REWRITING ON OPEN STRUCTURED GRAPHS

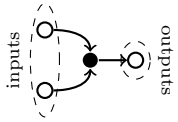
DANIEL CICALA

Abstract. It was shown in [1] that given a topos T, we can form a bicategory **MonicSp**(**Csp**(**T**)) whose 0-cells are the T-objects, 1-cells are cospans in T, and 2-cells are spans of cospans with monic legs. Taking T to be the category **Graph** of (directed) graphs and graph morphisms, we consider the full sub-bicategory **Rewrite** of **MonicSp**(**Csp**(**Graph**)) on the edgeless graphs. This bicategory ought to be considered as having open graphs and ways of rewriting them. We then demonstrate that we can capture categories of graphs with extra structure by considering certain slice categories of **Graph**. We can then construct a symmetric monoidal, compact closed bicategory of open structured graphs and their rewritings. Finally, we apply this construction to categorify the PROP whose morphisms are diagrams in the zx-calculus developed by Coecke and Duncan.

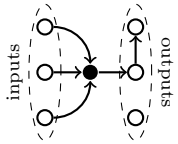
1. Introduction

2. Rewriting open graphs

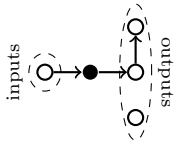
Intuitively, the notion of an open graph is rather simple. Take a directed graph and declare some of the nodes to be inputs and others to be outputs, for instance



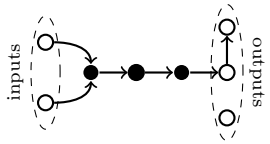
Whenever there is a bijection between the inputs of one graph and the outputs of another, we can connect them in a way described by the bijection. This process is provides a way to turn a pair of compatible open graphs into a single open graph. Indeedn, we cannot connect



to the above open graph, though we can connect



to form the open graph



This is made precise using cospans and pushout as follows. Consider the functor $N\colon \mathbf{Set} \to \mathbf{Graph}$ that assigns the edgeless graph with node set X to a set 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. We will denote this open graph by G when the legs of the cospan do not need to be explicit. Also, call the left leg $N(X)$ of the cospan the *inputs* of G and the right leg $N(Y)$ the *outputs* of G. Suppose we have another open graph G' with inputs $N(Y)$ and outputs $N(Z)$. Then we can compose the cospans

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

This is certainly not an open graph, but pushing out over the span $G \leftarrow N(Y) \to G'$ induces the open graph

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

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.

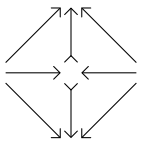
But we can do better! Indeed, we have only just described the first layer of a symmetric monoidal and compact closed bicategory. This bicategory was introduced by the author in [1] under the name **Rewrite**. It was shown that **Rewrite** is symmetric monoidal and compact closed in a joint work with Courser . Before moving on to consider graphs equipped with some additional structure, we briefly recall the story of **Rewrite**.

cite here

Given a topos $\mathbf{T}$, there is a symmetric monoidal and compact closed bicategory $\mathbf{MonicSp}(\mathbf{Csp}(\mathbf{T}))$ consisting of

- (0-cells) objects of $\mathbf{T}$,
- (1-cells) cospans in $\mathbf{T}$, and
- (2-cells) isoclasses of monic spans of cospans in $\mathbf{T}$.

The 2-cells are diagrams



where '$\rightarrowtail$' denotes a monic $\mathbf{T}$-morphism. Because $\mathbf{MonicSp}(\mathbf{Csp}(\mathbf{T}))$ is such a long name, we will shorten it to $\mathbf{MSC}(\mathbf{T})$.

Letting $\mathbf{T}$ be the topos **Graph** of directed graphs, there is a symmetric monoidal and compact closed sub-bicategory of $\mathbf{MonicSp}(\mathbf{Csp}(\mathbf{Graph}))$ called **Rewrite** whose 0-cells are exactly the edgeless graphs, and full in 1-cells and 2-cells. The idea of **Rewrite** is that the 1-cells are open graphs and the 2-cells are the ways

to rewrite a graph into another while preserving the input and output nodes. By rewriting, we mean double pushout graph rewriting . This means that restricting ourselves to monic spans in the 2-cells is not overly restrictive since many authors only consider monic double pushout rewriting rules.

The 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 symmetric monoidal and compact closed sub-bicategories on some collection of open graphs and rewriting rules. Of course, the collection that one would use depends on their interest. To illustrate, Example in shows that we obtain a categorified version of **2Cob**, the category of , with the following generators:

Why care about **Rewrite**? Graphical calculi typically use some version of open graphs as a semantics and equations between graphical terms are given by rewrite rules . However, equations are evil when looking through a categorical lens. Morally, equations ought to be replaced by isomorphisms. The process of replacing equations with isomorphisms and sets by categories is known as categorification. In this program, we are interested in categorifying certain categories into bicategories. The categories of interest are those whose morphisms are associated to open graphs of some sort. For instance there are a number of categories of open networks which we seek to categorify by interpreting rewrite rules as 2-cells instead of as providing equations between morphisms. However, this requires some work.

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 isotopy in 4 dimensions . Amending **Rewrite** to account for this shortcoming is a direction of future research. Currently, our interest lies in expanding the idea of **Rewrite** to account for 'open structured graphs', which we introduce in the next section.

DPO citation

put frob algebra example here

cite Kenny/me

describe 2cob here

input generators

citeation here

cite papers here

expand on this

3. Open structured graphs

So far, we have seen that there is a symmetric monoidal and compact closed bicategory $\mathbf{MSC}(\mathbf{T})$ for any topos $\mathbf{T}$. Moreover, when $\mathbf{T} \coloneqq \mathbf{Graph}$, we get this nice symmetric monoidal and compact closed sub-bicategory **Rewrite**. Because many graphical languages use graphs with additional structure, it would be nice to give a similar construction for these cases. To this end, we first discuss a method of providing additional structure in a manner that is amenable to this construction.

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

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

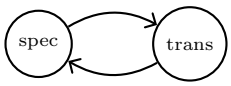
The way to think about how a graph G over S gives structure to G is that S classifies the nodes and edges of G via the fibres of the morphism. This method is being investigated by Baez and Courser in an upcoming paper . However, here is

cite it

an example to illustrate the concept.

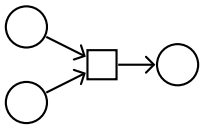
Example 3.1. A Petri net (without marking or weighted edges) is a bipartite graph whose partitions are called *species* and *transitions*. Let S_{PN} be the graph

(1)

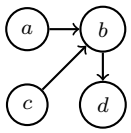


A Petri net is a graph over S_{PN}. Indeed, given a graph G over S_{PN}, the fibres of the nodes 'spec', 'trans' are the species and transitions, respectively. The arrows of S_{PN} force G to be bipartite. For example, the Petri net

(2)



can be realized as the graph



over S_{PN} via the assignment $a, c, d \mapsto$ spec and $b \mapsto$ trans.

Now, we want to go a bit further and study open graphs over S (for any graph S). That is, we want to equip a graph over S with inputs and outputs. To achieve this, we will use cospans as in the case of **Rewrite** discussed above. Though, the added structure introduces new considerations.

find citation

Recall that an over category in a topos is still a topos . In particular, given a graph S, we have the topos $\mathbf{Graph}/S$ of graphs over S. It follows that $\mathbf{MSC}(\mathbf{Graph}/S)$ is a symmetric monoidal and compact closed bicategory. However, there is not a unique (up to isomorphism) way to be an edgeless graph due to the possibility of there being many graph morphisms to S. For instance, the graph G_2 with two nodes and no edges can be a graph over S_{PN} in four different ways because there are four graph morphisms $G_2 \to S_{\mathrm{PN}}$.

However, there is a way to functorially choose edgeless graphs to serve as inputs and outputs for graphs over S. Namely, fix a node $*$ of S and define a functor $N\colon \mathbf{Set} \to \mathbf{Graph}/S$ by assigning $N(X) \to S$ to each set X where $N(X)$ is the edgeless graph with nodes X and each node is mapped to $*$.

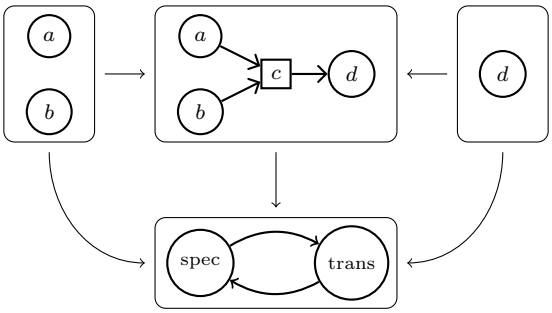
We will pause here to expand the Petri net example above to an 'open Petri net'.

Example 3.2. An open Petri net is a special case of an open graph as defined earlier where only species can be an input or output. However, we want define this using graphs over S_{PN} from (1). Define a functor $N\colon \mathbf{Set} \to \mathbf{Graph}/S_{\mathrm{PN}}$ by sending a set X to $N(X) \to S_{\mathrm{PN}}$ where $N(X)$ is the edgeless graph on the set X and the morphisms is constant over *spec*. An open Petri net is a cospan of the form

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

FIGURE 1. Generators for the PROP

$N(X) \to P \leftarrow N(Y)$ for sets X, Y and a Petri net P over S_{PN}. If P is the Petri net from (2), $X \coloneqq \mathbf{2}$, and $Y \coloneqq \mathbf{1}$, then the picture looks like



where the horizontally aligned graph morphisms are defined according to the labeling and the graph morphisms to S_{PN} send nodes $a, b, d \mapsto$ spec and $d \mapsto$ trans. Here, we have inputs a, b and output d.

go onto composition here?

Now that we have a notion of open graphs over S, we need to compose compatible such graphs as we did for open graphs. Again, this is done using pushouts in the category $\mathbf{Graph}/S$.

And so in a similar manner to open graphs, given a graph S and a functor $N\colon \mathbf{Set} \to \mathbf{Graph}/S$ whose object image lies within the collection of edgeless graphs over S, we obtain a symmetric monoidal bicategory $\mathbf{Rewrite}_{N,S}$ contained in $\mathbf{MSC}(\mathbf{Graph}/S)$. The 0-cells of $\mathbf{Rewrite}_{N,S}$ are those graphs over S with the form $N(X)$ for a set X. Then $\mathbf{Rewrite}_{N,S}$ is full in the 1-cells and 2-cells. Similar to **Rewrite**, the usefulness of $\mathbf{Rewrite}_{\mathbf{N},\mathbf{S}}$ is as an ambient context in which to generate sub-structures on a desired collection of open graphs over S and rewrite rules. This will be demonstrated on the zx-calculus .

cite coerke duncan

4. The zx-calculus as open structured graphs

The zx-calculus was developed by Coecke and Duncan as an intuitive graphical language in which to reason about quantum observables.

cite

There are five *basic elements* to the zx-calculus depicted in Figure 1. These should be read from left to right.

introduce this calculus

- a wire with a single input and output,
- green nodes with a non-negative integer number of inputs and outputs and paired with a phase $\alpha \in [-\pi, \pi)$,
- red nodes with a non-negative integer number inputs and outputs and paired with a phase $\beta \in [-\pi, \pi)$,

- a yellow node with a single input and output, and
- a black box node with no inputs or outputs

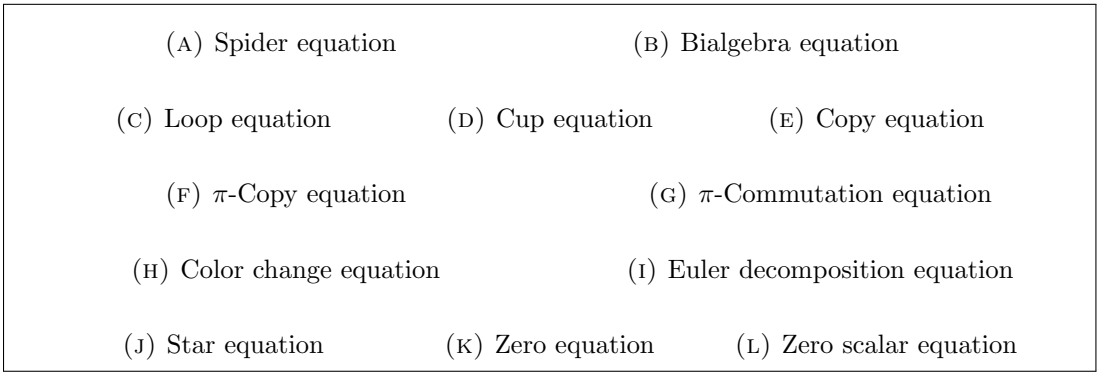
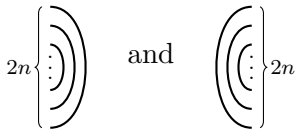


FIGURE 2. Relations in the PROP **ZX**

In the spirit of compositionality, we wish to view these elements as morphisms in some category. Indeed we will now construct a dagger compact category **ZX** generated, in part, by the basic elements.

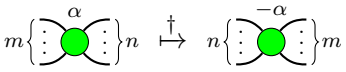
give credit for this category

The objects of **ZX** will be the non-negative integers. The morphisms will be generated by those depicted in Figure 1, though we also include



to give the compact structure on the object n. Their is a monoidal product on **ZX** given by addition on objects and disjoint union of diagrams on the morphisms. We include another generator

to provide symmetry for the monoidal product. The dagger structure is obtained by swapping inputs and outputs then, for the red and green 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, though to obtain **ZX**, we also include the relations depicted in Figure 2. Note that, to this list of relations, we also include equations obtained

- by exchanging red and green nodes,
- by daggering, and

- up to ambient isotopy in 4-space.

Also, note that we denote the empty graph, which is the monoidal unit, by $\emptyset$ plus red and green nodes which are not labeled with a phase are to be considered as having a phase of 0.

Because th zx-calculus we cooked up to model quantum observables, there is an interpretation functor

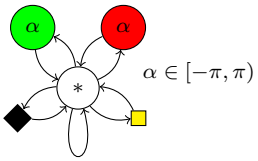
$$[\![-]\!] \colon \mathbf{ZX} \to \mathbf{FinHilb}_{\mathbb{C}}$$

where $\mathbf{FinHilb}_{\mathbb{C}}$ is the category of finite dimensional complex Hilbert spaces and linear maps. However, we are not interested in this functor. Instead, we shall use categorify **ZX** using open structured graphs.

cite this

Let S_{zx} be the graph

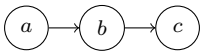
(3)



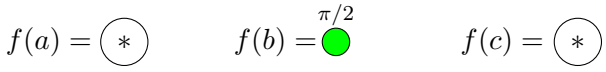
We have not drawn the entirety of S_{zx}. Actually, for every $\alpha \in [-\pi, \pi)$, there is a green node and a red node, both which have a single arrow to and from node α.

Now we have the topos $\mathbf{Graph}/S_{\text{zx}}$ of S_{zx}-structured graphs. Define a functor $N_{\text{zx}} \colon \mathbf{Set} \to \mathbf{Graph}/S_{\text{zx}}$ by sending a set X to the edgeless S_{zx}-structured graph with nodes X that are constant over $*$. We can now consider the SMCC bicategory $\mathbf{Rewrite}_{N,S_{\text{zx}}}$ contained in $\mathbf{MonicSp}(\mathbf{Csp}(\mathbf{Graph}/S_{\text{zx}}))$ that is full over objects of the form $N_{\text{zx}}(X)$. Our next step is to generate a sub-bicategory inside $\mathbf{Rewrite}_{N_{\text{zx}},S_{\text{zx}}}$ that will categorify **ZX**. For this, we will first need to introduce notation.

Suppose that we have a graph morphism $f \colon G \to S_{\text{zx}}$. Each node x of G will be drawn the in the same way as $f(x)$ is drawn in (3). The exception is nodes in the fibre of $*$ which will be drawn with an open circle. There is no need to label the arrows of G because f is completely determined by its behavior on the nodes. For instance, if G is the graph

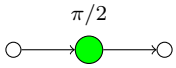


and f is given by



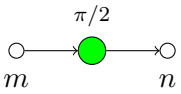
then we draw the over object $f \colon G \to S_{\text{zx}}$ as

(4)

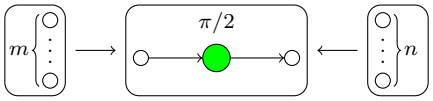


However, a 1-cell in $\mathbf{Rewrite}_{N_{\text{zx}},S_{\text{zx}}}$ is not just a graph over S_{zx}. An object is a cospan of graphs over S_{zx} such that both codomains are in the image of the node

functor N_{zx}. To denote (4) as an object from $N_{\text{zx}}(X) \to N_{\text{zx}}(Y)$, we write



where $m = |N_{\text{zx}}(X)|$ and $n = |N_{\text{zx}}(Y)|$ as a shorthand for the cospan



of graphs over S_{zx}. In this example, the graph maps in the cospan are clear and, indeed, suppressing the behavior of the maps in our shorthand should not lead to confusion.

We are now ready to give a presentation of **ZX**.

put more language here

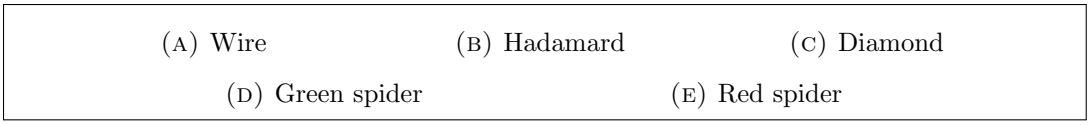


FIGURE 3. Generating 1-cells

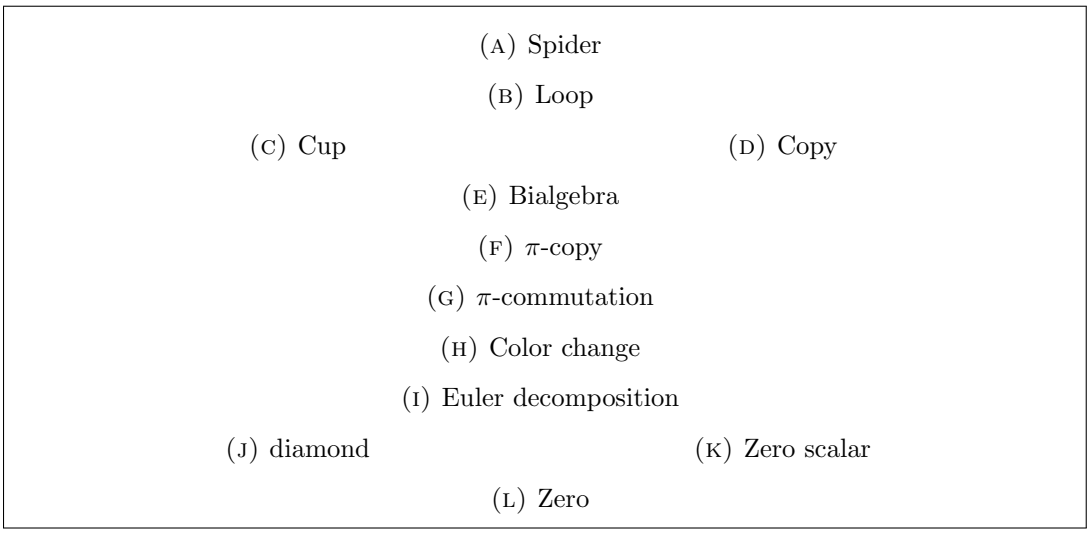


FIGURE 4. Generating 2-cells

Theorem 4.1. *The decategorification of the bicategory* **zxRewrite** *to a category is equivalent to the category* **ZX**.

prove this sucker

Proof. □

References

[1] D. Cicala, Spans of cospans. Available as arXiv:1611.07886.