \otimes e \\
{\scriptstyle B_{c,d\otimes e}}\downarrow & & \downarrow{\scriptstyle B_{c,d}\otimes e} \\
(d \otimes e) \otimes c & & (d \otimes c) \otimes e \\
{\scriptstyle \alpha^{-1}_{d,e,c}}\uparrow & & \downarrow{\scriptstyle \alpha_{d,c,e}} \\
d \otimes (e \otimes c) & \xleftarrow[d\otimes B_{c,e}]{} & d \otimes (c \otimes e)
\end{array}$$

$$\begin{array}{ccc}
(c \otimes d) \otimes e & \xrightarrow{\alpha_{c,d,e}} & c \otimes (d \otimes e) \\
{\scriptstyle B_{c\otimes d,e}}\downarrow & & \downarrow{\scriptstyle c\otimes B_{d,e}} \\
e \otimes (c \otimes d) & & c \otimes (e \otimes d) \\
{\scriptstyle \alpha_{e,c,d}}\uparrow & & \downarrow{\scriptstyle \alpha^{-1}_{c,e,d}} \\
(e \otimes c) \otimes d & \xleftarrow[B_{c,e}\otimes d]{} & (c \otimes e) \otimes d
\end{array}$$

commute. Then the corresponding diagrams for α^F and B^F commute.

$$\begin{array}{ccc}
(c,x) \otimes_F ((d,y) \otimes_F (e,z)) & \xrightarrow{(\alpha^F)^{-1}_{(c,x),(d,y),(e,z)}} & ((c,x) \otimes_F (d,y)) \otimes_F (e,z) \\
{\scriptstyle B^F_{(c,x),(d,y)\otimes_F(e,z)}}\downarrow & & \downarrow{\scriptstyle B^F_{(c,x),(d,y)}\otimes(e,z)} \\
((d,y) \otimes_F (e,z)) \otimes_F (c,x) & & ((d,y) \otimes_F (c,x)) \otimes_F (e,z) \\
{\scriptstyle (\alpha^F)^{-1}_{(d,y),(e,z),(c,x)}}\uparrow & & \downarrow{\scriptstyle \alpha^F_{(d,y),(c,x),(e,z)}} \\
(d,y) \otimes_F ((e,z) \otimes_F (c,x)) & \xleftarrow[(d,y)\otimes_F B^F_{(c,x),(e,z)}]{} & (d,y) \otimes_F ((c,x) \otimes_F (e,z))
\end{array}$$

$$\begin{array}{ccc}
((c,x) \otimes_F (d,y)) \otimes_F (e,z) & \xrightarrow{\alpha^F_{(c,x),(d,y),(e,z)}} & (c,x) \otimes_F ((d,y) \otimes_F (e,z)) \\
{\scriptstyle B^F_{(c,x)\otimes_F(d,y),(e,z)}}\downarrow & & \downarrow{\scriptstyle (c,x)\otimes_F B^F_{(d,y),(e,z)}} \\
(e,z) \otimes_F ((c,x) \otimes_F (d,y)) & & (c,x) \otimes_F ((e,z) \otimes_F (d,y)) \\
{\scriptstyle \alpha^F_{(e,z),(c,x),(d,y)}}\uparrow & & \downarrow{\scriptstyle (\alpha^F)^{-1}_{(c,x),(e,z),(d,y)}} \\
((e,z) \otimes_F (c,x)) \otimes_F (d,y) & \xleftarrow[B^F_{(c,x),(e,z)}\otimes_F(d,y)]{} & ((c,x) \otimes_F (e,z)) \otimes_F (d,y)
\end{array}$$

■

6.15. THEOREM. *There exists a unique functor, the* **braided Grothendieck construction**

$$\int : \mathsf{bmICat} \to \mathsf{bmCat},$$

that sends any object F to the braided monoidal category $\int F$ given in Thm. 6.14 and sends any morphism $G\colon F \to F'$ to the lax braided monoidal functor $\widehat{G}\colon \int F \to \int F'$ defined above.

PROOF. Uniqueness follows because the theorem specifies $\int$ on objects and morphisms. From Thm. 6.8 we already know that $\widehat{G}$ is lax monoidal for any morphism in mICat and that $\int$ preserves composition and identities. Thus, for existence all we need to show is that $\widehat{G}$ is in fact braided.

Since G is braided monoidal, the following diagram commutes:

$$\begin{array}{ccc} Gc \otimes' Gd & \xrightarrow{B'_{Gc,Gd}} & Gd \otimes' Gc \\ {\scriptstyle \Gamma_{c,d}}\downarrow & & \downarrow{\scriptstyle \Gamma_{d,c}} \\ G(c \otimes d) & \xrightarrow[GB_{c,d}]{} & G(d \otimes c) \end{array}$$

Thus, the corresponding diagram commutes:

$$\begin{array}{ccc} \widehat{G}(c,x) \otimes_{F'} \widehat{G}(d,y) & \xrightarrow{B^{F'}_{\widehat{G}(c,x),\widehat{G}(d,y)}} & \widehat{G}(d,y) \otimes_{F'} \widehat{G}(c,x) \\ {\scriptstyle \widehat{\Gamma}_{(c,x),(d,y)}}\downarrow & & \downarrow{\scriptstyle \widehat{\Gamma}_{(d,y),(c,x)}} \\ \widehat{G}((c,x) \otimes_F (d,y)) & \xrightarrow[\widehat{G}B^F_{(c,x),(d,y)}]{} & \widehat{G}((d,y) \otimes_F (c,x)) \end{array}$$

This shows that $\widehat{G}$ is braided. ∎

6.16. THE SYMMETRIC GROTHENDIECK CONSTRUCTION. Finally we turn to the symmetric monoidal case.

6.17. DEFINITION. *Let* smCat *be the category with small symmetric monoidal categories as objects and lax symmetric monoidal functors as morphisms.*

We defined the category smICat in Def. 5.8. So, we are ready to state our main result:

6.18. THEOREM. *There exists a unique functor, the* **symmetric Grothendieck construction**

$$\int : \mathsf{smICat} \to \mathsf{smCat}$$

that acts on objects and morphisms as in Thm. 6.15.

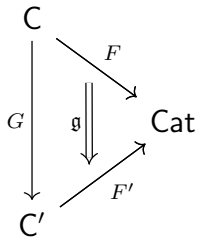
PROOF. In Thm. 6.15 we obtained a functor $\int : \mathsf{bmICat} \to \mathsf{bmCat}$, so for both existence and uniqueness we need only check that if an object $F\colon \mathsf{C} \to \mathsf{Cat}$ of bmICat has C symmetric then $\int F$ is symmetric as well. This is straightforward from the formula for the braiding in $\int F$. ∎

We conclude with a strict version of the above result, which we use in the next section.

6.19. Definition. *Let* ssmCat *be the category with small strict symmetric monoidal categories as objects and strict symmetric monoidal functors as morphisms.*

6.20. Definition. *Let* $\mathsf{ssmlCat}$ *be the category where:*

- *objects are pairs* (C, G) *where* C *is a small strict symmetric monoidal category and* $G\colon \mathsf{C} \to \mathsf{Cat}$ *is a lax symmetric monoidal functor.*
- *morphisms from* (C_1, G_1) *to* (C_2, G_2) *are pairs* $(G, \mathfrak{g})$ *where* $G\colon \mathsf{C}_1 \to \mathsf{C}_2$ *is a strict symmetric monoidal functor and* $\mathfrak{g}\colon F \Rightarrow F' \circ G$ *is a symmetric monoidal natural transformation:*



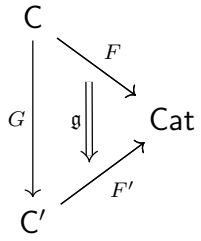
6.21. Theorem. *There is a unique functor, the* **strict symmetric Grothendieck construction**

$$\textstyle\int\colon \mathsf{ssmlCat} \;\to\; \mathsf{ssmCat},$$

that acts on objects and morphisms as in Thm. 6.18.

Proof. To prove this it suffices to check these claims:

1. If $F\colon \mathsf{C} \to \mathsf{Cat}$ is an object of smlCat with C a *strict* symmetric monoidal category, then $\int F$ is a strict symmetric monoidal category.
2. If



   is a morphism in smlCat with G a *strict* symmetric monoidal functor, then $\widehat{G}\colon \int F \to \int F$ is a strict symmetric monoidal functor.

The first of these follows from Prop. 6.9, while the second follows from Prop. 6.10. ∎

6.22. A 2-CATEGORICAL PERSPECTIVE. We have explained how to use the Grothendieck construction to build a symmetric monoidal category $\int F$ from a lax symmetric monoidal functor $F\colon \mathsf{C} \to \mathsf{Cat}$. In what follows, all we really need are the explicit formulas for how this works. Still, it seems worthwhile to set this result in a larger context. This requires a 2-categorical perspective on fibrations and indexed categories. A full review of the prerequisites would be rather lengthy, so we only recall a few facts. The main ideas go back to Grothendieck [9], but they have been developed and explained by many subsequent authors [6, 10, 12, 13, 22], whose works can be consulted for details. To conform to this literature we now replace C with C^{op}, which lets us work with fibrations rather than opfibrations. This makes no real difference for the categories C we are mainly interested in, which are groupoids.

A functor $F\colon \mathsf{C}^{\mathrm{op}} \to \mathsf{Cat}$ is sometimes called a 'split indexed category'. The Grothendieck construction builds from this a category $\int F$. The objects and morphism of $\int F$ are pairs whose first component belongs to C. Projecting onto this first component gives a functor

$$p\colon \textstyle\int F \to \mathsf{C}.$$

This functor is equipped with some extra structure that makes it into a 'split fibration'. The idea is that an object of $\int F$ is an object of C equipped with extra structure, and the splitting gives a well-behaved way to take this extra structure and pull it back along any morphism in C. For example, given a bijection of finite sets, and a simple graph on the domain of this bijection, we can pull it back to the codomain.

Conversely, from a split fibration we can recover a split indexed category. Indeed, split fibrations are essentially 'the same' as split indexed categories. To express this fact clearly, we need a couple of 2-categories that are nicely explained by Jacobs [12, Sec. 1.7]. First, we need the 2-category ICat of split indexed categories, where an object is a functor $F\colon \mathsf{C}^{\mathrm{op}} \to \mathsf{Cat}$ for an arbitrary small category C, a morphism is a pseudonatural transformation, and a 2-morphism is a modification. Second, we need the 2-category $\mathsf{Fib}_{\mathbf{split}}$ where an object is a split fibration, a morphism is a morphism of fibrations that preserves the splitting, and a 2-morphism is a fibered natural transformation. The map sending any functor $F\colon \mathsf{C} \to \mathsf{Cat}$ to the split fibration $p\colon \int F \to \mathsf{C}$ then extends to an equivalence of 2-categories

$$G\colon \mathsf{ICat} \xrightarrow{\sim} \mathsf{Fib}_{\mathbf{split}}.$$

There is also a 2-functor

$$\mathrm{dom}\colon \mathsf{Fib}_{\mathbf{split}} \longrightarrow \mathsf{Cat}$$

that sends a split fibration $F\colon \mathsf{D} \to \mathsf{C}$ to its domain category D, and the composite

$$\textstyle\int = \mathrm{dom} \circ G\colon \mathsf{ICat} \longrightarrow \mathsf{Cat}$$

is none other than the Grothendieck construction, now regarded as a 2-functor rather than a mere functor. For details see Jacobs [12, Thm. 1.10.7].

A 'pseudomonoid' is a generalization of a monoidal category which makes sense in any 2-category with finite products (or even more generally). We can also define braided and

symmetric pseudomonoids, which generalize braided and symmetric monoidal categories [7]. For any 2-category $\mathcal{C}$ with finite products, let:

- $\mathsf{m}\mathcal{C}$ be the 2-category of pseudomonoids in $\mathcal{C}$, monoidal morphisms, and monoidal transformations,
- $\mathsf{bm}\mathcal{C}$ be the 2-category of braided pseudomonoids in $\mathcal{C}$, braided monoidal morphisms, and braided monoidal transformations,
- $\mathsf{sm}\mathcal{C}$ be the 2-category of symmetric pseudomonoids in $\mathcal{C}$, symmetric monoidal morphisms, and symmetric monoidal transformations.

All these concepts are defined by McCrudden [18]. The notation here is compatible with that introduced earlier in this section, except that now we are working 2-categorically. That is, mCat, bmCat and smCat are 2-categories whose underlying categories were already defined in Defs. 6.5, 6.12 and 6.17. More interesting is that the 2-category $\mathsf{Fib}_{\mathbf{split}}$ has finite products, so the equivalent 2-category ICat does as well, and one can prove that the 2-categories $\mathsf{mICat}, \mathsf{bmICat}$ and smICat are 2-categories whose underlying categories match those defined in Defs. 6.6, 6.13 and 5.8, at least after replacing C with C^{op}. We leave the verification of this as an exercise for the reader.

The equivalence $G\colon \mathsf{ICat} \to \mathsf{Fib}_{\mathbf{split}}$ preserves products, and so does the 2-functor $\mathrm{dom}\colon \mathsf{Fib}_{\mathbf{split}} \to \mathsf{Cat}$. Thus, so does the composite 2-functor $\int = \mathrm{dom} \circ G$. It thus induces 2-functors

$$\int\colon \mathsf{mICat} \to \mathsf{mCat}$$

$$\int\colon \mathsf{bmICat} \to \mathsf{bmCat}$$

$$\int\colon \mathsf{smICat} \to \mathsf{smCat}.$$

These 2-functors match those given in Thms. 6.8, 6.15, and 6.18, at least after replacing C with C^{op}.

The reader conversant with fibrations may wonder why we are restricting attention to *split* indexed categories and *split* fibrations. Only the split case seems relevant to network models. However, everything we have just done also works for general indexed categories and fibrations. This is shown by Vasilakopolou and the second author in a paper that develops the 2-categorical approach sketched here [20].

7. Operads from network models

Next we describe the operad associated to a network model. There is a standard method of constructing an untyped operad from an object x in a strict symmetric monoidal category C. Namely, we define the set of n-ary operations to be $\hom_{\mathsf{C}}(x^{\otimes n}, x)$, and compose these operations using composition in C. This gives the so-called **endomorphism operad** of x. Here we use the generalization of this idea to the typed case, using *all* the objects of C as the types of the operad.

We assume familiarity with typed operads: these are often called 'colored' operads, with the types called 'colors' [23]. In what follows we let $\mathrm{Ob}(\mathsf{C})$ be the set of objects of a small category C.

7.1. Proposition. *If C is a small strict symmetric monoidal category then there is an $\mathrm{Ob}(\mathsf{C})$-typed operad $\mathrm{op}(\mathsf{C})$ for which:*

- *the set of operations $\mathrm{op}(\mathsf{C})(c_1, \ldots, c_k; c)$ is defined to be $\hom_{\mathsf{C}}(c_1 \otimes \cdots \otimes c_k, c)$,*
- *given operations*
$$f \in \hom_{\mathsf{C}}(c_1 \otimes \cdots \otimes c_k; c)$$
and
$$g_i \in \hom_{\mathsf{C}}(c_{ij_1} \otimes \cdots \otimes c_{ij_i}, c_i)$$
for $1 \leq i \leq k$, their composite is defined by
$$f \circ (g_1, \ldots, g_k) = f \circ (g_1 \otimes \cdots \otimes g_k). \tag{9}$$
- *identity operations are identity morphisms in C, and*
- *the action of S_k on k-ary operations is defined using the braiding in C.*

Proof. The various axioms of a colored operad can be checked for $\mathrm{op}(\mathsf{C})$ using the corresponding laws in the definition of a strict symmetric monoidal category. The associativity axiom for $\mathrm{op}(\mathsf{C})$ follows from associativity of composition and the functoriality of the tensor product in C. The left and right unit axioms for $\mathrm{op}(\mathsf{C})$ follow from the unit laws for composition and the functoriality of the tensor product in C. The two equivariance axioms for $\mathrm{op}(\mathsf{C})$ follow from the laws governing the braiding in C. ∎

Given a network model $F\colon \mathsf{S}(C) \to \mathsf{Cat}$, we can use the strict symmetric Grothendieck construction of Thm. 6.21 to define a strict symmetric monoidal category $\int F$. We can then use Prop. 7.1 to build an operad $\mathrm{op}(\int F)$.

7.2. Definition. *Given a network model $F\colon \mathsf{S}(C) \to \mathsf{Cat}$, define the* **network operad** O_F *to be* $\mathrm{op}(\int F)$.

If $F\colon \mathsf{S}(C) \to \mathsf{Mon}$, the objects of $\int F$ correspond to objects of $\mathsf{S}(C)$, which are formal expressions of the form
$$c_1 \otimes \cdots \otimes c_k$$
with $k \in \mathbb{N}$ and $c_i \in C$. Thus, the network operad O_F is a typed operad where the types are expressions of this form: that is, ordered k-tuples of elements of C.

Now suppose that F is a one-colored network model, so that $F\colon \mathsf{S} \to \mathsf{Mon}$. Then the objects of S are simply natural numbers, so O_F is an $\mathbb{N}$-typed operad. Given $n_1, \ldots, n_k, n \in \mathbb{N}$, we have
$$\mathrm{O}_F(n_1, \ldots, n_k; n) = \hom_{\int F}(n_1 + \cdots + n_k, n).$$

By the definition of the Grothendieck construction, a morphism in this homset is a pair consisting of a bijection $\sigma\colon n_1 + \cdots + n_k \to n$ and an element of the monoid $F(n)$. So, we have

$$\mathrm{O}_F(n_1, \ldots, n_k; n) = \begin{cases} S_n \times F(n) & \text{if } n_1 + \cdots n_k = n \\ \emptyset & \text{otherwise.} \end{cases} \tag{10}$$

Here is the basic example:

7.3. Example. [**Simple network operad**] If $\mathrm{SG}\colon \mathsf{S} \to \mathsf{Mon}$ is the network model of simple graphs in Ex. 2.4, we call O_{SG} the **simple network operad**. By Eq. 10, an operation in $\mathrm{O}_{\mathrm{SG}}(n_1, \ldots, n_k; k)$ is an element of S_n together with a simple graph having $\mathbf{n} = \{1, \ldots, n\}$ as its set of vertices.

The operads coming from other one-colored network models work similarly. For example, if $\mathrm{DG}\colon \mathsf{S} \to \mathsf{Mon}$ is the network model of directed graphs from Ex. 2.5, then an operation in $\mathrm{O}_{\mathrm{SG}}(n_1, \ldots, n_k; n)$ is an element of S_n together with a directed graph having $\mathbf{n}$ as its set of vertices.

In Thm. 2.3 we gave a pedestrian description of one-colored network models. We can describe the corresponding network operads in the same style:

7.4. Theorem. *Suppose F is a one-colored network model. Then the network operad O_F is the $\mathbb{N}$-typed operad for which the following hold:*

1. *The sets of operations are given by*

$$\mathrm{O}_F(n_1, \ldots, n_k; n) = \begin{cases} S_n \times F(n) & \textit{if } n_1 + \cdots n_k = n \\ \emptyset & \textit{otherwise.} \end{cases}$$

2. *Composition of operations is given as follows. Suppose that*

$$(\sigma, g) \in S_n \times F(n) = \mathrm{O}_F(n_1, \ldots, n_k; n)$$

and for $1 \leq i \leq k$ we have

$$(\tau_i, h_i) \in S_{n_i} \times F(n_i) = \mathrm{O}_F(n_{i1}, \ldots, n_{ij_i}; n_i).$$

Then

$$(\sigma, g) \circ ((\tau_1, h_1), \ldots, (\tau_k, h_k)) = (\sigma(\tau_1 + \cdots + \tau_k), g \cup \sigma(h_1 \sqcup \cdots \sqcup h_k))$$

where $+$ is defined in Eq. 1, while $\cup$ and $\sqcup$ are defined in Thm. 2.3.

3. *The identity operation in $\mathrm{O}_F(n; n)$ is $(1, e_n)$, where 1 is the identity in S_n and e_n is the identity in the monoid $F(n)$.*

4. *The right action of the symmetric group S_k on $\mathrm{O}_F(n_1, \ldots, n_k; n)$ is given as follows. Given $(\sigma, g) \in \mathrm{O}_F(n_1, \ldots, n_k; n)$ and $\tau \in S_k$, we have*

$$(\sigma, g)\tau = (\sigma\tau, g).$$

PROOF. To prove these we apply Prop. 7.1, which describes the operad $\mathrm{op}(\mathsf{C})$ coming from a strict symmetric monoidal category C, to the case $\mathsf{C} = \int F$. Item 1 is simply Eq. 10. To prove item 2 we first use Eq. 9, which defines composition of operations in $\mathrm{op}(\mathsf{C})$ in terms of composition and tensoring of morphisms in C. Then we use Eq. 2, which says how to compose morphisms in $\int F$, and Eq. 3, which says how to tensor them. Item 3 comes from how identity operations in $\mathrm{op}(C)$ and identity morphisms in $\int F$ are defined. Similarly, item 4 comes from how the symmetric group actions in $\mathrm{op}(C)$ and the braiding in $\int F$ are defined. ∎

The construction of operads from symmetric monoidal categories described in Prop. 7.1 is functorial, so the construction of operads from network models is as well. To discuss this functoriality we need a couple of categories. The first is ssmCat, defined in Def. 6.19. Second:

7.5. DEFINITION. *Let* $\mathbf{Op}$ *be the category with typed operads as objects and with a morphism from the* T*-typed operad* O *to the* T'*-typed operad* O' *being a function* $F\colon T \to T'$ *together with maps*

$$F\colon O(t_1, \ldots, t_n; t) \to O'(F(t_1), \ldots, F(t_n); F(t))$$

preserving the composition of operations, identity operations and the symmetric group actions.

7.6. PROPOSITION. *There exists a unique functor* $\mathrm{op}\colon \mathsf{ssmCat} \to \mathbf{Op}$ *defined on objects as in Prop. 7.1 and sending any strict symmetric monoidal functor* $F\colon \mathsf{C} \to \mathsf{C}'$ *to the operad morphism* $\mathrm{op}(F)\colon \mathrm{op}(\mathsf{C}) \to \mathrm{op}(\mathsf{C}')$ *that acts by* F *on types and also on operations:*

$$\mathrm{op}(F) = F\colon \hom_C(c_1 \otimes \cdots \otimes c_n, c) \to \hom_{\mathsf{C}'}(F(c_1) \otimes \cdots \otimes F(c_n), F(c)).$$

PROOF. This is a straightforward verification. ∎

7.7. THEOREM. *There exists a unique functor*

$$\mathrm{O}\colon \mathsf{NetMod} \to \mathbf{Op}$$

sending any network model $F\colon \mathsf{S}(C) \to \mathsf{Cat}$ *to the operad* $\mathrm{O}_F = \mathrm{op}(\int G)$ *and any morphism of network models* $(G, \mathfrak{g})\colon (C, F) \to (C', F'G')$ *to the morphism of operads* $\mathrm{O}_G = \mathrm{op}(\widehat{G})$.

PROOF. There is a functor

$$\textstyle\int\colon \mathsf{NetMod} \to \mathsf{ssmCat}$$

given by restricting the strict symmetric monoidal Grothendieck construction of Thm. 6.21 to NetMod. Composing this with the functor

$$\mathrm{op}\colon \mathsf{ssmCat} \to \mathbf{Op}$$

constructed in Prop. 7.6 we obtain a functor $\mathrm{O}\colon \mathsf{NetMod} \to \mathbf{Op}$ with the properties stated in the theorem. Since these properties specify how O acts on objects and morphisms, it is unique. ∎

8. Algebras of network operads

Our interest in network operads comes from their use in designing and tasking networks of mobile agents. The operations in a network operad are ways of assembling larger networks of a given kind from smaller ones. To describe how these operations act in a concrete situation we need to specify an algebra of the operad. The flexibility of this approach to system design takes advantage of the fact that a single operad can have many different algebras, related by homomorphisms. We have already discussed these ideas elsewhere [2, 3], and plan to write a more detailed treatment, so here we simply describe a few interesting algebras of network operads.

Recall from the introduction that an algebra A of a typed operad O specifies a set $A(t)$ for each type $t \in T$ such that the operations of O can be applied to act on these sets. That is, each algebra A specifies:

- for each type $t \in T$, a set $A(t)$, and
- for any types $t_1, \ldots, t_n, t \in T$, a function

$$\alpha \colon O(t_1, \ldots, t_n; t) \to \hom(A(t_1) \times \cdots \times A(t_n), A(t))$$

obeying some rules that generalize those for the action of a monoid on a set [23]. All the examples in this section are algebras of network operads constructed from one-colored network models $F \colon \mathsf{S} \to \mathsf{Mon}$. This allows us to use Thm. 7.4, which describes O_F explicitly.

The most basic algebra of such a network operad O_F is its 'canonical algebra', where it acts on the kind of network described by the network model F:

8.1. Example. [**The canonical algebra**] Let $F \colon \mathsf{S} \to \mathsf{Mon}$ be a one-colored network model. Then the operad O_F has a **canonical algebra** A_F with

$$A_F(n) = F(n)$$

for each $n \in N$, the type set of O_F. In this algebra any operation

$$(\sigma, g) \in \mathrm{O}_F(n_1, \ldots, n_k; n) = S_n \times F(n)$$

acts on a k-tuple of elements

$$h_i \in A_F(n_i) = F(n_i) \qquad (1 \leq i \leq k)$$

to give

$$\alpha(\sigma, g)(h_1, \ldots, h_k) = g \cup \sigma(h_1 \sqcup \cdots \sqcup h_k) \in A(n).$$

Here we use Thm. 2.3, which gives us the ability to overlay networks using the monoid structure $\cup \colon F(n) \times F(n) \to F(n)$, take their 'disjoint union' using maps $\sqcup \colon F(m) \times$

$F(m') \to F(m+m')$, and act on $F(n)$ by elements of S_n. Using the equations listed in this theorem one can check that α obeys the axioms of an operad algebra.

When we want to work with networks that have more properties than those captured by a given network model, we can equip elements of the canonical algebra with extra attributes. Three typical kinds of network attributes are vertex attributes, edge attributes, and 'global network' attributes. For our present purposes, we focus on vertex attributes. Vertex attributes can capture internal properties (or states) of agents in a network such as their locations, capabilities, performance characteristics, etc.

8.2. Example. [**Independent vertex attributes**] For any one-colored network model $F\colon \mathsf{S} \to \mathsf{Mon}$ and any set X, we can form an algebra A_X of the operad O_F that consists of networks whose vertices have attributes taking values in X. To do this, we define

$$A_X(n) = F(n) \times X^n.$$

In this algebra, any operation

$$(\sigma, g) \in \mathrm{O}_F(n_1, \ldots, n_k; n) = S_n \times F(n)$$

acts on a k-tuple of elements

$$(h_i, x_i) \in F(n_i) \times X^{n_i} \qquad (1 \leq i \leq k)$$

to give

$$\alpha_X(\sigma, g) = (g \cup \sigma(h_1 \sqcup \cdots \sqcup h_k), \sigma(x_1, \ldots, x_k)).$$

Here $(x_1, \ldots, x_k) \in X^n$ is defined using the canonical bijection

$$X^n \cong \prod_{i=1}^{k} X^{n_i}$$

when $n_1 + \cdots + n_k = n$, and $\sigma \in S_n$ acts on X^n by permutation of coordinates. In other words, α_X acts via α on the $F(n_i)$ factors while permuting the vertex attributes X^n in the same way that the vertices of the network $h_1 \sqcup \cdots \sqcup h_k$ are permuted.

One can easily check that the projections $F(n) \times X^n \to F(n)$ define a homomorphism of O_F-algebras, which we call

$$\pi_X\colon A_X \to A.$$

This homomorphism 'forgets the vertex attributes' taking values in the set X.

8.3. Example. [**Simple networks with a rule obeyed by edges**] Let O_{SG} be the simple network operad as explained in Ex. 7.3. We can form an algebra of the operad O_{SG} that consists of simple graphs whose vertices have attributes taking values in some set X, but where an edge is permitted between two vertices only if their attributes obey some condition. We specify this condition using a symmetric function

$$p\colon X \times X \to \mathbb{B}$$

where $\mathbb{B} = \{F, T\}$. An edge is not permitted between vertices with attributes $(x_1, x_2) \in X \times X$ if this function evaluates to F.

To define this algebra, which we call A_p, we let $A_p(n) \subseteq \mathrm{SG}(n) \times X^n$ be the set of pairs (g, x) such that for all edges $\{i, j\} \in g$ the attributes of the vertices i and j make p true:

$$p(x(i), x(j)) = T.$$

There is a function

$$\tau_p \colon A_X(n) \to A_p(n)$$

that discards edges $\{i, j\}$ for which $p(x(i), x(j)) = F$. Recall that $A_X(n) = \mathrm{SG}(n) \times X^n$, and recall from Ex. 3.2 that we can regard $\mathrm{SG}(n)$ as the set of functions $g \colon \mathcal{E}(n) \to \mathbb{B}$. Then we define τ_p by

$$\tau_p(g, x) = (\overline{g}, x)$$

where

$$\overline{g}\{i, j\} = \begin{cases} g\{i, j\} & \text{if} \quad p(x(i), x(j)) = T \\ \\ F & \text{if} \quad p(x(i), x(j)) = F. \end{cases}$$

We can define an action α_p of O_{SG} on the sets $A_p(n)$ with the help of this function. Namely, we take α_p to be the composite

$$\begin{array}{c} \mathrm{O}_{\mathrm{SG}}(n_1, \ldots, n_k; n) \times A_p(n_1) \times \cdots \times A_p(n_k) \\ \big\downarrow \\ \mathrm{O}_{\mathrm{SG}}(n_1, \ldots, n_k; n) \times A_X(n_1) \times \cdots \times A_X(n_k) \\ \big\downarrow{\scriptstyle \alpha_X} \\ A_X(n) \\ \big\downarrow{\scriptstyle \tau_p} \\ A_p(n) \end{array}$$

where the action α_X was defined in Ex. 8.2. One can check that α_p makes the sets $A_p(n)$ into an algebra of O_{SG}, which we call A_p. One can further check that the maps τ define a homomorphism of O_{SG}-algebras, which we call

$$\tau_p \colon A_X \to A_p.$$

8.4. Example. [**Range-limited networks**] We can use the previous examples to model range-limited communications between entities in a plane. First, let $X = \mathbb{R}^2$ and form the algebra A_X of the simple network operad O_{SG}. Elements of $A_X(n)$ are simple graphs with vertices in the plane.

Then, choose a real number $L \geq 0$ and let d be the usual Euclidean distance function on the plane. Define $p \colon X \times X \to \mathbb{B}$ by setting $p(x, y) = T$ if $d(x, y) \leq L$ and $p(x, y) = F$ otherwise. Elements of $A_p(n)$ are simple graphs with vertices in the plane such that no edge has length greater than L.

8.5. Example. [**Networks with edge count limits**] Recall the network model for multigraphs MG^+, defined in Ex. 2.6 and clarified in Ex. 3.3. An element of $\mathrm{MG}^+(n)$ is a multigraph on the set $\mathbf{n}$, namely a function $g\colon \mathcal{E}(n) \to \mathbb{N}$ where $\mathcal{E}(n)$ is the set of 2-element subsets of $\mathbf{n}$. If we fix a set X we obtain an algebra A_X of $\mathrm{O}_{\mathrm{MG}^+}$ as in Ex. 8.2. The set $A_X(n)$ consists of multigraphs on $\mathbf{n}$ where the vertices have attributes taking values in X.

Starting from A_X we can form another algebra where there is an upper bound on how many edges are allowed between two vertices, depending on their attributes. We specify this upper bound using a symmetric function

$$b\colon X \times X \to \mathbb{N}.$$

To define this algebra, which we call A_b, let $A_b(n) \subseteq \mathrm{MG}^+(n) \times X^n$ be the set of pairs (g, x) such that for all $\{i, j\} \in \mathcal{E}(n)$ we have

$$g(i, j) \leq b(x(i), x(j)).$$

Much as in Ex. 8.3 there is function

$$\pi\colon A_X(n) \to A_b(n)$$

that enforces this upper bound: for each $g \in A_X(n)$ its image $\pi(g)$ is obtained by reducing the number of edges between vertices i and j to the minimum of $g(i, j)$ and $\beta(i, j)$:

$$\pi(g)(i, j) = g(i, j) \min \beta(i, j).$$

We can define an action α_b of O_{MG} on the sets $A_b(n)$ as follows:

$$\begin{array}{c} \mathrm{O}_{\mathrm{MG}}(n_1, \ldots, n_k; n) \times A_p(n_1) \times \cdots \times A_p(n_k) \\ \big\downarrow \\ \mathrm{O}_{\mathrm{MG}}(n_1, \ldots, n_k; n) \times A_X(n_1) \times \cdots \times A_X(n_k) \\ \big\downarrow{\scriptstyle \alpha_X} \\ A_X(n) \\ \big\downarrow{\scriptstyle \pi} \\ A_p(n) \end{array}$$

One can check that α_b indeed makes the sets $A_b(n)$ into an algebra of $\mathrm{O}_{\mathrm{MG}^+}$, which we call A_b, and that the maps π_p define a homomorphism of $\mathrm{O}_{\mathrm{MG}^+}$-algebras, which we call

$$\pi_p\colon A_X \to A_b.$$

8.6. Example. [**Range-limited networks, revisited**] We can use Ex. 8.5 to model entities in the plane that have two types of communication channel, one of which has range L_1 and another of which has a lesser range $L_2 < L_1$. To do this, take $X = \mathbb{R}^2$ and define $b\colon X \times X \to \mathbb{N}$ by

$$b(x,y) = \begin{cases} 0 & \text{if } d(x,y) > L_1 \\ 1 & \text{if } L_2 < d(x,y) \leq L_1 \\ 2 & \text{if } d(x,y) \leq L_2 \end{cases}$$

Elements of $A_b(n)$ are multigraphs with vertices in the plane having no edges between vertices whose distance is $> L_1$, at most one edge between vertices whose distance is $\leq L_1$ but $> L_2$, and at most two edges between vertices whose distance is $\leq L_2$.

Moreover, the attentive reader may notice that the action α_b of $\mathrm{O}_{\mathrm{MG}^+}$ for this specific choice of b factors through an action of $\mathrm{O}_{\Gamma_{\mathbb{B}_2}}$, where $\Gamma_{\mathbb{B}_2}$ is the network model defined in Ex. 3.5. That is, operations $\mathrm{O}_{\Gamma_{\mathbb{B}_2}}(n_1, \ldots, n_k; n) = S_n \times \Gamma_{\mathbb{B}_2}(n)$ where $\Gamma_{\mathbb{B}_2}(n)$ is the set of multigraphs on $\mathbf{n}$ *with at most* 2 *edges between vertices* are sufficient to compose these range-limited networks. This is due to the fact that the values of this $b\colon X \times X \to \mathbb{N}$ are at most 2.

These examples indicate that vertex attributes and constraints can be systematically added to the canonical algebra to build more interesting algebras, which are related by homomorphisms. Ex. 8.2 illustrates how adding extra attributes to the networks in some algebra A can give networks that are elements of an algebra A' equipped with a homomorphism $\pi\colon A' \to A$ that forgets these extra attributes. Ex. 8.5 illustrates how imposing extra constraints on the networks in some algebra A can give an algebra A' equipped with a homomorphism $\tau\colon A \to A'$ that imposes these constraints: this works only if there is a well-behaved systematic procedure, defined by τ, for imposing the constraints on any element of A to get an element of A'.

The examples given so far scarcely begin to illustrate the rich possibilities of network operads and their algebras. Their connection to Petri nets is explored in [4], but there is much more to do.

In particular, it is worth noting that all the specific examples of network models described here involve commutative monoids. However, noncommutative monoids are also important. Suppose, for example, that we wish to model entities with a limited number of point-to-point communication interfaces—e.g., devices with a finite number p of USB ports. More formally, we wish to act on sets of degree-limited networks $A_{\deg}(n) \subset \mathrm{SG}(n) \times \mathbb{N}^n$ made up of pairs (g, p) such that the degree of each vertex i, $\deg(i)$, is at most the degree-limiting attribute of i: $\deg(i) \leq p(i)$. Naïvely, we might attempt to construct a map $\tau_{\deg}\colon A_{\mathbb{N}} \to A_{\deg}$ as in Ex. 8.5 to obtain an action of the simple network operad O_{SG}. However, this is turns out to be impossible. For example, if attempt to build a network from devices with a single USB port, and we attempt to connect multiple USB cables to one of these devices, the relevant network operad must include a rule saying which attempts, if any, are successful. Since we cannot prioritize links from some vertices over others—which would break the symmetry built into any network model—the

order in which these attempts are made must be relevant. Since the monoids $\mathrm{SG}(n)$ are commutative, they cannot capture this feature of the situation.

The solution is to use a class of noncommutative monoids dubbed 'graphic monoids' by Lawvere [16]: namely, those that obey the identity $aba = ab$. These allow us to construct a one-colored network model $\Gamma\colon \mathsf{S} \to \mathsf{Mon}$ whose network operad O_Γ acts on A_{deg}. For our USB device example, the relation $aba = ab$ means that first attempting to connect some USB cables between some devices (a), second attempting to connect some further USB cables (b), and third attempting to connect some USB cables *precisely as attempted in the first step* (a, again) has the same result as only performing the first two steps (ab).

In fact, one-colored network models constructed from graphic monoids appear to be sufficient to model a wide array of constraints on the structural design and behavioral tasking of agents. For more on network models arising from noncommutative monoids, see [19].

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, Tom Mifflin, John Paschkewitz, Thy Tran and Didier Vergamini for many helpful discussions. Christina Vasilakopoulou provided crucial assistance with Sec. 6.22. JB also thanks the Centre for Quantum Technologies, where some of this work was done.

References

[1] J. C. Baez and B. S. Pollard, A compositional framework for reaction networks, *Rev. Math. Phys.* **29**, 1750028. Available as arXiv:1704.02051.

[2] J. C. Baez, J. D. Foley, J. Moeller and B. S. Pollard, Operads for communication networks, 2016, 37 pages, Technical report, DARPA CASCADE project.

[3] J. C. Baez, J. D. Foley, J. Moeller and B. S. Pollard, Compositional tasking, 2017, 27 pages, Technical report, DARPA CASCADE project.

[4] J. C. Baez, J. Foley and J. Moeller, Network models from Petri nets with catalysts, *Compositionality* **1**, 4 (2019). Available as arXiv:1904.03550.

[5] F. Bergeron, G. Labelle, and P. Leroux, *Combinatorial Species and Tree-like Structures*, Cambridge, Cambridge Univ. Press, 1998.

[6] F. Borceux, *Handbook of Categorical Algebra*, Cambridge Univ. Press, Cambridge, 1994.

[7] B. Day and R. Street, Monoidal bicategories and Hopf algebroids, *Adv. Math.* **129** (1997), 99–157.

[8] N. Gambino and A. Joyal, On operads, bimodules and analytic functors, *Memoirs AMS* **249**, 2017. Available as arXiv:1405.7270

[9] A. Grothendieck, Catégories fibrées et descente, in *SGA1: Revêtements étales et groupe fondamental*, Lecture Notes in Mathematics **224**, Springer, Berlin, pp. 145–194. Available as arXiv:math/0206203.

[10] C. Hermida, Some properties of Fib as a fibred 2-category, *J. Pure Appl. Algebra*, **134** (1999), 83–109.

[11] M. Hyland and A. J. Power, Pseudo-commutative monads and pseudo-closed 2-categories, *Jour. Pure Appl. Alg.* **175** (2002), 141–185.

[12] B. Jacobs, *Categorical Logic and Type Theory*, Elsevier, Amderstam, 1999.

[13] P. Johnstone, *Sketches of an Elephant: A Topos Theory Compendium*, Vol. 1, Oxford Univ. Press, Oxford, 2002.

[14] A. Joyal, Une théorie combinatoire des séries formelles, *Adv. Math.* **42** (1981), 1–82.

[15] A. Joyal, Foncteurs analytiques et espèces des structures, in Lecture Notes in Mathematics **1234**, Springer, Berlin, 1986, pp. 126–159.

[16] F. W. Lawvere, Qualitative distinctions between some toposes of generalized graphs, *Contemp. Math.* **92** (1989), 261–299.

[17] S. Mac Lane, *Categories for the Working Mathematician*, 2nd edition, Springer, Berlin, 1998.

[18] P. McCrudden, Balanced coalgebroids, *Th. Appl. Cat.* **7** (2000), 71–147.

[19] J. Moeller, Noncommutative network models, *Math. Struct. Comp. Sci.* **30** (2020), 14–32. Available as arXiv:1804.07402.

[20] J. Moeller and C. Vasilakopoulou, Monoidal Grothendieck construction, *Theor. Appl. Cat.*, to appear. Available as arXiv:1809.00727.

[21] V. Sassone, *Strong Concatenable Processes: an Approach to the Category of Petri Net Computations*, *BRICS Report Series* **33** (1994).

[22] C. Vasilakopoulou, *Generalization of Algebraic Operations via Enrichment*, Chap. I.5: Fibrations and Cofibrations, Ph.D. Thesis, University of Cambridge, 2014. Available as arXiv:1411.3038.

[23] D. Yau, *Colored Operads*, American Mathematical Society, Providence, Rhode Island, 2016.

Department of Mathematics, University of California, Riverside CA, 92521, USA
Centre for Quantum Technologies, National University of Singapore, 117543, Singapore
Metron, Inc., 1818 Library St., Suite 600, Reston, VA 20190, USA
NIST, Mail Stop 8970, 100 Bureau Drive, Gaithersburg, MD 20899, USA
Email: `baez@math.ucr.edu`
`foley@metsci.com`
`moeller@math.ucr.edu`
`blake.pollard@nist.gov`

This article may be accessed at `http://www.tac.mta.ca/tac/`

THEORY AND APPLICATIONS OF CATEGORIES will disseminate articles that significantly advance the study of categorical algebra or methods, or that make significant new contributions to mathematical science using categorical methods. The scope of the journal includes: all areas of pure category theory, including higher dimensional categories; applications of category theory to algebra, geometry and topology and other areas of mathematics; applications of category theory to computer science, physics and other mathematical sciences; contributions to scientific knowledge that make use of categorical methods.
Articles appearing in the journal have been carefully and critically refereed under the responsibility of members of the Editorial Board. Only papers judged to be both significant and excellent are accepted for publication.

SUBSCRIPTION INFORMATION Individual subscribers receive abstracts of articles by e-mail as they are published. To subscribe, send e-mail to `tac@mta.ca` including a full name and postal address. Full text of the journal is freely available at `http://www.tac.mta.ca/tac/`.

INFORMATION FOR AUTHORS LaTeX2e is required. Articles may be submitted in PDF by email directly to a Transmitting Editor following the author instructions at `http://www.tac.mta.ca/tac/authinfo.html`.