A symmetric monoidal and compact closed bicategorical syntax for graphical calculi

Daniel Cicala

21 July 2017

University of California at Riverside

motivation

motivation

a question

Is there a general framework for systems comprised of open networks and rewriting?

Loosely, by *open network* we mean a graphical language with inputs and outputs

motivation

a question

Is there a general framework for systems comprised of open networks and rewriting?

Loosely, by *open network* we mean a graphical language with inputs and outputs

goals

Today, we will

construct such a bicategorical framework

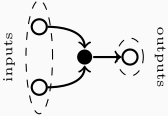
— and —

illustrate its use on the zx-calculus

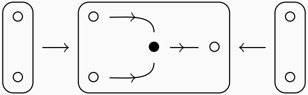
modeling open networks & rewrites

modeling open networks

Open networks can be modeled with cospans, eg



vs

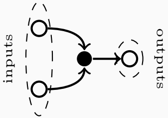


In general, for a network G with inputs X and outputs Y

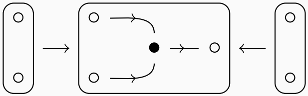
$$X \to G \leftarrow Y$$

modeling open networks

Open networks can be modeled with cospans, eg



vs

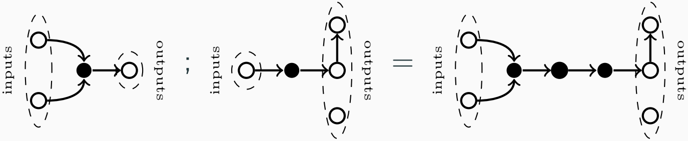


In general, for a network G with inputs X and outputs Y

$$X \to G \leftarrow Y$$

modeling open networks

Compatible open networks can be connected, e.g.



This is made precise with pushouts:

$$(X \to G \leftarrow Y);(Y \to H \leftarrow Z) = (X \to G +_Y H \leftarrow Z)$$

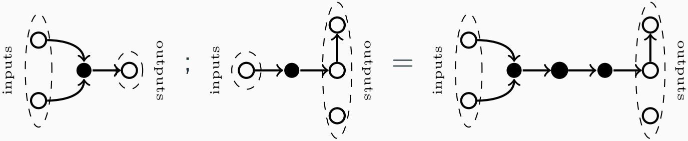
This induces a category with

(objects)	input and output types
(morphisms)	open networks possibly modulo relations.

Can we categorify this with relations as 2-cells?

modeling open networks

Compatible open networks can be connected, e.g.



This is made precise with pushouts:

$$(X \to G \leftarrow Y); (Y \to H \leftarrow Z) = (X \to G +_Y H \leftarrow Z)$$

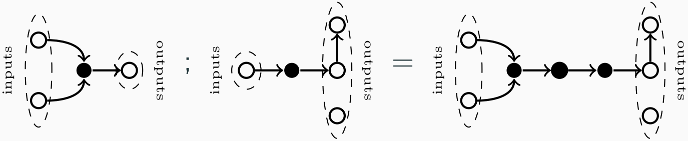
This induces a category with

(objects) input and output types
(morphisms) open networks possibly modulo relations.

Can we categorify this with relations as 2-cells?

modeling open networks

Compatible open networks can be connected, e.g.



This is made precise with pushouts:

$$(X \to G \leftarrow Y);(Y \to H \leftarrow Z) = (X \to G +_Y H \leftarrow Z)$$

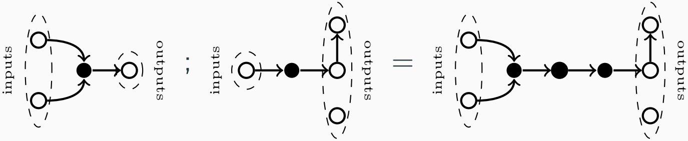
This induces a category with

(objects)	input and output types
(morphisms)	open networks possibly modulo relations.

Can we categorify this with relations as 2-cells?

modeling open networks

Compatible open networks can be connected, e.g.



This is made precise with pushouts:

$$(X \to G \leftarrow Y); (Y \to H \leftarrow Z) = (X \to G +_Y H \leftarrow Z)$$

This induces a category with

(objects)	input and output types
(morphisms)	open networks possibly modulo relations.

Can we categorify this with relations as 2-cells?

modeling rewrite rules

Using graph-like structures, we give relations by rewrite rules.

In particular, we use **double pushout rewriting** where a rule

$$L \rightsquigarrow R$$

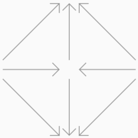
is given by a span

$$L \leftarrow K \rightarrow R$$

So what we want is

rewrite rules (spans) between **open networks** (cospans).

Thus **spans of cospans**:



modeling rewrite rules

Using graph-like structures, we give relations by rewrite rules.

In particular, we use **double pushout rewriting** where a rule

$$L \rightsquigarrow R$$

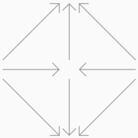
is given by a span

$$L \leftarrow K \rightarrow R$$

So what we want is

> **rewrite rules** (spans) between **open networks** (cospans).

Thus **spans of cospans**:



modeling rewrite rules

Using graph-like structures, we give relations by rewrite rules.

In particular, we use **double pushout rewriting** where a rule

$$L \rightsquigarrow R$$

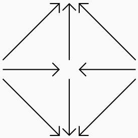
is given by a span

$$L \leftarrow K \rightarrow R$$

So what we want is

rewrite rules (spans) between **open networks** (cospans).

Thus **spans of cospans**:



combining open networks & rewrite rules

combining open networks & rewrite rules

The components we are working with are

- inputs and outputs
- open networks, i.e. cospans between inputs and outputs
- rewrites of open networks, i.e. spans of cospans

DID WE JUST DESCRIBE A BICATEGORY?

combining open networks & rewrite rules

The components we are working with are

- inputs and outputs
- open networks, i.e. cospans between inputs and outputs
- rewrites of open networks, i.e. spans of cospans

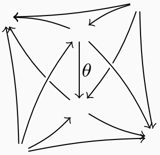
Did we just describe a bicategory?

combining open networks & rewrite rules

Theorem (C.)

Let $\mathbf{T}$ *be a topos. There is a bicategory* $\mathbf{MonicSp}(\mathbf{Csp}(\mathbf{T}))$ *with*

(0-cells) *objects of* $\mathbf{T}$
(1-cells) *cospans in* $\mathbf{T}$
(2-cells) *monic spans of cospans in* $\mathbf{T}$ *up to isomorphism*

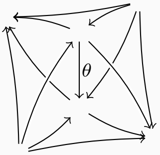


The hypothesis are used in the interchange rule.

DPO rewriting often assumes monic span legs

Theorem (C.)
Let $\mathbf{T}$ *be a topos. There is a bicategory* $\mathbf{MonicSp}(\mathbf{Csp}(\mathbf{T}))$ *with*

- **(0-cells)** *objects of* $\mathbf{T}$
- **(1-cells)** *cospans in* $\mathbf{T}$
- **(2-cells)** *monic spans of cospans in* $\mathbf{T}$ *up to isomorphism*



The hypothesis are used in the interchange rule.

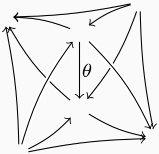
DPO rewriting often assumes monic span legs

combining open networks & rewrite rules

Theorem (C.)

Let **T** *be a topos. There is a bicategory* **MonicSp**(**Csp**(**T**)) *with*

> **(0-cells)** *objects of* **T**
>
> **(1-cells)** *cospans in* **T**
>
> **(2-cells)** *monic spans of cospans in* **T** *up to isomorphism*



The hypothesis are used in the interchange rule.

DPO rewriting often assumes monic span legs

combining open networks & rewrite rules

In case monic span legs are too strict...

Theorem (C.)
Let C be a category with finite limits and colimits. There is a bicategory $\mathbf{Sp}(\mathbf{Csp}(\mathbf{C}))$ *with*

(0-cells) *objects of* $\mathbf{C}$,
(1-cells) *cospans in* $\mathbf{C}$,
(2-cells) *spans of cospans in* $\mathbf{C}$,
up to sharing a domain and codomain.

combining open networks & rewrite rules

In case monic span legs are too strict...

Theorem (C.)
Let C be a category with finite limits and colimits. There is a bicategory $\mathbf{Sp}(\mathbf{Csp}(\mathbf{C}))$ *with*

> **(0-cells)** *objects of* $\mathbf{C}$,
> **(1-cells)** *cospans in* $\mathbf{C}$,
> **(2-cells)** *spans of cospans in* $\mathbf{C}$,
> *up to sharing a domain and codomain.*

combining open networks & rewrite rules

Theorem (C. & Courser)

Consider the topos T and the finitely complete and cocomplete category C to be symmetric monoical via $+$ and 0.

Then the bicategories $\mathbf{MonicSp}(\mathbf{Csp}(\mathbf{T}))$ *and* $\mathbf{Sp}(\mathbf{Csp}(\mathbf{C}))$ *are symmetric monoidal and compact closed (á la Mike Stay).*

$\mathbf{MonicSp}(\mathbf{Csp}(\mathbf{T}))$ and $\mathbf{Sp}(\mathbf{Csp}(\mathbf{C}))$ are too big!

We need to pare them down

Let's illustrate this process with the zx-calculus

combining open networks & rewrite rules

$\mathbf{MonicSp}(\mathbf{Csp}(\mathbf{T}))$ and $\mathbf{Sp}(\mathbf{Csp}(\mathbf{C}))$ are too big!

We need to pare them down

Let's illustrate this process with the zx-calculus

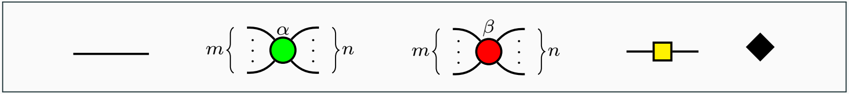
the zx-calculus

the zx-calculus – generators

The zx-calculus[1] is a syntax used in categorical quantum mechanics.

It models certain quantum processes

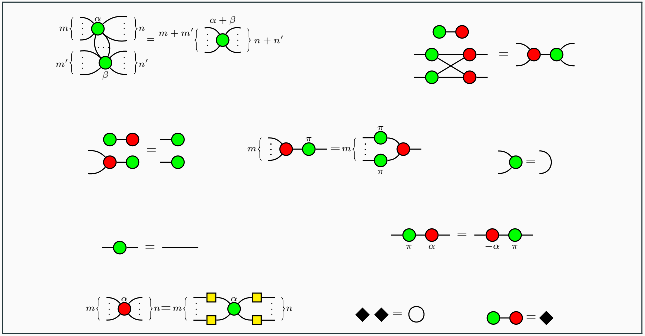
It is generated by the diagrams



[1] B Coecke & R Duncan (2011) *Interacting quantum observables: categorical algebra and diagrammatics*. New J. Phys., 13 (4), 043016.

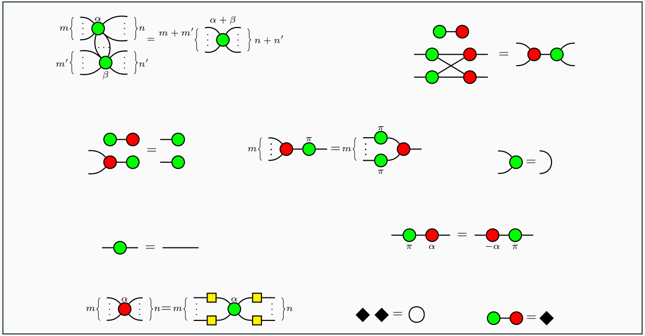
the zx-calculus – generators

and the relations



How can we realize these as open graph-like structures?

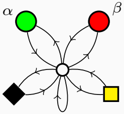
and the relations



How can we realize these as open graph-like structures?

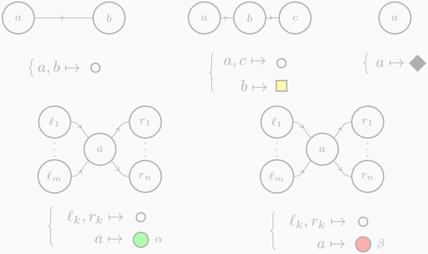
the zx-calculus – coloring the nodes

We want directed graphs with colored nodes. To this end, we define a graph S_{zx}



$\alpha, \beta \in [-\pi, \pi)$

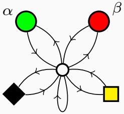
The generating zx-diagrams are almost **graphs over** S_{zx}



But these still lack inputs and outputs!

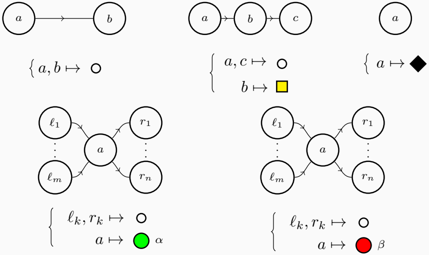
the zx-calculus – coloring the nodes

We want directed graphs with colored nodes. To this end, we define a graph S_{zx}



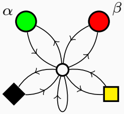
$\alpha, \beta \in [-\pi, \pi)$

The generating zx-diagrams are almost **graphs over** S_{zx}



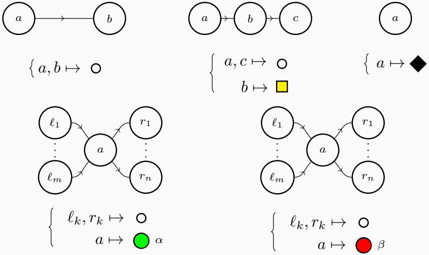
But these still lack inputs and outputs!

We want directed graphs with colored nodes. To this end, we define a graph S_{zx}



$\alpha, \beta \in [-\pi, \pi)$

The generating zx-diagrams are almost **graphs over** S_{zx}

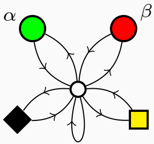


But these still lack inputs and outputs!

Define a functor

$$N\colon \mathbf{FinSet} \to \mathbf{Graph} \downarrow S_{\mathsf{zx}}$$

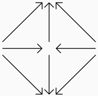
by sending a set x to the edgeless graph with node set x equipped with the map constant over the node ○ of



$\alpha, \beta \in [-\pi, \pi)$

the zx-calculus – constructing inputs and outputs

Rewrite is the SMCC sub-bicategory of $\mathbf{Sp}(\mathbf{Csp}(\mathbf{Graph} \downarrow S_{zx}))$

	Rewrite	conceit
(0-cells)	$N(x)$	input/output type
(1-cells)	$N(x) \to G \leftarrow N(y)$	open graphs over S_{zx}
(2-cells)		all DPO rewrite rules

Rewrite is still too big. <u>WHAT IS IT GOOD FOR?</u>

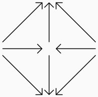
– an ambient space in which to generate SMCC bicategories –

To categorify the zx-calculus, we will translate

- zx-diagrams into open graphs over S_{zx}
- relations into DPO rewrite rules

the zx-calculus – constructing inputs and outputs

Rewrite is the SMCC sub-bicategory of $\mathbf{Sp}(\mathbf{Csp}(\mathbf{Graph} \downarrow S_{zx}))$

	Rewrite	conceit
(0-cells)	$N(x)$	input/output type
(1-cells)	$N(x) \to G \leftarrow N(y)$	open graphs over S_{zx}
(2-cells)		all DPO rewrite rules

Rewrite is still too big. <u>What is it good for?</u>

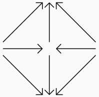
– an ambient space in which to generate SMCC bicategories –

To categorify the zx-calculus, we will translate

- zx-diagrams into open graphs over S_{zx}
- relations into DPO rewrite rules

the zx-calculus – constructing inputs and outputs

Rewrite is the SMCC sub-bicategory of $\mathbf{Sp}(\mathbf{Csp}(\mathbf{Graph} \downarrow S_{\text{zx}}))$

	Rewrite	conceit
(0-cells)	$N(x)$	input/output type
(1-cells)	$N(x) \to G \leftarrow N(y)$	open graphs over S_{zx}
(2-cells)		all DPO rewrite rules

Rewrite is still too big. <u>WHAT IS IT GOOD FOR?</u>

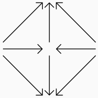
– an ambient space in which to generate SMCC bicategories –

To categorify the zx-calculus, we will translate

- zx-diagrams into open graphs over S_{zx}
- relations into DPO rewrite rules

the zx-calculus – constructing inputs and outputs

Rewrite is the SMCC sub-bicategory of **Sp**(**Csp**(**Graph** $\downarrow S_{\text{zx}}$))

	Rewrite	conceit
(0-cells)	$N(x)$	input/output type
(1-cells)	$N(x) \to G \leftarrow N(y)$	open graphs over S_{zx}
(2-cells)		all DPO rewrite rules

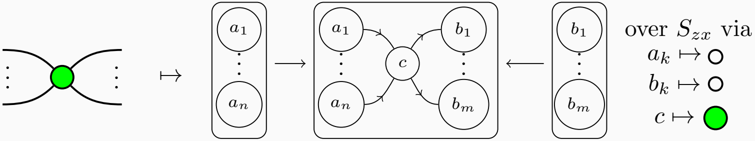
Rewrite is still too big. <u>WHAT IS IT GOOD FOR?</u>

– *an ambient space in which to generate SMCC bicategories* –

To categorify the zx-calculus, we will translate

- zx-diagrams into open graphs over S_{zx}
- relations into DPO rewrite rules

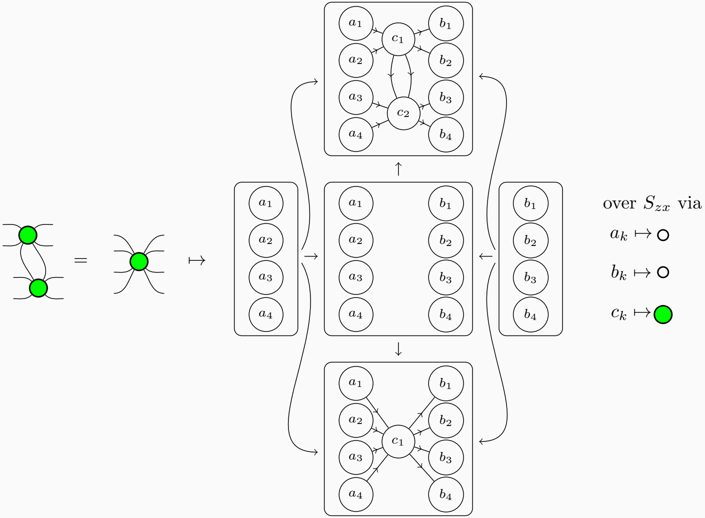
Translate zx-diagrams into 1-cells of **Rewrite**



etc.

the zx-calculus – translating to Rewrite

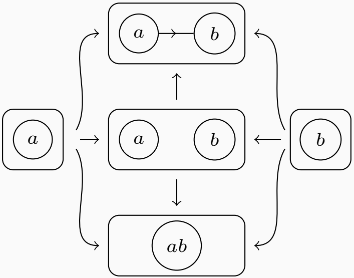
Translate zx-relations into 2-cells of **Rewrite**



etc.

the zx-calculus – translating to Rewrite

To force the wire to act like the identity, we add the 2-cell



over S_{zx} via

$a \mapsto$ ○

$b \mapsto$ ○

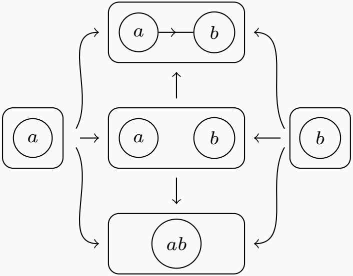
These 1-cells and 2-cells generate an SMCC sub-bicategory

<u>**zx**</u>

of **Rewrite**.

the zx-calculus – translating to Rewrite

To force the wire to act like the identity, we add the 2-cell



over S_{zx} via

$a \mapsto \circ$

$b \mapsto \circ$

These 1-cells and 2-cells generate an SMCC sub-bicategory

$$\underline{\mathbf{zx}}$$

of **Rewrite**.

the zx-calculus – a bicategory

Denote by **zx** the category with

(objects) $\mathbb{N}$

(morphisms) zx-diagrams modulo zx-relations

Theorem (C.)

Let $||\underline{\mathbf{zx}}||$ *be the category with*

(objects) *the 0-cells of* $\underline{\mathbf{zx}}$

(morphisms) *the 1-cells of* $\underline{\mathbf{zx}}$ *up to the 2-cells*

Then $||\underline{\mathbf{zx}}||$ *is equivalent to* **zx**

This equivalence is witnessed by the functor described in the above translation process.

Denote by **zx** the category with

(objects)	$\mathbb{N}$
(morphisms)	zx-diagrams modulo zx-relations

Theorem (C.)

Let $||\underline{\mathbf{zx}}||$ *be the category with*

(objects)	*the 0-cells of* $\underline{\mathbf{zx}}$
(morphisms)	*the 1-cells of* $\underline{\mathbf{zx}}$ *up to the 2-cells*

Then $||\underline{\mathbf{zx}}||$ *is equivalent to* **zx**

This equivalence is witnessed by the functor described in the above translation process.

the zx-calculus – a bicategory

Denote by **zx** the category with

(objects)	$\mathbb{N}$
(morphisms)	zx-diagrams modulo zx-relations

Theorem (C.)

Let $||\underline{\mathbf{zx}}||$ be the category with

(objects)	*the 0-cells of* $\underline{\mathbf{zx}}$
(morphisms)	*the 1-cells of* $\underline{\mathbf{zx}}$ *up to the 2-cells*

Then $||\underline{\mathbf{zx}}||$ is equivalent to **zx**

This equivalence is witnessed by the functor described in the above translation process.

in conclusion

conclusion

The benefits of this framework is...

- this process is sufficiently general to work with other graphical languages

- it gives a syntax that is bicategorical with symmetric monoidal and compact closed structure

- it should be straightforward, in concept, to include iterated rewrites

conclusion

The benefits of this framework is...

- this process is sufficiently general to work with other graphical languages

- it gives a syntax that is bicategorical with symmetric monoidal and compact closed structure

- it should be straightforward, in concept, to include iterated rewrites

conclusion

The benefits of this framework is...

- this process is sufficiently general to work with other graphical languages
- it gives a syntax that is bicategorical with symmetric monoidal and compact closed structure
- it should be straightforward, in concept, to include iterated rewrites

thank you