Wiring diagrams as normal forms for computing in symmetric monoidal categories

Evan Patterson David I. Spivak Dmitry Vagner

Applications of category theory often involve symmetric monoidal categories (SMCs), in which abstract processes or operations can be composed in series and parallel. However, in 2020 there remains a dearth of computational tools for working with SMCs. We present an "unbiased" approach to implementing symmetric monoidal categories, based on an operad of directed, acyclic wiring diagrams. Because the interchange law and other laws of a SMC hold identically in a wiring diagram, no rewrite rules are needed to compare diagrams. We discuss the mathematics of the operad of wiring diagrams, as well as its implementation in the software package Catlab.

1 Introduction

The syntax for an algebraic structure is often derived from its traditional axiomatization, without additional thought. A symmetric monoidal category (SMC) is defined through operations of composition, identity, monoidal product, monoidal unit, and braiding, subject to various laws. Once it is decided how to assign symbols to these operations, such as $\fatsemi$ for composition and $\otimes$ for the monoidal product, a symbolic syntax for constructing objects and morphisms follows immediately. So, given morphisms, say $f\colon x \to x \otimes y$ and $g\colon y \otimes z \to z$, a new morphism can be constructed via such expressions as

$$(f \otimes \mathrm{id}_z) \fatsemi (\mathrm{id}_x \otimes g).$$

Symbolic syntax has a long tradition in algebra. Its utility derives, on the one hand, from its ease in writing and typesetting, and on the other, from its immediacy given an axiomatization of an algebraic structure as a generalized algebraic theory.

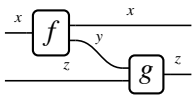
But these are not the only desiderata for mathematical syntax. In general, the mathematical objects denoted by two different expressions may be equal under the axioms. A good syntax narrows the gap between a mathematical object and its representation by avoiding redundancy. For example, since monoidal products are associative in a strict SMC, the expressions $f \otimes (g \otimes h)$ and $(f \otimes g) \otimes h$ denote the same morphism; thus, it is standard practice to eliminate parentheses around the monoidal product, writing simply $f \otimes g \otimes h$.

Encoding algebraic equations into a simplified yet unambiguous syntax has important cognitive and computational benefits. For humans, it substitutes visual inspection for equational reasoning, playing to our cognitive strengths. For computers, it reduces possibly complex algorithms for checking *equality* to a simple test of *identity* on a suitable data structure. From this perspective, the ideal syntax provides a *normal form*, making two expressions identical if and only if they denote equal mathematical objects.

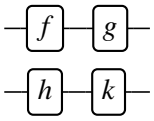
What is the right syntax for symmetric monoidal categories? Beginning with the Penrose graphical notation for tensors [Pen71], it was gradually understood that morphisms in a monoidal category are best depicted by a *two-dimensional* syntax, with one axis representing composition and the other representing

David I. Spivak and Jamie Vicary (Eds.): Applied Category Theory 2020 (ACT2020)
EPTCS ??, 2020, pp. 1–16, doi:10.4204/EPTCS.??.??

monoidal product. For example, the above expression $(f \otimes \mathrm{id}_z) \fatsemi (\mathrm{id}_x \otimes g)$ becomes the *string diagram*



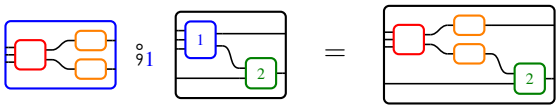
As string diagrams, both sides of the *interchange law* $(f \fatsemi g) \otimes (h \fatsemi k) = (f \otimes h) \fatsemi (g \otimes k)$ of a monoidal category have the same representation, namely:



String diagrams were first put on a rigorous footing by Joyal and Street, who showed that the diagrammatic language is sound and complete for the equations between morphisms deducible from the axioms of a strict symmetric monoidal category [JS91]. Diagrammatic languages are now known for other kinds of monoidal categories, such as traced monoidal categories and hypergraph categories [Sel10; FS19a].

For Joyal and Street, string diagrams are geometric figures in the plane or in a higher-dimensional Euclidean space. This perspective, while intuitively appealing, is of little computational use, since geometric objects are not readily translated into data structures. For computational purposes, we would like to extract the combinatorial data defining a string diagram, much like a graph does for a graph embedding or graph drawing.

Wiring diagrams were introduced in [RS13]. Though they are often depicted graphically, wiring diagrams are combinatorial, rather than geometric, objects. They also differ from string diagrams in that they deal only with the syntax of composition, and do not include explicit morphisms from a given SMC. More precisely, wiring diagrams—say for representing compositions in symmetric monoidal categories—are organized as the morphisms of a typed operad $\mathscr{W}$. Composition of morphisms corresponds to nesting of wiring diagrams, for example:



(1)

Note that in the symbolic syntax of a symmetric monoidal category, there are infinitely-many ways to represent any of the wiring diagrams in (1), for example the middle one could be represented by tensoring with an arbitrary number of monoidal units, but the wiring diagram represents exactly one element of the set $\mathscr{W}(X_1, X_2; Y)$, where X_1 and X_2 are the inner boxes and Y is the outer box.

A given model of $\mathscr{W}$, which we can think of as a symmetric monoidal category $\mathscr{C}$, is represented as a functor $H \colon \mathscr{W} \to \mathsf{Set}$, which we refer to as a $\mathscr{W}$-algebra.[1] Each box (object in $\mathscr{W}$) is sent by H to the set of all $\mathscr{C}$-morphisms of that shape, and each wiring diagram (morphism in $\mathscr{W}$) is sent by H to the function that takes morphisms in $\mathscr{C}$ and composes them accordingly.

Operads of wiring diagrams can be used to model various sorts of categories, such as traced or hypergraph categories [SSR17; FS19a]—whose string diagram languages we mentioned above—or even just

[1] For now, we elide the fact that wires should be annotated by types corresponding to the objects of $\mathscr{C}$.

ordinary categories. Here we will focus on the operad $\mathscr{W}$ of wiring diagrams for symmetric monoidal categories.

The aim of this paper is to show how $\mathscr{W}$ can serve as a foundation for the computer algebra of symmetric monoidal categories, both practically and theoretically. Since the interchange law and other axioms of an SMC hold identically within $\mathscr{W}$, wiring diagrams are nearly normal forms for morphisms in a free SMC, as will be explained. With this motivation, Section 2 describes an implementation of the operad of wiring diagrams in Catlab.jl, a Julia package for applied category theory [Pc20]. The operad structure is taken as fundamental in Catlab and used to implement a diagrammatic syntax for symmetric monoidal categories. The wiring diagram operad is then extended to SMCs with extra structure, such as cartesian monoidal categories, biproduct categories, and traced monoidal categories. In contrast to systems like Quantomatic [KZ15] and Cartographer [SWZ19], Catlab is not a graphical editor or a proof assistant; rather, it provides data structures and algorithms for computing with in symmetrical monoidal categories for scientific and engineering applications.

The remainder of the paper is about the mathematics of the operad of wiring diagrams. After some preliminaries on biproducts and spans in Section 3, an operad of wiring diagrams is constructed in Section 4. The directed wiring diagrams are defined to satisfy an acyclicity condition, making them suitable for symmetric monoidal categories. They are built using the categorical matrix calculus, which extends the notion of an adjacency matrix of a directed graph. Finally, Section 5 shows how algebras on the wiring diagram operad give rise to symmetric monoidal categories and, conversely, how symmetric monoidal categories yield algebras on this operad. These two constructions are not inverse equivalences; however, beginning with an SMC, the roundtrip does return an equivalent SMC. We will make the correspondence more precise in Theorems 5.2 and 5.3.

2 Wiring diagrams in Catlab

In the Julia package Catlab, wiring diagrams are implemented as combinatorial data structures. Wiring diagrams are akin to directed graphs but possess extra structure; namely each box—which plays the role of a node—has an explicit set of input and output ports. Every wiring diagram has an underlying directed graph, obtained by forgetting this extra structure. For implementation purposes, Catlab exploits this hierarchy by building its data structure for wiring diagrams on top of graph data structures that already exist in the Julia ecosystem.

The functionality for wiring diagrams is implemented in several layers. The bottom layer is the core data structure and a low-level imperative interface for mutating it. Operadic composition—or substitution—of wiring diagrams is defined using this interface. Finally, the operadic interface is used to define a syntax for morphisms in symmetric monoidal categories. The three layers are summarized in Sections 2.1, 2.2, and 2.3 respectively.

2.1 The wiring diagram data structure

The data structure for wiring diagrams is comprised of several data types. From the user's perspective, these types are:

1. `WiringDiagram`: a quadruple consisting of (i) a list of input port types and (ii) a list of output port types, both for the outer box; (iii) a list of boxes, where the first and second entries are special

values representing the input and output types of the outer box and the remaining entries are boxes inside the diagram, of type `Box`; and (iv) a set of wires, of type `Wire`.

2. `Box`: a triple consisting of (i) a label or value for the box, (ii) a list of input port types, (iii) and a list of output port types. Thus, a morphism $f : x \otimes y \to z$ would be represented as `Box(:f, [:x,:y], [:z])`, where the Julia syntax `:x` denotes a symbol named "x".
3. `Wire`: a source-target pair, where both the source and target are pairs of numbers identifying a box in the diagram and an input or output port on that box. So, a wire from output port p on box i to input port q on box j is `Wire((i,p) => (j,q))`.

As a small but complete example, the wiring diagram corresponding to the composite $f \fatsemi g : x \to z$ of morphisms $f : x \to y$ and $g : y \to z$ is implemented as:

```
WiringDiagram([:x], [:z],
  [ 1 => {inputs}, 2 => {outputs}, 3 => Box(:f, [:x], [:y]), 4 => Box(:g, [:y], [:z]) ],
  [ Wire((1,1) => (3,1)), Wire((3,1) => (4,1)), Wire((4,1) => (2,1)) ])
```

For performance reasons, the wires in a wiring diagram are not actually stored as a set. Instead, underlying each wiring diagram is a simple directed graph, as implemented by the Julia package LightGraphs.jl [BFc17]. The vertices in the graph are numbered consecutively from 1 to $n+2$, where n is the number of boxes in the wiring diagram. Vertices 1 and 2, labelled $\{$`inputs`$\}$ and $\{$`outputs`$\}$ above, refer to the inputs and outputs of the outer box, respectively; the remaining vertices refer to boxes inside the diagram. There is an edge between two vertices if and only if there is at least one wire between the corresponding boxes.

Traversals on the wiring diagram are then delegated to the underlying directed graph. For example, to find the in-neighbors or out-neighbors of a box, one simply finds the in-neighbors or out-neighbors of the corresponding vertex in the underlying graph. The graph data structure is designed to do this efficiently. Wiring diagrams can be accessed and mutated through a simple imperative interface. Boxes, ports, and wires can be retrieved individually or iterated over, and boxes and wires can be added and removed from an existing diagram.

2.2 Implementing the operad of wiring diagrams

Operadic composition of wiring diagrams as in (1) is supported through the function `ocompose`, with two signatures:

```
ocompose(f::WiringDiagram, gs::Vector{<:WiringDiagram})::WiringDiagram
ocompose(f::WiringDiagram, i::Int, g::WiringDiagram)::WiringDiagram
```

The first signature corresponds to full (May-style) operadic composition $\circ$ and the second corresponds to partial (Markl-style) operadic composition $\circ_i$. Both methods are one-line wrappers around the procedure `substitute`, which substitutes wiring diagrams for one or more boxes in another wiring diagram. Mathematically, `substitute` simultaneously performs one or more non-overlapping partial operadic compositions.

Substitution of wiring diagrams is implemented using the imperative interface. In outline, the algorithm proceeds as follows:

1. Create a copy `d` of the original wiring diagram.
2. Add to `d` all the boxes from all the diagrams to be substituted.

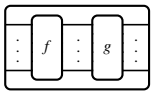
3. Extend each wire in each substituted diagram to new a wire in `d`. This subroutine branches into four different cases, shown in pseudo-Julia code in Listing 1.
4. Remove from `d` all the boxes from the original diagram that were to be substituted, which in turn removes any extraneous wires created during step 3.

```
function substitute_wires!(d::WiringDiagram, v::Int, sub::WiringDiagram)
  for wire in wires(sub)
    # Case 1: Passing wire.
    if {wire source and target are on outer box}
      for in_wire in {wires in d incoming to source outer port}
        for out_wire in {wires in d outgoing from target outer port}
          {add wire to d fusing in_wire -> wire -> out_wire}
        end
      end
    # Case 2: Incoming wire.
    elseif {wire source is on outer box}
      for in_wire in {wires in d incoming to source outer port}
        {add wire to d fusing in_wire -> wire}
      end
    # Case 3: Outgoing wire.
    elseif {wire target is on outer box}
      for out_wire in {wires in d outgoing from target outer port}
        {add wire to d fusing wire -> out_wire}
      end
    # Case 4: Fully internal wire.
    else
      {add wire to d}
    end
  end
end
```

Listing 1: Pseudo-Julia code for main subroutine in substitution algorithm. Each wire substituted into the new diagram is either passing, incoming, outgoing, or fully internal. The inner loops under each of the cases are needed because a port may have many or no incident wires, representing copying, merging, deleting, or creating.

2.3 Wiring diagrams as a syntax for SMCs

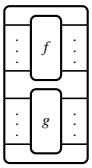
Having implemented the operad of wiring diagrams, it is now straightforward to define an alternate syntax for symmetric monoidal categories using wiring diagrams. That is, we construct a symmetric monoidal category whose morphisms are wiring diagrams in which each box has been filled with a morphism. To compute the series composition $f \fatsemi g$ of morphisms f and g, one forms the following wiring diagram



and then performs an operadic composition. Apart from exception handling and formatting, the following Julia code for the procedure `compose` is identical to the implementation in Catlab.

```
function compose(f::WiringDiagram, g::WiringDiagram)::WiringDiagram
  @assert length(codom(f)) == length(dom(g))
  h = WiringDiagram(dom(f), codom(g))
  fv, gv = add_box!(h, f), add_box!(h, g)
  add_wires!(h, [[ (input_id(h),i) => (fv,i) for i in 1:length(dom(f)) ];
                 [ (fv,i) => (gv,i) for i in 1:length(codom(f)) ];
                 [ (gv,i) => (output_id(h),i) for i in 1:length(codom(g)) ]])
  substitute(h, [fv,gv])
end
```

Similarly, to compute the parallel composition $f \otimes g$ of two morphisms f and g, form the generic diagram with two boxes composed in parallel and then perform an operadic composition:



The wiring diagram syntax provides a normal form for morphisms in a free symmetric monoidal category, up to (simple directed) graph isomorphism. Specifically, two morphisms represented by wiring diagrams are equal if and only if there is an isomorphism of the underlying graphs making the diagrams identical as Julia data structures. Although the graph isomorphism problem is not known to be solvable in polynomial time, in practice it can usually be solved efficiently. Moreover, the labels on the boxes drastically restrict the possible matchings. Wiring diagrams thus constitute an effective normal form for morphisms in a free SMC.

3 Biproducts, matrix calculus, and categories of spans

We will formally present a wiring diagram by encoding all the interconnections between its boxes as a single span. Since the category of spans enjoys a biproduct structure, we can represent wiring diagrams as matrices and their compositions using matrix algebra. We begin with a brief detour into biproduct categories.

3.1 Morphisms as matrices

Let $(\mathscr{C}, \oplus, \mathbf{0})$ be a biproduct category, i.e., a monoidal category for which $\oplus$ is both a product and a coproduct. Such categories can equivalently be seen as monoidal categories with a homomorphic supply of bimonoids [FS19b]. Given a morphism

$$\mathbf{t}_1 \oplus \cdots \oplus \mathbf{t}_m \xrightarrow{f} \mathbf{v}_1 \oplus \cdots \oplus \mathbf{v}_n$$

in such a category, we can extract the *component* ${}_{\mathbf{t}_i} f_{\mathbf{v}_j}$ corresponding to a choice of direct summand $\mathbf{t}_i$ in the domain and $\mathbf{v}_j$ in the codomain, by pre- and post-composing f with canonical inclusions and projections:

$$\mathbf{t}_i \xrightarrow{\iota_{\mathbf{t}_i}} \mathbf{t}_1 \oplus \cdots \oplus \mathbf{t}_n \xrightarrow{f} \mathbf{v}_1 \oplus \cdots \oplus \mathbf{v}_m \xrightarrow{\pi_{\mathbf{v}_j}} \mathbf{v}_j. \tag{2}$$

This procedure defines a map of type $\mathscr{C}\big(\bigoplus_{i:I}\mathbf{t}_i, \bigoplus_{j:J}\mathbf{v}_j\big) \to \mathscr{C}(\mathbf{t}_i, \mathbf{v}_j)$ for finite sets I and J, to which we can define a section by swapping the inclusion and projections in (2). We say that this section is the *embedding* of the component since it produces a morphism for which all other components are the zero morphisms $\mathbf{0} : \mathbf{t}_k \to \mathbf{v}_l$, i.e. the unique morphism of the form $\mathbf{t}_k \to \mathbf{0} \to \mathbf{v}_l$.

Any biproduct category is enriched in commutative monoids, with the sum $f+g$ of two morphisms of type $\mathbf{t} \to \mathbf{v}$ given by $\mathbf{t} \xrightarrow{\Delta} \mathbf{t}\oplus\mathbf{t} \xrightarrow{f\oplus g} \mathbf{v}\oplus\mathbf{v} \xrightarrow{\nabla} \mathbf{v}$. Here Δ and ∇ are the diagonal and codiagonal morphisms arising from the universal property of product and coproduct. By the naturality of these maps, composition distributes over sums. This operation allows us to define the leftward map in the bijection $\mathscr{C}\big(\bigoplus_{i:I}\mathbf{t}_i, \bigoplus_{j:J}\mathbf{v}_j\big) \cong \prod_{(i,j):I\times J}\mathscr{C}(\mathbf{t}_i, \mathbf{v}_j)$ by simply summing across the embeddings of each component. By the universal property of the biproduct, a morphism is fully specified by its components. More formally, given a choice of direct sum decomposition on both domain and codomain, we can represent f as the block matrix

$$f = \begin{bmatrix} \mathbf{t}_1 f_{\mathbf{v}_1} & \cdots & \mathbf{t}_1 f_{\mathbf{v}_n} \\ \vdots & \ddots & \vdots \\ \mathbf{t}_m f_{\mathbf{v}_1} & \cdots & \mathbf{t}_m f_{\mathbf{v}_n} \end{bmatrix}$$

meaning that we can reason about morphisms by simultaneously breaking up both domain and codomain into cases. As an example of this reasoning, we recover matrix multiplication from the composition of two morphisms

$$\mathbf{s}_1 \oplus \cdots \oplus \mathbf{s}_k \xrightarrow{f} \mathbf{t}_1 \oplus \cdots \oplus \mathbf{t}_m \xrightarrow{g} \mathbf{v}_1 \oplus \cdots \oplus \mathbf{v}_n.$$

The ${}_{\mathbf{s}_i}(fg)_{\mathbf{v}_j}$ component is defined as the composite $(\iota_{\mathbf{s}_i} f)(g\pi_{\mathbf{v}_j})$, where $\iota_{\mathbf{s}_i} f$ is given by a sum of morphisms of type $\mathbf{s}_i \to \mathbf{t}_1 \oplus \cdots \oplus \mathbf{t}_m$ and $g\pi_{\mathbf{v}_j}$ is given by a sum of morphisms of type $\mathbf{t}_1 \oplus \cdots \oplus \mathbf{t}_m \to \mathbf{v}_j$:

$$\iota_{\mathbf{s}_i} f = ({}_{\mathbf{s}_i}f_{\mathbf{t}_1})(\iota_{\mathbf{t}_1}) + \cdots + ({}_{\mathbf{s}_i}f_{\mathbf{t}_m})(\iota_{\mathbf{t}_m}) \qquad \text{and} \qquad g\pi_{\mathbf{v}_j} = (\pi_{\mathbf{t}_1})({}_{\mathbf{t}_1}g_{\mathbf{v}_j}) + \cdots + (\pi_{\mathbf{t}_m})({}_{\mathbf{t}_m}g_{\mathbf{v}_j}).$$

Composing these sums, applying distributivity, and then noting that $\iota_{\mathbf{t}_k}\pi_{\mathbf{t}_l} = \delta_{k,l}$ (the Kronecker delta), we arrive at the familiar formula for the component of a matrix multiplication

$${}_{\mathbf{s}_i}(fg)_{\mathbf{v}_j} = \sum_{i,j}({}_{\mathbf{s}_i}f_{\mathbf{t}_k})(\iota_{\mathbf{t}_k}\pi_{\mathbf{t}_l})({}_{\mathbf{t}_l}g_{\mathbf{v}_j}) = \sum_{k}({}_{\mathbf{s}_i}f_{\mathbf{t}_k})({}_{\mathbf{t}_k}g_{\mathbf{v}_j}).$$

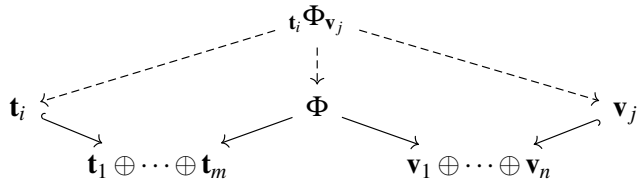
We can narrativize this expression as answering the question "how can we go from $\mathbf{s}_i$ to $\mathbf{v}_j$?" with the response "through any one of the $\mathbf{t}_k$," demonstrating that the sum can be interpreted as a disjunction. The capacity that biproduct categories possess for declaratively formalizing such reasoning motivates us to situate wiring diagrams in a certain biproduct category, which we now proceed to define.

3.2 Spans

Let $\mathscr{C}$ be an extensive category, i.e., one for which the coproduct functor $\mathscr{C}/x \times \mathscr{C}/y \to \mathscr{C}/(x+y)$ is an equivalence of categories. For example, $\mathscr{C}$ could be any elementary topos. Recall the bicategory of $\mathscr{C}$-spans, whose objects are $\mathscr{C}$-objects, morphisms $X \to Y$ are spans $Y \leftarrow S \to X$ in $\mathscr{C}$, and 2-morphisms from $Y \leftarrow S \to X$ to $Y \leftarrow S' \to X$ are $\mathscr{C}$-arrows $S \to S'$ commuting with the legs of the spans. Let $\mathsf{Sp}\mathscr{C}$ be the category whose objects are $\mathscr{C}$-objects and whose morphisms are isomorphism classes of spans in the

bicategory of $\mathscr{C}$-spans. The coproduct in $\mathscr{C}$ is then the biproduct in $\mathsf{Sp}\mathscr{C}$, which we denote as $\oplus$. This means that we can compute with spans using the matrix calculus of the previous subsection.

It is worthwhile to observe how the matrix calculus manifests itself in this concrete setting. Given a span $\Phi : \mathbf{t}_1 \oplus \cdots \oplus \mathbf{t}_m \to \mathbf{v}_1 \oplus \cdots \oplus \mathbf{v}_n$ and indices $1 \leq i \leq m$ and $1 \leq j \leq n$, we can define a component sub-span ${}_{\mathbf{t}_i}\Phi_{\mathbf{v}_j} : \mathbf{t}_i \to \mathbf{v}_j$ via the following limit:



The multiplication $\mathbf{t} \oplus \mathbf{t} \to \mathbf{t}$ and unit $\mathbf{0} \to \mathbf{t}$ morphisms are given by the spans

$$\mathbf{t} \oplus \mathbf{t} \xleftarrow{\text{id}} \mathbf{t} \oplus \mathbf{t} \xrightarrow{\nabla} \mathbf{t} \qquad\qquad \mathbf{0} \xleftarrow{\text{id}} \mathbf{0} \xrightarrow{!} \mathbf{t}$$

and the comultiplication and counit are their transposes.

Let τ be a set, which we interpret as a set of types. We now specialize to the slice category $\mathscr{C} \coloneqq \mathsf{Set}/\tau$, for which an object is a function $x \to \tau$, regarded as a *τ-typed set*; when we speak of its elements, we mean elements of x. Note that $\mathscr{C} \coloneqq \mathsf{Set}/\tau$ is extensive, so the above discussion applies. In the setting of $\mathsf{Sp}\mathscr{C}$ we will refer to elements of the domain and codomain of a map as *ports*, to elements of the apex as *wires*, and to the legs as *attachment maps*.

The sum of two spans $\mathbf{t} \xleftarrow{f_-} \alpha \xrightarrow{f_+} \mathbf{v}$ and $\mathbf{t} \xleftarrow{g_-} \beta \xrightarrow{g_+} \mathbf{v}$ is given by the span

$$\mathbf{t} \xleftarrow{f_- \nabla g_-} \alpha \oplus \beta \xrightarrow{f_+ \nabla g_+} \mathbf{v}$$

where, given $f : \mathbf{t} \to \mathbf{v}$ and $g : \mathbf{u} \to \mathbf{v}$, $f \nabla g : \mathbf{t} \oplus \mathbf{u} \to \mathbf{v}$ is the composition $\iota_{\mathbf{t}} f + \iota_{\mathbf{u}} g$. We hence interpret this enrichment in commutative monoids as a disjunction: in $\alpha \oplus \beta$, the $\mathbf{t}$ ports attach to the $\mathbf{v}$ ports via *either* α-wires *or* β-wires. Of course, the zero span is $\mathbf{t} \leftarrow \mathbf{0} \to \mathbf{v}$, also denoted $\mathbf{0}$.

In linear algebra, block matrices of linear maps correspond to direct sum decompositions, while ordinary matrices of *scalars* correspond to maximal decompositions into one-dimensional subspaces. Similarly, in $\mathsf{Sp}(\mathsf{Set}/\tau)$, we have maximal decompositions into singleton sets. What serves as a basis in this context is simply a choice of ordering on a set, which we represent via enumeration as a tuple. In such a decomposition, every entry of the corresponding matrix represents a subspan whose domain and codomain are both singletons. In this case, the attachment maps are trivial, and hence we can, without loss of information, represent such entries via the set of wires that connect the domain and codomain singletons. Furthermore, for the sake of notational convenience, if a span Φ is given by a diagonal matrix, i.e. consists of a direct sum of singleton spans with apex sets $S_1, \ldots, S_n$, we will write $S_1 \oplus \cdots \oplus S_n$ for the total span.

Example 3.1. The $(r,s,t) \to (x,y,z)$ matrix

$$\begin{bmatrix} \mathbf{0} & \{B\} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \{C\} & \mathbf{0} & \{A,D\} \end{bmatrix}$$

represents the span $\{r,s,t\} \xleftarrow{f_-} \{A,B,C,D\} \xrightarrow{f_+} \{x,y,z\}$ given by

$$\begin{aligned} f_-(A) &= t & f_-(B) &= r & f_-(C) &= t & f_-(D) &= t \\ f_+(A) &= z & f_+(B) &= y & f_+(C) &= x & f_+(D) &= z \end{aligned}$$

We note that the span matrix is like an adjacency matrix, except that rather than merely indicating the presence of connection via a truth value, the entries indicate the set of all such connections.

We note that the composite of the span $\{s\} \leftarrow \alpha \rightarrow \{t\}$ and the span $\{t\} \leftarrow \beta \rightarrow \{v\}$ is the span $\{s\} \leftarrow \alpha \times \beta \rightarrow \{v\}$. We can hence perform matrix multiplication as in the following example.

Example 3.2. The composite of the span matrices

$$(j,k) \xrightarrow{\begin{bmatrix} \mathbf{0} & \{N\} & \{L,M\} \\ \{O,P\} & \mathbf{0} & \{Q\} \end{bmatrix}} (r,s,t) \xrightarrow{\begin{bmatrix} \{F,G\} & \{B\} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \{A,D\} \\ \{C\} & \mathbf{0} & \{E\} \end{bmatrix}} (x,y,z)$$

is given by the following $(j,k) \to (x,y,z)$ matrix

$$\begin{bmatrix} \{L,M\} \times \{C\} & \mathbf{0} & \{N\} \times \{A,D\} + \{L,M\} \times \{E\} \\ \{O,P\} \times \{F,G\} + \{Q\} \times \{C\} & \{O,P\} \times \{B\} & \{Q\} \times \{E\} \end{bmatrix}$$

Given our interpretation of apexes as wires, allowing non-bijective attachment maps corresponds to allowing wires to split, merge, terminate, and initiate. For the purposes of defining wiring diagrams for (strict) symmetric monoidal categories, we will (in Definition 4.1) restrict the legs of our spans to bijections; however, the more general definition can be readily used to define wiring diagrams for SMCs in which objects are supplied with monoids and/or comonoids. In particular, this bijectivity restriction is lifted in Listing 1, where ports can attach to multiple other ports on either side.

We denote the category of τ-typed finite sets and typed bijections between them as Bij_τ and the associated category of spans by $\mathsf{Sp}(\mathsf{Bij}_\tau)$. One might remark that spans of bijections are equivalent simply to bijections and wonder why Definition 4.1 uses the former. Indeed, since $\mathsf{Sp}(\mathsf{Bij}_\tau)$ is equivalent to Bij_τ, we could have in principle simply defined such wiring diagrams in the latter category. However, our composition formula mirrors the case analysis of Listing 1 by leveraging the matrix calculus of the biproduct category $\mathsf{Sp}(\mathsf{Set}/\tau)$.

4 The operad of acyclic wiring diagrams

We are now equipped to characterize unbiased compositions of morphisms in a strict symmetric monoidal category. We make the strictness assumption for a couple of reasons. First, any monoidal category is monoidally equivalent to a strict one and hence no loss of generality is incurred. Second, the graphical languages of string and wiring diagrams prohibits bracketing of parallel wires and hides unit wires, corresponding to strict associativity and unitality.

In addition to requiring that our symmetric monoidal category be strict, we define the monoidal product in an unbiased manner, as in [DM82]. Thus, we index the hom-sets by a pair of typed finite

sets—rather than typed finite ordinals—whose monoidal product give the domain and codomain respectively. In the context of computer science, this choice corresponds to a dictionary-like representation of domain and codomain rather than the more traditional list-like representation. We note that this does *not* imply that we enforce commutativity of the monoidal product. For instance, the typed finite sets $f,g : \{a_1,a_2\} \rightrightarrows \tau$ given by $f(a_1) = t_1, f(a_2) = t_2$ and $g(a_1) = t_2, g(a_2) = t_1$ are distinct, though isomorphic.

Fix a set τ, whose elements we think of as types. Next we will define the operad $\mathscr{W}_\tau$ of *acyclic wiring diagrams*. Roughly speaking, morphisms in $\mathscr{W}_\tau$ specify compositions (serial, parallel, etc.) of morphisms in an arbitrary strict unbiased SMC $\mathscr{C}$ that has been equipped with a function $\tau \to \mathrm{Ob}(\mathscr{C})$, i.e., for which an object of $\mathscr{C}$ has been chosen for each element of τ.

Definition 4.1. Let τ be a set, elements of which we call *types*. We define the τ-typed operad $\mathscr{W}_\tau$ of acyclic wiring diagrams as follows.

- an object, called a *box*, $\mathbf{t}$ is a τ-typed signed set; i.e. a pair $(\mathbf{t}_-, \mathbf{t}_+)$ of τ-typed sets, where a τ-typed set is an object of Set/τ; we call $\mathbf{t}_-$ the *inputs* and $\mathbf{t}_+$ the *outputs*.
- a morphism, called a *wiring diagram*, Φ of type $\mathbf{t}^1,\dots,\mathbf{t}^n \to \mathbf{v}$ is a span in Bij_τ

$$\mathbf{v}_- \oplus \mathbf{t}_+ \xleftarrow{\Phi_{\mathrm{src}}} \omega \xrightarrow{\Phi_{\mathrm{tgt}}} \mathbf{t}_- \oplus \mathbf{v}_+$$

  where $\mathbf{t}_\pm \coloneqq \bigoplus_{i=1}^n \mathbf{t}^i_\pm$. We call ω-elements *wires* and enforce the following:
  - ⋆ *progress condition*: imposing $\mathbf{t}^i \prec \mathbf{t}^j$ whenever there is a wire $A \in \omega$ for which both $\Phi_{\mathrm{src}}(A) \in \mathbf{t}^i_+$ and $\Phi_{\mathrm{tgt}}(A) \in \mathbf{t}^j_-$, the result must be a partial order on the $\mathbf{t}^i$.
- the identity morphism $\mathsf{inert}_{\mathbf{t}} : \mathbf{t} \to \mathbf{t}$ is given by the identity span

$$\mathbf{t}_- \oplus \mathbf{t}_+ \xleftarrow{1} \mathbf{t}_- \oplus \mathbf{t}_+ \xrightarrow{1} \mathbf{t}_- \oplus \mathbf{t}_+$$

  we note that the block matrix form in the summands is given by the identity matrix.
- given wiring diagrams $\Psi : \mathbf{s}^{i_1},\dots,\mathbf{s}^{i_{m_i}} \to \mathbf{t}^i$ and $\Phi : \mathbf{t}^1,\dots,\mathbf{t}^n \to \mathbf{v}$, we now define their i^{th} partial composite (sometimes called "circle-i" composition)

$$\Psi \circ_i \Phi : \mathbf{t}^1,\dots,\mathbf{s}^{i_1},\dots,\mathbf{s}^{i_{m_i}},\dots \mathbf{t}^n \to \mathbf{v}$$

  Letting $\mathbf{t}^{\neg i}_\pm \coloneqq \bigoplus_{j\neq i} \mathbf{t}^j_\pm$, we make the following abbreviations

$$\mathbf{u}_- \coloneqq \mathbf{v}_- \oplus \mathbf{t}^{\neg i}_+ \qquad\qquad \mathbf{u}_+ \coloneqq \mathbf{t}^{\neg i}_- \oplus \mathbf{v}_+$$

  The composite $\Psi \circ_i \Phi$ is then given by the block matrix of type $\mathbf{u}_- \oplus \mathbf{s}_+ \to \mathbf{s}_- \oplus \mathbf{u}_+$

$$\Psi \circ_i \Phi = \begin{bmatrix} (\mathbf{u}_- \Phi_{\mathbf{t}^i_-})(_{\mathbf{t}^i_-}\Psi_{\mathbf{s}_-}) & \mathbf{u}_- \Phi_{\mathbf{u}_+} + (\mathbf{u}_- \Phi_{\mathbf{t}^i_-})(_{\mathbf{t}^i_-}\Psi_{\mathbf{t}^i_+})(_{\mathbf{t}^i_+}\Phi_{\mathbf{u}_+}) \\ \mathbf{s}_+ \Psi_{\mathbf{s}_-} & (\mathbf{s}_+ \Psi_{\mathbf{t}^i_+})(_{\mathbf{t}^i_+}\Phi_{\mathbf{u}_+}) \end{bmatrix} \tag{3}$$

  If $\prec^\Phi$ and $\prec^\Psi$ are the orderings induced on the $\mathbf{t}^i$ and $\mathbf{s}^j$ by Φ and Ψ respectively, then one can define an ordering $\prec$ on $\{\mathbf{t}^1,\dots,\mathbf{s}^{k_1},\dots,\mathbf{s}^{k_{m_k}},\dots \mathbf{t}^n\}$ as follows

$$\mathbf{s}^{k_i} \prec \mathbf{s}^{k_j} \coloneqq \mathbf{s}^{k_i} \prec^\Psi \mathbf{s}^{k_j} \qquad\qquad \mathbf{s}^{k_i} \prec \mathbf{t}^j \coloneqq \mathbf{t}^k \prec^\Phi \mathbf{t}^j$$
$$\mathbf{t}^j \prec \mathbf{s}^{k_i} \coloneqq \mathbf{t}^j \prec^\Phi \mathbf{t}^k \qquad\qquad \mathbf{t}^i \prec \mathbf{t}^j \coloneqq \mathbf{t}^i \prec^\Phi \mathbf{t}^j$$

  Thus $\Psi \circ_i \Phi$ satisfies the progress condition.

We remark that the composite formula (3) is a declarative version of the imperative substitution algorithm presented in Listing 1. In particular, the four cases in the algorithm correspond to the four entries in the composite matrix: incoming wire, passing wire, fully internal wire, and outgoing wire. Furthermore, the products within each component, e.g. $({}_{\mathbf{u}_-}\Phi_{\mathbf{t}^i_-})({}_{\mathbf{t}^i_-}\Psi_{\mathbf{t}^i_+})({}_{\mathbf{t}^i_+}\Phi_{\mathbf{u}_+})$, correspond to the fusing together of wires. Finally, the sum in the upper right-hand entry codifies the fact that the composite wiring diagram still includes all of the wires, ${}_{\mathbf{u}_-}\Phi_{\mathbf{u}_+}$, that did not interact with input box $\mathbf{t}_i$.

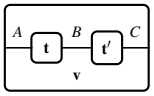
Each of the entries in the fully decomposed matrix representation of a bijective span as in Definition 4.1 is either the empty set or a singleton; we will write $\mathbf{0}$ in the former case and write A for the singleton set $\{A\}$. Similarly, we will write (A,B) for $\{A\} \times \{B\}$.

Example 4.2. We now define wiring diagrams corresponding to core SMC operations. We call these *$\mathscr{W}$-representations* for any set τ; though we introduce them in an example, they will play an important role in the theory.

- **Symmetry.** Consider wires ω, no inner boxes, and outer box $\mathbf{t} = (\omega, \omega)$. For any permutation $\sigma : \omega \xrightarrow{\sim} \omega$, the $\mathscr{W}$-representation of symmetry is given by the 0-ary wiring diagram $\mathsf{sym}^{\sigma}_{\omega} : () \to \mathbf{t}$ represented by the permutation matrix of σ. We let $\mathsf{unit}_{\omega} := \mathsf{sym}^{\mathrm{id}}_{\omega}$. For instance here is the wiring diagram and matrix form in the case where σ is the transposition of two elements.

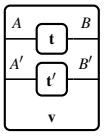
$$\begin{bmatrix} \mathbf{0} & A \\ B & \mathbf{0} \end{bmatrix}$$

- **Sequential composition.** Consider wires $\omega = \{A, B, C\}$, inner boxes $\mathbf{t}, \mathbf{t}'$, and outer box $\mathbf{v}$. The $\mathscr{W}$ representation of sequential composition (shown left) is given by the wiring diagram $\mathsf{seq}_{(A,B,C)} : \mathbf{t}, \mathbf{t}' \to \mathbf{v}$ with matrix of type $\mathbf{v}_- \oplus \mathbf{t}_+ \oplus \mathbf{t}'_+ \to \mathbf{t}_- \oplus \mathbf{t}'_- \oplus \mathbf{v}_+$ shown right:



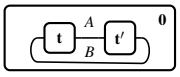
$$\begin{bmatrix} A & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & B & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & C \end{bmatrix}$$

- **Parallel composition.** Consider wires $\omega = \{A, A', B, B'\}$, inner boxes $\mathbf{t}, \mathbf{t}'$, and outer box $\mathbf{v}$. The $\mathscr{W}$ representation of parallel composition is given by the wiring diagram $\mathsf{para}_{\left[\begin{smallmatrix} A & B \\ A' & B' \end{smallmatrix}\right]} : \mathbf{t}, \mathbf{t}' \to \mathbf{v}$ with matrix of type $\mathbf{v}_- \oplus \mathbf{t}_+ \oplus \mathbf{t}'_+ \to \mathbf{t}_- \oplus \mathbf{t}'_- \oplus \mathbf{v}_+$, where both $\mathbf{v}_-$ and $\mathbf{v}_+$ are two-dimensional.



$$\begin{bmatrix} A & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A' & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & B & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & B' \end{bmatrix}$$

Example 4.3. **Non-example.** Consider the following wiring diagram Φ, which has a loop.

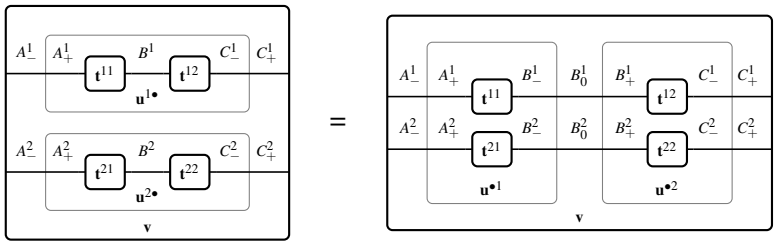


This wiring diagram is of type $\mathbf{t}, \mathbf{t}' \to \mathbf{0}$, with wires $\{A, B\}$, given by the span

$$\Phi_{\mathrm{src}}(A) = \mathbf{t}_+ \qquad \Phi_{\mathrm{tgt}}(A) = \mathbf{t}'_- \qquad \Phi_{\mathrm{src}}(B) = \mathbf{t}'_+ \qquad \Phi_{\mathrm{tgt}}(B) = \mathbf{t}_-$$

The ordering $\prec$ from Definition 4.1 is not a partial order: both $\mathbf{t} \prec \mathbf{t}'$ and $\mathbf{t}' \prec \mathbf{t}$. Therefore this diagram fails to satisfy the progress condition of Definition 4.1.

Example 4.4. **Interchange Law.** To demonstrate composition, we will prove that $\mathscr{W}$-representations satisfy the interchange law.



We wish to show the following diagram commutes:

$$\begin{array}{ccc} \mathbf{t}^{11},\mathbf{t}^{21},\mathbf{t}^{12},\mathbf{t}^{22} & \xrightarrow{\mathsf{seq}_{\omega^1},\mathsf{seq}_{\omega^2}} & \mathbf{u}^{1\bullet},\mathbf{u}^{2\bullet} \\ \Big\downarrow{\scriptstyle \mathsf{para}_{\eta^1},\mathsf{para}_{\eta^2}} & & \Big\downarrow{\scriptstyle \mathsf{para}_{\eta}} \\ \mathbf{u}^{\bullet 1},\mathbf{u}^{\bullet 2} & \xrightarrow[\mathsf{seq}_{\omega}]{} & \mathbf{v} \end{array}$$

where $\mathbf{t}^{11},\mathbf{t}^{12},\mathbf{t}^{21},\mathbf{t}^{22}$ are inner boxes, $\mathbf{u}^{1\bullet},\mathbf{u}^{2\bullet},\mathbf{u}^{\bullet 1},\mathbf{u}^{\bullet 2}$ are intermediary boxes, $\mathbf{v}$ is the outer box, and the subscripts on the morphisms correspond to the following wire tuples:

$$\omega^1 = (A^1_+, B^1, C^1_-) \qquad \omega^2 = (A^2_+, B^2, C^2_-) \qquad \omega = (A^1_- \oplus A^2_-, B^1_0 \oplus B^2_0, C^1_+ \oplus C^2_+)$$

$$\eta^1 = \begin{bmatrix} A^1_+ & B^1_- \\ A^2_+ & B^2_- \end{bmatrix} \qquad \eta^2 = \begin{bmatrix} B^1_+ & C^1_- \\ B^2_+ & C^2_- \end{bmatrix} \qquad \eta = \begin{bmatrix} A^1_- & C^1_+ \\ A^2_- & C^2_+ \end{bmatrix}$$

The composite spans are represented as matrices of type

$$\mathbf{v}_- \oplus \mathbf{t}^{11}_+ \oplus \mathbf{t}^{21}_+ \oplus \mathbf{t}^{12}_+ \oplus \mathbf{t}^{22}_+ \to \mathbf{t}^{11}_- \oplus \mathbf{t}^{21}_- \oplus \mathbf{t}^{12}_- \oplus \mathbf{t}^{22}_- \oplus \mathbf{v}_+$$

where $\mathbf{v}_\pm$ are two-dimensional; we now compute these composites to be as follows:

$$\mathsf{seq}_{\omega^1},\mathsf{seq}_{\omega^2} \fatsemi \mathsf{para}_{\eta} = \begin{bmatrix} (A^1_-,A^1_+) & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (A^2_-,A^2_+) & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & B^1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & B^2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (C^1_-,C^1_+) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (C^2_-,C^2_+) \end{bmatrix}$$

$$\mathsf{para}_{\eta^1},\mathsf{para}_{\eta^2} \fatsemi \mathsf{seq}_{\omega} = \begin{bmatrix} (A^1_-,A^1_+) & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (A^2_-,A^2_+) & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (B^1_-,B^1_0,B^1_+) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & (B^2_-,B^2_0,B^2_+) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (C^1_-,C^1_+) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (C^2_-,C^2_+) \end{bmatrix}$$

These matrices of sets are isomorphic since they have isomorphic sets (singleton and empty sets) in corresponding entries.

5 Operad algebras and symmetric monoidal categories

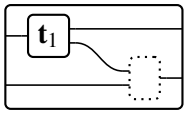
With the operad $\mathscr{W}_\tau$ of acyclic wiring diagrams (whose wires are labeled by a chosen set τ) in hand, we now formalize how it captures the compositional structure of symmetric monoidal categories. Recall that a $\mathscr{W}_\tau$-algebra is an operad functor:

$$H\colon \mathscr{W}_\tau \to \mathsf{Set},$$

where Set is the operad corresponding to the symmetric monoidal category $(\mathsf{Set}, \times, 1)$. We will now show precisely how such algebras correspond to strict SMC's. Before doing so, we will first characterize $\mathscr{W}_\tau$ in terms of generators.

Lemma 5.1. *The operad $\mathscr{W}_\tau$ is generated by* $\mathsf{sym}, \mathsf{seq}, \mathsf{para}$ *defined in Example 4.2, and these satisfy the usual axioms (parallel and sequential unitality and associativity, interchange, and permutation).*

Sketch of proof. We proceed by induction on the number of inner boxes. Let $\Phi\colon \mathbf{t}_1, \dots, \mathbf{t}_n \to \mathbf{v}$ be a morphism. If $n = 0$ then this is just some wire permutation sym_σ. For $n \geq 0$, the progress condition gives a partial ordering on boxes. Without loss of generality, suppose that $\mathbf{t}_1$ is minimal. Then we can rewrite the diagram in the form



(4)

where all the wires shown can represent multiple wires (possibly none). The wiring diagram in (4) can be written as a sequential composite of parallel composites. An example axiom relating series and parallel composition was proved in Example 4.4. □

We now show how $\mathscr{W}$-algebras give rise to SMC's.

Theorem 5.2. *Let τ be a set. There is a fully faithful functor from the category of $\mathscr{W}_\tau$-algebras to that of strict SMC's whose objects are τ-typed finite sets and whose morphisms are identity-on-objects symmetric monoidal functors.*

Sketch of proof. Let $H\colon \mathscr{W}_\tau \to \mathsf{Set}$ be a functor. We need to define an SMC $(\mathscr{C}, \otimes, I)$ with objects τ-typed finite sets; in particular, we need to define the hom-set for a given pair of objects $(\mathbf{t}_-, \mathbf{t}_+) \in \mathrm{Ob}(\mathscr{C})$. But since objects in $\mathscr{W}_\tau$ are exactly such pairs, we may simply use H:

$$\mathscr{C}(\mathbf{t}_-, \mathbf{t}_+) := H(\mathbf{t}_-, \mathbf{t}_+). \tag{5}$$

We will obtain the identities, composition, and monoidal structure of $\mathscr{C}$, by applying H to various wiring diagrams.

For any object $A \in \mathrm{Ob}(\mathscr{C})$, consider the box $\mathbf{t} = (A, A)$ and the 0-ary wiring diagram $\mathsf{unit}_A\colon () \to \mathbf{t}$ given by 

. We obtain a function $H(\mathsf{unit}_A)\colon 1 \to H(A,A)$, and we define the identity on A to be the image of the unique element.

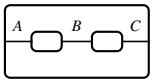
Given objects $A, B, C \in \mathrm{Ob}(\mathscr{C})$, we need a function $\mathrm{Hom}(A,B) \times \mathrm{Hom}(B,C) \to \mathrm{Hom}(A,C)$. We obtain it by applying H to the wiring diagram $\mathsf{seq}_{(A,B,C)} : (A,B), (B,C) \to (A,C)$, giving

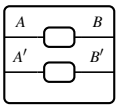
$$\mathring{9}_{A,B,C} := H(\mathsf{seq}_{(A,B,C)})\colon H(A,B) \times H(B,C) \to H(A,C).$$

Similarly, given objects $A,A',B,B' \in \mathrm{Ob}(\mathscr{C})$, parallel composition is defined by

$$\otimes_{A,A',B,B'} \coloneqq H\left(\mathsf{para}_{\left[\begin{smallmatrix} A & B \\ A' & B' \end{smallmatrix}\right]}\right) \colon H(A,B) \times H(A',B') \to H(A \otimes A', B \otimes B').$$

These two compositions are respectively depicted below.





The interchange law was established in Example 4.4, and the others, unitality and associativity of serial composition, and unitality, associativity, and symmetry of parallel composition, are similar.

It remains to show that for any two $\mathscr{W}_\tau$-algebras H,H' with associated SMCs $\mathscr{C},\mathscr{C}'$, we have a bijection between the set of natural transformations $H \to H'$ and that of identity-on-objects symmetric monoidal functors $\mathscr{C} \to \mathscr{C}'$. A natural transformation $\alpha \colon H \to H'$ defines a function $H(\mathbf{t}_-,\mathbf{t}_+) \to H'(\mathbf{t}_-,\mathbf{t}_+)$, which by (5) defines the required functor on hom-sets. This functor respects identity, composition, and the symmetric monoidal structure by the naturality of α with respect to the morphisms in $\mathscr{W}_\tau$ that correspond to these structures. □

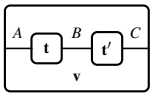
The SMC resulting from the procedure in Theorem 5.2 is free-on-objects. In the next theorem we will prove that the functor from Theorem 5.2 is 2-essentially surjective: every symmetric monoidal category is equivalent to one coming from an $\mathscr{W}$-algebra.

Theorem 5.3. *To every SMC $(\mathscr{C},\otimes,I)$ there is an associated $\mathscr{W}_{\mathrm{Ob}(\mathscr{C})}$-algebra $H_\mathscr{C}$. If we then apply the construction of Theorem 5.2 to $H_\mathscr{C}$ to obtain an SMC $\mathscr{C}'$, there is an induced equivalence of symmetric monoidal categories $\mathscr{C}' \to \mathscr{C}$.*

Sketch of proof. Let $(\mathscr{C},\otimes,I)$ be a symmetric monoidal category. We define a functor $H_\mathscr{C} \colon \mathscr{W}_{\mathrm{Ob}(\mathscr{C})} \to \mathsf{Set}$ as follows. Suppose given a box, i.e. an object $(\mathbf{t}_-,\mathbf{t}_+)$ in $\mathscr{W}_{\mathrm{Ob}(\mathscr{C})}$, where $\mathbf{t}_- \colon S_- \to \mathrm{Ob}(\mathscr{C})$ and $\mathbf{t}_+ \colon S_+ \to \mathrm{Ob}(\mathscr{C})$ are the typed finite sets. To it we assign the hom-set

$$H_\mathscr{C}(\mathbf{t}_-,\mathbf{t}_+) \coloneqq \mathscr{C}\left(\bigotimes_{s\in S_-} \mathbf{t}_-(s), \bigotimes_{s\in S_+} \mathbf{t}_+(s)\right). \tag{6}$$

By Lemma 5.1, $\mathscr{W}_{\mathrm{Ob}(\mathscr{C})}$ is generated by the morphisms $\mathsf{para},\mathsf{seq}$, and sym corresponding to parallel composition, series composition, and permutation, and that these morphisms satisfy well-known relations. Thus to give the action of $H_\mathscr{C}$ on morphisms, it suffices to say how it acts on these generators, and show that relations hold. For the morphism $\mathsf{seq}_{(A,B,C)}$ in $\mathscr{W}_{\mathrm{Ob}(\mathscr{C})}$ representing series composition



we need to give a function $H_\mathscr{C}(A,B) \times H_\mathscr{C}(B,C) \to H_\mathscr{C}(A,C)$. By definition this is just a function $\mathscr{C}(A,B) \times \mathscr{C}(B,C) \to \mathscr{C}(A,C)$, and of course we use the composition function $\fatsemi_{A,B,C}$ from $\mathscr{C}$ as a category. The case for parallel composition (resp. symmetry) is similar: one uses the monoidal product (resp. symmetry) from $\mathscr{C}$. These satisfy the required diagrammatic relations because by definition $\mathscr{C}$ satisfies the laws of monoidal categories.

Finally we consider the roundtrip, starting with $\mathscr{C}$, constructing $H_{\mathscr{C}}$ as above, and then applying the construction of Theorem 5.2 to obtain a new SMC $\mathscr{C}'$. The objects of $\mathscr{C}'$ are the $\mathrm{Ob}(\mathscr{C})$-typed finite sets, and there is a surjection $\mathrm{Ob}(\mathscr{C}') \to \mathrm{Ob}(\mathscr{C})$ given by sending $\mathbf{t}\colon S \to \mathrm{Ob}(\mathscr{C})$ to $\bigotimes_{s\in S}\mathbf{t}(s)$. At this point, (5) and (6) provide a bijection $\mathscr{C}'(\mathbf{t}_-,\mathbf{t}_+) \to \mathscr{C}(\bigotimes_{s\in S_-}\mathbf{t}_-(s), \bigotimes_{s\in S_+}\mathbf{t}_+(s))$, which one can check is part of a surjective-on-objects fully faithful symmetric monoidal functor. □

Acknowledgments

David Spivak acknowledges support from AFOSR grants FA9550-19-1-0113 and FA9550-17-1-0058. The authors also appreciate the helpful reviews of the ACT2020 program committee, which improved the quality of this paper.

References

[BFc17] Seth Bromberger, James Fairbanks, and other contributors. *LightGraphs.jl: an optimized graphs package for the Julia programming language*. 2017. DOI: 10.5281/zenodo.889971 (cit. on p. 4).

[DM82] Pierre Deligne and James S. Milne. "Tannakian categories". In: *Hodge cycles, motives, and Shimura varieties*. 1982, pp. 101–228. URL: https://www.jmilne.org/math/xnotes/tc2018.pdf (cit. on p. 9).

[FS19a] Brendan Fong and David I. Spivak. "Hypergraph categories". In: *Journal of Pure and Applied Algebra* 223.11 (2019), pp. 4746–4777. DOI: 10.1016/j.jpaa.2019.02.014 (cit. on p. 2).

[FS19b] Brendan Fong and David I. Spivak. "Supplying bells and whistles in symmetric monoidal categories". In: (2019). arXiv: 1908.02633 (cit. on p. 6).

[JS91] André Joyal and Ross Street. "The geometry of tensor calculus, I". In: *Advances in mathematics* 88.1 (1991), pp. 55–112 (cit. on p. 2).

[KZ15] Aleks Kissinger and Vladimir Zamdzhiev. "Quantomatic: A proof assistant for diagrammatic reasoning". In: *International Conference on Automated Deduction*. 2015, pp. 326–336. DOI: 10.1007/978-3-319-21401-6_22 (cit. on p. 3).

[Pc20] Evan Patterson and other contributors. *Catlab.jl: a framework for applied and computational category theory*. 2020. DOI: 10.5281/zenodo.598366 (cit. on p. 3).

[Pen71] Roger Penrose. "Applications of negative dimensional tensors". In: *Combinatorial mathematics and its applications* 1 (1971), pp. 221–244 (cit. on p. 1).

[RS13] Dylan Rupel and David I. Spivak. "The operad of temporal wiring diagrams: formalizing a graphical language for discrete-time processes". In: (2013). arXiv: 1307.6894 (cit. on p. 2).

[Sel10] Peter Selinger. "A survey of graphical languages for monoidal categories". In: *New structures for physics*. 2010, pp. 289–355. DOI: 10.1007/978-3-642-12821-9_4. arXiv: 0908.3347 (cit. on p. 2).

[SSR17] David I. Spivak, Patrick Schultz, and Dylan Rupel. "String diagrams for traced and compact categories are oriented 1-cobordisms". In: *Journal of Pure and Applied Algebra* 221.8 (2017), pp. 2064–2110. DOI: 10.1016/j.jpaa.2016.10.009. arXiv: 1508.01069 (cit. on p. 2).

[SWZ19] Pawel Sobocinski, Paul W Wilson, and Fabio Zanasi. "CARTOGRAPHER: A Tool for String Diagrammatic Reasoning (Tool Paper)". In: *8th Conference on Algebra and Coalgebra in Computer Science (CALCO 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2019. DOI: 10.4230/LIPIcs.CALCO.2019.20 (cit. on p. 3).