CATEGORIFYING THE ZX-CALCULUS

DANIEL CICALA

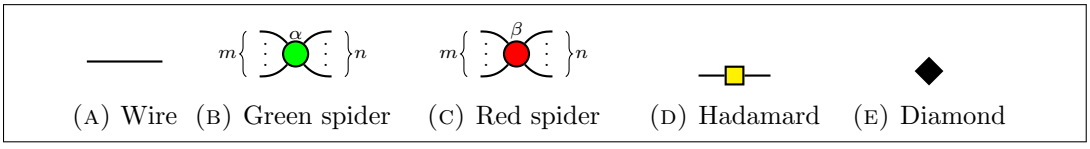
Abstract. This paper presents a symmetric monoidal and compact closed bicategory that categorifies the zx-calculus developed by Coecke and Duncan. The 1-cells in this bicategory are certain graph morphisms that correspond to the string diagrams of the zx-calculus, while the 2-cells are rewrite rules.

1. Introduction

Compositionality is becoming increasingly recognized as a viable method to model complex systems such as those found in physics [1], computer science [28], and biology [3]. The idea is to study smaller, simpler systems and ways of connecting them together. The word *compositionality* suggests that category theory can play a key role, and indeed it does. Systems are morphisms and connections are morphism composition.

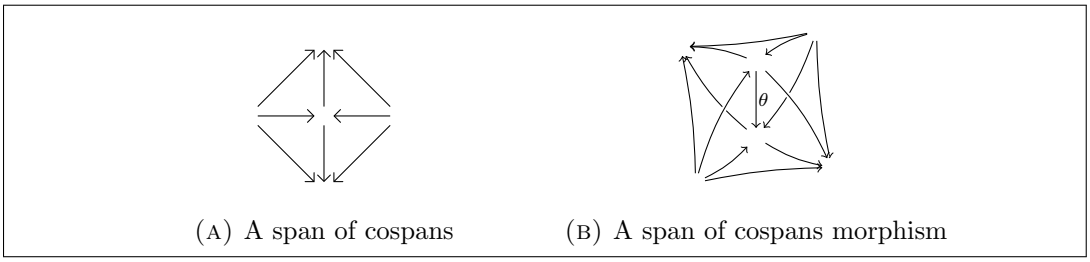
This paper looks at one example of compositionality in action: the zx-calculus. The backstory dates to Penrose's tensor networks [26] and, more recently, to the relationship between graphical languages and monoidal categories explored by Joyal, Street, and Selinger [22, 29]. Abramsky and Coecke capitalized on this relationship when inventing a categorical framework for quantum physics [1]. With a categorically oriented point of view, Coecke and Duncan presented early results on a diagrammatic language in which to reason about complementary quantum observables [7]. After a fruitful period of development [9, 10, 13, 18, 19, 20, 25], the first complete presentation of the zx-calculus was published [8].

The zx-calculus contains five basic diagrams, depicted in Figure 1. These can be combined in various ways to form larger, more complex diagrams. Observe that these diagrams have dangling wires on the left and right. Think of those on the left as inputs and those on the right as outputs. Formalizing this perspective, we let these diagrams generate the morphisms of a dagger compact category **zx** whose objects are the non-negative integers, the meaning of which is number of inputs



(A) Wire (B) Green spider (C) Red spider (D) Hadamard (E) Diamond

Figure 1. Generators for the category **zx**



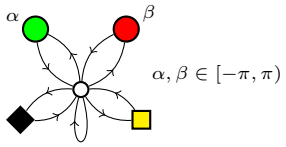
(A) A span of cospans (B) A span of cospans morphism

FIGURE 2. A generic 2-cell in $\mathbf{Sp}(\mathbf{Csp}(\mathbf{C}))$

and outputs. Section 2 contains a formal introduction to the zx-calculus. This is a merely a brief primer and contains nothing new. There, we also touch on various software tools, such as Quantomatic [4, 17] or Globular [4], used to compute with the types of diagrams found in the zx-calculus.

Our goal in this paper is to introduce a symmetric monoidal and compact closed (SMCC) bicategory $\underline{\mathbf{zx}}$ that categorifies $\mathbf{zx}$ in the sense that the 1-cells of $\underline{\mathbf{zx}}$ will correspond to the zx-diagrams. Building this correspondence, however, requires a bit of labor. We begin Section 3 by discussing open graphs and starting construction on a bicategory that suitably houses these open graphs. With this in mind, we slightly modify past work of the author and Courser [5, 6] to produce an SMCC-bicategory with graphs as 0-cells, cospans of graphs as 1-cells, and certain isomorphism classes of spans of cospans (see Figure 2) of graphs as 2-cells. This has a 1-full and 2-full sub-bicategory **Rewrite** consisting of non-negative integers as 0-cells, open graphs as 1-cells, and rewrite rules of open graphs as 2-cells. The SMCC bicategory **Rewrite** is a nice ambient space in which to generate systems modeled on open graphs. In this paper, we exploit **Rewrite** to model the zx-calculus.

In Section 4, we discuss *open graphs over* S_{zx}. That is, we pick a graph S_{zx}



in such a way that graph morphisms $G \to S_{\text{zx}}$ correspond exactly to the zx-morphisms. These form an SMCC bicategory with graphs over S_{zx} as 0-cells, cospans of graphs over S_{zx} as 1-cells, and certain isomorphism classes of spans of these cospans as 2-cells. This contains a sub-bicategory named **zxRewrite** that is analogous to **Rewrite**, if not as immediate. Indeed, to define this, we first introduce the functor $N_{\text{zx}} \colon \mathbf{Set}_0 \to (\mathbf{Graph} \downarrow S_{\text{zx}})$ on a skeleton of the category **Set**. Define $N_{\text{zx}}(X)$ to be the edgeless graph with nodes X and is constant on the S_{zx}-node ○. Then **zxRewrite** is 1-full and 2-full on the 0-cells of form $N_{\text{zx}}(X)$.

Now, **zxRewrite** is a space in which we can generate SMCC sub-bicategories. In Section 5, we give a presentation for a sub-bicategory $\underline{\mathbf{zx}}$ of **zxRewrite** by choosing

1-cell corresponding to diagrams of the zx-calculus and 2-cells corresponding to relations that hold between diagrams. After constructing $\underline{\mathbf{zx}}$, we decategorify it to a 1-category decat($\underline{\mathbf{zx}}$) by identifying 1-cells whenever there is a 2-cell between them. Though this seems asymmetrical, the dual nature of spans allows this to actually give an equivalence relation, not merely generate one. Our main result is Theorem 5.4, in which we construct a dagger compact functor decat($\underline{\mathbf{zx}}$) $\to \mathbf{zx}$ that is an equivalence of categories.

The author would like to thank his advisor John Baez for many helpful ideas and discussions that contributed to this paper.

2. The zx-calculus

One of the most fascinating features of quantum physics is the incompatibility of observables. Roughly, an observable is a measurable quantity of some system, for instance the spin of a photon. Incompatibility is in stark contrast to classical physics in which measurable quantities are compatible, in that they can have arbitrarily precise values at the same time. Arguably, the most famous example of incompatibility is Heisenberg's uncertainty principal which places limits to the precision that one can simultaneously measure a pair of observables: position and momentum. There are different levels of incompatibility amongst pairs of observables. When such a pair is maximally incompatible, in the sense that knowing one with complete precision implies total uncertainty of the other, we say they are *complementary observables*.

Historically, the typical framework in which one might study observables is Hilbert space. In a Hilbert space, each vector represents the state of a given system. Even though this formalism has been incredibly fruitful, computations in Hilbert spaces can be difficult and non-intuitive. The zx-calculus was developed by Coecke and Duncan [8] as a high-level language to facilitate such computation. It was immediately used to generalize both *quantum circuits* [24] and the *measurement calculus* [15]. Its validity was further justified when Duncan and Perdrix presented a non-trivial method of verifying measurement-based quantum computations with the zx-calculus [19]. At its core, the zx-calculus is an intuitive graphical language in which to reason about complementary observables.

The five *basic diagrams* in the zx-calculus are depicted in Figure 1 and are to be read from left to right. They are

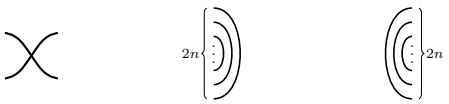
- a *wire* with a single input and output,
- *green spiders* with a non-negative integer number of inputs and outputs and paired with a phase $\alpha \in [-\pi, \pi)$,
- *red spiders* with a non-negative integer number inputs and outputs and paired with a phase $\beta \in [-\pi, \pi)$,
- the *Hadamard node* with a single input and output, and
- a *black diamond node* with no inputs or outputs.

The wire plays the role of an identity, much like a plain wire in an electrical circuit, or straight pipe in a plumbing system. The green and red spiders arise from a pair of complementary observables. Incredibly, observables correspond to certain

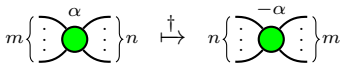
commutative Frobenius algebras A living in a dagger symmetric monoidal category $\mathbf{C}$. Moreover, a pair of complementary observables gives a pair of Frobenius algebras whose operations interact via laws like those of a Hopf algebra [11, 12]. This is particularly nice because Frobenius algebras have beautiful representations as string diagrams. Hence the depiction of the spiders. Now, if I is the monoidal unit of $\mathbf{C}$, there is an isomorphism $\mathbf{C}(I, A) \to \mathbf{C}(A, A)$ of commutative monoids that gives rise to a group on A known as the *phase group*. It is from this that we get the phases on the spiders. The Hadamard node embodies the Hadamard gate. The diamond follows from the notion of *coherence*, which exists between observable structures if certain equations are satisfied. A deeper exploration of these notions goes beyond the scope of this paper. For interested readers, there are already excellent treatments of these topics [8].

In the spirit of compositionality, we will present a category **zx** whose morphisms are generated by the five diagrams in Figure 1. Therefore, we will go ahead and call the basic diagrams in Figure 1 *basic* **zx**-*morphisms* and call a diagram an **zx**-*morphism* if it is generated by a basic one.

Observe that, on the basic **zx**-morphisms, there is a non-negative number of dangling wires on the right and left sides. We will refer to the wires on the left as *inputs* and those on the right as *outputs*. With this in mind, we now present the dagger compact category **zx** introduced by Coecke and Duncan [8] and further studied by Backens [2]. The objects of **zx** are the non-negative integers. The morphisms are generated by the basic **zx**-morphisms, though we take the wire to be the identity on 1. Composition in **zx** is performed by separately enumerating the inputs and outputs of a pair of compatible diagrams and connecting the outputs of the first diagram to the inputs of the second diagram accordingly. The monoidal product on **zx** is given by addition of numbers and the disjoint union of **zx**-morphisms. From this, we obtain the identity on any n by taking the disjoint union of n wires. The symmetry and compactness of the monoidal product provide a braiding, evaluation, and coevaluation morphisms, respectively,



The evaluation and coevalutation maps are of type $2n \to 0$ and $0 \to 2n$ for each object $n \geq 1$ and the empty diagram for $n = 0$. The dagger structure is obtained by swapping inputs and outputs then, for the spider diagrams, multiplying the phase by -1. For instance,



The dagger acts trivially on the wire, Hadamard, and diamond elements.

Thus far, we have a presentation for a free dagger compact category. However, there are relations that between **zx**-morphisms. The emergence of these relations is technical and the interested reader should read about the genesis of the zx-calculus

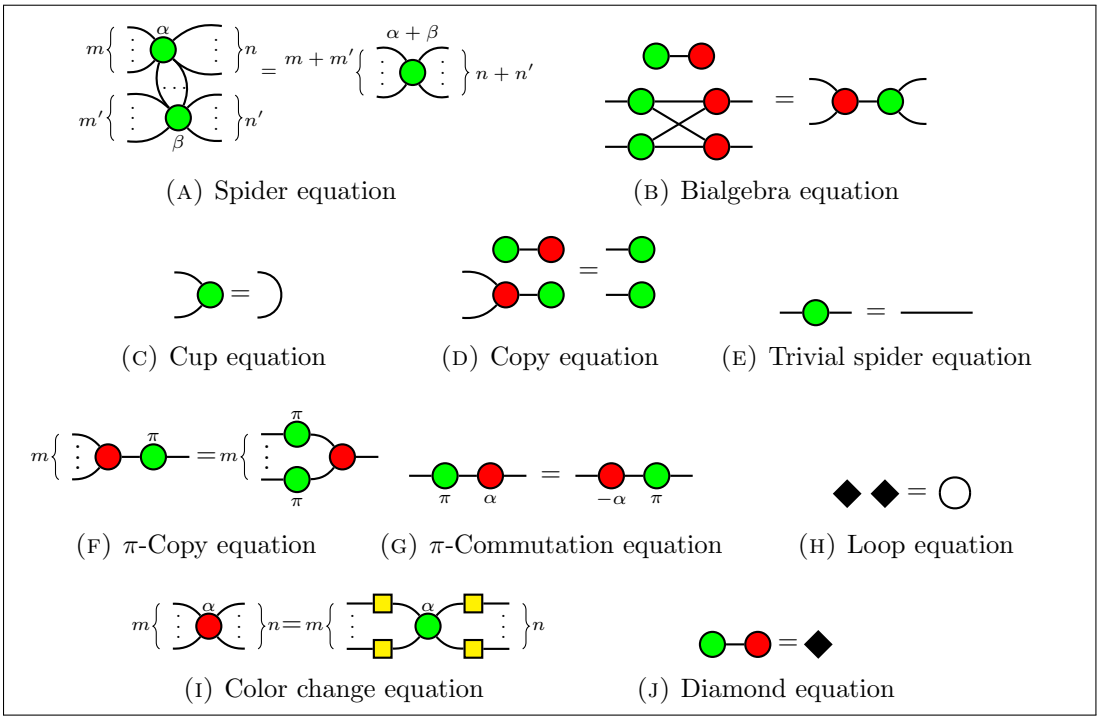


FIGURE 3. Relations in the category **zx**

[8] to learn the story. To the generating relations of **zx** depicted in Figure 3, we add equations obtained by exchanging red and green nodes, daggering, and taking the diagrams up to ambient isotopy in 4-space. These listed relations will be called *basic*. Note that we denote the empty graph, which is the monoidal unit, by $\emptyset$. Also, spiders with no phase indicated have a phase of 0.

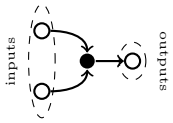
A major advantage of using these string diagrams, besides their intuitive nature, is that computations are more easily encoded into computer programs. Indeed, software programs such as Quantomatic [4, 17] and Globular [4] are graphical proof assistants tailor made for such graphical reasoning. The logic of these programs are encapsulated by double pushout rewrite rules. However, the algebraic structure of **zx** and other graphical calculi do not contain the rewrite rules as explicit elements. It is the author's hope that introducing rewrite rules into the algebraic structures embodying graphical calculi, a positive contribution will be made to these programs.

3. REWRITING OPEN GRAPHS

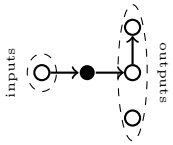
Now that we have seen depictions of the **zx**-morphisms, it is clear that they are reminiscent of directed graphs. There is reason to be optimistic that graphs can model the zx-calculus using graphs. However, our hope is tempered by the fact that there are clear differences between graphs and **zx**-morphisms. For instance, graphs

do not have inputs or outputs. In this section, we reconcile this particular difference by presenting the notion of an open graph.

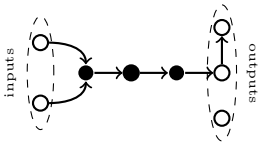
Intuitively, the notion of an open graph is rather simple. Take a directed graph and declare some of the nodes to be inputs and others to be outputs, for instance



Whenever there is a bijection between the inputs of one graph and the outputs of another, we can identify them as described by the bijection. This process is provides a way to turn a pair of compatible open graphs into a single open graph. For instance, to the above open graph, we can connect



to form the open graph



This is made precise using cospans and pushout as follows.

Definition 3.1. Consider the functor $N\colon \mathbf{Set}_0 \to \mathbf{Graph}$, on a skeleton of **Set**, that is given by the letting $N(X)$ be the edgeless graph with nodes X. An *open graph* is then a cospan in the category **Graph** of the form $N(X) \to G \leftarrow N(Y)$ for sets X and Y.

The left leg $N(X)$ of the cospan are the *inputs* and the right leg $N(Y)$ are the *outputs*. Suppose we have another open graph G' with inputs $N(Y)$ and outputs $N(Z)$. Then we can compose cospans

$$N(X) \to G \leftarrow N(Y) \to G' \leftarrow N(Z).$$

by pushing out over $G \leftarrow N(Y) \to G'$ to get

$$N(X) \to G +_{N(Y)} G' \leftarrow N(Z).$$

By taking isomorphism classes of these pushouts, we obtain a category whose objects are those in the image of N and morphisms are open graphs. But we can do better!

Thus far, we have only just described the first layer of a symmetric monoidal and compact closed (SMCC) bicategory named **Rewrite**$'$. This bicategory was introduced by the author, though without the prime [5]. It was shown that **Rewrite**$'$ is SMCC in a joint work with Courser [6]. The monoidal structure is induced from the coproduct of graphs. However, for our purposes, there are technical reasons to

make a slight adjustment to **Rewrite′**. To create our revised version, **Rewrite**, we begin by defining a new bicategory.

Given a category $\mathbf{C}$, we can talk about spans of cospans in $\mathbf{C}$. They and their morphisms are simply commuting diagrams of a certain shape illustrated in Figure 2. We are interested in morphism classes of spans of cospans, by which we mean the equivalence relation generated on the spans of cospans by relating S and S' if there is a morphism of spans of cospans $S \to S'$. One can alternatively conceive a morphism class as an isomorphism class of spans of cospans in the image of the inclusion functor $\mathbf{C} \hookrightarrow \widehat{\mathbf{C}}$ where $\widehat{\mathbf{C}}$ is the groupoid generated by $\mathbf{C}$.

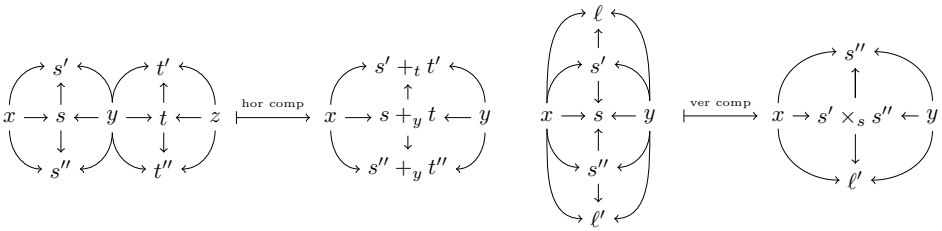
Theorem 3.2. *If* $\mathbf{C}$ *has finite limits and colimits, there is an SMCC bicategory* $\mathbf{Sp}(\mathbf{Csp}(\mathbf{C}))$ *consisting of objects of* $\mathbf{C}$ *as* 0*-cells, cospans in* $\mathbf{C}$ *as* 1*-cells, and morphism classes of spans of cospans in* $\mathbf{C}$ *as* 2*-cells.*

Proof. By appropriately modifying the authors work in [5], one can show that $\mathbf{Sp}(\mathbf{Csp}(\mathbf{C}))$ is a bicategory. A slight change in arguments found in the authors joint work with Courser [6], we get that $\mathbf{Sp}(\mathbf{Csp}(\mathbf{C}))$ is symmetric monoidal and compact closed. □

Showing that $\mathbf{Sp}(\mathbf{Csp}(\mathbf{C}))$ is symmetric monoidal relied heavily on Shulman's technique involving symmetric monoidal double categories [30]. The monoidal product is induced by the coproduct on $\mathbf{C}$.

Recall that 2-cells can be composed in two ways in a bicatgory. Horizontal composition in $\mathbf{Sp}(\mathbf{Csp}(\mathbf{C}))$ uses pushouts and vertical composition uses pullbacks, as illustrated here:

(1)



Definition 3.3. Define **Rewrite** to be the 1-full and 2-full SMCC sub-bicategory of $\mathbf{Sp}(\mathbf{Csp}(\mathbf{Graph}))$ whose 0-cells are exactly graphs in the image of the functor $N\colon \mathbf{Set}_0 \to \mathbf{Graph}$.

The concept of **Rewrite** is that the 1-cells are open graphs and the 2-cells are rewrite rules that preserves the input and output nodes. By rewrite rules, we mean those taken from the double pushout graph rewriting approach [14].

Our motivation for constructing **Rewrite** is not necessarily to study it directly, but rather for it to serve as an ambient context in which to generate SMCC subbicategories on some collection of open graphs and rewriting rules. This is worth considering because many graphical calculi use some version of open graphs as a semantics and with equations between graphical terms given by rewrite rules

[16, 21, 27]. There are drawbacks to this approach. For example, working with open graphs is only useful to model graphical calculi whose terms are equal up to ambient isotopy in 4-space. This limits the current approach to only symmetric monoidal (bi)categories as Selinger's shows in his work [29] excellently describing various graphical calculi and the role ambient isotopy plays in their use.

4. Open graphs over S_{ZX}

So far, in Section 3, we saw that there is an SMCC bicategory $\mathbf{Sp}(\mathbf{Csp}(\mathbf{C}))$. In particular, we discussed its sub-bicategory **Rewrite**. While, the **zx**-morphisms are like open graphs with multi-sorted nodes, **Rewrite** is not good enough for our purposes. It can capture the openness, but is not strong enough to see the multi-sortedness of the node. We need some new construction, analogous to **Rewrite**, with this additional structure. In this section, we determine exactly what structure is needed and produce the desired SMCC bicategory, **zxRewrite**.

Definition 4.1. Let S be a graph. By a *graph over* S, we mean a graph morphism $G \to S$. A morphism between graphs over S is a graph morphism $G \to G'$ such that

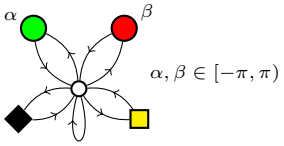
$$\begin{array}{ccc} G & \to & G' \\ & \searrow \quad \swarrow & \\ & S & \end{array}$$

commutes.

Using graphs over S, for a particular S, will provide the multi-sorted nodes we seek. Essentially, G absorbs the structure of S via the fibres of the morphism. We illustrate this with the following example.

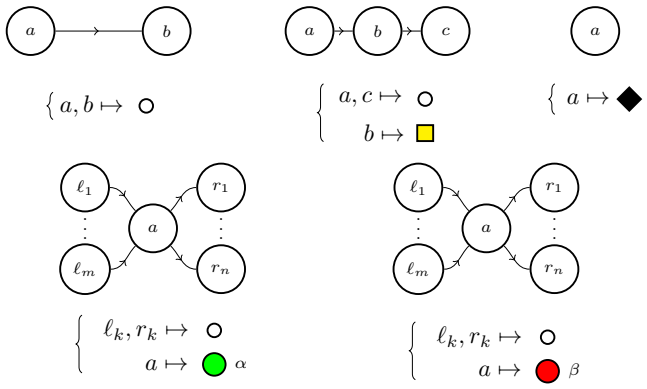
Example 4.2. Let S_{ZX} be the graph

(2)



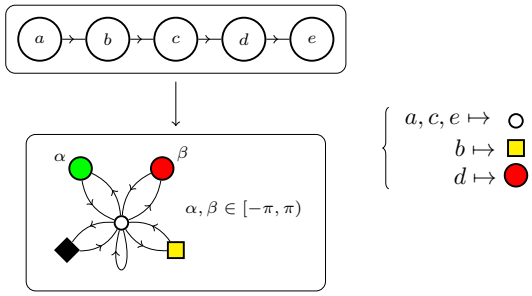
We have not drawn the entirety of S_{ZX}. In actuality, the green and red nodes run through $[-\pi, \pi)$ and all of them have a single arrow to and from node ○.

Most of the structure of the basic **zx**-morphisms is captured by graphs over S_{zx}:



Note that the functions are completely determined by their behavior on the nodes because there is at most one arrow between any two nodes in S_{zx}. The role each of the nodes in S_{zx} plays in providing structure is evident except, perhaps, for the node ○. Observe that four of the basic **zx**-morphisms have dangling wires on either end. Because edges of directed graphs must be attached to a pair of nodes, we use this node to anchor the dangling edges.

Example 4.3. At this point, we can think of the basic elements of the zx-calculus as graphs over S_{zx}. But we can actually do this for any **zx**-morphism, such as



This corresponds with the **zx**-morphism

Above, we mentioned that the graphs over S_{zx} from Example 4.2 capture 'most' of the structure of the basic **zx**-morphisms. The difference is, basic **zx**-morphisms are composable. Therefore, if we aim to describe **zx**-morphisms with graphs over S_{zx}, we need to be able to connect graphs over S_{zx} in a way that captures composition. We can achieve this by equipping graphs over S_{zx} with inputs and outputs, just as we did for open graphs. Again, we use cospans, though the added structure introduces new considerations.

Consider the over category $\mathbf{Graph} \downarrow S_{\text{zx}}$ of graphs over S_{zx}. By Theorem 3.2, this gives us an SMCC bicategory $\mathbf{Sp}(\mathbf{Csp}(\mathbf{Graph} \downarrow S_{\text{zx}}))$ within which we would

like to construct a sub-bicategory in a similar vein to **Rewrite**. However, there is a problem. Recall that the objects of **Rewrite** have the form $N(X)$ where $N\colon \mathbf{Set}_0 \to \mathbf{Graph}$ is the functor sending a set to the edgeless graph on that set. In **Graph**, there is a unique way to be an edgeless graph. But in $\mathbf{Graph} \downarrow S_{\mathrm{zx}}$, there is no unique way to be edgeless because there may be more than one graph morphism to S_{zx}. For instance, the graph with two nodes and no edges can be a graph over S_{zx} in $5^2 = 25$ ways. This issue can be rectified by functorially choosing which edgeless graphs over S_{zx} will serve as inputs and outputs.

Definition 4.4. Define a functor

$$N_{\mathrm{zx}}\colon \mathbf{Set}_0 \to \mathbf{Graph} \downarrow S_{\mathrm{zx}}$$

by $X \mapsto (N_{\mathrm{zx}}(X) \to S_{\mathrm{zx}})$ where $N_{\mathrm{zx}}(X)$ is the edgeless graph with nodes X that are constant over ○. An *open graph over* S_{zx} is a cospan in $\mathbf{Graph} \downarrow S_{\mathrm{zx}}$ of the form

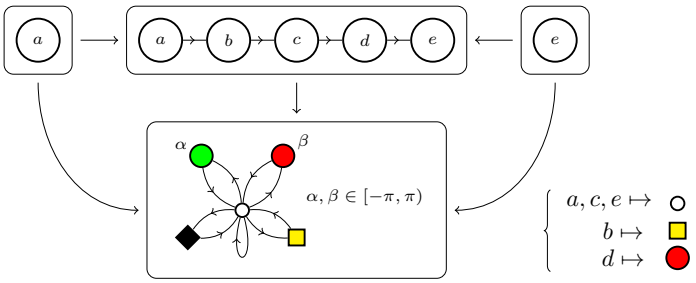
$$N_{\mathrm{zx}}(X) \to G \leftarrow N_{\mathrm{zx}}(Y).$$

With N_{zx} in hand, we can expand the notion of graphs over S_{zx} to open graphs over S_{zx}. At this point, we are ready to define the analogue to **Rewrite**.

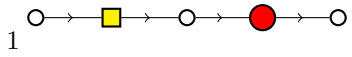
Definition 4.5. Define a bicategory **zxRewrite** as the symmetric monoidal and compact closed sub-bicategory of $\mathbf{Sp}(\mathbf{Csp}(\mathbf{Graph} \downarrow S_{\mathrm{zx}}))$ that is 1-full and 2-full on the objects of the form $N_{\mathrm{zx}}(X)$ for sets X.

Unpacking this definition, the 0-cells of **zxRewrite** are those edgeless graphs over S_{zx} in the image of N_{zx}. The 1-cells are exactly the open graphs over S_{zx}. The 2-cells are all the ways to rewrite one open graph over S_{zx} into another in a way that preserves the inputs and outputs. To better understand this bicategory, we give an example of an open graph over S_{zx}. Along with this example, we introduce new notation in order to keep the remaining diagrams rather compact, which is better illustrated than described.

Example 4.6. Consider the graph over S_{zx} in Example 4.3. Make this an open graph as follows:



There is a single input, node a, and a single output, node e. Denote this by

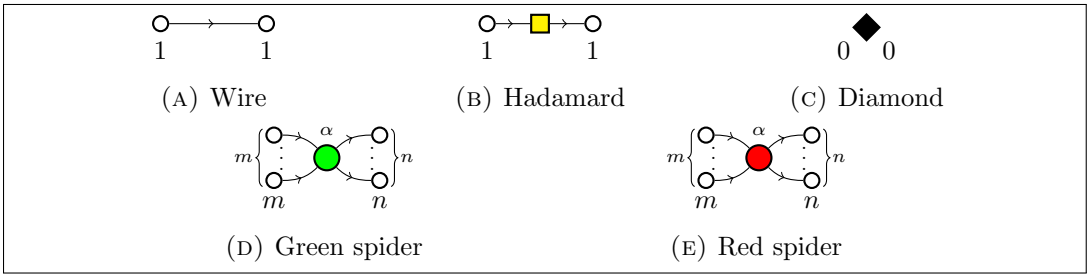


The input nodes are aligned on the far left and the output nodes on the far right. The 1's in the corners refer to the cardinality of the input and output node sets. This may seem unnecessary or perhaps even redundant, but it will bring clarity to several situations arising later on. Hence, we will side with consistency and write these cardinalities every time. Also, it is safe to assume that the left and right hand cospan maps are always injections. Even though this is not required by **zxRewrite**, it is true throughout this paper. Of course, this notation sheds away a fair amount information regarding the graph morphisms involved in the cospan, though this information should not be missed.

As mentioned earlier, our interest in the category **Rewrite** is as an ambient context in which to generate symmetric monoidal and compact closed bicategories from some collection of 1-cells and 2-cells. The same can be said for our new bicategory **zxRewrite**. Presently, we are interested in 1-cells that correspond to the basic **zx**-morphisms from Figure 1 and 2-cells that correspond to the basic relations from Figure 3. Our claim is that the bicategory generated by these 1-cells and 2-cells categorifies **zx**.

5. A categorification of **zx**

The basic **zx**-morphisms in Figure 1 are naturally presented as open graphs over S_{zx} as shown in Figure 4. We will refer to these five diagrams as *basic*. To clarify the double instances of m and n written in the spider diagrams, those written on the bottom of the diagram refer to the cardinalities of the cospan legs, and those written beside the brackets count how many nodes are there. The injectivity of the cospan legs of these generators ensure that these numbers will be the same for the basic diagrams.



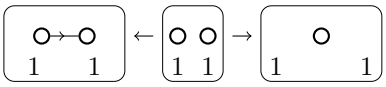
(A) Wire (B) Hadamard (C) Diamond

(D) Green spider (E) Red spider

Figure 4. Generating 1-cells for the bicategory $\underline{\mathbf{zx}}$

Just as the open graphs over S_{zx} in Figure 4 capture the generating **zx**-morphisms, we also want to introduce the basic relations into our framework. These relations are presented in Figure 5, but we also include 2-cells obtained by exchanging red and green nodes, swapping inputs and outputs for each graph over S_{zx}, or by turning

the spans around. There is also an additional 2-cell



added to account for the fact that the wire 1-cell is not the identity on 1 here. This 2-cell will allow us to replace a wire with an identity. All of these 2-cells, we call *basic*. We are now ready to define the bicategory which categorifies the zx-calculus.

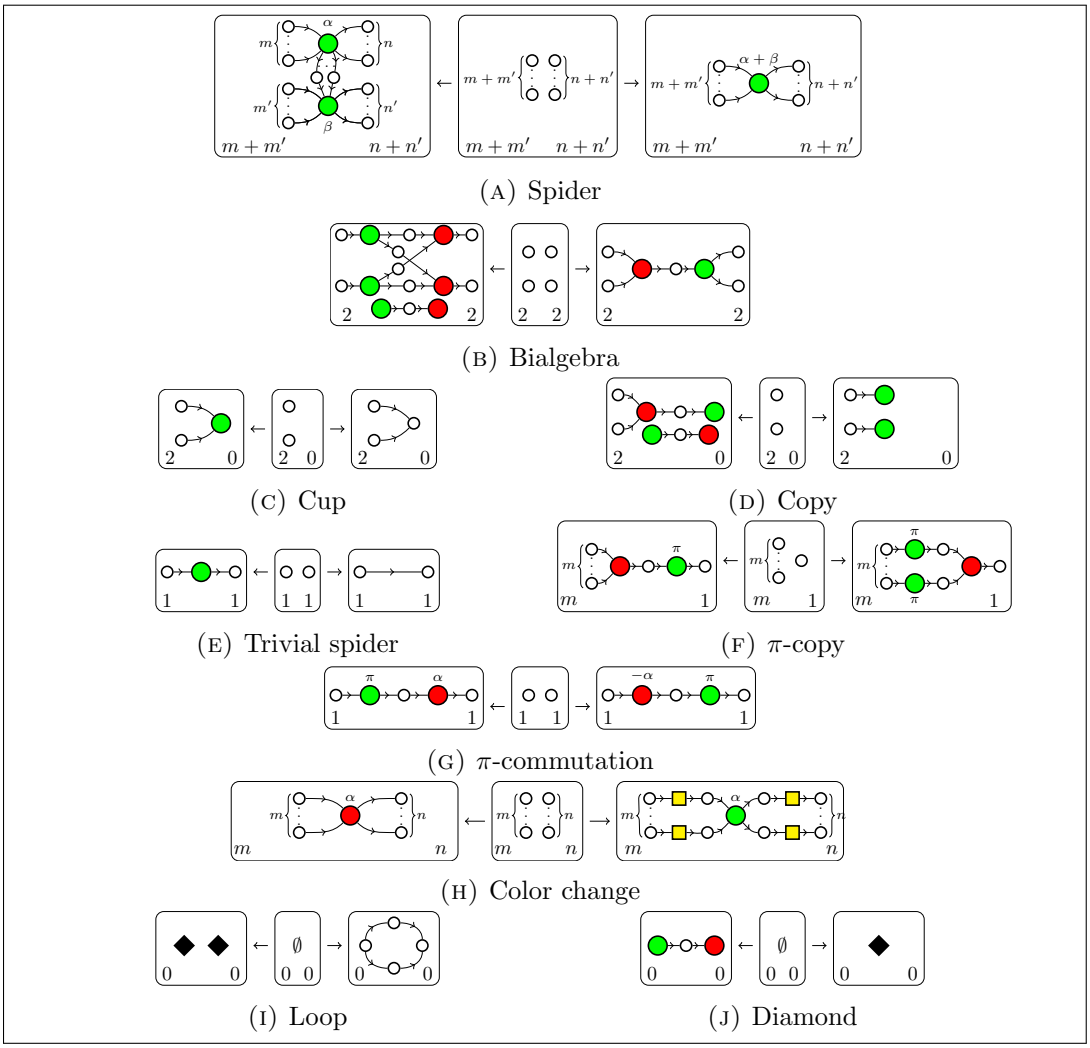
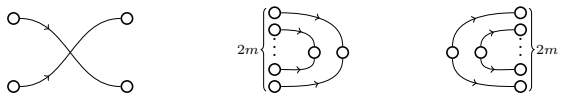


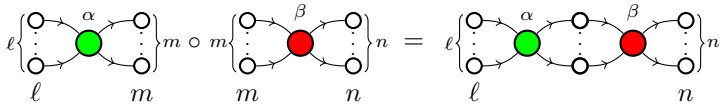
FIGURE 5. Generating 2-cells for the bicategory **zx**

Definition 5.1. Define **zx** to be the symmetric monoidal and compact closed sub-bicategory of **zxRewrite** generated by the basic 1-cells and basic 2-cells.

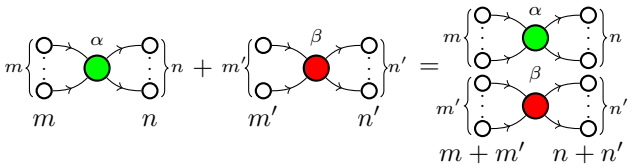
Because $\underline{\mathbf{zx}}$ is symmetric monoidal and compact closed, it also contains a twist 1-cell, plus evaluation and coevaluation 1-cells for the compact structure on m

(3) 

Horizontal and vertical composition acts just as in (1). For example, we can compose spider diagrams with the same number of inputs and outputs:



Tensoring 1-cells is simply disjoint union:

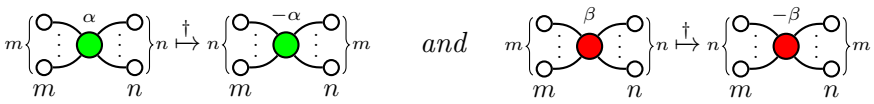


Having defined $\underline{\mathbf{zx}}$, we can turn our focus towards presenting the main theorem. We start with the following definition.

Definition 5.2. Define decat($\underline{\mathbf{zx}}$) to be the category whose objects are the 0-cells of $\underline{\mathbf{zx}}$ and whose arrows the 1-cells of $\underline{\mathbf{zx}}$ modulo the equivalence relation $\sim$ given by: $f \sim g$ if and only if there is a 2-cell $f \Rightarrow g$ in $\underline{\mathbf{zx}}$.

To be clear, the above actually is an equivalence relation and doesn't merely generate one. This follows from the symmetry of spans and vertical composition.

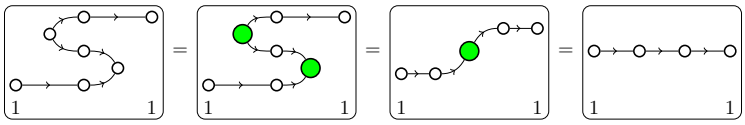
Theorem 5.3. *The category* decat($\underline{\mathbf{zx}}$) *is dagger compact via the identity on objects functor described by*



and

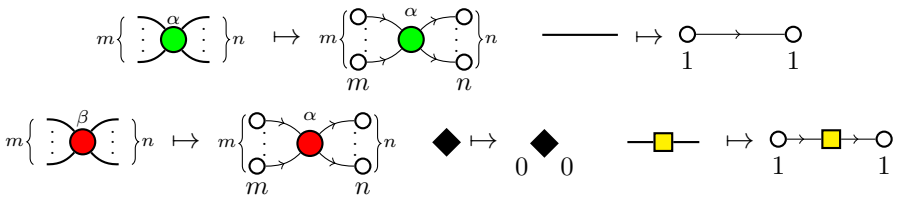
as well as by identity on the wire, Hadamard, and diamond morphisms.

Proof. Compact closedness follows from the self duality of objects via the evaluation and coevaluation maps from (3). Moreover, we can derive the snake equation



where the equalities follow from the evident 2-cells in $\underline{\mathbf{zx}}$. The extra relation introduced in Definition 5.1 ensures that the string of wires is the identity. Showing that the described functor is a dagger functor is a matter of checking some easy to verify details. □

Theorem 5.4. *The identity on objects, dagger compact functor* $E\colon \mathbf{zx} \to \mathrm{decat}(\underline{\mathbf{zx}})$ *given by*



is an equivalence of categories.

Proof. Essential surjectivity follows immediately from E being identity on objects. Fullness follows from the fact that the morphism generators for $\mathrm{decat}(\underline{\mathbf{zx}})$ are all in the image of E.

Faithfulness is more involved. Let f, g be $\mathbf{zx}$-morphisms. Let $\widetilde{Ef}$, $\widetilde{Eg}$ be the representatives of Ef, Eg obtained by directly translating the graphical representation of f, g to open graphs of S_{zx} as in Examples 4.3 and 4.6. For faithfulness, it suffices to show that the existence of a 2-cell $\widetilde{Ef} \Rightarrow \widetilde{Eg}$ in $\underline{\mathbf{zx}}$ implies that $f = g$.

Observe that any 2-cell α in $\underline{\mathbf{zx}}$ can be written, not necessarily uniquely, as sequence $\alpha_1 \square \cdots \square \alpha_n$ of length n where each α_i is a basic 2-cell and each box is filled in with '$\circ_{\mathrm{h}}$', '$\circ_{\mathrm{v}}$', or '$+$'. By '$\circ_{\mathrm{h}}$' and '$\circ_{\mathrm{v}}$', we mean horizontal and vertical composition. We will induct on sequence length. If $\alpha\colon \widetilde{Ef} \Rightarrow \widetilde{Eg}$ is a basic 2-cell, then there is clearly a corresponding basic relation equating f and g. Suppose we have a sequence of length $n{+}1$ such that the left-most square is a '$+$'. Rhen we have a 2-cell $\alpha_1 + \alpha_2\colon Ef \Rightarrow Eg$ where α_1 is a basic 2-cell and α_2 can be written with length n. By fullness, we can write $\alpha_1 + \alpha_2\colon Ef_1 + EF_2 \Rightarrow Eg_1 + Eg_2$ where $\alpha_i\colon Ef_i \Rightarrow Eg_i$. This gives that $f_i = g_i$ and the result follows. A similar argument handles the cases when the left-most operation is vertical or horizontal composition. □

References

[1] S. Abramsky & B. Coecke (2004) *A categorical semantics of quantum protocols.* In Logic in Computer Science. Proceedings of the 19th Annual IEEE Symposium on (pp. 415-425). IEEE. Chicago. Also available as arXiv:quant-ph/0402130.

[2] M. Backens (2016) *Completeness and the ZX-calculus.* Available as arXiv:1602.08954.

[3] J. Baez, B. Fong, & B. Pollard (2016) *A compositional framework for Markov processes.* Journal of Mathematical Physics, 57(3), 033301. Also available as arXiv:1508.06448.

[4] K. Bar, A. Kissinger, & J. Vicary (2016) *Globular: an online proof assistant for higher-dimensional rewriting.* 1st International Conference on Formal Structures for Computation and Deduction, vol. 52, pp. 111. http://globular.science

[5] D. Cicala (2016) *Spans of cospans.* Available as arXiv:1611.07886.

[6] D. Cicala & K. Courser, *Bicategories of spans and cospans.* In preparation.

[7] B. Coecke & R. Duncan (2008) *Interacting Quantum Observables*. In: Aceto L., Damgrd I., Goldberg L.A., Halldrsson M.M., Inglfsdttir A., Walukiewicz I. (eds) Automata, Languages and Programming. ICALP 2008. Lecture Notes in Computer Science, vol 5126. Springer, Berlin, Heidelberg

[8] B. Coecke & R. Duncan (2011) *Interacting quantum observables: categorical algebra and diagrammatics*. New Journal of Physics, 13(4), 043016. Also available as arXiv:0906.4725.

[9] B. Coecke & B. Edwards (2011) *Toy quantum categories*. In: Proceedings of Quantum Physics and Logic/Development of Computational Models (QPL-DCM). Electronic Notes in Theoretical Computer Science, 271(1), pp. 29-40. Also available as arXiv:0808.1037.

[10] B. Coecke, B. Edwards, & R. Spekkens (2011) *Phase groups and the origin of non-locality for qubits*. Electronic Notes in Theoretical Computer Science, 271(2), pp. 15-36. Also available as arXiv:1003.5005.

[11] B. Coecke & D. Pavlovic (2006) *Quantum measurements without sums*. Available as arXiv:quant-ph/0608035.

[12] B. Coecke, D. Pavlovic, & J. Vicary (2013) *A new description of orthogonal bases*. Mathematical Structures in Computer Science, 23(03), pp. 555-567. Also available as arXiv:0810.0812.

[13] B. Coecke & S. Pedrix (2010) *Environment and classical channels in categorical quantum mechanics*. In: Proceedings of the 19th EACSL Annual Conference on Computer Science Logic (CSL), Lecture Notes in Computer Science 6247, pp. 230-244, Springer-Verlag. Also available as arXiv:1004.1598

[14] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, & M. Löwe (1997). *Algebraic Approaches to Graph Transformation-Part I: Basic Concepts and Double Pushout Approach*. In Handbook of Graph Grammars, pp. 163-246.

[15] V. Danos, E. Kashefi, & P. Panangaden, (2007) *The measurement calculus*. Journal of the ACM (JACM), 54(2), 8. Also available as arXiv:0704.1263

[16] L. Dixon, R. Duncan, & A. Kissinger (2010) *Open graphs and computational reasoning*. Available as arXiv:1007.3794.

[17] L. Dixon, R. Duncan, & A. Kissinger *Quantomatic*. https://sites.google.com/site/quantomatic/

[18] R. Duncan & S. Perdrix (2009) *Graph States and the Necessity of Euler Decomposition*. In: Ambos-Spies K., Lwe B., Merkle W. (eds) Mathematical Theory and Computational Practice. CiE 2009. Lecture Notes in Computer Science, vol 5635. Springer, Berlin, Heidelberg.

[19] R. Duncan, & S. Perdrix (2010) In: Abramsky S., Gavoille C., Kirchner C., Meyer auf der Heide F., Spirakis P.G. (eds) *Rewriting measurement-based quantum computations with generalised flow*. ICALP 2010. Lecture Notes in Computer Science, vol 6199. Springer, Berlin, Heidelberg.

[20] J. Evans, R. Duncan, A. Lang, & P. Panangaden (2009) *Classifying all mutually unbiased bases in Rel*. Available as arXiv:0909.4453

[21] B. Fong (2016) *The Algebra of Open and Interconnected Systems*. Available as arXiv:1609.05382.

[22] A. Joyal & R. Street (1991) *The geometry of tensor calculus, I*. Advances in Mathematics, 88(1), pp. 55-112. Also available at http://www.sciencedirect.com/science/article/pii/000187089190003P.

[23] A. Kissinger & V. Zamdzhiev (2015) *Quantomatic: A proof assistant for diagrammatic reasoning*. In International Conference on Automated Deduction (pp. 326-336). Springer International Publishing. Also available as arXiv:1503.01034.

[24] M. Nielsen & I. Chuang (2000) *Quantum Computation and Quantum Information*. Cambridge University Press.

[25] D. Pavlovic (2009) *Quantum and classical structures in nondeterministic computation*. Lecture Notes in Computer Science, 5494, pp. 143-157, Springer. Also available as arxiv:0812.2266

[26] R. Penrose (1971) *Applications of negative dimensional tensors.* Combinatorial mathematics and its applications. 221244. Also available at http://homepages.math.uic.edu/ kauffman/Penrose.pdf

[27] B. Pollard (2016) *Open Markov processes: A compositional perspective on non-equilibrium steady states in biology.* Entropy, 18(4), pp. 140. Also available as arXiv:1601.00711.

[28] V. Sassone, & P. Sobociński (2005) *A congruence for Petri nets.* Electronic Notes in Theoretical Computer Science, 127(2), pp. 107-120. Also available as http://www.sciencedirect.com/science/article/pii/S1571066105001222

[29] P. Selinger (2010) *A survey of graphical languages for monoidal categories.* New Structures for Physics, pp. 289-355. Springer Berlin Heidelberg. Also available as arXiv:0908.3347.

[30] M. Shulman (2010) *Constructing symmetric monoidal bicategories.* Available as arXiv:1004.0993.