UNIVERSITY OF CALIFORNIA
RIVERSIDE

Rewriting Structured Cospans: A Syntax For Open Systems

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Mathematics

by

Daniel Cicala

June 2019

Dissertation Committee:

Dr. John C. Baez, Chairperson
Dr. Wee Liang Gan
Dr. Jacob Greenstein

The Dissertation of Daniel Cicala is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

First and foremost, I would like to thank my advisor John Baez. In these past few years, I have learned more than I could have imagined about mathematics and the job of doing mathematics. I also want to thank the past and current Baez Crew for the many wonderful discussions. I am indebted to Math Department at the University of California, Riverside, which has afforded me numerous opportunities to travel to conferences near and far.

Almost certainly, I would never have had a chance to pursue my doctorate had it not been for my parents who were there for me through every twist and turn on this, perhaps, too scenic route that I traveled.

Most importantly, this project would have been impossible without the full-hearted support of my love, Elizabeth.

I would also like to acknowledge the previously published material in this dissertation. The interchange law in Section 3.1 was published in [15]. The material in Sections 3.2 and 3.3 appear in [16]. Also, the ZX-calculus example in Section 4.3 appears in [18].

Elizabeth. It's finally over, baby!

ABSTRACT OF THE DISSERTATION

Rewriting Structured Cospans: A Syntax For Open Systems

by

Daniel Cicala

Doctor of Philosophy, Graduate Program in Mathematics
University of California, Riverside, June 2019
Dr. John C. Baez, Chairperson

The concept of a system has proliferated through natural and social sciences. While myriad theories of systems exist, there is no mathematical general theory of systems. In this thesis, we take a first step towards formulating such a theory. Our focus is on developing a syntax for compositional systems equipped with a rewriting theory. We pull from category theory and linguistics to accomplish this. The basic syntactical unit is a structured cospan and rewriting is introduced via the double pushout method. Two versions of rewriting are proposed: one that tracks intermediate steps and another disregards them. Benefits and drawbacks of both versions are discussed. We apply our results to the decomposition of closed systems, obtaining a structurally inductive viewpoint of rewriting such systems.

Contents

List of Figures

Chapter 0

Introduction

Systems exist everywhere and there are many different languages used to describe them. The diversity of languages reflect those who study systems. Physicists, chemists, biologists, ecologists, economists, sociologists, linguists, mathematicians, computer scientists all work with systems and all have their own idiosyncratic methods to describe them. This parallels diversity in the natural languages where location and communication needs are but two factors contributing to a language's development.

Just as linguists glean knowledge about humanity from studying languages, we can glean knowledge about our world from studying languages of systems. Still, no fully general mathematical theory of systems exists. Should it?

We say 'yes'. To develop a fully general mathematical theory of systems is a worthy pursuit. Successfully creating a formal language of systems can bestow many gifts. For instance, with a better understanding of systems, engineers get a better toolkit for their designs. One such engineered system, the power grid—a keystone to our way of life—is vulnerable due to increased energy demands inflicted by climate change [50]. A better understanding of systems eases translation across disciplines. By placing, say, systems ecology [52] and the programming language R [48] in the same formalism, ecological models can be more faithfully translated into mathematical models. A better understanding of systems directs us to new paths of inquiry. An abstract understanding of systems places them into a “space of systems” where they can be compared and contrasted. With this space, we can craft analogies and narratives. This new perspective should present questions previously not apparent. So yes, aspiring to a general mathematical theory of systems is worthwhile.

Often, one studies *a* system. The social network described by Facebook is a single system frequently studied. Another is the logistics of shipping Amazon packages the world over. In reality, systems rarely exist in isolation. The Facebook network is affected by other social media networks. Amazon's shipping networks are affected by the economics of oil prices. That is, systems interact with each other to form new systems and this ought to be a component of an honest general mathematical systems theory. One way systems interact is to not exert any influence over each other, which should evoke to a mathematician the disjoint union operation. But to exert influence necessitates each system to have points on which the interaction can occur. For example, a point of interaction of a building's electrical system is an outlet, where one can connect a blender forming a composite electrical-blender system. A point of interaction with a pulley system is a dangling rope that one can pull, upon which we obtain the composite pulley-musculoskeletal system.

When connecting systems together, one may veer into the *principle of compositionality.* Compositionality is present when the whole of a system is equal to the sum of its parts. This can be exploited to great effect when analyzing complicated systems by allowing for its decomposition into simpler pieces. For instance, the physical system of two pendu-

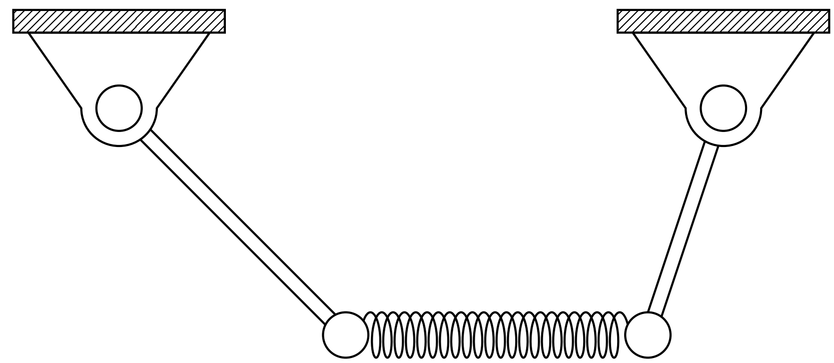


Figure 0.1: A compositional physical system

lums connected together with a spring (see Figure 0.1) can be fully analyzed by separately considering the two pendulums and the spring. In mathematical terms, this amounts to coupling the corresponding differential equations.

Compositionality lies in contrast to so-called *emergent systems* where new features burst into existence upon connection. Life is believed to have emerged from complex systems of ribonucleic acid (better known as RNA). No sign of life is present in a single RNA molecule but somehow life appears in a system comprising only RNA.

The two methods of interaction described above, disjoint union and connecting along points of interaction, have clear analogies to fundamental mathematical concepts: addition and composition. From the many areas of mathematics, the one that stands out in its singular focus on addition and composition is the theory of monoidal categories. Category theory takes as fundamental the composition of 'arrows' and endowing a category with a 'monoidal structure' allows us to "add" the arrows together. Therefore, monoidal categories are an excellent foundation on which to base a general mathematical theory of compositional systems.

What is this thesis about?

Here, we take first steps in towards building a theory of compositional systems. What do these first steps look like? In short, we are setting up a syntax for compositional systems.

The term 'syntax' appears most often in linguistics where it refers to rules and principles that an arrangement of words must satisfy to be a well-formed sentence. It means roughly the same for us except that we are working with compositional systems, not words and sentences. In this analogy, compositional systems correspond to both words and sentences in that, instead of building sentences by arranging words, we are building larger systems by connecting smaller systems. To do so, we need a set of rules and principals governing how to connect systems together.

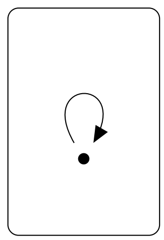
The yin to syntax's yang is *semantics*. This concept, also from linguistics, refers to the meaning of a sentence. In our context, semantics refers to the *behavior* of a system. Resistor circuits are a nice example to highlight the distinction between syntax and semantics. First, recall that resistors wired in series have the same resistance as a single resistor with the aggregate resistance. Now, while a circuit with a 25Ω and 35Ω resistors wired in series is syntactically different from a circuit with a single 60Ω resistor, their resistance is equal meaning they have the same semantics. While semantics is important to any theory of systems, we do not directly consider it in this thesis. However, we do consider it indirectly.

Granting that syntax and semantics are separate entities, it is often useful for syntax to *reflect* semantics. We do not want to say that the two resistor circuits are *equal*. That is too strong. But we do want to establish a formal *relationship* between them. More than that, we want a way to propagate this relationship through a suitable space of circuits so that every circuit with resistors wired in series relates to the circuit with a single resistor in their place. Of course, our method of propagating such a relationship must be abstract enough to handle more systems than just resistor circuits.

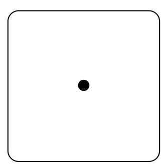
Again we turn to linguistics, this time the study of formal languages. These are different from natural languages like English, Italian, or Afrikaans that ebb and flow under so many social forces. Formal languages are designed and can be controlled. They can approximate natural languages. This makes them useful in studying natural languages. However, the "formal languages" we are interested in do not contain words and sentences. The formal languages we are interested in are systems connected together.

From the study of formal languages comes *rewriting theory*. Originally used to generate well-formed sentences, rewriting has since evolved through being studied by mathematicians, logicians, and computer scientists for whom it provides a mechanism to replace terms with distinct but equivalent terms. As mentioned above, rewriting is syntactic but meant to reflect semantics. This means that rewriting relates syntactical terms if they behave in the same way. For example, a programming language that can perform addition would have a 'rewrite rule' saying that '2+2' can be rewritten into '4' because they mean the same thing. There would *not* be a rule rewriting '2+2' into '5' because they never mean the same thing. Moreover, rewriting theory provides a way to extend this rule to longer strings containing '2+2', for instance, the string '(3*(2+2))/(2+2+3)' can be rewritten into

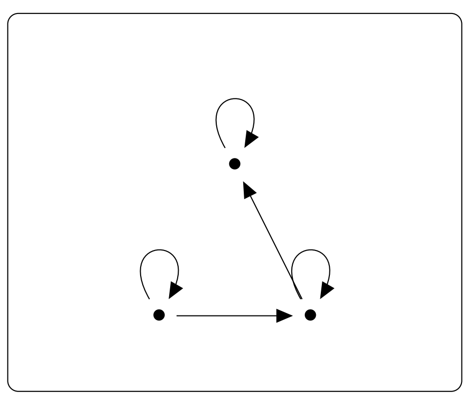
‘(3*4)/(4+3)’. Crucially, rewriting also prevents erroneous applications such as rewriting ‘2+2(x+y)’ into ‘4(x+y)’. The first expansion of rewriting theory beyond the realm of characters and words was into combinatorial graphs where rewrite rules tell us when one graph can replace another. If we were modeling the internet as a directed graph with websites as nodes and a link from one website to another as edges, then we are likely uninterested in self-loops, which represent a webpage that links to itself. So we can introduce a rule that deletes self-loops. Informally, this would say that the graph



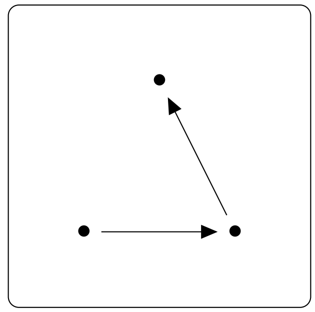
can be rewritten into the graph



This rule can be extended to remove loops from more complicated graphs like



being rewritten into



To formalize this requires abstract mathematics, namely category theory. Fortunately, because the category theory involved in rewriting graphs is so abstract, we can use it to rewrite syntax developed for compositional systems.

What does rewriting do for us? It allows us to simplify our syntax, whether that syntax is based on characters or combinatorial graphs or other types of systems. The ability to simplify syntax is a powerful tool for any would-be analyst simply because of how complex syntactical terms can grow. The graph model of the internet is massive with over 1.5 billion nodes, each an individual website.

Our goal in this thesis is to present a syntax for compositional systems proposed by Baez and Courser [5] called 'structured cospans' and combine it with a theory of rewriting.

A road map for the thesis

The larger goal of creating a general mathematical theory for compositional systems is still aspirational, but we stride within these several chapters, developing a syntax and rewriting theory. To assist the reader in navigating these chapters, we sketch their contents and give the highlights. We visualize the dependencies between the chapters with Figure 0.2.

In Chapter 1, we present a syntax for compositional systems. Baez and Courser introduced this syntax under the name 'structured cospans'. A cospan is a diagram in a category with shape

$$a \xrightarrow{f} b \xleftarrow{g} c$$

where a, b, c are objects in the category and f, g are arrows in the category. For a structured cospan, we have a specific interpretation in mind: the object b is a system with inputs a and outputs c. The arrows f and g maps the inputs and outputs to the system.

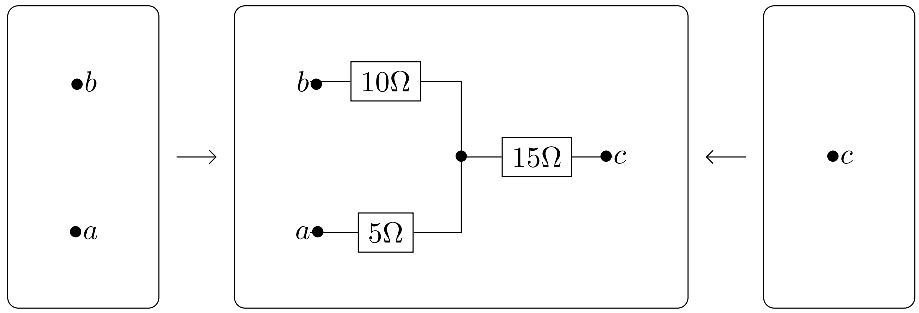
To formalize this perspective, our starting data is an adjunction

$$\mathsf{A} \underset{R}{\overset{L}{\rightleftarrows}} \mathsf{X} \quad (L \dashv R)$$

between topoi A and X. We interpret A as a topos whose objects are the *interface types*; that is the objects that can serve as inputs or outputs to our systems, and X as a topos whose objects are the system types. Often, A is the topos Set of sets and functions. And X can be whatever system we are working with, for example a category whose objects are resistor circuits. The functor $L\colon \mathsf{A} \to \mathsf{X}$ translates the interface types into degenerate system types so that they can interact via a structured cospan, which is a cospan of the form

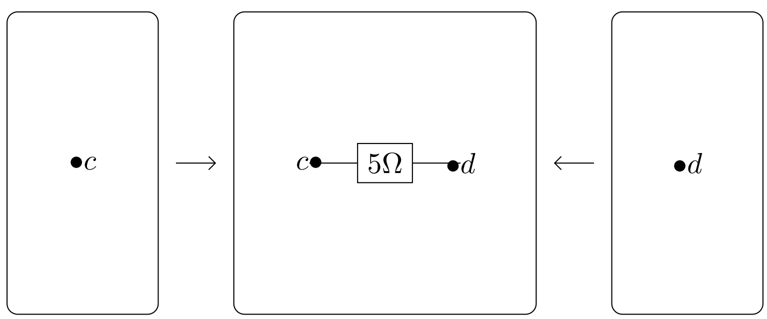
$$La \xrightarrow{f} x \xleftarrow{g} Lb$$

This structured cospan is a system x with inputs La and outputs Lb. A resistor circuit as a structured cospan would look like

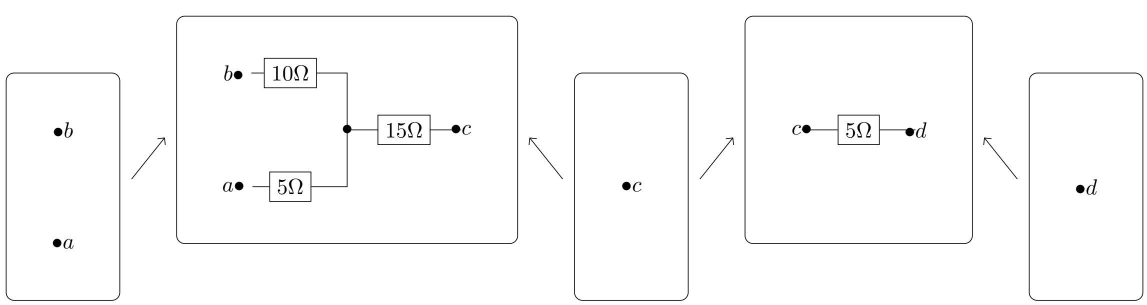


The left-hand graph $L(\{a, b\})$ gives the inputs and the right-hand graph $L(\{c\})$ gives the outputs.

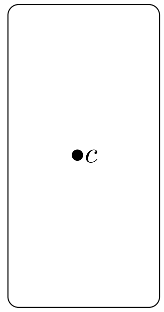
We devote Section 1.1 to composing structured cospans. As is standard in cospan categories, composition uses pushout. For example, any resistor circuit with a single input, say



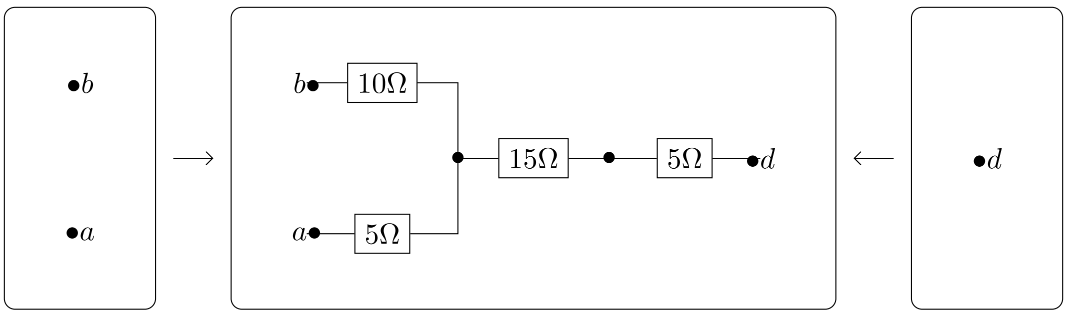
can be connected to the resistor circuit above that has a single output as follows



We then pushout over the common interface



to get the single structured cospan



that represents a single circuit with input nodes a, b and output node d.

Starting with the adjunction $L\colon \mathsf{A} \rightleftarrows \mathsf{X}\colon R$, where A and X are symmetric monoidal categories with their respective coproducts, we then package structured cospans into a compact closed category $({}_L\mathsf{Csp}, \otimes, 0_A)$ whose objects are the interface types, that is objects of A, and the arrows of type $a \to b$ are the structured cospans $La \to x \leftarrow Lb$.

Our stated goal is to introduce a rewriting theory to structured cospans. To do this, we must ensure that structured cospans are sufficiently nice to accommodate rewriting. This entails designing a topos where structured cospans are the objects. Constructing this topos is the topic of Section 1.2. We define a category ${}_L\mathsf{StrCsp}$ whose objects are structured cospans and whose arrows between the structured cospans $La \to x \leftarrow Lb$ and $La' \to x' \leftarrow Lb'$ are commuting diagrams

$$\begin{array}{ccccc} La & \longrightarrow & x & \longleftarrow & Lb \\ {\scriptstyle Lf}\downarrow & & {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle g} \\ La' & \longrightarrow & x' & \longleftarrow & Lb' \end{array}$$

in X. The main result of this section is

Theorem 8. For any adjunction

$$\mathsf{A} \underset{R}{\overset{L}{\rightleftarrows}} \mathsf{X} \quad (L \dashv R)$$

between topoi, the category ${}_L\mathsf{StrCsp}$ is a topos.

This result is the keystone that stabilizes the combination of structured cospans and rewriting. Because of this fact, structured cospans do accommodate a rewriting theory. By this, we mean that the local Church–Rosser and concurrency properties hold. We do not investigate these properties in this thesis, but Corradini, et. al. thoroughly discuss these properties [25]. We also show in Theorem 9 that constructing ${}_L\mathsf{StrCsp}$ is functorial in L.

Viewing structured cospans through the two categories ${}_L\mathsf{Csp}$ and ${}_L\mathsf{StrCsp}$ in which they appear, we note that they play two roles. In ${}_L\mathsf{Csp}$, structured cospans form the arrows. In ${}_L\mathsf{StrCsp}$, structured cospans form the objects. We combine these two perspectives into a single framework using double categories in Section 1.3. The final section of Chapter 1 sets the groundwork for rewriting structured cospans by defining spans of structured cospans.

In Chapter 2, we discuss the theory of rewriting with just enough detail to provide the reader with an appreciation for the subject and enough tools to read this text. We begin with its linguistic beginnings but quickly move to the axiomatization of the double pushout method of rewriting. The axioms of rewriting theory are captured in their full generality by so-called 'adhesive categories'. However, this is too general for our needs, so we restrict to rewriting in a topos, a type of adhesive category.

By fixing a topos T, we learn how to apply a rewrite rule, which manifests as a span

$$\ell \leftarrow k \rightarrow r$$

in T. We interpret this rule to say ℓ can be rewritten into r. We apply this rule by identifying a copy of ℓ inside another object ℓ' via an arrow $\ell \to \ell'$ of T and there are objects k' and r' of T fitting into a 'double pushout diagram'

$$\begin{array}{ccccc} \ell & \longleftarrow & k & \longrightarrow & r \\ \downarrow & & \downarrow & & \downarrow \\ \ell' & \longleftarrow & k' & \longrightarrow & r' \end{array}$$

We then say that ℓ' can be rewritten to r'. The double pushout diagram encodes that we first identify a copy of ℓ in ℓ', remove and replace it by r, and this results in r'. In this way, an initial set of rewrite rules propagate throughout T by collecting all possible applications of all the initial rules.

In Chapter 3, we introduce the first of two styles of rewriting structured cospans. A 'fine rewrite rule' of structured cospans is a diagram with shape

$$\begin{array}{ccccc} La & \longrightarrow & x & \longleftarrow & La' \\ \cong\uparrow & & \uparrow & & \uparrow\cong \\ Lb & \longrightarrow & y & \longleftarrow & Lb' \\ \cong\downarrow & & \downarrow & & \downarrow\cong \\ Lc & \longrightarrow & z & \longleftarrow & Lc' \end{array}$$

taken up to isomorphism. The marked arrows are monic and an isomorphism to another fine rewrite of structured cospans

$$\begin{array}{ccccc}
La & \longrightarrow & x & \longleftarrow & La' \\
{\scriptstyle\cong}\uparrow & & \uparrow & & \uparrow{\scriptstyle\cong} \\
Lb & \longrightarrow & y' & \longleftarrow & Lb' \\
{\scriptstyle\cong}\downarrow & & \downarrow & & \downarrow{\scriptstyle\cong} \\
Lc & \longrightarrow & z & \longleftarrow & Lc'
\end{array}$$

is an invertible arrow $y \to y'$ such that the evident diagrams commute. Admittedly, we are being rather brusque by saying 'evident', though Definition 14 spells this out in detail. The main result of this section is the construction of a double category ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ whose objects are interface types from A, horizontal arrows are structured cospans, and squares are fine rewrites of structured cospans. This result is listed as Proposition 25. Proving the interchange law is quite technical, so we devote all of Section 3.1 to this. In Section 3.2 we equip the double category ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ with a symmetric monoidal structure. In the final section of Chapter 3, we appease those readers who prefer bicategories to double categories. There, we extract from the double category ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ a compact closed bicategory ${}_L\mathbf{FineRewrite}$.

In Chapter 4, we introduce the counterpart to fine rewriting called 'bold rewriting'. A bold rewrite rule is the connected component of a diagram

$$\begin{array}{ccccc} La & \longrightarrow & x & \longleftarrow & La' \\ \cong\big\uparrow & & \big\uparrow & & \big\uparrow\cong \\ Lb & \longrightarrow & y & \longleftarrow & Lb' \\ \cong\big\downarrow & & \big\downarrow & & \big\downarrow\cong \\ Lc & \longrightarrow & z & \longleftarrow & Lc' \end{array}$$

By connected component, we mean the equivalence class generated by relating the above diagram to

$$\begin{array}{ccccc} La & \longrightarrow & x & \longleftarrow & La' \\ \cong\big\uparrow & & \big\uparrow & & \big\uparrow\cong \\ Lb & \longrightarrow & y' & \longleftarrow & Lb' \\ \cong\big\downarrow & & \big\downarrow & & \big\downarrow\cong \\ Lc & \longrightarrow & z & \longleftarrow & Lc' \end{array}$$

if there is an arrow $y \to y'$ such that the evident diagrams commute. This chapter largely mirrors that on fine rewriting. We define a double category ${}_L\mathbb{B}\mathbf{old}\mathbb{R}\mathbf{ewrite}$ whose objects are the interface types from A, whose horizontal arrows are structured cospans, and whose squares are bold rewrites. Again, we extract a bicategory from the double category. We show that this bicategory ${}_L\mathbf{BoldRewrite}$ is a bicategory of relations.

In the final section of Chapter 4, we illustrate bold rewriting with the ZX-calculus. This is a language consisting of string diagrams used to reason about a corner of quantum mechanics favored by quantum computer theorists. Coecke and Duncan, the inventors of the ZX-calculus, organized it into a dagger compact category whose arrows are the very diagrams that constitute the ZX-calculus. Using the machinery laid out in this chapter, we expand this dagger compact category to a symmetric monoidal double category that encodes the ZX-calculus. The benefit of this is that, instead of merely equating ZX-calculus diagrams when there exists a rewrite rule between them, the squares of our double category actually witness these equations. This should satisfy mathematical constructivists. Overall, the double category structure we build is richer than the category.

We complete this thesis with Chapter 5. Most academic work on systems focuses on *closed systems*, those with an empty interface. Physicists often represent a closed system with a phase space. Chemical reactions are worked out as if the rest of the world does not exist (or is reduced to a triviality). Petri nets do not interact with each other. Markov chains are never combined. One hope of this research program is to provide the mathematical resources to change this, so that *open networks* become the norm. Then the phase spaces of two different systems could be connected. Chemical reactions could more easily consider their environment. Petri nets and Markov chains could be composed together. This final chapter motivates using open systems to study closed systems.

Specifically, we construct a mechanism to rewrite closed systems using structural induction. That is, we can decompose a given closed system into open sub-systems each of which can be rewritten independently of each other. After simplifying each sub-system via this rewriting procedure, we reconnect them together into an equivalent version of the original closed system. In short, we introduce an inductive process that simplifies closed systems. This is characterized by the following theorem.

Theorem 74. Fix a adjunction $L\colon \mathsf{A} \rightleftarrows \mathsf{X}\colon R$ with monic counit. Let (X, P) be a grammar such that for every X-object x in the apex of a production of P, the Heyting algebra $\mathrm{Sub}(x)$ is well-founded. Given g, $h \in \mathsf{X}$, then $g \rightsquigarrow^* h$ in the rewriting relation for a grammar (X, P) if and only if there is a square

$$\begin{array}{ccccc} LR0 & \longrightarrow & g & \longleftarrow & LR0 \\ \uparrow & & \uparrow & & \uparrow \\ LR0 & \longrightarrow & d & \longleftarrow & LR0 \\ \downarrow & & \downarrow & & \downarrow \\ LR0 & \longrightarrow & h & \longleftarrow & LR0 \end{array}$$

in the double category $\mathrm{Lang}({}_L\mathsf{StrCsp}, P')$.

In less technical terms, this theorem says that, under suitable hypotheses, one closed system can be rewritten into another precisely when there is a square between their corresponding structured cospans. This square is built inductively from rewrites between open sub-systems.

This marks the end of the thesis proper. However, we anticipate that the results contained within may be of interest to a wide audience including certain network theorists, systems theorists, computer scientists, and mathematicians. Therefore, we organized the thesis so that background material is mostly confined to the appendices. This way, it will not distract those familiar with it and it is readily available to those readers who are not. Here are the topics of the appendices.

Appendix A.1 Enriched categories and bicategories. This material is used in Sections 3.3 and 4.2 where bicategories are extracted from double categories;

Appendix A.2 Internalization and double categories are useful throughout as double categories are a main character in our story. Also, this section covers internal monoids which are used to show that the bicategory of bold rewrites ${}_L\mathbf{BoldRewrite}$ is a bicategory of relations in Section 4.2;

Appendix A.3 Bicategories of relations, which are used in Section 4.2;

Appendix A.4 Duality in bicategories, which is used for the bicategories in both Sections 3.3 and 4.2;

Appendix A.5 Adhesive categories, which are the result of axiomatizing rewriting theory and, though useful throughout because of the central role played by rewriting in this thesis, we pack most of the required information into the next section of the appendix;

Appendix A.6 Topoi, which are used throughout.

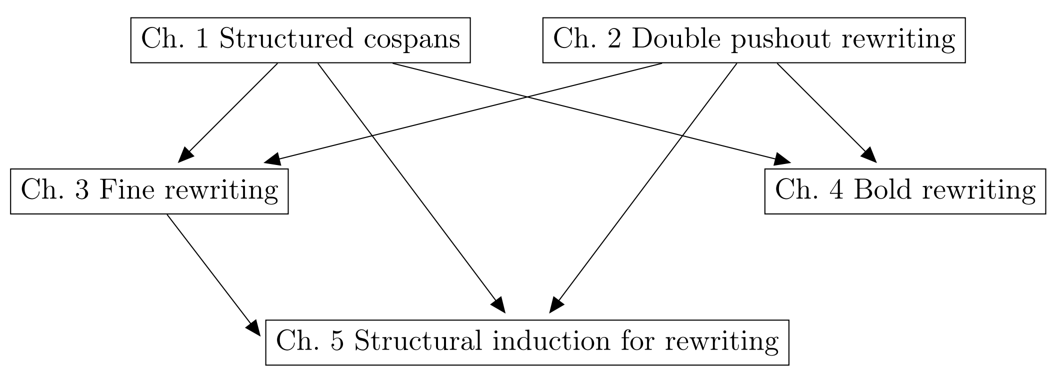


Figure 0.2: Chapter dependencies

Global notation and assumptions

As usual in mathematics, we systematically select notation to orient the reader. Here, we lay out the logic behind our notation.

Categorical structures Three types of categorical structure are used throughout:

- Categories and topoi, which we denote with the font A, X, C, T. A and X are used are topoi used to build structured cospans, C is a generic category, and T is a generic topos.
- Bicategories, which we denote with bold font $\mathbf{C}$. The two most important bicategories for us are **FineRewrite** and **BoldRewrite**.
- Double categories, which we denote with blackboard bold font $\mathbb{C}$. The two most important double categories for us are $\mathbb{F}$**ine**$\mathbb{R}$**ewrite** and $\mathbb{B}$**old**$\mathbb{R}$**ewrite**.

Objects Objects in a category are denoted by lower case letters. The most common categories we work with are labeled as A and X and we refer to their respective objects are $a, b, c, \ldots$ and $\ldots x, y, z$.

Arrows Both categories and graphs frequent these pages. To distinguish whether a drawing is of a graph or a diagram in a category, look at the arrow tips. An arrow in a category uses

$$\longrightarrow$$

while an arrow in a graph uses

$$-\!\!\!\blacktriangleright$$

Also, we reserve tailed arrows

$$\rightarrowtail$$

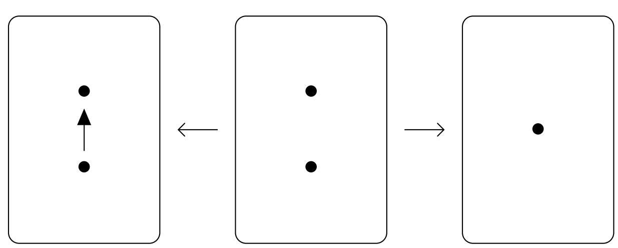
to mean a monic arrow in a category. We do not often refer to specific arrows, but when we do, we use lower case letters f, g, h, etc. Occasionally, if an arrow is of particular importance we distinguish it with a lower case Greek letter.

2-arrows We refer to 2-arrows in higher categories using Greek letters. In particular, when using λ, ρ, α without explicitly stating what they are, then they are monoidal coherence maps for left unity, right unity, and associativity.

Rewrite relation Central to the theory of rewriting is the 'rewriting relation'. This is built in two steps from a given rewriting system. First, $a \rightsquigarrow b$ says that a can be rewritten into b by applying a single rewrite rule. The rewriting relation, which we denote by $\rightsquigarrow^*$, is the reflexive and transitive closure of $\rightsquigarrow$.

Systems and networks Our work concerns both open and closed systems, the former more prominently. Therefore, when using the term system or network without a qualifier, we mean 'open' by default. Only when we explicitly say 'closed' do we mean a closed system or network.

Cospans of graphs Many graph morphisms are drawn throughout the following pages. Too much detail tends to clutter the drawings, so we leverage the geometry of the page to suggest the definition of the morphisms. Only in cases where this suggestion lacks clarity do we explicitly spell out the meanings. In Chapter 2, we see the drawing



which consists of three directed graphs each in a box and two graph morphisms. Note the differences between the arrow heads. Also, the definitions of these graph morphisms are not explicitly spelled out, but they are apparent nonetheless because of the location of the graph nodes on the page.

Chapter 1

Structured cospans

Researchers traditionally study *closed systems*, those that lack the ability to interact with outside agents. A research program initiated by John Baez centralizes the study of *open systems*, those with ability to interact with outside agents [6, 7, 8, 9].

In this thesis, our primary example of an open system is an *open graphs*. We use them throughout to illustrate new concepts and definitions. For this reason, we start with a set theoretical definition of open graphs and modify our understanding of them in parallel to building our structured cospan formalism. We use this approach to provide a concrete example to ground us through the development of our theory. Open graphs are not new [27, 35], but our structured cospan perspective is new.

Definition 1 (Open and closed graphs)**.** An **open graph** $G := (E, N, r, s, t, I, O)$ is a directed reflexive multi-graph (E, N, r, s, t) equipped with two non-empty subsets $I, O \subseteq N$ of nodes. We call elements of I the inputs of the graph and the elements of O the outputs of the graph. In the case that I and O are empty, then we call G a **closed graph**.

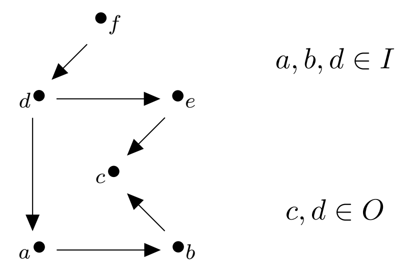
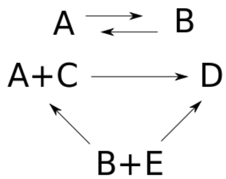


Figure 1.1: An open graph

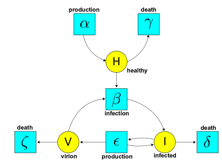
This definition deserves several remarks. First, note that a closed graph is simply a graph in the classical sense. We append the qualifier 'closed' to highlight the fact that it has no inputs or outputs. Second, the terms 'input' and 'output' *do not* imply causal structure or directionality. Finally, the author prefers reflexive graphs to non-reflexive graphs because *(i)* they are truncated simplicial sets so have nicer topological features, *(ii)* unlike graphs, the "points" of reflexive graphs (the nodes) correspond to maps from the terminal object, and *(iii)* the category of reflexive graphs RGraph is monadic over Set.

An open graph is illustrated in Figure 1.1. We suppress the reflexive loops in drawing reflexive graphs. In that figure, the nodes are a, b, c, d, e, and f. The input nodes are a, b, and d. The output nodes are c and d.

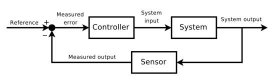
A non-exhaustive list of other systems of interest to Baez's research program are Petri nets [49], Markov processes [6], passive linear circuits [8], reaction networks [9], the ZX-calculus [18]. See Figure 1.2 for depictions of these various systems. These systems are traditionally studied as closed systems. To "open" them, they need an interface along which compatible systems can be connected. This is the purpose of introducing the input and output nodes.



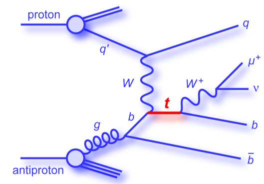
Chemical Reaction Network



Petri Net



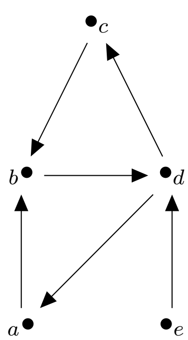
Control Network



Feynman Diagram

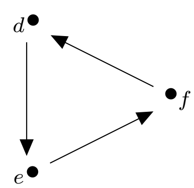
Figure 1.2: Various systems

Example 2 (Connecting open graphs)**.** We can connect together two open graphs when the inputs of one is equal to the outputs of the other. To illustrate this, consider the open graphs



$a, c, d \in$ `inputs`

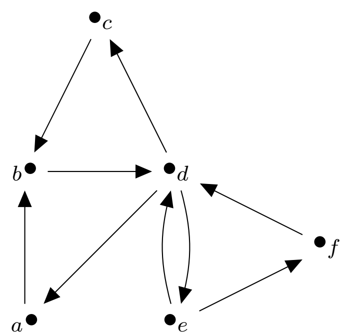
$d, e \in$ `outputs`



$d, e \in$ `inputs`

$e, f \in$ `outputs`

Connect these open graphs by gluing like-nodes together. This results in



$a, c, d \in I$

$e, f \in O$

The operation of gluing open graphs together can be defined set theoretically. However, we prefer to define this operation as a composition of morphisms in an appropriate category. This ensconces the gluing operation as fundamental. In this chapter, we discuss the formalism of structured cospans. These offer a language better equipped to describe open systems than do more traditional set theory styled definitions.

A **cospan** in a category is a pair of arrows

$$x \to y \leftarrow z$$

with common codomain. A structured cospan is a special sort of cospan. The rough idea of a structured cospan is that the common codomain is some system and the domains are the inputs and outputs of that system. In other words, we interpret a structured cospan as the diagram

$$\texttt{inputs} \xrightarrow{\iota} \texttt{system} \xleftarrow{\omega} \texttt{outputs}$$

where ι chooses the part of the system to serve as inputs and ω chooses the outputs. Section 1.1 is devoted to constructing a category whose arrows are the structured cospans.

The motivation for using composition to describe the connection of open systems also has a philosophical component. We study systems through the lens of compositionality.

A pithy description of compositionality is "the opposite of emergent". That is, the behavior of a compositional system is fully determined by the behavior of the sub-systems comprising it. Here are some examples of compositionality.

- Set functions are compositional. Given functions $f: X \to Y$ and $g: Y \to Z$, then we know everything about the composite function $g \circ f: X \to Z$.

- Given two computer programs, one that approximates a smooth solution to a given differential equation and another that outputs a visualization of a smooth function, then we know that the composite program renders a drawing of an approximate smooth solution to a given differential equation.

- If one manufacturing line inputs various wood pulp and outputs paper and another manufacturing line inputs paper and outputs notebooks, then the composite manufacturing line inputs wood-pulp and outputs notebooks.

Already, we have mentioned examples of systems we are interested in. Each of these examples are useful tools applied by various scientists or engineers. Naturally, each formalism has developed idiosyncrasies, inflating the differences between them. However, there remain clear qualitative similarities between the different formalisms that ought to be exploited to transport results determined with one formalism to results about another formalism. As cross-disciplinary collaboration increases, the importance of translating between formalisms grows. We propose the structured cospan serve as a medium of translation.

The analogy to languages runs deeper than mere translation. Indeed, languages have both syntactic and semantic content. Systems do too. We intend to clearly delineate between the two. William Lawvere's 'functorial semantics' [45] serves as inspiration. This is a categorical approach to universal algebra where algebraic theories are separated into two pieces: one category capturing the structure and properties of a type of algebraic object A and another category containing the "stuff" underlying an instance of A (e.g. the underlying set). A functor between the categories selects an instance of an algebraic object of type A. In our context, we separate open systems, not algebraic object types, into two categories. One category contains the system syntax and the other category the system semantics. In this perspective, categories with structured cospans for arrows serve as syntax and their compositionality manifests as a functor into a category of semantics.

In this chapter, we define structured cospans and two categories in which they appear. The first categories ${}_L\mathsf{Csp}$ was introduced by Baez and Courser [5] and encodes open systems are arrows. The second category ${}_L\mathsf{StrCsp}$ houses the morphisms of structured cospans which are used to define their rewriting. To ensure that structured cospans support a good theory of rewriting, we show that ${}_L\mathsf{StrCsp}$ is a topos. We close this chapter by combining ${}_L\mathsf{Csp}$ and ${}_L\mathsf{StrCsp}$ into a double category. Most of the work in this chapter appeared previously in [17].

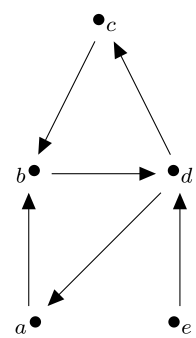
1.1 Structured cospans as a compositional framework

In this section, we define a structured cospan and fit them as arrows into a category. There are several technical components we need to consider, each serving a purpose. So instead of providing the definition here, we build up to it discussing each technicality along the way.

When thinking of a structured cospan, we have in mind a diagram

$$\texttt{inputs} \rightarrow \texttt{system} \leftarrow \texttt{outputs}$$

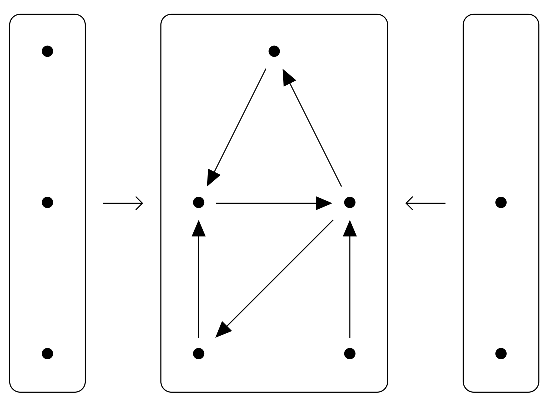
sitting in a category. Often, the inputs and outputs of a system will be sets. For sets to exist in the same category as the systems—as is needed to have the inputs, outputs, and system represented in the same diagram—we consider sets as degenerate systems. For instance, the open graph



$$a, c, d \in \texttt{inputs}$$

$$d, e \in \texttt{outputs}$$

presented using Definition 1 is realized as the structured cospan



(1.1)

Inside this picture, we have three graphs enclosed in the boxes. The left and right-most graphs are really just sets considered as edgeless graphs or, in our parlance, as "degenerate systems". The arrows between the graphs are graph morphisms defined as suggested by the layout. These arrows choose the components of the central graph to serve as inputs and outputs.

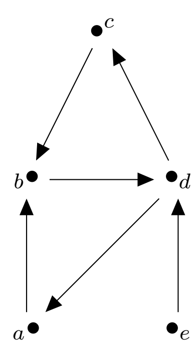
To model open graphs with structured cospans, we do not want to allow arbitrary graphs in the feet of the cospan. We only want sets qua edgeless graphs. To accomplish this, we define a functor

$$L\colon \mathsf{Set} \to \mathsf{RGraph} \tag{1.2}$$

that turns a set a into a graph La with node set a and no non-reflexive edges. Now, the open graph in (1.1) has form

$$La \to x \leftarrow Lb$$

where a is a three element set, b is a two element set, and x is the graph



The functor L in (1.2) is crucial to the definition of a structured cospan. To capture open systems more general than open graphs, we allow L to be of type $\mathsf{A} \to \mathsf{X}$ for categories A and X. Now, a structured cospan based on a functor $L\colon \mathsf{A} \to \mathsf{X}$ is a cospan in X of the form $La \to x \leftarrow Lb$. We do not use this as a definition because for rewriting we require more from L, A, and X.

One such need is to construct a category where structured cospans $La \to x \leftarrow Lb$ are arrows. Hence, given another structured cospan $Lb \to y \leftarrow Lc$, we need to define the composite. As is typical in cospan categories [11], we compose by pushout. That is, the composite of the structured cospans

$$La \to x \leftarrow Lb \quad \text{and} \quad Lb \to y \leftarrow Lc$$

is the structured cospan

$$La \to x +_{Lb} y \leftarrow Lc$$

Using this composition, we henceforth require X to have pushouts.

Let us unpack this composition. We have a pair of systems x and y, where the outputs of x are chosen by the arrow $Lb \to x$ and the inputs of y are chosen by the arrow $Lb \to y$. Considered together, we have a span $x \leftarrow Lb \to y$. The pushout of this span is

$$\begin{array}{ccc} Lb & \longrightarrow & y \\ \downarrow & & \downarrow \\ x & \longrightarrow & x +_{Lb} y \end{array}$$

A useful intuition of this pushout is that the system $x +_{Lb} y$ is obtained by gluing the image of Lb in x to the image of Lb in y. The composite system $x +_{Lb} y$ has inputs chosen by the composite $La \to x \to x +_{Lb} y$ and outputs chosen by the composite $Lc \to y \to x +_{Lb} y$. The composite structured cospan is then

$$La \to x +_{Lb} y \leftarrow Lc$$

From this composition, a functor $L \colon \mathsf{A} \to \mathsf{X}$ where X has pushouts gives a category whose objects are those of A and whose arrows of type $a \to b$ are structured cospans $La \to x \leftarrow Lb$. For our needs, however, we ask more of L, A, and X.

In Chapter 2, we introduce a theory of rewriting structured cospans. To do so, we need a *topos*—discussed in Appendix A.6—in which structured cospans are the objects.

We find this topos in Theorem 8 and so our theory requires the assumptions held there. Precisely, we need L to be a pullback preserving left adjoint and for both A and X to be topoi. Section 1.2 contains further discussion about how these assumptions figure into our goal of modeling systems. In the meantime, we fix these assumptions once and for all.

Fix a adjunction

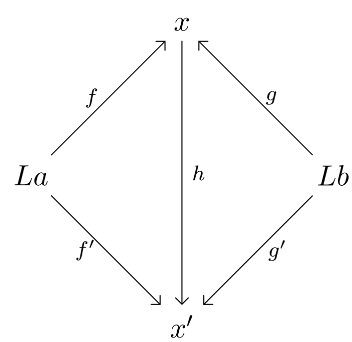
$$\mathsf{A} \underset{R}{\overset{L}{\underset{\bot}{\rightleftarrows}}} \mathsf{X}$$

with L preserving pullbacks. How does our theory of systems map onto this adjunction? Interpret the topos X as a category whose objects are systems and whose arrows are the homomorphism of systems. These systems are *closed*, in that they cannot interact with outside agents, specifically other systems of the same type. To provide a compositional structure to these systems, we introduce a topos A that we interpret as a category of interfaces types and their morphisms. By transporting the interface types along L, we can include them in the cospans with systems in X. The arrows of a structured cospan equip a system with its interface. Once equipped with a (non-empty) interface, a system is *open* in that they can interact with compatible systems. There is no explicit role for R. It is the properties of L that exists in light of L being an adjunction that we use. However, we can still interpret R as returning the maximal (by inclusion) interface of a system. The existence of R is a side-effect that we leverage in Theorem 8.

Using the adjunction $L \colon \mathsf{A} \rightleftarrows \mathsf{X} \colon R$ we construct a compositional framework having systems as arrows in a cospan category. Composition of arrows uses pushout which encodes connecting a pair of compatible systems. Because cospans are too general for our needs, we restrict our attention to structured cospans.

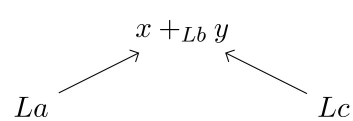
Definition 3 (Structured cospan)**.** A **structured cospan** is a cospan of the form $La \to x \leftarrow Lb$. When we want to emphasize L, we use the term L**-structured cospans**.

Structured cospans fit into two different categories that are central to our theory. The first one, that we meet now, was proved by Baez and Courser to actually be a category [5]. To start, we define an isomorphism of structured cospans from $La \to x \leftarrow Lb$ to $La \to x' \leftarrow Lb$ to be an invertible arrow $h \colon x \to x'$ in X that fits into the commuting diagram



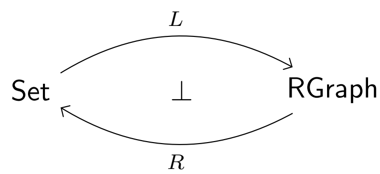
Definition 4. The category ${}_L\mathsf{Csp}$ has as objects the objects of A and arrows $a \to b$ are structured cospans $La \to x \leftarrow Lb$ up to isomorphism.

Composing $La \to x \leftarrow Lb$ with $Lb \to y \leftarrow Lc$ uses pushout

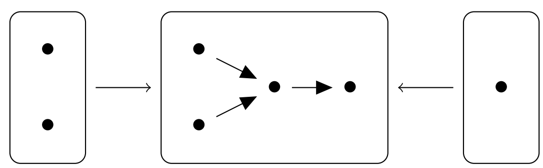


In a sense, pushouts glue objects together making it a sensible way to model system connection. The composition above is like connecting along Lb. Using structured cospans, we now improve our earlier definition of open graphs.

Example 5. There is a geometric morphism (see Definition 112)

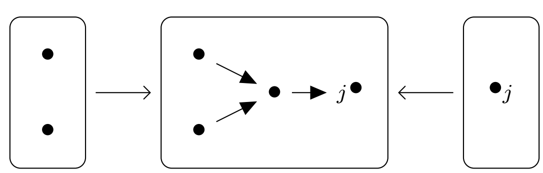


where Rx is the node set of graph x and La is the edgeless graph with node set a. An **open graph** is a cospan $La \to x \leftarrow Lb$ for sets a, b, and graph x. An illustrated example, with the reflexive loops suppressed, is

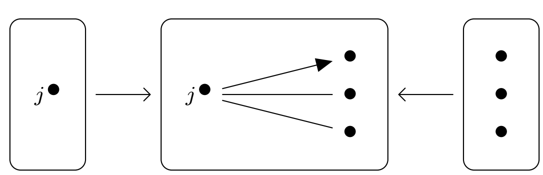


The boxed items are graphs and the arrows between boxes are graph morphisms defined as suggested by the illustration. In total, the three graphs and two graph morphisms make up a single open graph whose inputs and outputs are, respectively, the left and right-most graphs.

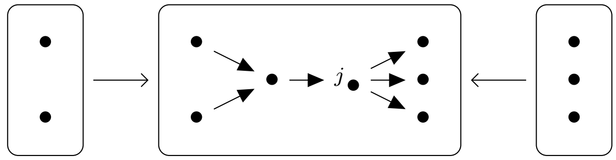
Open graphs are compositional. For instance, we can compose



with



to get the open graph



which is obtained by composing structured cospans. Note that this is composition in ${}_L\mathsf{Csp}$ for $L\colon \mathsf{Set} \to \mathsf{RGraph}$.

In general, interpret $La \to x \leftarrow Lb$ as consisting of a system x equipped with an interface comprised of inputs La and outputs Lb. The terms 'input' and 'output' do not imply any causal structure. They are merely meant to provide a way to connect a pair of systems along a proper subset of their interfaces. Decomposing the interface into inputs and outputs distinguish the portion of the interface that is used in a connection from the portion of the interface that is not used. The specific connection formed determines the interface decomposition and every possibility exists as an arrow in ${}_L\mathsf{Csp}$. This is reflected in the fact that ${}_L\mathsf{Csp}$ is compact closed (see Definition 100).

Proposition 6. $({}_L\mathsf{Csp}, \otimes, 0_{\mathsf{A}})$, where

$$\otimes\colon {}_L\mathsf{Csp} \times {}_L\mathsf{Csp} \to {}_L\mathsf{Csp}$$

$$a \otimes b \mapsto a + b$$

$$\left(La \xrightarrow{f} x \xleftarrow{g} Lb\right) \otimes \left(La' \xrightarrow{f'} x' \xleftarrow{g'} Lb'\right) \mapsto \left(L(a+a') \xrightarrow{f+f'} x + x' \xleftarrow{g+g'} L(b+b')\right)$$

is compact closed.

Proof. It is a matter of course to show that $({}_L\mathsf{Csp}, \otimes, 0_{\mathsf{A}})$ is a symmetric monoidal category. Though, we point out that we are being a bit casual with our definition of $\otimes$. The tensor product actually returns the structured cospan

$$L(a+a') \xrightarrow{\sigma_a^{-1}} La + La' \xrightarrow{f+f'} x + x' \xleftarrow{g+g'} Lb + Lb' \xleftarrow{\sigma_b^{-1}} L(b+b')$$

where σ is the structure map arising from the preservation of $+$ by L. The symmetry rests on the fact that both $(\mathsf{A}, +, 0_{\mathsf{A}})$ and $(\mathsf{X}, +, 0_{\mathsf{X}})$ are symmetric monoidal categories.

Regarding compactness, each object is self-dual. For an object a, the evaluation map $a \otimes a \to I$ is

$$L(a+a) \xrightarrow{L\nabla} La \xleftarrow{!} L0_{\mathsf{A}}$$

and the coevaluation map is

$$L0 \xrightarrow{!} La \xleftarrow{L\nabla} L(a+a)$$

where ∇ denotes the codiagonal. Checking the triangle identities are straightforward. ■

1.2 Structured cospans as objects

Lack and Sobocinski provided a way to rewrite objects in what are called adhesive categories [42]. To provide a theory of rewriting structured cospans using adhesive categories, we need a category in which structured cospans are the objects. This, of course, requires a notion of structured cospan morphism.

Definition 7. A morphism between L-structured cospans $La \to x \leftarrow Lb$ and $Lc \to y \leftarrow Ld$ is a triple of arrows (f, g, h) that fit into the commuting diagram

$$\begin{array}{ccccc} La & \longrightarrow & x & \longleftarrow & Lb \\ {\scriptstyle Lf}\downarrow & & {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle Lh} \\ Lc & \longrightarrow & y & \longleftarrow & Ld \end{array}$$

There is a category ${}_L\mathsf{StrCsp}$ whose objects are structured cospans and arrows are these morphisms.

We now come to the first of our main results: that ${}_L\mathsf{StrCsp}$ is a topos. This result is critical for our theory because, as each topos is adhesive [43], it allows the introduction of rewriting onto structured cospans.

Theorem 8. For any adjunction

$$\mathsf{A} \underset{R}{\overset{L}{\rightleftarrows}} \mathsf{X} \quad (L \dashv R)$$

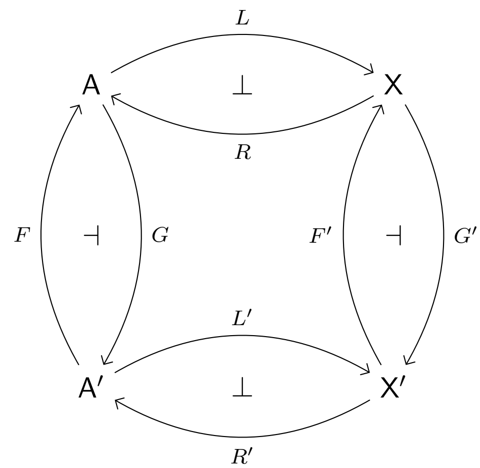
between topoi A and X, the category ${}_L\mathsf{StrCsp}$ is a topos.

Proof. Note that ${}_L\mathsf{StrCsp}$ is equivalent to the category whose objects are cospans of form $a \to Rx \leftarrow b$ and morphisms are triples (f, g, h) fitting into the commuting diagram

$$\begin{array}{ccccc} w & \longrightarrow & Ra & \longleftarrow & x \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle Rg} & & \downarrow{\scriptstyle h} \\ y & \longrightarrow & Rb & \longleftarrow & z \end{array}$$

This, in turn, is equivalent to the comma category $(\mathsf{A} \times \mathsf{A} \downarrow \Delta R)$, where $\Delta \colon \mathsf{A} \to \mathsf{A} \times \mathsf{A}$ is the diagonal functor. But this diagonal functor is right adjoint to the coproduct functor. Therefore, ΔR is also a right adjoint so $(\mathsf{A} \times \mathsf{A} \downarrow \Delta R)$ is an instance of Artin gluing [60], hence a topos. ■

We now show that constructing ${}_L\mathsf{StrCsp}$ is functorial in L. The codomain of this functor is comprised of topoi and adjoint pairs, the left of which preserves pullbacks. We call this category $\mathsf{AdjTopos}$. The domain this functor is the arrow category of $\mathsf{AdjTopos}$, which we denote by $[\bullet \to \bullet, \mathsf{AdjTopos}]$. In this category, the objects are adjunctions between topoi, the left adjoint preserving pullbacks, and an arrow from $L\colon \mathsf{A} \rightleftarrows \mathsf{X}\colon R$ to $L'\colon \mathsf{A}' \rightleftarrows \mathsf{X}'\colon R'$ is a pair of adjoints $F \dashv G$ and $F' \dashv G'$ fitting into a diagram

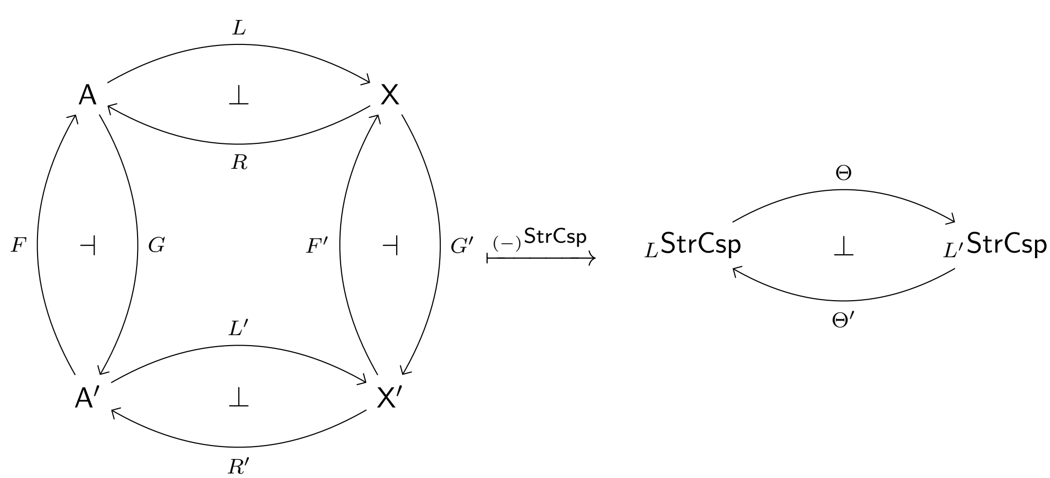


such that $LF = F'L'$ and $GR = R'G'$.

Theorem 9. There is a functor

$$_{(-)}\mathsf{StrCsp}\colon [\bullet \to \bullet, \mathsf{AdjTopos}] \to \mathsf{AdjTopos}$$

defined by



which is in turn given by

$$
\begin{array}{ccccc}
La & \xrightarrow{m} & x & \xleftarrow{n} & Lb \\
{\scriptstyle Lf}\downarrow & & \downarrow{\scriptstyle g} & & \downarrow{\scriptstyle Lh} \\
Lc & \xrightarrow[o]{} & y & \xleftarrow[p]{} & Ld
\end{array}
\quad \overset{\Theta}{\longmapsto} \quad
\begin{array}{ccccc}
L'G'a & \xrightarrow{Gm} & Gx & \xleftarrow{Gn} & L'G'b \\
{\scriptstyle L'G'f}\downarrow & & \downarrow{\scriptstyle Gg} & & \downarrow{\scriptstyle L'G'h} \\
L'G'c & \xrightarrow[Go]{} & Gy & \xleftarrow[Gp]{} & L'G'd
\end{array}
$$

and

$$
\begin{array}{ccccc}
L'a' & \xrightarrow{m'} & x' & \xleftarrow{n'} & L'b' \\
{\scriptstyle L'f'}\downarrow & & \downarrow{\scriptstyle g'} & & \downarrow{\scriptstyle L'h'} \\
L'c' & \xrightarrow{o'} & y' & \xleftarrow{p'} & L'd'
\end{array}
\quad \overset{\Theta'}{\longmapsto} \quad
\begin{array}{ccccc}
LF'a' & \xrightarrow{Fm} & Fx' & \xleftarrow{Fn'} & LF'b' \\
{\scriptstyle LF'f'}\downarrow & & \downarrow{\scriptstyle Fg'} & & \downarrow{\scriptstyle LF'h'} \\
LF'c' & \xrightarrow{Fo'} & Fy' & \xleftarrow{Fp'} & LF'd'
\end{array}
$$

Proof. In light of Theorem 8, it suffices to show that $\Theta \dashv \Theta'$ gives an adjunction and Θ preserves pushouts.

Denote the structured cospans

$$La \xrightarrow{m} x \xleftarrow{n} Lb$$

in ${}_L\mathsf{StrCsp}$ by ℓ and

$$L'a' \xrightarrow{m'} x' \xleftarrow{n'} L'b'$$

in ${}_{L'}\mathsf{StrCsp}$ by ℓ'. Denote the unit and counit for $F \dashv G$ by η, ε and for $F' \dashv G'$ by η', ε'. The assignments

$$\big((f,g,h)\colon \ell \to \Theta'\ell'\big) \mapsto \big((\varepsilon' \circ F'f, \varepsilon \circ Fg, \varepsilon' \circ F'h)\colon \Theta\ell \to \ell'\big)$$

$$\big((f',g',h')\colon \Theta\ell \to \ell'\big) \mapsto \big((G'f' \circ \eta', Gg' \circ \eta, G'h' \circ \eta')\colon \ell \to \Theta'\ell'\big)$$

give a bijection $\hom(\Theta\ell, \ell') \simeq \hom(\ell, \Theta'\ell')$. The naturality of ℓ and ℓ' rest on natural maps η, ε, η', and ε'. The left adjoint Θ' preserves finite pullbacks because they are taken pointwise and L, F, and F' all preserve finite limits. ■

The arrows ${}_L\mathsf{StrCsp} \to_{L'} \mathsf{StrCsp}$ that we are interested in act on the systems and their interfaces.

Definition 10. Fix a pair of structured cospan categories ${}_L\mathsf{StrCsp}$ and ${}_{L'}\mathsf{StrCsp}$ using the adjunctions

$$\mathsf{A} \underset{R}{\overset{L}{\rightleftarrows}} \mathsf{X} \quad (L \dashv R) \qquad \text{and} \qquad \mathsf{A}' \underset{R'}{\overset{L'}{\rightleftarrows}} \mathsf{X}' \quad (L' \dashv R')$$

with L and L' preserving pullbacks. A **structured cospan functor** of type

$${}_L\mathsf{StrCsp} \to_{L'} \mathsf{StrCsp}$$

is a pair of finitely continuous and cocontinuous functors $F\colon \mathsf{X} \to \mathsf{X}'$ and $G\colon \mathsf{A} \to \mathsf{A}'$ such that the diagrams

$$\begin{array}{ccc} \mathsf{A} & \xrightarrow{L} & \mathsf{X} \\ {\scriptstyle G}\downarrow & & \downarrow{\scriptstyle F} \\ \mathsf{A}' & \xrightarrow[L']{} & \mathsf{X}' \end{array} \qquad \begin{array}{ccc} \mathsf{A} & \xleftarrow{R} & \mathsf{X} \\ {\scriptstyle G}\downarrow & & \downarrow{\scriptstyle F} \\ \mathsf{A}' & \xleftarrow[R']{} & \mathsf{X}' \end{array}$$

commute.

Structured cospan categories and their morphisms form a category which we leave unnamed.

1.3 A double category of structured cospans

We use (pseudo) double categories (see Definition 85) to combine into a single instrument the competing perspectives of structured cospans as objects and as arrows.

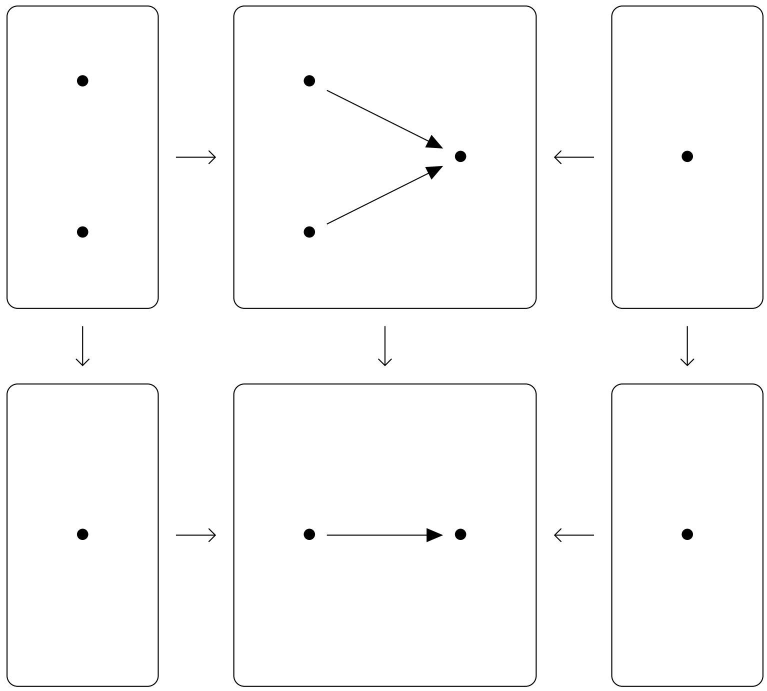
Definition 11 (Structured cospan double category)**.** There is a double category ${}_L\mathbb{S}\mathbf{trCsp}$ given by the following data:

- the objects are the A-objects
- the vertical arrows $a \to b$ are the A-arrows,
- the horizontal arrows $a \nrightarrow b$ are the cospans $La \to x \leftarrow Lb$, and
- the squares are the commuting diagrams

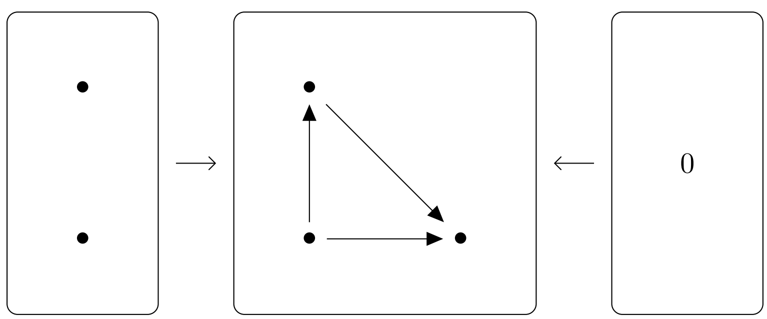
$$\begin{array}{ccccc} La & \longrightarrow & x & \longleftarrow & Lb \\ {\scriptstyle Lf}\downarrow & & \downarrow{\scriptstyle g} & & \downarrow{\scriptstyle Lh} \\ Lc & \longrightarrow & y & \longleftarrow & Ld \end{array}$$

Baez and Courser proved that this truly is a double category [15, Cor. 3.9]. Moreover, when A and X are cocartesian, their coproducts can be used to define a symmetric monoidal structure on ${}_L\mathbb{S}\mathbf{trCsp}$. The meaning of this structure is that the disjoint union of two systems can be considered a single system. The following example illustrates the squares and tensor product.

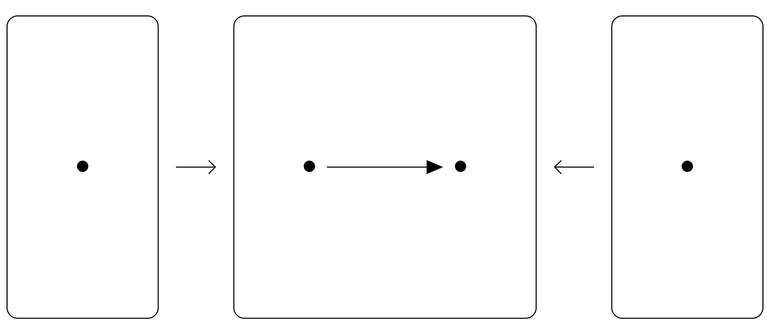
Example 12. Consider the double category ${}_L\mathbb{S}\mathbf{trCsp}$ where L is left adjoint to the underlying node functor $R\colon \mathsf{RGraph} \to \mathsf{Set}$. A square in this double category is a diagram in RGraph such as



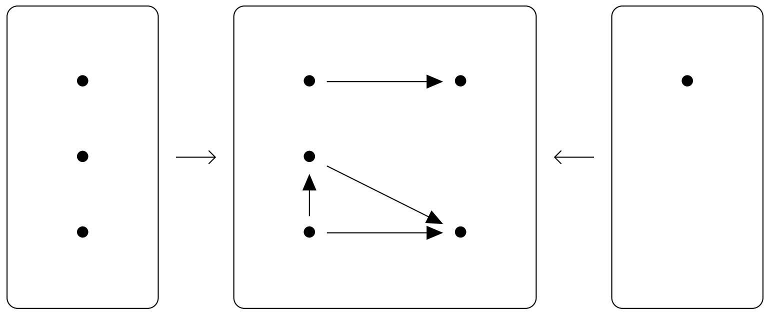
The tensor is the disjoint union of open graphs. For example, tensoring



together with



gives the open graph



This double category is explored further by Baez and Courser [5]. For us, it is a nice structure in which to simultaneously present the compositional role and the object role of structured cospans.

1.4 Spans of structured cospans

For this final section of the chapter, we define spans of structured cospans. These are the objects that serve as rewrite rules. We bring the two flavors of rewriting, fine and bold, to structured cospans in Chapter 3 and Chapter 4. This section segues to those two chapters.

We continue to work with a adjunction

$$\mathsf{A} \underset{R}{\overset{L}{\rightleftarrows}} \mathsf{X} \qquad (L \dashv R)$$

with L preserving pullbacks.

Definition 13. A **span of structured cospans** is a commuting diagram

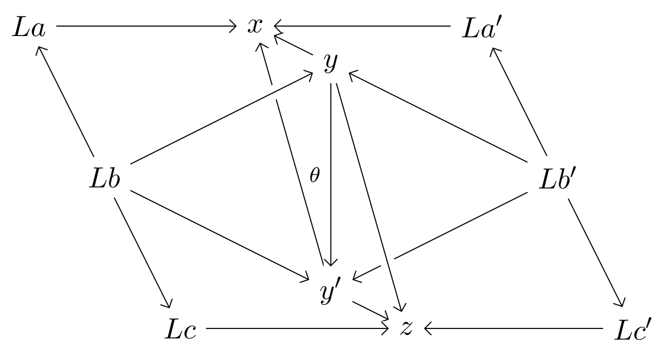
$$
\begin{array}{ccccc}
La & \longrightarrow & x & \longleftarrow & La' \\
\uparrow & & \uparrow & & \uparrow \\
Lb & \longrightarrow & y & \longleftarrow & Lb' \\
\downarrow & & \downarrow & & \downarrow \\
Lc & \longrightarrow & z & \longleftarrow & Lc'
\end{array}
$$

Spans of cospans (not structured cospans) were considered by Kissinger in his thesis [41] and also by Grandis and Paré in [36]. They did not fit them into a categorical structure as we do in latter chapters. For us, they will be squares in a double category for which we need to introduce horizontal composition $\circ_{\mathrm{h}}$ and vertical composition $\circ_{\mathrm{v}}$. The compositions use pushouts and pullbacks, which are only defined up to isomorphism. It follows that we will need to consider *classes* of spans of cospans, the specifics of which we put off until introducing the fine rewriting and bold rewriting of structured cospans. For now, we define a morphism of spans of structured cospans.

Definition 14. A **morphism of spans of structured cospans** from

$$
\begin{array}{ccccc}
La & \longrightarrow & x & \longleftarrow & Lb \\
\uparrow & & \uparrow & & \uparrow \\
Lc & \longrightarrow & y & \longleftarrow & Ld \\
\downarrow & & \downarrow & & \downarrow \\
Le & \longrightarrow & z & \longleftarrow & Lf
\end{array}
\quad \text{to} \quad
\begin{array}{ccccc}
La & \longrightarrow & x & \longleftarrow & Lb \\
\uparrow & & \uparrow & & \uparrow \\
Lc & \longrightarrow & y' & \longleftarrow & Ld \\
\downarrow & & \downarrow & & \downarrow \\
Le & \longrightarrow & z & \longleftarrow & Lf
\end{array}
$$

is an arrow $\theta\colon y \to y'$ that fits into a commuting diagram



If θ is invertible, then the morphism is an isomorphism.

We now have our syntactical device in hand. As previously stated, our goal is to incorporate rewriting. To do so, we spend the next chapter covering rewriting in a general setting before moving on to focus solely on rewriting structured cospans.

Chapter 2

Double pushout rewriting

Our primary aim is to develop a theory of rewriting for open systems. This goal fits into a larger program of studying the "linguistics" of open systems. By this we mean designating syntax and semantics. Rewriting lives on the syntactical side of this divide.

To develop an intuition for rewriting, we provide a sliver of its broader story. We chase from its beginnings in linguistics to double pushout graph rewriting to the modern day axioms of adhesive categories (see Appendix A.5). The most important example of an adhesive category for us is a topos. This fact highlights the importance of structured cospans forming a topos (Theorem 8) and it cements our ability to rewrite open systems.

2.1 A brief history of rewriting

We prefer to sketch the theory of rewriting rather than delve into details. For us, it is enough to build an intuition for rewriting prior to introducing it to open systems via structured cospans.

The theory of rewriting arose from Chomsky's work in formal languages [14]. He used rewriting as a device to generate well-formed sentences. While a well-formed sentence must be grammatically sound, it need not mean anything. Chomsky's [14] classic example of a grammatically sound but meaningless sentence is

'Colorless green ideas sleep furiously.'

That this sentence is syntactically good but semantically bad helps to highlight the difference between syntax and semantics. How does one use rewriting, in Chomsky's sense, to build that sentence?

We begin with a collection of rewrite rules:

1. a sentence is a noun phrase followed by a verb phrase;
2. a verb phrase consists of a verb and the option to follow with an adverb;
3. a noun phrase can be a noun with, optionally, a preceding determiner such as an article, demonstrative, quantifier, etc;
4. a noun phrase can be a noun with, optionally, a preceding adjective phrase or, optionally, a prepositional phrase.

These rules are denoted as follows:

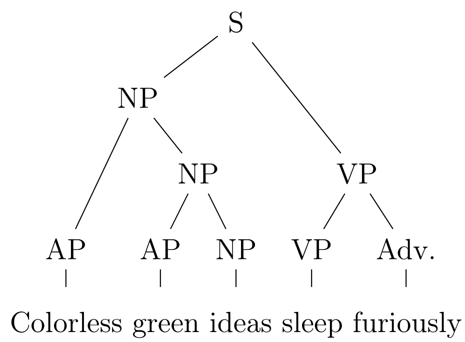
$$\text{S} \rightarrow \text{NP VP}$$

$$\text{VP} \rightarrow \text{VP (Adv.)}$$

$$\text{NP} \rightarrow \text{(Det.) NP}$$

$$\text{NP} \rightarrow \text{(AP) NP (PP)}$$

To derive a sentence, first apply a rule to S, then apply a rule to that first step's output, and so on. Eventually, no further rules are applicable at which point we are left with a grammatically sound sentence. The derivation of the above sentence is



The success of rewriting in linguistics led to its use in logic and mathematics. One evolution of rewriting into mathematics is an *Abstract Rewriting System*, a set A together with a binary relation $A \nrightarrow A$. An element of this relation (a, b) means that you can 'reduce' a to b. Often, one studies the transitive and reflexive closure of $A \nrightarrow A$ which we denote by adorning the arrow with an asterisk $A \nrightarrow^* A$. This so-called **rewriting relation** $\nrightarrow^*$ accounts for reflexive and multi-step reductions.

Example 15. The word problem can be expressed in terms of abstract rewriting. Let M be the set underlying a monoid, let FM be the free monoid on M, and let $\nrightarrow$ be a binary relation on FM given by $x_1 \cdots x_n \nrightarrow x$, with $x_i \in M$, whenever $x_1 \cdots x_n = x$ in the monoid M. The word problem asks, "if given words w, w' in FM, does $w \nrightarrow^* w'$ and $w' \nrightarrow^* w$"?

As just seen, we can determine whether syntactical expressions, such as words in a free monoid, are equivalent using rewriting. It is in this sense, not in generating sentences, that we are interested in rewriting.

We are particularly interested 'structured cospans', a syntactical device Baez and

Courser introduced [5] as a written language for open systems. In order to develop a theory of rewriting for structured cospans, we need more sophisticated machinery than abstract rewriting systems.

A first step in that direction is graph rewriting, invented by Ehrig, et. al. [34], where graphs are used in place of words and sentences. Rules are used to choose a subgraph and replace it with another equivalent[1] graph. Ehrig, et. al. encode rewrite rules in spans of graphs and apply a rule using pushouts. That is, a rule is a span of graphs

$$\ell \leftarrow k \rightarrow r$$

We interpret this rule to say *any instance of a sub-graph isomorphic to* ℓ *can be replaced by the graph* r.

Given such a rule and a graph g, how do we identify a copy of ℓ inside of g and then replace it with r? The answer lies in the following definition.

Definition 16 (Double pushout)**.** A **double pushout diagram** is a a pair of pushouts

$$\begin{array}{ccccc} \ell & \longleftarrow & d & \longrightarrow & r \\ \downarrow & \ulcorner & \downarrow & \urcorner & \downarrow \\ g & \longleftarrow & k & \longrightarrow & h \end{array}$$

that share an arrow as depicted.

While double pushout diagrams make sense in any category, graph rewriting restricts to the category Graph of directed graphs and their morphisms. So in the diagram above, each letter represents a graph and the arrows are graph morphisms. The rule being

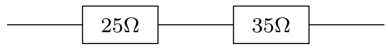
[1] We mean 'equivalent' in a semantic sense, thus varying with context.

applied is $\ell \leftarrow k \rightarrow r$ and the output of applying this rule to g is the graph h. The graph k is what holds fixed as r replaces ℓ and d is what holds fixed as h replaces g. A concrete example of this is given below in Equation (2.3).

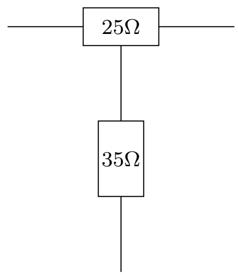
Observers noticed that the mechanisms did not require anything specific about graphs to work. Pushouts and spans are basic constructions in category theory, so it is reasonable to consider extending double pushout rewriting to a broader class of categories than just Graph. There were a number of attempts to axiomatize the important properties of graph rewriting, the most prominent example being 'high level replacement systems' [32], which we discuss in Section A.5. The drawback of HLRS's was the sheer number of axioms. Lack and Sobocinski eventually found a much shorter list of axioms. They called categories that satisfy their axioms 'adhesive categories' [42] (see Appendix A.5). Adhesive categories are currently the most general setting in which rewriting theory holds. However, we don't need the full generality of adhesive categories and instead focus on topoi, each of which is adhesive.

As mentioned earlier, there are different ways to interpret what a *rewriting* is. For instance, 'a rewriting is making a choice' or 'a rewriting is a simplification'. The interpretation for our needs is 'a rewriting is to replace by a behaviorally indistinguishable system'. The linguistic analogy is 'synonym'.

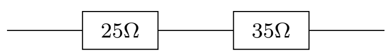
Though the focus of this thesis is on the syntax of open systems, the semantics of systems cannot fully be ignored. By the syntax of a system, we mean the rules followed by its diagrammatic representations. By the semantics of a system, we mean the behavior of a system. For example, consider resistor systems. It is a syntactic issue that the circuit diagram



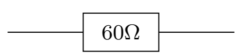
makes sense but



does not. A semantic consideration is that the resistor circuit



behaves in the same exact way as the circuit



This follows from Ohm's law. Syntactically, these are two different circuits. Building a rewriting theory into our structured cospan formalism provides our system syntax a mechanism to recognize semantically (i.e. behaviorly) indistinguishable systems.

2.2 Rewriting in topoi

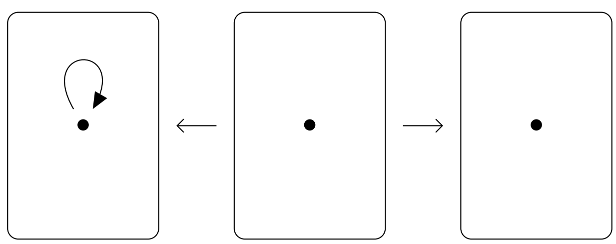
Fix a topos T. Rewriting starts with the notion of a **rewrite rule**, or simply **rule**. In its most general form, a rule is a span

$$\ell \leftarrow k \rightarrow r$$

in T. The arrows are left unnamed unless we need to refer to them. For us, rules come in two flavors. A **fine rule** is one in which both of the span arrows are monic. A **bold rule** is one without restriction on the arrows.

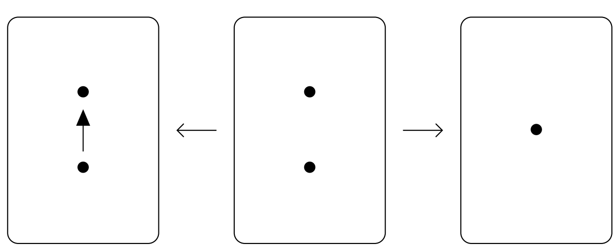
Remark 17. Both fine and bold approaches are considered in the rewriting literature, but often by the name 'linear' and 'non-linear', respectively. Fine rewriting is more common. Habel, Muller, and Plump compared these alternatives in the context of graph rewriting [37]. The distinction between the two cases does not appear in this chapter, and everything we say carries through in either case. We do take care to ensure that constructions are well-defined in the monic case.

The conceit of a rule is that r replaces ℓ while k identifies a subsystem of ℓ that remains fixed. For example, suppose we were modeling some system using graphs where self-loops were meaningless. In the introduction, we considered modeling the internet with a graph with websites as nodes and links as edges. If we did not care about websites with a link to itself, we would introduce a rule that replaces a node with a loop with a node



(2.1)

For another example, suppose we had another system modeled on graphs where an edge between two nodes is equivalent to having a single node. This is captured with the rule



(2.2)

This rule appears in the ZX-calculus example from Section 4.3. Observe that the first example is a fine rewrite and the second is a bold rewrite.

To *apply* a rule $\ell \leftarrow k \rightarrow r$ to an object g, we require an arrow $m\colon \ell \rightarrow g$ such that there exists a **pushout complement**, an object d fitting into a pushout diagram

$$\begin{array}{ccc} \ell & \longleftarrow & k \\ {\scriptstyle m}\big\downarrow & \urcorner & \big\downarrow \\ g & \longleftarrow & d \end{array}$$

A pushout complement need not exist, but when it does and the map $k \rightarrow \ell$ is monic, then it is unique up to isomorphism [42, Lem. 15].

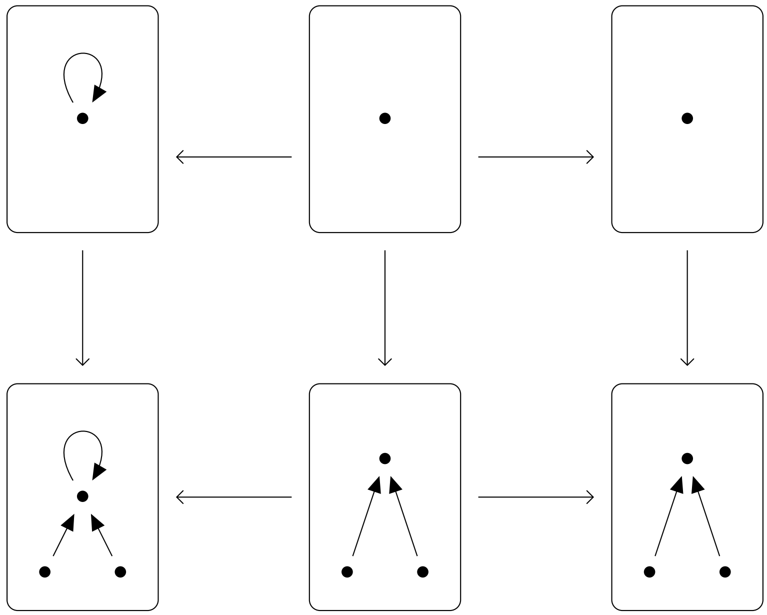
For each application of a rule, we derive a new rule.

Definition 18 (Derived rule)**.** A **derived rule** is any span $g \leftarrow d \rightarrow h$ fitting into the bottom row of the double pushout diagram

$$\begin{array}{ccccc} \ell & \longleftarrow & k & \longrightarrow & r \\ \big\downarrow & \urcorner & \big\downarrow & \ulcorner & \big\downarrow \\ g & \longleftarrow & d & \longrightarrow & h \end{array}$$

When the arrows of the rule $\ell \leftarrow k \rightarrow r$ are both monic, the arrows of the span $g \leftarrow d \rightarrow h$ are also monic because pushouts preserve monics in topoi [42, Lem. 12]. The intuition of this diagram is that $\ell \rightarrow g$ identifies a copy of ℓ in g and we replace that copy with r, resulting in a new object h.

To illustrate, let us return to a system modeled with graphs and where self-loops are meaningless. Then we can apply Rule (2.1) to any node with a loop. This application is captured with the double pushout diagram



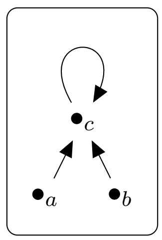
(2.3)

We identified a self-loop in the bottom left graph then applied the rule to remove it. The result is the bottom right graph. The reader can check that the two squares are pushouts.

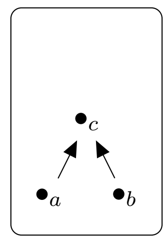
Usually when modeling a system, there is a set of rewrite rules that accompany it. For example, in resistor circuits there are parallel, series, and star rules. Just like in natural languages, we call a collection of rules a grammar.

Definition 19 (Grammar)**.** A topos T together with a finite set P of rules $\{\ell_j \leftarrow k_j \rightarrow r_j\}$ in T is a **grammar**. When the all rules in a grammar have monic arrows, we say the grammar is **fine**. Else, the grammar is **bold**. An arrow of (fine, bold) grammars $(\mathsf{S}, P) \rightarrow (\mathsf{T}, Q)$ is a pullback and pushout preserving functor $F \colon \mathsf{S} \rightarrow \mathsf{T}$ such that for each rule $\ell \xleftarrow{f} k \xrightarrow{g} r$ in P, the rule $F\ell \xleftarrow{Ff} Fk \xrightarrow{Fg} Fr$ is in Q. Together these form a category Gram.

A grammar is a seed. Like a seed, the grammar gives birth to something entirely new and more complex called the language. It is this language that we are interested more so than the grammar. We can certainly learn about the language from the grammar, but what we actually study is the 'rewrite relation' which informs us about how different components of the language relate. Every grammar (T, P) gives rise to a relation $\rightsquigarrow$ on the objects of T defined by $g \rightsquigarrow h$ whenever there exists a rule $g \leftarrow d \rightarrow h$ derived from a production in P. For instance, the above double pushout diagram would relate



to



But $\rightsquigarrow$ is too small to capture the full behavior of the language. For one, it is not true in general that $g \rightsquigarrow g$ holds. Also, $\rightsquigarrow$ does not capture multi-step rewrites. That is, there may be derived rules witnessing $g \rightsquigarrow g'$ and $g' \rightsquigarrow g''$ but not a derived rule witnessing $g \rightsquigarrow g''$. We want to relate a pair of objects if one can be rewritten into another with a finite sequence of derived rules. Therefore, we actually want the following.

Definition 20 (Rewrite relation)**.** To each grammar (T, P), assign a relation on the objects of T defined by setting $g \rightsquigarrow h$ whenever there is a rewrite rule $\ell \leftarrow k \rightarrow r$ in P and an object d of T that fit into a double pushout diagram

$$\begin{array}{ccccc} \ell & \longleftarrow & k & \longrightarrow & r \\ \downarrow & \ulcorner & \downarrow & \urcorner & \downarrow \\ g & \longleftarrow & d & \longrightarrow & h \end{array}$$

The **rewrite relation** $\rightsquigarrow^*$ is the transitive and reflexive closure of $\rightsquigarrow$.

Every grammar determines a unique rewrite relation in a functorial way. We devote

Section 5.2 to proving this fact, though, we restrict ourselves working with grammars of structured cospan categories.

Chapter 3

Fine rewriting and structured cospans

In this chapter, we introduce a theory of fine rewriting to structured cospans. Rewriting is *fine* when the rewrite rules are spans with monic legs. Our primary goal is to define a double category whose squares are fine rewrites of structured cospans. The rough idea is that this double category, denoted ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$, has interface types for objects, structured cospans for horizontal arrows, isomorphisms of interface objects for vertical arrows, and fine rewrite rules of structured cospans for squares. We prove in Proposition 25 that ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ actually is a double category. The first step to proving this is to ensure the fine rewrite rules are suitable squares for our double category, we define them as follows.

Definition 21 (Fine rewrite)**.** A **fine rewrite of structured cospans** is an isomorphism class of spans of structured cospans of the form

$$\begin{array}{ccccc} La & \longrightarrow & x & \longleftarrow & Lb \\ \cong\uparrow & & \uparrow & & \uparrow\cong \\ Lc & \longrightarrow & y & \longleftarrow & Ld \\ \cong\downarrow & & \uparrow & & \downarrow\cong \\ Le & \longrightarrow & z & \longleftarrow & Lf \end{array}$$

The marked arrows are monic.

In a double category, the squares have two composition operations. Horizontal composition uses pushout as is typical with cospan categories. The vertical composition uses pullback as is typical in span categories. But because there are no higher order arrows traversing the squares in a double category, and because pushouts and pullbacks are only defined up to isomorphism, we take isomorphism classes of structured cospan rewrite rules. With the squares of ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ defined, we can introduce the two composition operations.

Definition 22. The **horizontal composition** of fine rewrite rules is given by

$$\begin{array}{ccccc} La & \longrightarrow & v & \longleftarrow & La' \\ \uparrow & & \uparrow & & \uparrow \\ Lb & \longrightarrow & w & \longleftarrow & Lb' \\ \downarrow & & \downarrow & & \downarrow \\ Lc & \longrightarrow & x & \longleftarrow & Lc' \end{array} \quad \circ_{\mathrm{h}} \quad \begin{array}{ccccc} La' & \longrightarrow & v' & \longleftarrow & La'' \\ \uparrow & & \uparrow & & \uparrow \\ Lb' & \longrightarrow & w' & \longleftarrow & Lb'' \\ \downarrow & & \downarrow & & \downarrow \\ Lc' & \longrightarrow & x' & \longleftarrow & Lc'' \end{array} \quad :=$$

$$
\begin{array}{ccccc}
La & \longrightarrow & v +_{La'} v' & \longleftarrow & La'' \\
\uparrow & & \uparrow & & \uparrow \\
Lb & \longrightarrow & w +_{Lb'} w' & \longleftarrow & Lb'' \\
\downarrow & & \downarrow & & \downarrow \\
Lc & \longrightarrow & x +_{Lc'} x' & \longleftarrow & Lc''
\end{array}
$$

The **vertical composition** of fine rewrite rules is

$$
\begin{array}{ccccc}
La & \longrightarrow & v & \longleftarrow & La' \\
\uparrow & & \uparrow & & \uparrow \\
Lb & \longrightarrow & w & \longleftarrow & Lb' \\
\downarrow & & \downarrow & & \downarrow \\
Lc & \longrightarrow & x & \longleftarrow & Lc'
\end{array}
\quad \circ_{\mathrm{v}} \quad
\begin{array}{ccccc}
Lc & \longrightarrow & x & \longleftarrow & Lc' \\
\uparrow & & \uparrow & & \uparrow \\
Ld & \longrightarrow & y & \longleftarrow & Ld' \\
\downarrow & & \downarrow & & \downarrow \\
Le & \longrightarrow & z & \longleftarrow & Le'
\end{array}
\quad :=
$$

$$
\begin{array}{ccccc}
La & \longrightarrow & v & \longleftarrow & La' \\
\uparrow & & \uparrow & & \uparrow \\
L(b \times_c d) & \longrightarrow & w \times_x y & \leftarrow & L(b' \times_{c'} d') \\
\downarrow & & \downarrow & & \downarrow \\
Le & \longrightarrow & z & \longleftarrow & Le'
\end{array}
$$

We defined $\circ_{\mathrm{h}}$ and $\circ_{\mathrm{v}}$ using representatives of isomorphism classes, however this operation is well-defined. It is less clear, however, that these operations preserve the monic arrows in the fine rewrites of structured cospans. In Proposition 24, we show that horizontal and vertical composition do preserve these monic arrows. To prove this, we require the following lemma.

Lemma 23. The diagram

$$
\begin{array}{ccccc}
x & \longleftarrow & y & \longrightarrow & z \\
\downarrow & & \cong \downarrow & & \downarrow \\
x' & \longleftarrow & y' & \longrightarrow & z'
\end{array}
\tag{3.1}
$$

induces a pushout

$$
\begin{array}{ccc}
x + z & \xrightarrow{\rho} & x +_y z \\
\gamma \downarrow & & \downarrow \gamma' \\
x' + z' & \xrightarrow[\rho']{} & x' +_{y'} z
\end{array}
\tag{3.2}
$$

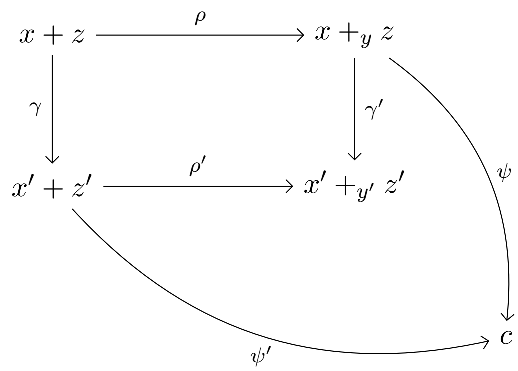
such that the canonical arrows γ and γ' are monic.

Proof. The universal property of coproducts implies that γ factors through $x' + z$ as in the diagram

$$
\begin{array}{ccccc}
x & \xrightarrow{\iota_x} & x + z & & \\
\downarrow & & \downarrow & & \\
x' & \xrightarrow[\iota_{x'}]{} & x' + z & \xleftarrow{\iota_z} & z \\
 & & \downarrow & & \downarrow \\
 & & x' + z' & \xleftarrow[\iota_{z'}]{} & z'
\end{array}
$$

It is straightforward to check that both squares are pushouts. By Lemma 116, it follows that γ is monic.

Diagram 3.2 commutes because of the universal property of coproducts. To see that it is a pushout, arrange a cocone



(3.3)

Denote by ι_x any map that includes x. Then $\psi'\iota_{x'}$, $\psi'\iota_{z'}$, and c form a cocone under the span $x' \leftarrow y' \rightarrow z'$ from the bottom face of Diagram 3.1. This induces the canonical map $\psi''\colon x' +_{y'} z' \rightarrow c$. It follows that $\psi'\iota_{x'} = \psi''\rho'\iota_{x'}$ and $\psi'\iota_{z'} = \psi''\rho'\iota_{z'}$. Therefore $\psi' = \psi''\rho'$ by the universal property of coproducts.

Furthermore, $\psi\rho\iota_z$, $\psi\rho\iota_z$, and c form a cocone under the span $x \leftarrow y \rightarrow z$ on the top face of Diagram 3.1. then $\psi\rho\iota_x = \psi'\gamma\iota_x = \psi''\rho'\gamma\iota_x = \psi''\psi'\rho\iota_x$ and $\psi\rho\iota_z = \psi'\gamma\iota_z = \psi''\rho'\gamma\iota_z = \psi''\gamma'\rho\iota_z$ meaning that both ψ and $\psi''\psi'$ satisfy the canonical map $x +_y z \rightarrow d$. Hence $\psi = \psi''\psi'$.

The universality of ψ'' with respect to Diagram 3.3 follows from the universality of γ'' with respect to $x' +_{y'} z'$. ■

Lemma 24. Horizontal and vertical composition of fine rewrites are fine rewrites.

Proof. We can see that the span of cospan obtained by horizontal composition of fine rewrites

$$
\begin{array}{ccccc}
La & \longrightarrow & v & \longleftarrow & La' \\
{\scriptstyle\cong}\uparrow & & \uparrow & & \uparrow{\scriptstyle\cong} \\
Lb & \longrightarrow & w & \longleftarrow & Lb' \\
{\scriptstyle\cong}\downarrow & & \downarrow & & \downarrow{\scriptstyle\cong} \\
Lc & \longrightarrow & x & \longleftarrow & Lc'
\end{array}
\quad \circ_{\mathrm{h}} \quad
\begin{array}{ccccc}
La' & \longrightarrow & v' & \longleftarrow & La'' \\
{\scriptstyle\cong}\uparrow & & \uparrow & & \uparrow{\scriptstyle\cong} \\
Lb' & \longrightarrow & w' & \longleftarrow & Lb'' \\
{\scriptstyle\cong}\downarrow & & \downarrow & & \downarrow{\scriptstyle\cong} \\
Lc' & \longrightarrow & x' & \longleftarrow & Lc''
\end{array}
\quad :=
$$

$$
\begin{array}{ccccc}
La & \longrightarrow & v +_{La'} v' & \longleftarrow & La'' \\
{\scriptstyle\cong}\uparrow & & \uparrow & & \uparrow{\scriptstyle\cong} \\
Lb & \longrightarrow & w +_{Lb'} w' & \longleftarrow & Lb'' \\
{\scriptstyle\cong}\downarrow & & \downarrow & & \downarrow{\scriptstyle\cong} \\
Lc & \longrightarrow & x +_{Lc'} x' & \longleftarrow & Lc''
\end{array}
$$

is again a fine rewrite, that is the arrows $w +_{Le} x \to u +_{Lb} v$ and $w +_{Le} x \to y +_{Lh} z$ are monic, by applying Lemma 23 to the diagrams

$$
\begin{array}{ccccc}
v & \longleftarrow & La' & \longrightarrow & v' \\
\uparrow & & \uparrow{\scriptstyle\cong} & & \uparrow \\
w & \longleftarrow & Lb' & \longrightarrow & w'
\end{array}
\qquad \text{and} \qquad
\begin{array}{ccccc}
w & \longleftarrow & Lb' & \longrightarrow & w' \\
\downarrow & & \downarrow{\scriptstyle\cong} & & \downarrow \\
x & \longleftarrow & Lc' & \longrightarrow & x'
\end{array}
$$

The result for vertical composition

$$
\begin{array}{ccccc}
La & \longrightarrow & v & \longleftarrow & La' \\
{\scriptstyle\cong}\uparrow & & \uparrow & & \uparrow{\scriptstyle\cong} \\
Lb & \longrightarrow & w & \longleftarrow & Lb' \\
{\scriptstyle\cong}\downarrow & & \downarrow & & \downarrow{\scriptstyle\cong} \\
Lc & \longrightarrow & x & \longleftarrow & Lc'
\end{array}
\quad \circ_{\mathrm{v}} \quad
\begin{array}{ccccc}
Lc & \longrightarrow & x & \longleftarrow & Lc' \\
{\scriptstyle\cong}\uparrow & & \uparrow & & \uparrow{\scriptstyle\cong} \\
Ld & \longrightarrow & y & \longleftarrow & Ld' \\
{\scriptstyle\cong}\downarrow & & \downarrow & & \downarrow{\scriptstyle\cong} \\
Le & \longrightarrow & z & \longleftarrow & Le'
\end{array}
\quad :=
$$

$$
\begin{array}{ccccc}
La & \longrightarrow & v & \longleftarrow & La' \\
\uparrow{\scriptstyle\cong} & & \uparrow & & \uparrow{\scriptstyle\cong} \\
L(b \times_c d) & \longrightarrow & w \times_x y & \longleftarrow & L(b' \times_{c'} d') \\
\downarrow{\scriptstyle\cong} & & \downarrow & & \downarrow{\scriptstyle\cong} \\
Le & \longrightarrow & z & \longleftarrow & Le'
\end{array}
$$

holds because pullback preserves monomorphisms. ■

With horizontal and vertical composition in hand, we construct the double category ${}_L\mathbb{F}$**ine**$\mathbb{R}$**ewrite**. Actually, we delay discussing the interchange law until Section 3.1 because it is difficult enough to warrant its own section.

Proposition 25. Let

$$
\mathsf{A} \underset{R}{\overset{L}{\rightleftarrows}} \mathsf{X} \qquad (L \dashv R)
$$

be a adjunction with L preserving pullbacks. There is a double category ${}_L\mathbb{F}$**ine**$\mathbb{R}$**ewrite** whose objects are the A-objects, horizontal arrows of type $a \to b$ are structured cospans $La \to x \leftarrow Lb$, vertical arrows are spans in A with invertible arrows, and squares are fine rewrites of structured cospans

$$
\begin{array}{ccccc}
La & \longrightarrow & x & \longleftarrow & La' \\
\uparrow{\scriptstyle\cong} & & \uparrow & & \uparrow{\scriptstyle\cong} \\
Lb & \longrightarrow & y & \longleftarrow & Lb' \\
\downarrow{\scriptstyle\cong} & & \downarrow & & \downarrow{\scriptstyle\cong} \\
Lc & \longrightarrow & z & \longleftarrow & Lc'
\end{array}
$$

Proof. This proof requires we check the axioms of a double category as laid out in Definition 85. For simplicity, we denote ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ by $\mathbb{R}$ in this proof.

The object category $\mathbb{R}_0$ is given by objects of A and isomorphism classes of spans in A such that each leg is an isomorphism. The arrow category $\mathbb{R}_1$ has as objects the structured cospans

$$La \to x \leftarrow La'$$

and as morphisms the fine rewrites of structured cospans.

The functor $U \colon \mathbb{R}_0 \to \mathbb{R}_1$ acts on objects by mapping a to the identity cospan on La and on morphisms by mapping $La \leftarrow Lb \to Lc$, whose legs are isomorphisms, to the square

$$\begin{array}{ccccc}
La & \longrightarrow & La & \longleftarrow & La \\
\uparrow & & \uparrow & & \uparrow \\
Lb & \longrightarrow & Lb & \longleftarrow & Lb \\
\downarrow & & \downarrow & & \downarrow \\
Lc & \longrightarrow & Lc & \longleftarrow & Lc
\end{array}$$

The functor $S \colon \mathbb{R}_1 \to \mathbb{R}_0$ acts on objects by sending $La \to x \leftarrow La'$ to a and on morphisms by sending a square

$$\begin{array}{ccccc}
La & \longrightarrow & x & \longleftarrow & La' \\
\uparrow & & \uparrow & & \uparrow \\
Lb & \longrightarrow & y & \longleftarrow & Lb' \\
\downarrow & & \downarrow & & \downarrow \\
Lc & \longrightarrow & z & \longleftarrow & Lc'
\end{array}$$

to the span $La \leftarrow Lb \rightarrow Lc$. The functor T is defined similarly sends an object

$$La \rightarrow x \leftarrow La'$$

of $\mathbb{R}_1$ to a' a square

$$\begin{array}{ccccc}
La & \longrightarrow & x & \longleftarrow & La' \\
\uparrow & & \uparrow & & \uparrow \\
Lb & \longrightarrow & y & \longleftarrow & Lb' \\
\downarrow & & \downarrow & & \downarrow \\
Lc & \longrightarrow & z & \longleftarrow & Lc'
\end{array}$$

to the span $La' \leftarrow Lb' \rightarrow Lc'$.

The horizontal composition functor

$$\odot \colon \mathbb{R}_1 \times_{\mathbb{R}_0} \mathbb{R}_1 \rightarrow \mathbb{R}_1$$

acts on objects by composing cospans with pushouts in the usual way. It acts on morphisms by

$$\begin{array}{ccccccccc}
La & \longrightarrow & v & \longleftarrow & La' & \longrightarrow & v' & \longleftarrow & La'' \\
\uparrow & & \uparrow & & \uparrow & & \uparrow & & \uparrow \\
Lb & \longrightarrow & w & \longleftarrow & Lb' & \longrightarrow & w' & \longleftarrow & Lb'' \\
\downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\
Lc & \longrightarrow & x & \longleftarrow & Lc' & \longrightarrow & x' & \longleftarrow & Lc''
\end{array}
\quad \overset{\odot}{\longmapsto} \quad
\begin{array}{ccccc}
La & \longrightarrow & v +_{La'} v' & \longleftarrow & La'' \\
\uparrow & & \uparrow & & \uparrow \\
Lb & \longrightarrow & w +_{Lb'} w' & \longleftarrow & Lb'' \\
\downarrow & & \downarrow & & \downarrow \\
Lc & \longrightarrow & x +_{Lc'} x' & \longleftarrow & Lc''
\end{array}$$

Section 3.1 is devoted to proving that $\odot$ is functorial, that is, it preserves composition. It is straightforward to check that the required equations are satisfied. The associator and unitors are given by natural isomorphisms that arise from universal properties. ■

And now, our double category of fine rewrites is defined. It remains to prove the interchange law, which we do next.

3.1 The interchange law

Here we prove the most technical part of the proof that ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ is a double category: the interchange law. This law relates the horizontal and vertical composition defined in the previous section.

Theorem 26. Given four fine rewrites of structured cospans

$$\alpha := \begin{array}{ccccc} La & \longrightarrow & u & \longleftarrow & Lb \\ \cong\uparrow & & \uparrow & & \uparrow\cong \\ Ld & \longrightarrow & w & \longleftarrow & Le \\ \cong\downarrow & & \downarrow & & \downarrow\cong \\ Lg & \longrightarrow & y & \longleftarrow & Lh \end{array} \qquad \alpha' := \begin{array}{ccccc} Lb & \longrightarrow & v & \longleftarrow & Lc \\ \cong\uparrow & & \uparrow & & \uparrow\cong \\ Le & \longrightarrow & x & \longleftarrow & Lf \\ \cong\downarrow & & \downarrow & & \downarrow\cong \\ Lh & \longrightarrow & z & \longleftarrow & Li \end{array}$$

$$\beta := \begin{array}{ccccc} Lg & \longrightarrow & y & \longleftarrow & Lh \\ \cong\uparrow & & \uparrow & & \uparrow\cong \\ Ld' & \longrightarrow & w' & \longleftarrow & Le' \\ \cong\downarrow & & \downarrow & & \downarrow\cong \\ La' & \longrightarrow & x' & \longleftarrow & Lb' \end{array} \qquad \beta' := \begin{array}{ccccc} Lh & \longrightarrow & z & \longleftarrow & Li \\ \cong\uparrow & & \uparrow & & \uparrow\cong \\ Le' & \longrightarrow & x' & \longleftarrow & Lf' \\ \cong\downarrow & & \downarrow & & \downarrow\cong \\ Lb' & \longrightarrow & v' & \longleftarrow & Lc' \end{array} \tag{3.4}$$

it is true that

$$(\alpha \circ_{\mathrm{h}} \alpha') \circ_{\mathrm{v}} (\beta \circ_{\mathrm{h}} \beta') = (\alpha \circ_{\mathrm{v}} \beta) \circ_{\mathrm{h}} (\alpha' \circ_{\mathrm{v}} \beta'). \tag{3.5}$$

We devote the remainder of this section proving Theorem 26. The first thing we do is deconstruct Equation (3.5), starting with the left hand side.

The horizontal compositions $\alpha \circ_{\mathrm{h}} \alpha'$ and $\beta \circ_{\mathrm{h}} \beta'$ are, respectively,

$$
\begin{array}{ccccc}
La & \longrightarrow & u +_{Lb} v & \longleftarrow & Lc \\
{\scriptstyle\cong}\uparrow & & \uparrow & & \uparrow{\scriptstyle\cong} \\
Ld & \longrightarrow & w +_{Le} x & \longleftarrow & Lf' \\
{\scriptstyle\cong}\downarrow & & \downarrow & & \downarrow{\scriptstyle\cong} \\
Lg & \longrightarrow & y +_{Lh} z & \longleftarrow & Lc'
\end{array}
\qquad
\begin{array}{ccccc}
Lg & \longrightarrow & y +_{Lh} z & \longleftarrow & Li \\
{\scriptstyle\cong}\uparrow & & \uparrow & & \uparrow{\scriptstyle\cong} \\
Ld' & \longrightarrow & w' +_{Le'} x' & \longleftarrow & Lf' \\
{\scriptstyle\cong}\downarrow & & \downarrow & & \downarrow{\scriptstyle\cong} \\
La' & \longrightarrow & x' +_{Lb'} v' & \longleftarrow & Lc'
\end{array}
$$

Lemma 24 ensures that the marked arrows above are monic. The vertical composition of these is

$$
(\alpha \circ_{\mathrm{h}} \alpha') \circ_{\mathrm{v}} (\beta \circ_{\mathrm{h}} \beta') =
\begin{array}{ccccc}
La & \longrightarrow & u +_{Lb} v & \longleftarrow & Lc \\
{\scriptstyle\cong}\uparrow & & \uparrow & & \uparrow{\scriptstyle\cong} \\
Ld \times_{Lg} Ld' & \longrightarrow & (w +_{Le} x) \times_{(y +_{Lh} z)} (w' +_{Le'} x') & \longleftarrow & Lf +_{Li} Lf' \\
{\scriptstyle\cong}\downarrow & & \downarrow & & \downarrow{\scriptstyle\cong} \\
La' & \longrightarrow & x' +_{Lb'} v' & \longleftarrow & Lc'
\end{array}
$$

Again, the marked arrows are monic due to Lemma 24. The outside, vertical arrows are isomorphisms because pullbacks preserve isomorphism.

To compute the right hand side of Equation (3.5), we start with the vertical composites $\alpha \circ_{\mathrm{v}} \beta$ and $\alpha' \circ_{\mathrm{v}} \beta'$, which are the respective diagrams

$$
\begin{array}{ccccc}
La & \longrightarrow & u & \longleftarrow & Lb \\
\cong \uparrow & & \uparrow & & \uparrow \cong \\
L(d \times_g d') & \longrightarrow & w \times_y w' & \longleftarrow & L(e \times_h e') \\
\cong \downarrow & & \downarrow & & \downarrow \cong \\
La' & \longrightarrow & x' & \longleftarrow & Lb'
\end{array}
$$

$$
\begin{array}{ccccc}
Lb & \longrightarrow & v & \longleftarrow & Lc \\
\cong \uparrow & & \uparrow & & \uparrow \cong \\
L(e \times_h e') & \longrightarrow & x \times_z x' & \longleftarrow & L(f \times_i f') \\
\cong \downarrow & & \downarrow & & \downarrow \cong \\
Lb' & \longrightarrow & v' & \longleftarrow & Lc'
\end{array}
$$

Lemma 24 ensures the marked arrows are monic. The horizontal composition of these is

$$
(\alpha \circ_{\mathrm{v}} \beta) \circ_{\mathrm{h}} (\alpha' \circ_{\mathrm{v}} \beta') =
\begin{array}{ccccc}
La & \longrightarrow & u +_{Lb} v & \longleftarrow & Lc \\
\cong \uparrow & & \uparrow & & \uparrow \cong \\
Ld \times_{Lg} Ld' & \longrightarrow & (w \times_y w') +_{L(e \times_h e')} (x \times_z x') & \longleftarrow & Lf +_{Li} Lf' \\
\cong \downarrow & & \downarrow & & \downarrow \cong \\
La' & \longrightarrow & x' +_{Lb'} v' & \longleftarrow & Lc'
\end{array}
$$

It follows that the proof of Theorem 26 comes down to finding an isomorphism

$$(w \times_y w') +_{L(e \times_h e')} (x \times_z x') \to (w +_{Le} x) \times_{(y +_{Lh} z)} (w' +_{Le'} x')$$

To simplify our diagrams, we introduce new notation. We write

$$p \coloneqq (w \times_y w') + (x \times_z x'), \quad p' \coloneqq (w \times_y w') +_{L(e \times_h e')} (x' \times_z x'),$$

$$q \coloneqq (w + x) \times_{y+z} (w' + x'), \quad q' \coloneqq (w +_{Lg} x) \times_{y +_{Lh} z} (w' +_{Li} x').$$

In this notation, the isomorphism we seek is

$$\theta' \colon p' \to q' \tag{3.6}$$

Also, because Lb, Le, Lh, Le', Lb', and therefore $L(e \times_h e')$ are all isomorphic, we simply write $L*$ to mean any of these. Each are interchangeable in the diagrams below, and adjusting this notation will not cause any false reasoning. While we do lose the ability to discern between these objects, context should help the reader determine this. Despite losing this ability, we gain a breezier exposition and a more readable proof.

Apply Lemma 23 to the diagram

$$\begin{array}{ccccc} w \times_y w' & \longleftarrow & L* & \longrightarrow & x \times_z x' \\ \downarrow & & \downarrow = & & \downarrow \\ y & \longleftarrow & L* & \longrightarrow & z \end{array}$$

to get the pushout

$$\begin{array}{ccc} p & \longrightarrow & p' \\ \downarrow \psi & & \downarrow \psi' \\ y + z & \longrightarrow & y +_{L*} z \end{array}$$

Similarly, we get pushouts

$$\begin{array}{ccc} p & \longrightarrow & p' \\ \downarrow \sigma & & \downarrow \sigma' \\ w + x & \longrightarrow & w +_{L*} x \end{array} \quad \text{and} \quad \begin{array}{ccc} p & \longrightarrow & p' \\ \downarrow \phi & & \downarrow \phi' \\ w' + x' & \longrightarrow & w' +_{L*} x' \end{array}$$

Now, p forms a cone over the cospan $w + x \to y + z \leftarrow w' + x'$ via the maps ψ, σ, and ϕ. And so, we get a canonical map $\theta \colon p \to q$.

Lemma 27. The commuting diagram

$$\begin{array}{ccccc} & & L* & & \\ & \nearrow & \downarrow & \nwarrow & \\ Lg & \longrightarrow & t & \longleftarrow & Li \end{array}$$

induces a canonical isomorphism between $Lg \times_{L*} Li$ and $Lg \times_t L*$.

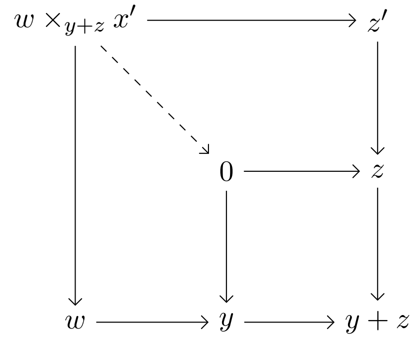
Proof. Via the projection maps, $Lg \times_{L*} Li$ forms a cone over the cospan $Lg \to t \leftarrow Li$ and, also, $Lg \times_t Li$ forms a cone over the cospan $Lg \to L* \leftarrow Li$, though the latter requires the monic $L* \rightarrowtail t$ to do so. Universality implies that the induced maps are mutual inverses and they are the only such pair. ■

Lemma 28. The map $\theta \colon p \to q$ is an isomorphism.

Proof. Because colimits are stable under pullback [47, Thm. 4.7.2], we get an isomorphism

$$\gamma \colon (w \times_{y+z} w') + (w \times_{y+z} x') + (x \times_{y+z} w') + (x \times_{y+z} x') \to q.$$

But $w \times_{y+z} x'$ and $w' \times_{y+z} x$ are initial. To see this, recall that in a topos, all maps to the initial object are isomorphisms. Now, consider the diagram



whose lower right square is a pullback because coproducts are disjoint in topoi. Similarly, $x \times_{y+z} w'$ is initial. Hence we get a canonical isomorphism

$$\gamma' \colon (w \times_{y+z} w') + (x \times_{y+z} x') \to q \tag{3.7}$$

that factors through γ. But Lemma 27 gives unique isomorphisms

$$w \times_y w' \cong w \times_{y+z} w' \text{ and } x \times_z x' \cong x \times_{y+z} x'.$$

This produces a canonical isomorphism

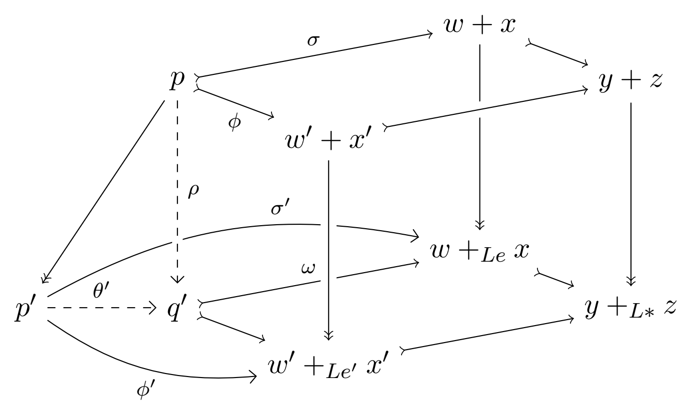
$$\gamma'' \colon p \to (w \times_{y+z} w') + (x \times_{y+z} x').$$

One can show that $\theta = \gamma' \circ \gamma''$ using universal properties. ■

Having shown that $\theta \colon p \to q$ is an isomorphism, we can write p in place of

$$(w + x) \times_{(y+z)} (w' + x')$$

in the following diagram



(3.8)

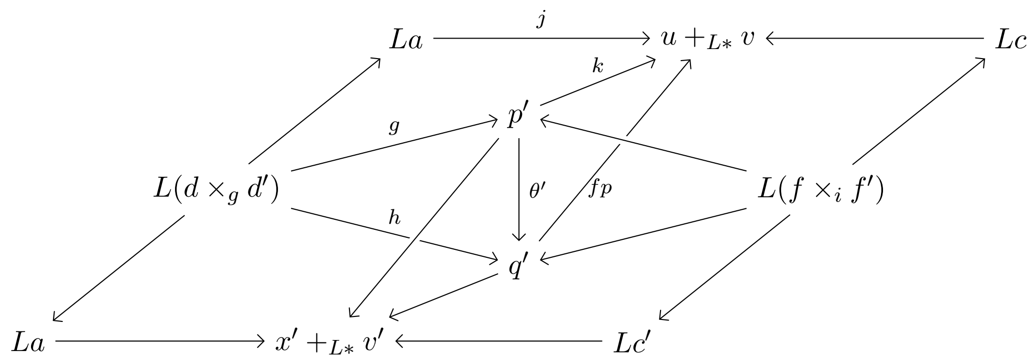
where θ' from Equation (3.6) finally appears. It and ρ are the canonical maps arising from the pullback on the bottom. Observe that ρ factors through θ' in the above diagram. This follows from the universal property of pullbacks.

Lemma 29. The map $\theta' \colon p' \to q'$ is an isomorphism.

Proof. Because we are working in a topos, it suffices to show that θ' is both monic and epic. It is monic because σ' is monic.

To see that θ' is epic, it suffices to show that ρ is epic. The front and rear right faces of (3.8) are pushouts by Lemma 23. Then because the top and bottom squares of (3.8) are pullbacks consisting of only monomorphisms, Lemma 117 implies that the front and rear left faces are pushouts. However, as pushouts over monomorphisms, Lemma 116 tells us they are pullbacks. But in a topos, regular epimorphisms are stable under pullback, and so ρ is epic. ∎

It remains to show that θ' serves as an isomorphism between fine rewrites. This amounts to showing that

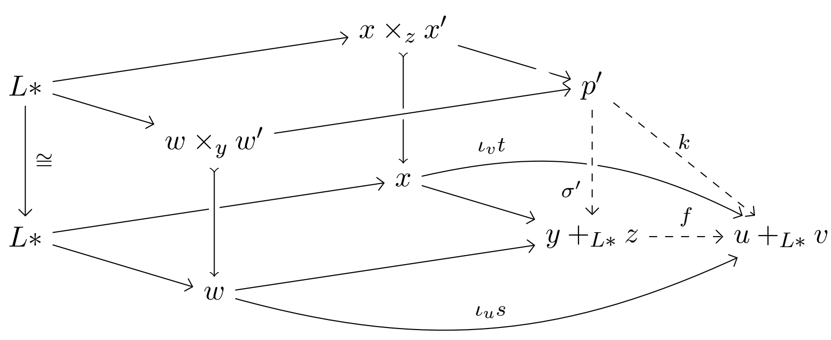


(3.9)

commutes. Here g and k are induced from applying vertical composition before horizontal, h from applying horizontal composition before vertical, j is from composing in either order, f is from horizontal composition as given in Definition 22 and ω is from (3.8). The top and bottom face commute by construction.

Lemma 30. The inner triangles of diagram (3.9) commute. That is, we have $k = f\rho\theta'$ and $h = \theta' g$.

Proof. To see that $k = f\omega\theta'$, consider the diagram



The bottom face is exactly the pushout diagram from which f was obtained. Universality implies that $k = f\sigma'$ and, as seen in (3.8), $\sigma' = \rho\theta'$.

That $h = \theta' g$ follows from

$$f\rho h = j = kg = f\rho\theta' g$$

and the fact that $f\rho$ is monic. ■

Of course, we have only shown that two of the four inner triangles commute, but we can replicate our arguments to show the remaining two commute as well. This lemma was the last step in proving Theorem 26, the interchange law.

3.2 A symmetric monoidal structure

The double category ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ can be equipped with a symmetric monoidal structure lifted from the cocartesian structure on A and X. Proving this amounts to checking the axioms of Definition 86.

Lemma 31. ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ is a symmetric monoidal double category.

Proof. We denote ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ by $\mathbb{R}$ for convenience. Let us first show that the category of objects $\mathbb{R}_0$ and the category of arrows $\mathbb{R}_1$ are symmetric monoidal categories.

We obtain the monoidal structure $(\otimes_0, 0_{\mathsf{A}})$ on $\mathbb{R}_0$ by lifting the cocartesian structure on A to the objects and by defining

$$(a \xleftarrow{f} b \xrightarrow{g} c) \otimes_0 (a' \xleftarrow{f'} b' \xrightarrow{g'} c') := (a + a' \xleftarrow{f+g} b + b' \xrightarrow{f'+g'} c + c')$$

on morphisms. Universal properties provide the associator and unitors as well as the coherence axioms. This monoidal structure is clearly symmetric.

Next, we have the category $\mathbb{R}_1$ whose objects are the structured cospans and morphisms are their fine rewrites. We obtain a symmetric monoidal structure

$$(\otimes_1, L0_{\mathsf{A}} \to L0_{\mathsf{A}} \leftarrow L0_{\mathsf{A}})$$

on the objects via

$$(La \to x \leftarrow La') \otimes_1 (Lb \to y \leftarrow Lb') := (L(a+b) \to x+y \leftarrow L(a'+b'))$$

and on the morphisms by

$$\begin{array}{ccccc} La & \longrightarrow & v & \longleftarrow & La' \\ \uparrow & & \uparrow & & \uparrow \\ Lb & \longrightarrow & w & \longleftarrow & Lb' \\ \downarrow & & \downarrow & & \downarrow \\ Lc & \longrightarrow & x & \longleftarrow & Lc' \end{array} \quad \otimes_1 \quad \begin{array}{ccccc} La'' & \longrightarrow & v' & \longleftarrow & La''' \\ \uparrow & & \uparrow & & \uparrow \\ Lb'' & \longrightarrow & w' & \longleftarrow & Lb''' \\ \downarrow & & \downarrow & & \downarrow \\ Lc'' & \longrightarrow & x' & \longleftarrow & Lc''' \end{array} \quad :=$$

$$\begin{array}{ccccc} L(a+a'') & \longrightarrow & v+v' & \longleftarrow & L(a'+a''') \\ \uparrow & & \uparrow & & \uparrow \\ L(b+b'') & \longrightarrow & w+w' & \longleftarrow & L(b'+b''') \\ \downarrow & & \downarrow & & \downarrow \\ L(c+c'') & \longrightarrow & x+x' & \longleftarrow & (c'+c''') \end{array}$$

Again, universal properties provide the associator, unitors, and coherence axioms. Hence both $\mathbb{R}_0$ and $\mathbb{R}_1$ are symmetric monoidal categories.

It remains to find globular isomorphisms $\mathfrak{x}$ and $\mathfrak{u}$ and their coherence. To find $\mathfrak{x}$, fix horizontal 1-morphisms

$$La \to x \leftarrow La', \qquad La' \to x' \leftarrow La'',$$

$$Lb \to y \leftarrow Lb', \qquad Lb' \to y' \leftarrow Lb''.$$

The globular isomorphism $\mathfrak{x}$ is an invertible 2-morphism with domain

$$L(a+b) \to (x+y) +_{L(a'+b')} (x'+y') \leftarrow L(a''+b'')$$

and codomain

$$L(a+b) \to (x +_{La'} y) + (x' +_{Lb'} y') \leftarrow L(a''+b'')$$

This comes down to finding an isomorphism in X between the apexes of the above cospans. Such an isomorphism exists, and is unique, because both apexes are colimits of the non-connected diagram

$$La \nearrow x \nwarrow La' \nearrow x' \nwarrow La'' \quad Lb \nearrow y \nwarrow Lb' \nearrow y' \nwarrow Lb''$$

Moreover, the resulting globular isomorphism is a fine rewrite of structured cospans because the universal maps are isomorphisms. The globular isomorphism $\mathfrak{u}$ is similar.

Finally, we check that the coherence axioms, namely (a)-(k) of Definition 86, hold. These are straightforward, though tedious, to verify. For instance, if we have

$$M_1 = La \nearrow x \nwarrow La' \quad M_2 = La' \nearrow x' \nwarrow La'' \quad M_3 = La'' \nearrow x'' \nwarrow La'''$$

$$N_1 = Lb \nearrow y \nwarrow Lb' \quad N_2 = Lb' \nearrow y' \nwarrow Lb'' \quad N_3 = Lb'' \nearrow y'' \nwarrow Lb'''$$

then following Diagram (5) around the top right gives the sequence of cospans

$$((M_1 \otimes N_1) \odot (M_2 \otimes N_2)) \odot (M_3 \otimes N_3) =$$

$$L(a+b) \longrightarrow ((x+y) +_{L(a'+b')} (x'+y')) +_{L(a''+b'')} (x''+y'') \longleftarrow L(a''+b'')$$

$$((M_1 \odot M_2) \otimes (N_1 \odot N_2)) \odot (M_3 \otimes N_3) =$$

$$L(a+b) \longrightarrow ((x +_{La'} x') + (y +_{Lb'} y')) +_{L(a''+b'')} (x''+y'') \longleftarrow L(a''+b'')$$

$$((M_1 \odot M_2) \odot M_3) \otimes ((N_1 \odot N_2) \odot N_3) =$$

$$L(a+b) \longrightarrow ((x +_{La'} x') +_{La''} x'') + ((y +_{Lb'} y') +_{Lb''} y'') \longleftarrow L(a''+b'')$$

$$(M_1 \odot (M_2 \odot M_3)) \otimes (N_1 \odot (N_2 \odot N_3)) =$$

$$L(a+b) \longrightarrow (x +_{La'} (x' +_{La''} x'')) + (y +_{Lb'} (y' +_{Lb''} y'')) \longleftarrow L(a''+b'')$$

Following the diagram (5) around the bottom left gives another sequence of cospans

$$((M_1 \otimes N_1) \odot (M_2 \otimes N_2)) \odot (M_3 \otimes N_3) =$$

$$L(a+b) \longrightarrow ((x+y) +_{L(a'+b')} (x'+y')) +_{L(a''+b'')} (x''+y'') \longleftarrow L(a''+b'')$$

$$(M_1 \otimes N_1) \odot ((M_2 \otimes N_2) \odot (M_3 \otimes N_3)) =$$

$$L(a+b) \longrightarrow (x+y) +_{L(a'+b')} ((x'+y') +_{L(a''+b'')} (x''+y'')) \longleftarrow L(a'''+b''')$$

$$(M_1 \otimes N_1) \odot ((M_2 \odot M_3) \otimes (N_2 \odot N_3)) =$$

$$L(a+b) \longrightarrow (x+y) +_{L(a'+b')} ((x' +_{La''} x'') + (y' +_{Lb''} y'')) \longleftarrow L(a'''+b''')$$

$$(M_1 \odot (M_2 \odot M_3)) \otimes (N_1 \odot (N_2 \odot N_3)) =$$

$$L(a+b) \longrightarrow (x +_{La'} (x' +_{La''} x'')) + (y +_{Lb'} (y' +_{Lb''} y'')) \longleftarrow L(a'''+b''')$$

Putting these together gives the following commutative diagram.

$$
\begin{array}{ccccc}
L(a+b) & \longrightarrow & (x +_{La'} (x' +_{La''} x'')) + (y +_{Lb'} (y' +_{Lb''} y'')) & \longleftarrow & L(a''' + b''') \\
\uparrow & & \uparrow & & \uparrow \\
L(a+b) & \longrightarrow & ((x +_{La'} x') +_{La''} x'') + ((y +_{Lb'} y') +_{Lb''} y'') & \longleftarrow & L(a''' + b''') \\
\uparrow & & \uparrow & & \uparrow \\
L(a+b) & \longrightarrow & ((x +_{La'} x') + (y +_{Lb'} y')) +_{L(a''+b'')} (x'' + y'') & \longleftarrow & L(a''' + b''') \\
\uparrow & & \uparrow & & \uparrow \\
L(a+b) & \longrightarrow & ((x + y) +_{L(a'+b')} (x' + y')) +_{L(a''+b'')} (x'' + y'') & \longleftarrow & L(a''' + b''') \\
\downarrow & & \downarrow & & \downarrow \\
L(a+b) & \longrightarrow & (x + y) +_{L(a'+b')} ((x' + y') +_{L(a''+b'')} (x'' + y'')) & \longleftarrow & L(a''' + b''') \\
\downarrow & & \downarrow & & \downarrow \\
L(a+b) & \longrightarrow & (x + y) +_{L(a'+b')} ((x' +_{la''} x'') + (y' +_{La''} y'')) & \longleftarrow & L(a''' + b''') \\
\downarrow & & \downarrow & & \downarrow \\
L(a+b) & \longrightarrow & (x +_{La'} (x' +_{La''} x'')) + (y +_{Lb'} (y' +_{Lb''} y'')) & \longleftarrow & L(a''' + b''')
\end{array}
$$

The vertical 1-morphisms on the left and right are the the respective identity spans on $L(a+b)$ and $L(a''' + b''')$. The vertical 1-morphisms in the center are isomorphism classes of monic spans where each leg is given by a universal map between two colimits of the same diagram. The horizontal 1-morphisms are given by universal maps into coproducts and pushouts. The top cospan is the same as the bottom cospan, making a bracelet-like figure in which all faces commute. The other diagrams witnessing coherence are given in a similar fashion. ■

3.3 A compact closed bicategory of spans of cospans

Double categories have many nice features yet are not as established in the world of higher categories as bicategories. For those who more comfortable with bicategories, we write this section to discuss a bicategory of fine rewrites of structured cospans. Intuitively, it is straightforward to pass from the double category ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ to a bicategory of fine rewrites. By only accepting the squares of ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ that fix the inputs and outputs, that is disallow permutations, then the only vertical arrows left are identities. But a double category with only identity vertical arrows is virtually a bicategory. Care is needed, though, because to actually remove a bicategory of fine rewrites from ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ requires more rigor than simply picking out only the vertical arrows that are the identity.

More than a bicategory, we can actually extract a compact closed bicategory from the symmetric monoidal double category ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$. To obtain a symmetric monoidal bicategory from ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$, we use machinery developed by Shulman [58]. To show that this bicategory is also compact closed, we use work by Stay [59].

First, let us extract the 'horizontal bicategory' of ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$, so named because we remove the vertical arrows.

Definition 32. Define ${}_L\mathbf{FineRewrite}$ to be the bicategory whose objects are the objects of A, 1-arrows are structured cospans, and 2-arrows are fine rewrite rules of form

$$\begin{array}{ccccc} La & \longrightarrow & x & \longleftarrow & Lb \\ {\scriptstyle \mathrm{id}}\uparrow & & \uparrow & & \uparrow{\scriptstyle \mathrm{id}} \\ La & \longrightarrow & y & \longleftarrow & Lb \\ {\scriptstyle \mathrm{id}}\downarrow & & \downarrow & & \downarrow{\scriptstyle \mathrm{id}} \\ La & \longrightarrow & z & \longleftarrow & Lb \end{array}$$

That this is a double category follows from Shulman's construction mentioned in Definition 89. Had we used the same notation as that definition, we would let ${}_L\mathbf{FineRewrite} :=$

$\mathcal{H}({}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite})$.

Shulman's construction continues to be useful, as we use it to show that ${}_L\mathbf{FineRewrite}$ is symmetric monoidal. The first step towards this is showing that ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ is isofibrant (see Definition 88).

Lemma 33. The symmetric monoidal double category ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ is isofibrant.

Proof. The companion of a vertical 1-morphism

$$f = (a \xleftarrow{\theta} b \xrightarrow{\psi} c)$$

is given by

$$\widehat{f} = (La \xrightarrow{L\theta^{-1}} Lb \xleftarrow{L\psi^{-1}} Lc)$$

The required 2-arrows are given by

$$\begin{array}{ccccc} La & \longrightarrow & Lb & \longleftarrow & Lc \\ \uparrow & & \uparrow & & \uparrow \\ Lb & \longrightarrow & Lc & \longleftarrow & Lc \\ \downarrow & & \downarrow & & \downarrow \\ Lc & \longrightarrow & Lc & \longleftarrow & Lc \end{array} \quad \text{and} \quad \begin{array}{ccccc} La & \longrightarrow & La & \longleftarrow & La \\ \uparrow & & \uparrow & & \uparrow \\ La & \longrightarrow & La & \longleftarrow & Lb \\ \downarrow & & \downarrow & & \downarrow \\ La & \longrightarrow & Lb & \longleftarrow & Lc \end{array}$$

The conjoint of f is given by $\check{f} = \widehat{f}^{\mathrm{op}}$. ■

Because the symmetric monoidal double category ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ is isofibrant, Theorem 90 extracts a symmetric monoidal bicategory ${}_L\mathbf{FineRewrite}$ comprised of the same objects, structured cospans as arrows, and isomorphism classes of fine rewrites of structured cospans with form

$$
\begin{array}{ccccc}
La & \longrightarrow & v & \longleftarrow & La' \\
\uparrow{\scriptstyle \mathrm{id}} & & \uparrow & & \uparrow{\scriptstyle \mathrm{id}} \\
La & \longrightarrow & w & \longleftarrow & La' \\
\downarrow{\scriptstyle \mathrm{id}} & & \downarrow & & \downarrow{\scriptstyle \mathrm{id}} \\
La & \longrightarrow & x & \longleftarrow & La'
\end{array}
$$

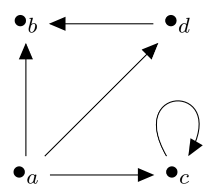
The difference between these fine rewrites and the squares of $_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ is that the vertical arrows are identities. This is necessary given that bicategories have no vertical arrows. However, the isofibrancy condition ensures that information carried by the vertical arrows is encoded the horizontal arrows.

Theorem 34. $_L$**FineRewrite** is a symmetric monoidal bicategory.

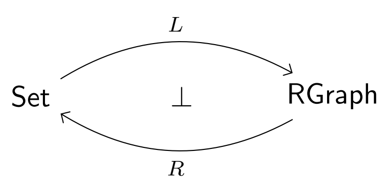
Proof. Lemma 33 states that $_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ is isofibrant. The result then follows from Theorem 90. ■

It remains to show that this bicategory is compact closed. This structure of $_L$**FineRewrite** is another benefit of bicategories over double categories. Currently, there is no notion of compact closedness for double categories. However, it is a nice feature to have in a category that serves as the syntax for open systems with inputs and outputs. Here, we mention again that the terms 'inputs' and 'outputs' do not imply a causal structure. Instead, they partition the interface of an open system into two parts, the purpose of which manifests when composing a pair of systems. If we connect an open system, considered as an structured cospan $La \to x \leftarrow La'$, to another system, then La is parts of the connection and La' is not or vice versa. That is, partitioning an interface into inputs and outputs allows a portion of the interface to be part of a connection and the remain portion to be left out of the connection. Compact closedness formalizes the viewpoint that *how* an interface is partitioned is arbitrary. Indeed, every possible partition of the interface exists as an arrow in ${}_L\mathbf{FineRewrite}$. That is, given a system x with interface i, then for any two subobjects a, a' of i such that $a + a' \cong i$, there is an an arrow $La \to x \leftarrow La'$ in ${}_L\mathbf{FineRewrite}$.

Example 35. Denote by x the graph



with interface $\{a, c, d\}$. Then x appears as an arrow in ${}_L\mathbf{FineRewrite}$ where L is from

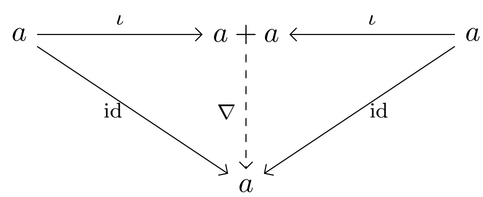


as all of the following

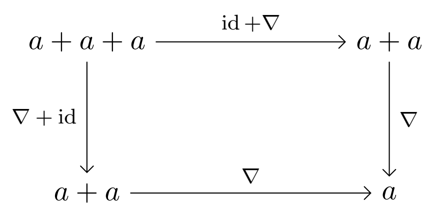
$$\begin{array}{cc}
\{a, c, d\} \to x \leftarrow 0 & \{a, c\} \to x \leftarrow \{d\} \\
\{a, d\} \to x \leftarrow \{c\} & \{c, d\} \to x \leftarrow \{a\} \\
\{a\} \to x \leftarrow \{c, d\} & \{c\} \to x \leftarrow \{a, d\} \\
\{d\} \to x \leftarrow \{a, c\} & 0 \to x \leftarrow \{a, b, c\}
\end{array}$$

The ability to change an input to an output and vice versa comes from the compact closed structure. We take the remainder of this section to show that ${}_L\mathbf{FineRewrite}$ is compact closed.

We start with the following lemma. For this lemma, we introduce the notation $\nabla\colon a + a \to a$ for the folding map, which arises from the coproduct diagram

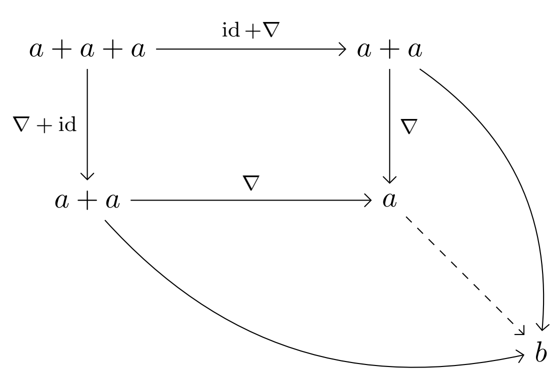


Lemma 36. In a category with coproducts, the diagram



is a pushout square.

Proof. Suppose that we have two maps $f, g\colon a + a \to b$ forming a cocone over the span inside the above diagram. Let the arrow $\iota_{\mathrm{m}}\colon a \to a + a + a$ include a into the middle copy. Observe that $\iota_{\mathrm{l}} \coloneqq (\nabla + a) \circ \iota_{\mathrm{m}}$ and $\iota_r \coloneqq (a + \nabla) \circ \iota_{\mathrm{m}}$ are, respectively, the left and right inclusions $a \to a + a$. Then $f \circ \iota_{\mathrm{l}} = g \circ \iota_{\mathrm{r}}$ is a map $a \to b$, which we claim is the unique map making



commute. Indeed, given $h\colon a \to b$ such that $f = h \circ \nabla = g$, then $g \circ \iota_{\mathrm{r}} = f \circ \iota_{\mathrm{l}} = h \circ \nabla \circ \iota_{\mathrm{l}} = h$.

■

In the following theorem, we will make a slight abuse of notation by writing ∇ to mean

$$L(a+a) \to La + La \xrightarrow{\nabla} La.$$

Here, $L(a+a) \to La+La$ is the structure map which is invertible because, as a left adjoint, L preserves coproducts.

Theorem 37. The symmetric monoidal bicategory ${}_L\mathbf{FineRewrite}$ is compact closed.

Proof. First we show that each object is its own dual. For an object a, define the counit $\varepsilon\colon a+a \to 0$ and unit $\eta\colon 0 \to a+a$ to be the following cospans:

$$\varepsilon := (L(a+a) \xrightarrow{\nabla} La \leftarrow 0), \qquad \eta := (0 \to La \xleftarrow{\nabla} L(a+a)).$$

Next we define the cusp isomorphisms, α and β. Note that α is a 2-morphism whose domain is the composite

$$a \xrightarrow{\iota_{\mathrm{l}}} a+a \xleftarrow{\mathrm{id}+\nabla} a+a+a \xrightarrow{\nabla+\mathrm{id}} a+a \xleftarrow{\iota_{\mathrm{r}}} a$$

and whose codomain is the identity cospan on a. From Lemma 36 we have the equations $\nabla + \mathrm{id} = \iota_{\mathrm{l}} \circ \nabla$ and $\mathrm{id} + \nabla = \iota_{\mathrm{r}} \circ \nabla$ from which it follows that the domain of α is the identity cospan on a, and the codomain of β is also the identity cospan on a obtained as the composite

$$a \xrightarrow{\iota_{\mathrm{r}}} a+a \xleftarrow{\nabla+\mathrm{id}} a+a+a \xrightarrow{X+\nabla} a+a \xleftarrow{\iota_{\mathrm{l}}} a$$

Take α and β each to be the isomorphism class determined by the identity 2-morphism on a, which in particular is a monic span of cospans. Thus we have a dual pair $(a, a, \varepsilon, \eta, \alpha, \beta)$.

By Theorem 103, there exists a cusp isomorphism β' such that $(a, a, \varepsilon, \eta, \alpha, \beta')$ is a coherent dual pair, and thus ${}_L\mathbf{FineRewrite}$ is compact closed. ■

Chapter 4

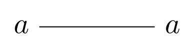
Bold rewriting and structured cospans

We contrast this section with the previous section on fine rewriting with an example. In the fine rewriting of structured cospans, we ask for rewrite rules with the monic arrows as in the diagram

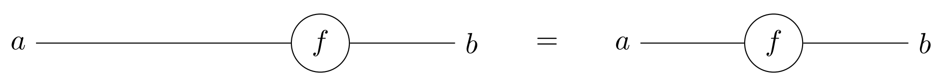
$$\begin{array}{ccccc} La & \longrightarrow & x & \longleftarrow & La' \\ \cong\uparrow & & \uparrow & & \uparrow\cong \\ Lb & \longrightarrow & y & \longleftarrow & Lb' \\ \cong\downarrow & & \downarrow & & \downarrow\cong \\ Lc & \longrightarrow & z & \longleftarrow & Lc' \end{array}$$

There are situations, however, where requiring those monic arrows is untenable. Consider, for instance, the string calculi so frequently use to reason in monoidal categories. For this example, we permit ourselves to ignore details and subtleties so that we do not muddy the point we mean to illustrate. For a detailed and complete look at string calculi, Selinger's survey [57] provides an excellent overview.

Given a monoidal category $(\mathsf{C}, \otimes, I)$, objects are represented by certain isotopy classes of strings and arrows are represented by nodes. This is illustrated in Figure 4.1. The diagrams read from left to right. Now, to draw a string for an identity arrow, we do not include the node, giving the diagram



to represent $\mathrm{id}\colon a \to a$. Composing with id another arrow should result in nothing changing, as captured in this equation

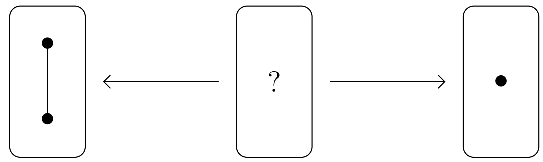


From this, we observe that the length of the string does not matter. This accords with defining strings up to isotopy. In particular, we want to have a string be equivalent to a point. In the parlance of this thesis, we want to be able to rewrite a string, with two distinct endpoints, into a single point. Yet, this is not possible to do with a fine rewrite rule.

A string representing an arrow $f\colon a \to b$ A string representing an arrow $g\colon a \otimes b \to a' \otimes b'$

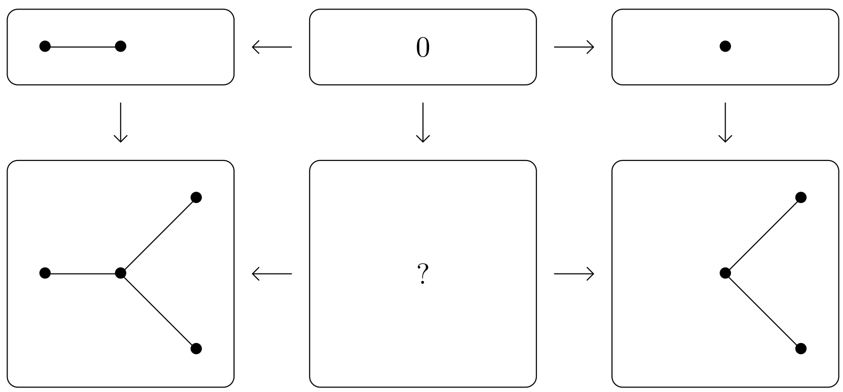
Figure 4.1: String diagrams

Indeed, suppose we are working with strings in some topos of spaces and we want to finely rewrite a string into a point. Such a rewrite rule would be a span

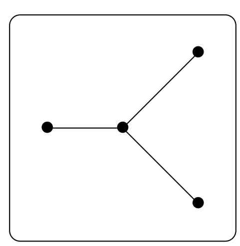


(4.1)

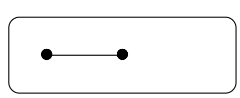
with '?' replaced by a subobject of both the string on the left and point on the right. Thus, '?' must either be empty or a point. Choosing the empty string does not scale. A simple counter example is



To see this more clearly, we reframe the question to take advantage of the fact that pushing out over 0 is the same as taking a disjoint union. So we can ask whether

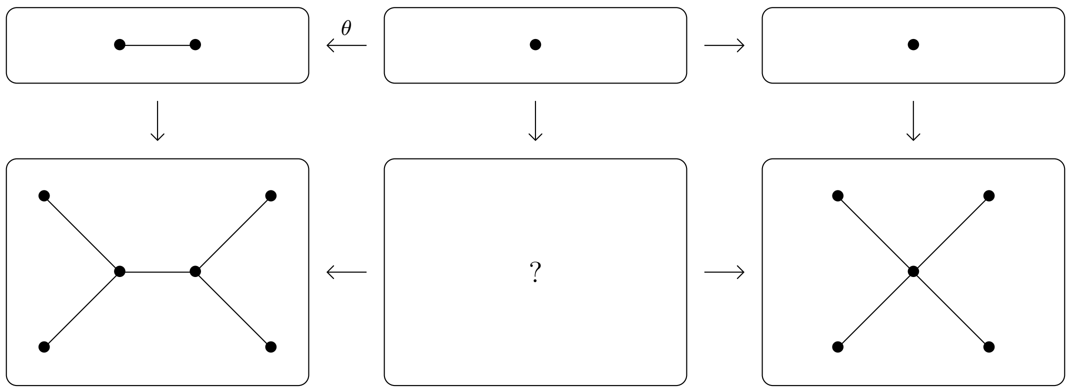


is the disjoint union of

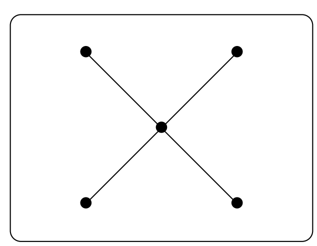


and something else. Of course, it is not.

But maybe the issue was pushing out over 0 in the first place. What about replacing 0 with a point? A simple counter example to illustrate the failure of this idea is



where we define θ to choose the left or the right point; the failure will occur regardless of the choice. Again, there is nothing that we can place into the center, bottom square to give a double pushout diagram. To see why, we use the fact that if we could fill in '?', we already know what it must be. The right square must also be a pushout. This forces us to fill the blank with the graph



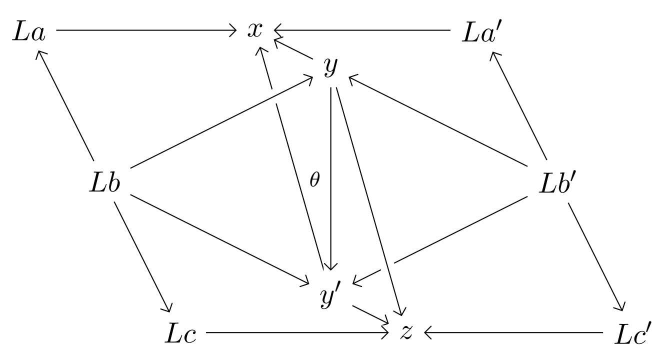
But then the left square is not a pushout.

And so, fine rewriting can be insufficient. In this chapter, we define bold rewriting of structured cospans to handle situations like this one found in string calculi. We see that, though it largely mirrors the fine rewriting of structured cospans, it has its own character: the bicategory we extract is a bicategory of relations. At the end of the chapter, we illustrate bold rewriting using the string calculus from quantum computer science known as the ZX-calculus.

4.1 A double category of bold rewrites of structured cospans

In this section, we define a double category ${}_L\mathbb{B}\mathbf{old}\mathbb{R}\mathbf{ewrite}$ whose objects are interface types, whose vertical arrows are spans of interface types with invertible legs, whose horizontal arrows are structured cospans, and whose squares are bold rewrites of structured cospans. The only difference between the definitions of ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ and ${}_L\mathbb{B}\mathbf{old}\mathbb{R}\mathbf{ewrite}$ is in the squares. The objects, horizontal arrows, and vertical arrows are the same in each case. This winds up having an interesting effect on the horizontal bicategory of ${}_L\mathbb{B}\mathbf{old}\mathbb{R}\mathbf{ewrite}$ which we explore in Section 4.2. Before turning to that, we need to properly define ${}_L\mathbb{B}\mathbf{old}\mathbb{R}\mathbf{ewrite}$, Fortunately, most of the work has been done when constructing ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$, so we begin by defining the squares.

Recall from Definition 13 that a morphism of spans of structured cospans is an arrow θ that fits into a commuting diagram



Using a morphism of structured cospans, we can define the connected components of structured cospans. We first define a relation $\sim$ setting

$$
\begin{array}{ccccc}
La & \longrightarrow & x & \longleftarrow & La' \\
\uparrow & & \uparrow & & \uparrow \\
Lb & \longrightarrow & y & \longleftarrow & Lb' \\
\downarrow & & \downarrow & & \downarrow \\
Lc & \longrightarrow & z & \longleftarrow & Lc'
\end{array}
\quad \sim \quad
\begin{array}{ccccc}
La & \longrightarrow & x & \longleftarrow & La' \\
\uparrow & & \uparrow & & \uparrow \\
Lb & \longrightarrow & y' & \longleftarrow & Lb' \\
\downarrow & & \downarrow & & \downarrow \\
Lc & \longrightarrow & z & \longleftarrow & Lc'
\end{array}
$$

if there is a morphism from the rewriting on the left side of $\sim$ to that on the right. A **connected component of structured cospans** is an equivalence class generated by $\sim$. The coarseness of the classes of squares is the most important distinction between fine and bold rewriting.

Definition 38 (Bold rewrite)**.** A **bold rewrite of structured cospans** is a connected component of structured cospans whose representative has the form

$$
\begin{array}{ccccc}
La & \longrightarrow & x & \longleftarrow & La' \\
\cong\uparrow & & \uparrow & & \uparrow\cong \\
Lb & \longrightarrow & y & \longleftarrow & Lb' \\
\cong\downarrow & & \downarrow & & \downarrow\cong \\
Lc & \longrightarrow & z & \longleftarrow & Lc'
\end{array}
$$

The horizontal and vertical compositions for bold rewrites of structured cospans are defined in the same way as for fine rewrites. The classes are different, but the operation on the class representatives work in the same way.

Definition 39. The **horizontal composition** $\circ_{\mathrm{h}}$ of bold rewrites of structured cospans are defined by the operation

$$
\begin{array}{ccccc}
La & \longrightarrow & x & \longleftarrow & La' \\
\uparrow & & \uparrow & & \uparrow \\
Lb & \longrightarrow & y & \longleftarrow & Lb' \\
\downarrow & & \downarrow & & \downarrow \\
Lc & \longrightarrow & z & \longleftarrow & Lc'
\end{array}
\quad \circ_{\mathrm{h}} \quad
\begin{array}{ccccc}
La' & \longrightarrow & x' & \longleftarrow & La'' \\
\uparrow & & \uparrow & & \uparrow \\
Lb' & \longrightarrow & y' & \longleftarrow & Lb'' \\
\downarrow & & \downarrow & & \downarrow \\
Lc' & \longrightarrow & z' & \longleftarrow & Lc''
\end{array}
\quad :=
$$

$$
\begin{array}{ccccc}
La & \longrightarrow & x +_{La'} x' & \longleftarrow & La'' \\
\uparrow & & \uparrow & & \uparrow \\
Lb & \longrightarrow & y +_{Lb'} y' & \longleftarrow & Lb'' \\
\downarrow & & \downarrow & & \downarrow \\
Lc & \longrightarrow & z +_{Lc'} z' & \longleftarrow & Lc''
\end{array}
$$

The **vertical composition** of bold rewrites of structured cospans is defined by

$$
\begin{array}{ccccc}
La & \longrightarrow & v & \longleftarrow & La' \\
\uparrow & & \uparrow & & \uparrow \\
Lb & \longrightarrow & w & \longleftarrow & Lb' \\
\downarrow & & \downarrow & & \downarrow \\
Lc & \longrightarrow & x & \longleftarrow & Lc'
\end{array}
\quad \circ_{\mathrm{h}} \quad
\begin{array}{ccccc}
Lc & \longrightarrow & x & \longleftarrow & Lc' \\
\uparrow & & \uparrow & & \uparrow \\
Ld & \longrightarrow & y & \longleftarrow & Ld' \\
\downarrow & & \downarrow & & \downarrow \\
Le & \longrightarrow & z & \longleftarrow & Le'
\end{array}
\quad :=
$$

$$
\begin{array}{ccccc}
La & \longrightarrow & v & \longleftarrow & La' \\
\uparrow & & \uparrow & & \uparrow \\
Lb \times_{Lc} Ld & \longrightarrow & w \times_x y & \longleftarrow & Lb' \times_{Lc'} Ld' \\
\downarrow & & \downarrow & & \downarrow \\
Le & \longrightarrow & z & \longleftarrow & Le'
\end{array}
$$

Unlike for fine rewrites of structured cospans, the interchange law is straightforward to prove. The coarser classes of rewrites of structured cospans vastly simplifies concocting the isomorphism.

Lemma 40. Let

$$
\alpha := \begin{array}{ccccc}
La & \longrightarrow & v & \longleftarrow & La' \\
\uparrow & & \uparrow & & \uparrow \\
Lb & \longrightarrow & w & \longleftarrow & Lb' \\
\downarrow & & \downarrow & & \downarrow \\
Lc & \longrightarrow & x & \longleftarrow & Lc'
\end{array}
\qquad
\alpha' := \begin{array}{ccccc}
La' & \longrightarrow & v' & \longleftarrow & La'' \\
\uparrow & & \uparrow & & \uparrow \\
Lb' & \longrightarrow & w' & \longleftarrow & Lb'' \\
\downarrow & & \downarrow & & \downarrow \\
Lc' & \longrightarrow & x' & \longleftarrow & Lc''
\end{array}
$$

$$
\beta := \begin{array}{ccccc}
Lc & \longrightarrow & x & \longleftarrow & Lc' \\
\uparrow & & \uparrow & & \uparrow \\
Ld & \longrightarrow & y & \longleftarrow & Ld' \\
\downarrow & & \downarrow & & \downarrow \\
Le & \longrightarrow & z & \longleftarrow & Le'
\end{array}
\qquad
\beta' := \begin{array}{ccccc}
Lc' & \longrightarrow & x' & \longleftarrow & Lc'' \\
\uparrow & & \uparrow & & \uparrow \\
Ld' & \longrightarrow & y' & \longleftarrow & Ld'' \\
\downarrow & & \downarrow & & \downarrow \\
Le' & \longrightarrow & z' & \longleftarrow & Le''
\end{array}
$$

be bold rewrites of structured cospans. Then

$$(\alpha \circ_{\mathrm{h}} \alpha') \circ_{\mathrm{v}} (\beta \circ_{\mathrm{h}} \beta') = (\alpha \circ_{\mathrm{v}} \beta) \circ_{\mathrm{h}} (\alpha' \circ_{\mathrm{v}} \beta').$$

That is, the interchange law holds.

Proof. The left hand side of the equation is the bold rewrite of structured cospans

$$\begin{array}{ccccc}
La & \longrightarrow & v +_{La'} v' & \longleftarrow & La'' \\
\uparrow & & \uparrow & & \uparrow \\
Lb \times_{Lc} Ld & \longrightarrow & (w +_{Lb'} w') \times_{(x +_{Lc'} x')} (y \times_{Ld'} y') & \longleftarrow & Lb \times_{Lc'} Ld' \\
\downarrow & & \downarrow & & \downarrow \\
Le & \longrightarrow & z +_{Le'} z' & \longleftarrow & Le''
\end{array}$$

while the right hand side is

$$\begin{array}{ccccc}
La & \longrightarrow & v +_{La'} v' & \longleftarrow & La'' \\
\uparrow & & \uparrow & & \uparrow \\
Lb \times_{Lc} Ld & \longrightarrow & (w \times_x y) +_{(Lb' \times_{Lc'} Ld')} (w' \times_{x'} y') & \longleftarrow & Lb \times_{Lc'} Ld' \\
\downarrow & & \downarrow & & \downarrow \\
Le & \longrightarrow & z +_{Le'} z' & \longleftarrow & Le''
\end{array}$$

To show that these are equal as bold rewrites of structured cospans, it suffices to find a *morphism* between them. Precisely, we need a morphism

$$(w \times_x y) +_{(Lb' \times_{Lc'} Ld')} (w' \times_{x'} y') \to (w +_{Lb'} w') \times_{(x +_{Lc'} x')} (y \times_{Ld'} y')$$

We can obtain the two objects as follows. Let C be the walking cospan category $\{\bullet \to \bullet \leftarrow \bullet\}$ and let S be the walking span category $\{\bullet \leftarrow \bullet \to \bullet\}$. Then $\mathsf{C} \times \mathsf{S}$ is the walking cospan of spans category

$$
\boxed{\begin{array}{ccccc}
\bullet & \longleftarrow & \bullet & \longrightarrow & \bullet \\
\downarrow & & \downarrow & & \downarrow \\
\bullet & \longleftarrow & \bullet & \longrightarrow & \bullet \\
\uparrow & & \uparrow & & \uparrow \\
\bullet & \longleftarrow & \bullet & \longrightarrow & \bullet
\end{array}}
$$

Let $F\colon \mathsf{C} \times \mathsf{S} \to \mathsf{X}$ be the functor that returns the diagram

$$
\begin{array}{ccccc}
w & \longleftarrow & Lb' & \longrightarrow & w' \\
\downarrow & & \downarrow & & \downarrow \\
x & \longleftarrow & Lc' & \longrightarrow & x' \\
\uparrow & & \uparrow & & \uparrow \\
y & \longleftarrow & Ld' & \longrightarrow & y'
\end{array}
$$

which is the middle of the diagram obtained by gluing α, β, α', and β' together along their coinciding edges. There is a canonical morphism of type

$$\operatorname{colim}_{\mathsf{S}} \lim_{\mathsf{C}} F \to \lim_{\mathsf{C}} \operatorname{colim}_{\mathsf{S}} F$$

where the domain is the image of F under the composite functor

$$\mathsf{X}^{\mathsf{C}\times\mathsf{S}} \xrightarrow{\cong} (\mathsf{X}^{\mathsf{C}})^{\mathsf{S}} \xrightarrow{\lim_{\mathsf{C}}} \mathsf{X}^{\mathsf{S}} \xrightarrow{\operatorname{colim}_{\mathsf{S}}} \mathsf{X}$$

and the domain is the image of F under the composite functor

$$\mathsf{X}^{\mathsf{C}\times\mathsf{S}} \xrightarrow{\cong} (\mathsf{X}^{\mathsf{S}})^{\mathsf{C}} \xrightarrow{\operatorname{colim}_{\mathsf{S}}} \mathsf{X}^{\mathsf{C}} \xrightarrow{\lim_{\mathsf{C}}} \mathsf{X}.$$

One can check that this canonical morphism gives the morphism of bold rewrites of structured cospans we need. ■

4.2 A bicategory of relations for bold rewriting of structured cospans

There are two philosophies in rewriting. One is that we care about how one object is rewritten into another, and so we keep track of certain data to describe the rewriting. The other perspective is that we do not care about *how* an object is rewritten into another, only that the rewriting is possible. Bold rewriting of structured cospans belongs to the latter philosophy. This is realized explicitly through the fact that the horizontal bicategory forms a bicategory of relations, specifically that it is locally posetal. Appendix A.3 discusses the theory of such bicategories.

The first goal of this section is to define the bicategory in question. We take the same approach as finding the horizontal bicategory of fine rewrites of structured cospans in Section 3.3. After extracting the bicategory, we show that it is a bicategory of relations (see Definition 98).

This next theorem is proved with virtually the same argument as Lemma 31.

Theorem 41. ${}_L\mathbb{B}\mathbf{old}\mathbb{R}\mathbf{ewrite}$ is a symmetric monoidal double category.

From here, we prove a series of lemmas that, when put together, prove that the horizontal bicategory ${}_L\mathbf{BoldRewrite}$ of ${}_L\mathbb{B}\mathbf{old}\mathbb{R}\mathbf{ewrite}$ is a bicategory of relations. The first lemma in this string is proved by replicating the proof of Lemma 33 and the second follows from Theorem 90.

Lemma 42. ${}_L\mathbb{B}\textbf{old}\mathbb{R}\textbf{ewrite}$ is isofibrant.

Lemma 43. ${}_L\textbf{BoldRewrite}$ is a symmetric monoidal bicategory.

In the following lemma, we use $\nabla := [\mathrm{id}, \mathrm{id}] \colon a + a \to a$ to denote the codiagonal map and ! to denote a canonical arrow from the initial object.

Lemma 44. For each object a of ${}_L\textbf{BoldRewrite}$, define operations

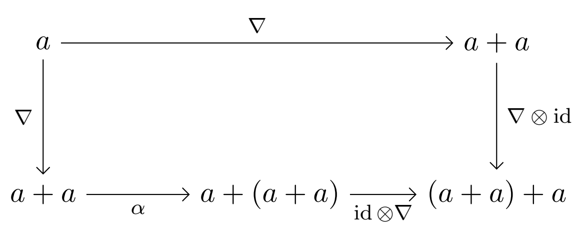
$$\Delta_a \colon a \to a + a \quad \text{and} \quad \varepsilon_a \colon a \to 0$$

to be the structured cospans

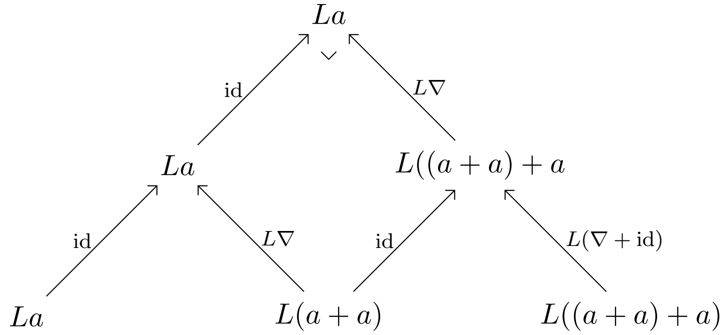
$$La \xrightarrow{\mathrm{id}} La \xleftarrow{L\nabla_a} L(a+a) \quad \text{and} \quad La \xrightarrow{\mathrm{id}} La \xleftarrow{!} L0$$

respectively. Then $(a, \Delta_a, \varepsilon_a)$ is a cocommutative comonoid.

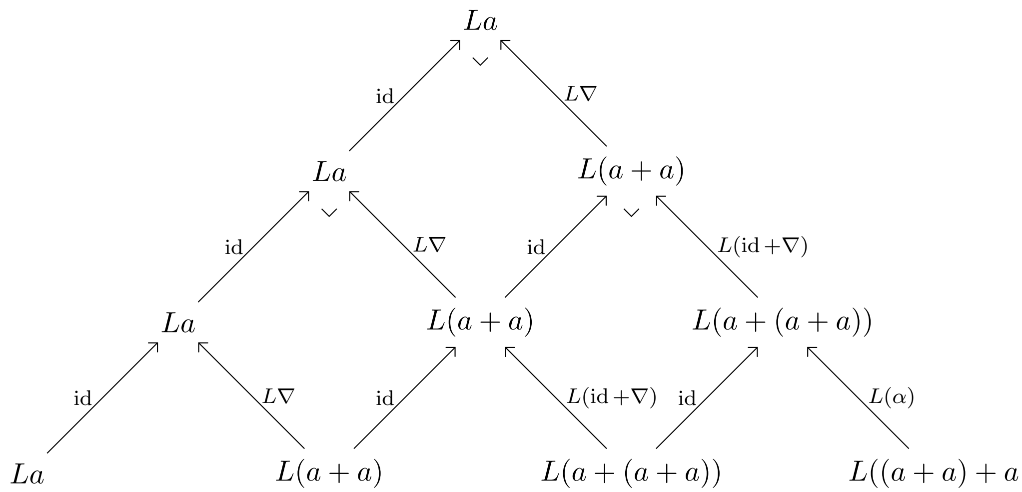
Proof. Proving this amounts to showing that the coassociativity, counitality, and cocommutativity diagrams commute. The coassociativity diagram



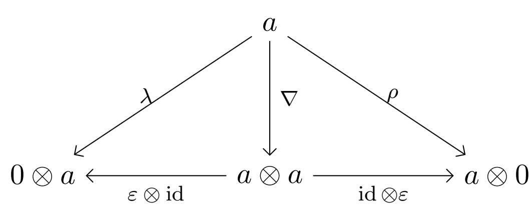
commutes because the top path, which is the composite



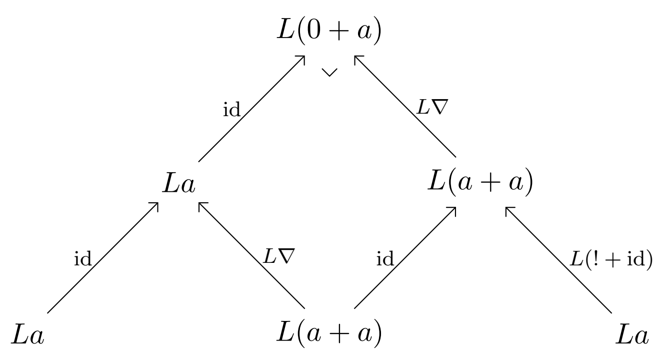
equals the bottom path, which is the composite



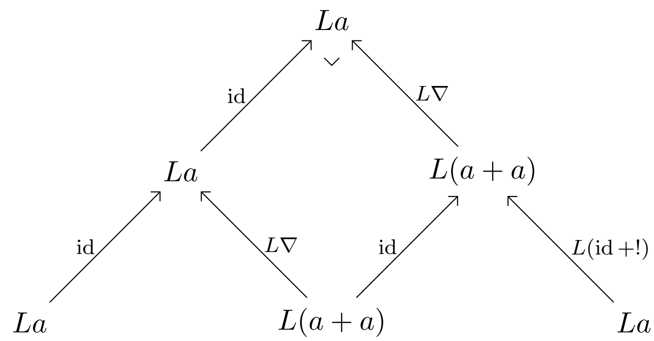
The counitality diagram



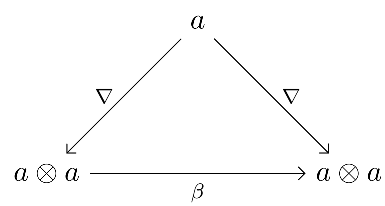
commutes because the composite



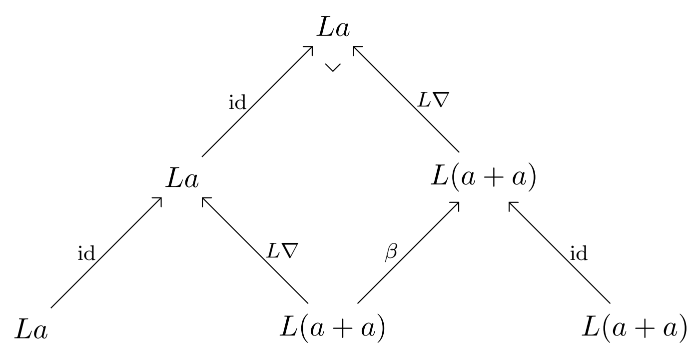
is equal to the left unitor and



is the right unitor. Finally, the cocommutative diagram



commutes because the composite $\nabla\beta$ is given by



which is exactly the comuliplication. ■

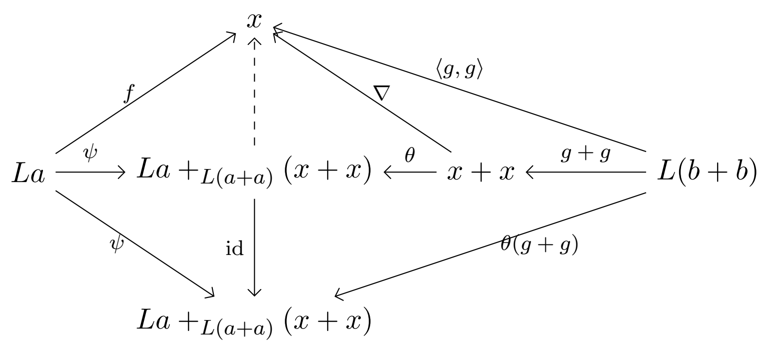
In the following lemma, we follow the convention of writing $f \leq g$ to represent a 2-arrow from f to g in a locally posetal bicategory. This notation is faithful to the fact that the hom-categories are actually hom-posets. This is discussed further in Section A.3.

Lemma 45. Let $(a, \Delta_a, \varepsilon_a)$ and $(b, \Delta_b, \varepsilon_b)$ be cocommutative comonoid objects in ${}_L\mathbf{BoldRewrite}$. Every structured cospan $La \to x \leftarrow Lb$ in ${}_L\mathbf{BoldRewrite}$ is a lax comonoid homomorphism.

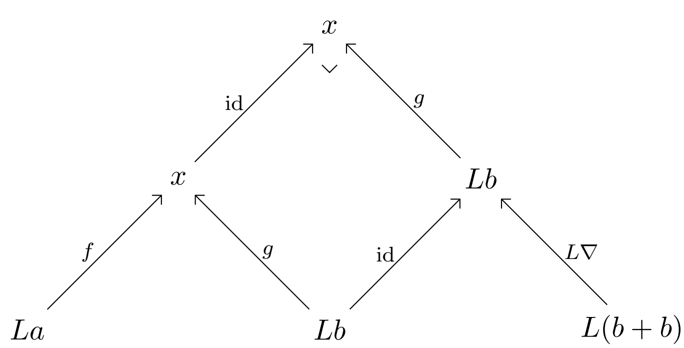
That is,

$$\Delta_b x \leq (x+x)\Delta_a \quad \text{and} \quad \varepsilon_b x \leq \varepsilon a$$

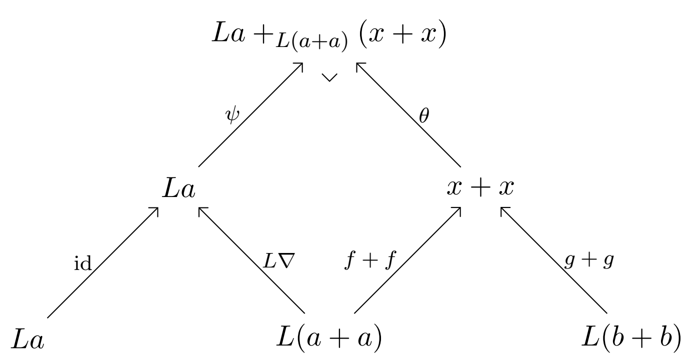
Proof. The first 2-arrow is



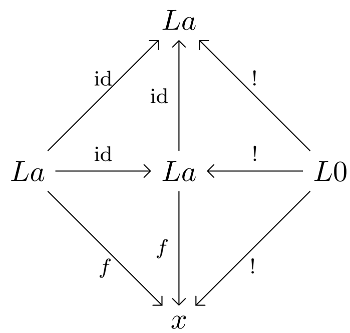
where the dashed line is the universal arrow formed in reference to f and ∇. The source of this 2-arrow is the composite



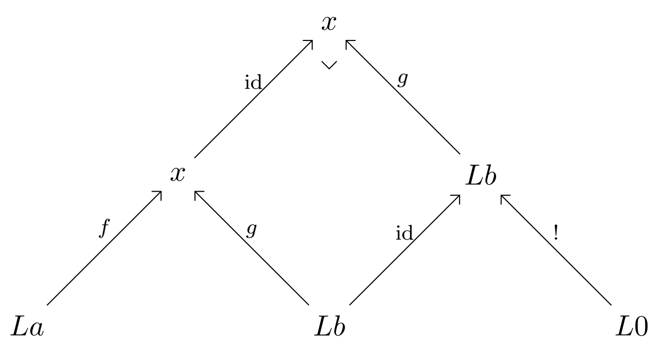
and the target is the composite



The second is witnessed by the 2-arrow



where target 2-arrow is the composite



■

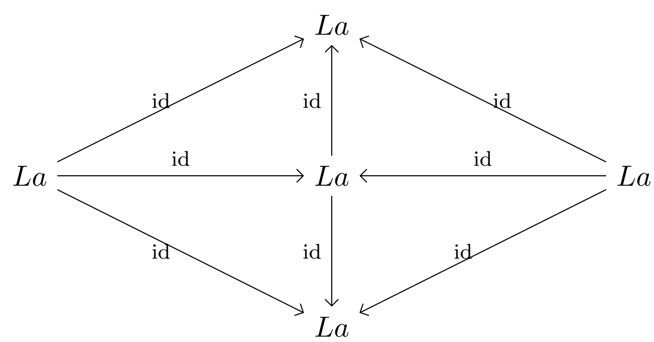
Lemma 46. For any object a in ${}_L\mathbf{BoldRewrite}$, each cocommutative comonoid structure map

$$\nabla := \left(La \xrightarrow{\mathrm{id}} La \xleftarrow{L\nabla_a} L(a+a)\right) \quad \text{and} \quad \varepsilon := \left(La \xrightarrow{\mathrm{id}} La \xleftarrow{!} L0\right)$$

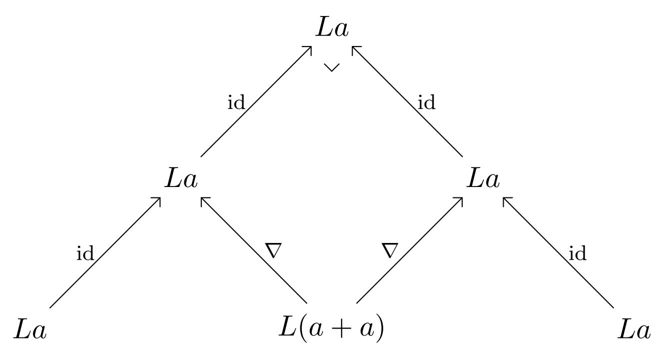
has a right adjoint (see Definition 95), respectively,

$$\nabla^* := \left(L(a+a) \xrightarrow{L\nabla_a} La \xleftarrow{\mathrm{id}} La\right) \quad \text{and} \quad \varepsilon^* := \left(L0 \xrightarrow{!} La \xleftarrow{\mathrm{id}} La\right).$$

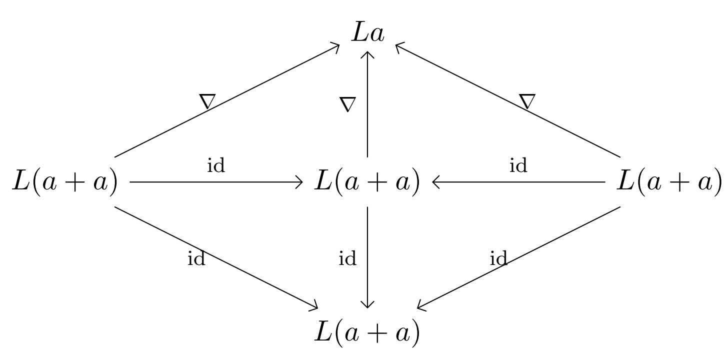
Proof. The unit of the adjunction $\nabla \dashv \nabla^*$ is



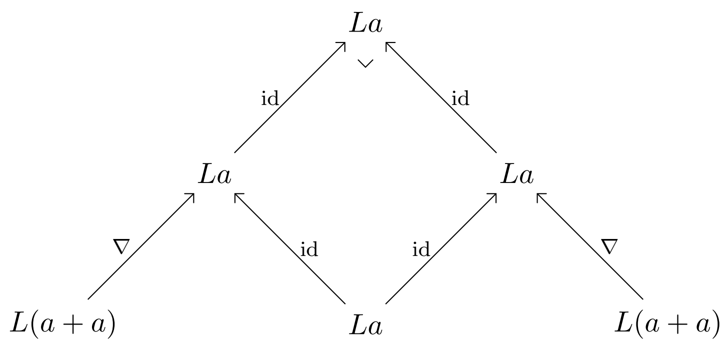
where the target is the composite 1-arrow



The counit of $\nabla \dashv \nabla^*$ is the 2-arrow

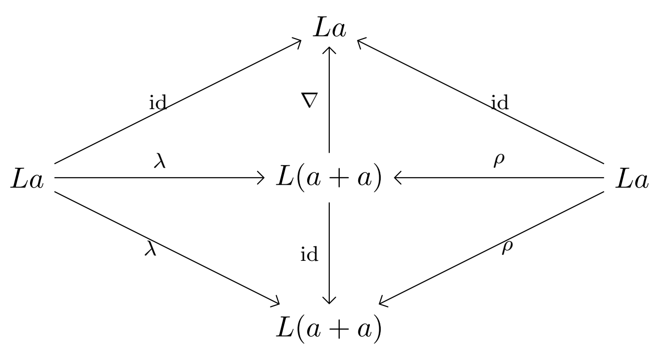


where the source is the composite 1-arrow

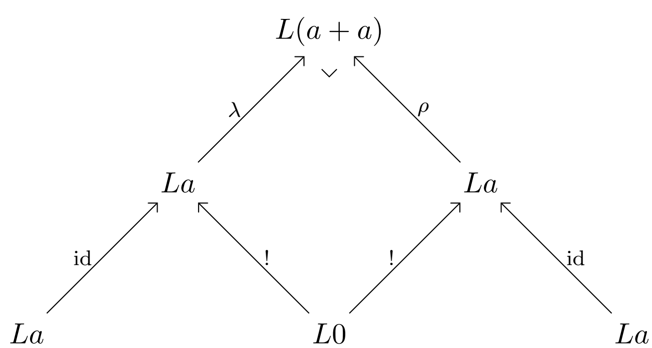


Checking the triangle identities is straightforward.

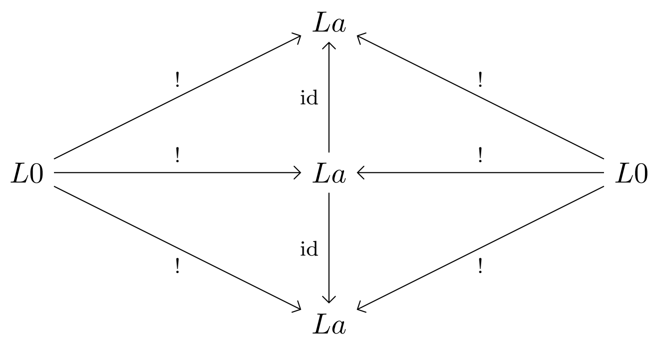
The unit of the adjunction $\varepsilon \dashv \varepsilon^*$ is the 2-arrow



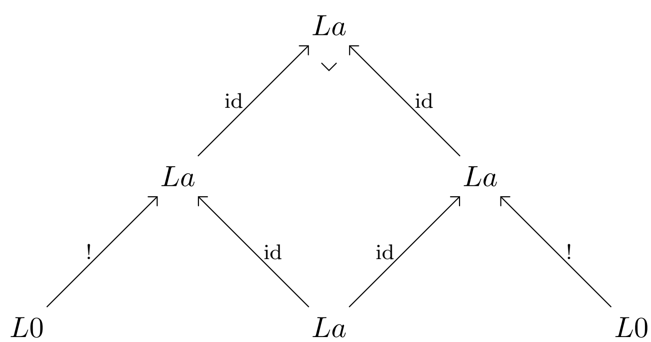
where the target is the composite 1-arrow



the counit of $\varepsilon \dashv \varepsilon^*$ is the 2-arrow



where the source is the composite 1-arrow



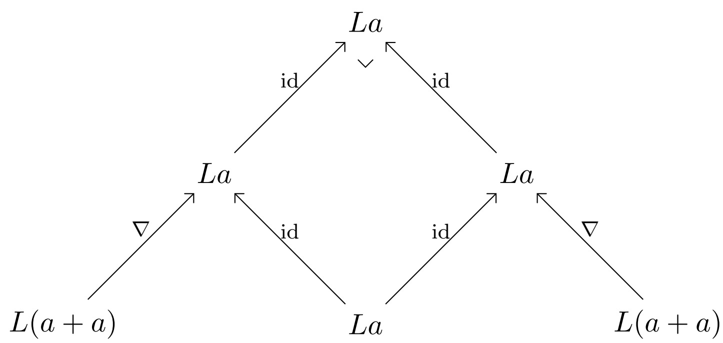
Again, the triangle equations are straightforward to check. ■

The following lemma refers to a 'Frobenius monoid', a monoid and comonoid that satisfy some nice properties that we spell out in Definition 82.

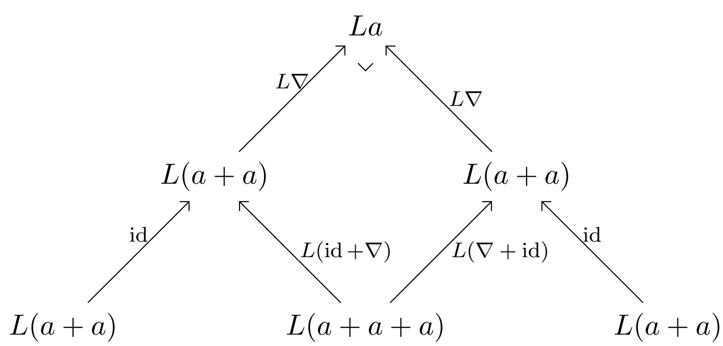
Lemma 47. For any object a of ${}_L$**BoldRewrite**, $(a, \nabla^*, \varepsilon^*, \nabla, \varepsilon)$ is a Frobenius monoid. In particular,

$$\nabla\nabla^* = (\nabla^* \otimes \mathrm{id})\,(\mathrm{id} \otimes \nabla) \tag{4.2}$$

Proof. The left-hand side of Equation 4.2 is given by the composite



The right-hand side is given by



These both compose to $L(a+a) \xrightarrow{L\nabla} La \xleftarrow{L\nabla} L(a+a)$. ■

The following structure theorem follows from this string of lemmas.

Theorem 48. ${}_L$**BoldRewrite** is a bicategory of relations.

4.3 The ZX-calculus

Perhaps one of the most interesting features of quantum mechanics is the incompatibility of observables. Roughly, an observable is a measurable quantity of some system, for instance the spin of a photon. In classical physics, measureable quantities are comparable, meaning that we can obtain arbitrarily precise values at the same time. For example, given a Porsche speeding down the highway, we can simultaneously measure its velocity and its mass with arbitrary precision. Knowledge about its velocity does not preclude us from obtain information about its mass. The situation is quite different in quantum mechanics. Given two measurable quantities, knowledge of one may prevent us from obtaining knowledge about the other. This is illustrated by the famous Heisenberg uncertainty principle which quantifies the limits of precision to which one can simultaneously measure the position and momentum of a particle. In general, the strength of this restriction depends on the situation. The most extreme case is that knowing one quantity with total precision implies total uncertainty about the other quantity. Such a pair of observables are called **complementary**.

Historically, a quantum physicist would reason about observables, complementary or otherwise, using Hilbert spaces. Given the rapid progress of quantum physics in the twentieth century, this framework seems to have worked quite well for scientists. Working with Hilbert spaces, however, is challenging even for skilled researchers. But the language of quantum physics is now relevant to a wider audience since the dawn of quantum computing. Given the challenge of working with Hilbert spaces, perhaps developing a simpler language is worth pursuing.

Such a high-level language was invented by Coecke and Duncan [24]. This language, called the ZX-calculus, was immediately used to generalize both quantum circuits [51] and the measurement calculus [26]. Its validity was further justified when Duncan and Perdrix presented a non-trivial method of verifying measurement-based quantum computations [31]. At its core, the ZX-calculus is an intuitive graphical language in which to reason about complementary observables.

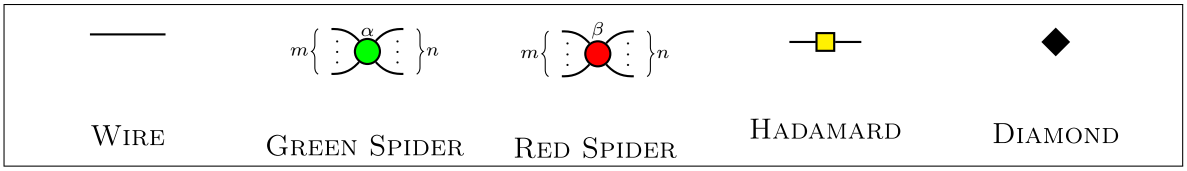


Figure 4.2: Generators for the ZX-calculus diagrams

In this section, we illustrate our framework with the ZX-calculus. The backstory of the ZX-calculus dates to Penrose's tensor networks [54] and, more recently, to the relationship between graphical languages and monoidal categories [39, 57]. Abramsky and Coecke capitalized on this relationship when inventing a categorical framework for quantum physics [1]. Soon after, Coecke and Duncan introduced a diagrammatic language in which to reason about complementary quantum observables [19]. After a fruitful period of development [20, 23, 30, 31, 29, 53], a full presentation of the ZX-calculus was published [24]. The completeness of the ZX-calculus for stabilizer quantum mechanics was later proved by Backens [4].

The ZX-calculus begins with the five diagrams depicted in Figure 4.2. On each diagram, the dangling wires on the left are **inputs** and those on the right are **outputs**. By connecting inputs to outputs, we can form larger diagrams, which we call **ZX-diagrams**. These diagrams generate the arrows of a dagger compact category ZX whose objects, the non-negative integers, count the inputs and outputs of a diagram. Below, we give a presentation of ZX along with a brief discussion on the origins of its generating arrows (Figure 4.2) and relations (Figure 4.3).

Our goal with this example is to generate, using the machinery laid out in this chapter, a bicategory of relations **ZX** to provide a syntax for the ZX-calculus. We show that **ZX** extends ZX in a way we make precise below.

The five **basic diagrams** in the ZX-calculus are depicted in Figure 4.2 and are to be read from left to right. They are

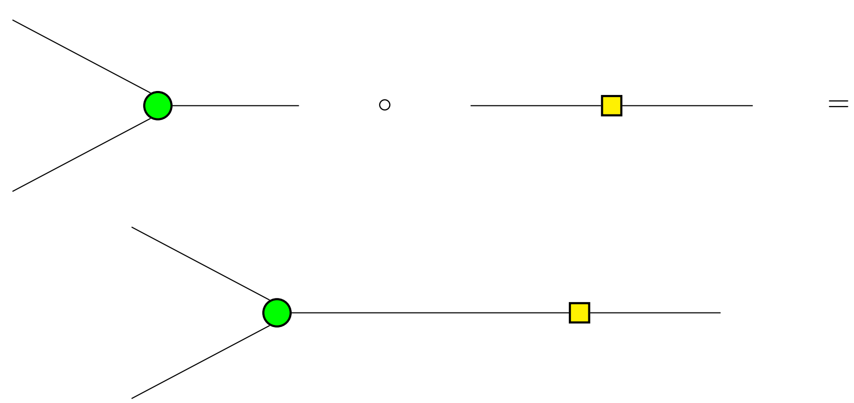
- a **wire** with a single input and output,
- **green spiders** with a non-negative integer number of inputs and outputs and paired with a phase $\alpha \in [-\pi, \pi)$,
- **red spiders** with a non-negative integer number inputs and outputs and paired with a phase $\beta \in [-\pi, \pi)$,
- the **Hadamard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

The wire plays the role of an identity, much like a wire without resistance in an electrical circuit, or straight pipe in a plumbing system. The green and red spiders each arise from a pair of complementary observables. In categorical quantum mechanics [1], observables correspond to certain commutative Frobenius algebras A living in a dagger symmetric monoidal category $(\mathsf{C}, \otimes, I)$, the classic example $\mathsf{C} := \mathsf{FinHilb}$ being the category of finite dimensional Hilbert spaces and linear maps. A pair of complementary observables gives a pair of Frobenius algebras whose operations interact via laws like those of a Hopf algebra [21, 22]. This is particularly nice because Frobenius algebras have beautiful string diagram representations. There is an morphism $\mathsf{C}(I, A) \to \mathsf{C}(A, A)$ of commutative monoids that gives rise to a group structure on A known as the **phase group**, which Coecke and Duncan detail [24, Def. 7.5]. The phases on the green and red spider diagrams arise from this group. The Hadamard node embodies the Hadamard gate. The diamond is a scalar obtained when connecting a green and red node together. A deeper exploration of these notions goes beyond the scope of this paper. For those interested, the original paper on the topic [24] is an excellent place learn more.

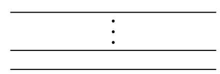
In the spirit of compositionality, we present a category ZX whose arrows are generated by the five basic diagrams. We sketched ZX at the beginning of this section, but we now detail the construction.

We start by allowing the basic ZX-diagrams from Figure 4.2 generate the arrows of a free dagger compact category whose objects are the non-negative integers. We then subject the arrows (ZX-diagrams) to the relations given in Figure 4.3, to which we add equations obtained by exchanging red and green nodes, daggering, and taking diagrams up to ambient isotopy in 4-space. These listed relations are called **basic**. Spiders with no phase indicated have a phase of 0.

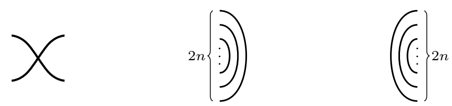
This category, denoted as ZX, was introduced by Coecke and Duncan [24] and further studied by Backens [4]. To compose in ZX, connect compatible diagrams along a bijection between inputs and the outputs. For example



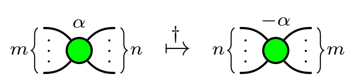
A monoidal structure is given by adding numbers and taking the disjoint union of ZX-diagrams. The identity on n is the disjoint union of n wires:



The symmetry and compactness of the monoidal product provide a braiding, evaluation, and coevaluation morphisms: respectively,



The evaluation and coevalutation arrows are of type $2n \to 0$ and $0 \to 2n$ for each object $n \geq 1$ and the empty diagram for $n = 0$. On the spider diagrams, the dagger structure swaps inputs and outputs then multiplies the phase by -1:



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

A major advantage of using string diagrams, apart from their intuitive nature, is that computations are more easily programmed into computers. Indeed, graphical proof assistants like Quantomatic [10, 28] and Globular [10] were 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. On the other hand, the framework developed in this thesis explicitly includes the rewrite rules.

To model the ZX-calculus using structured cospans, we need an appropriate adjunction $L\colon \mathsf{A} \rightleftarrows \mathsf{X} \colon R$. Determining the correct pieces to fill in requires some discussion. Before providing the details, we sketch the process. Let $\mathsf{A} := \mathsf{FinSet}$ be the topos of finite

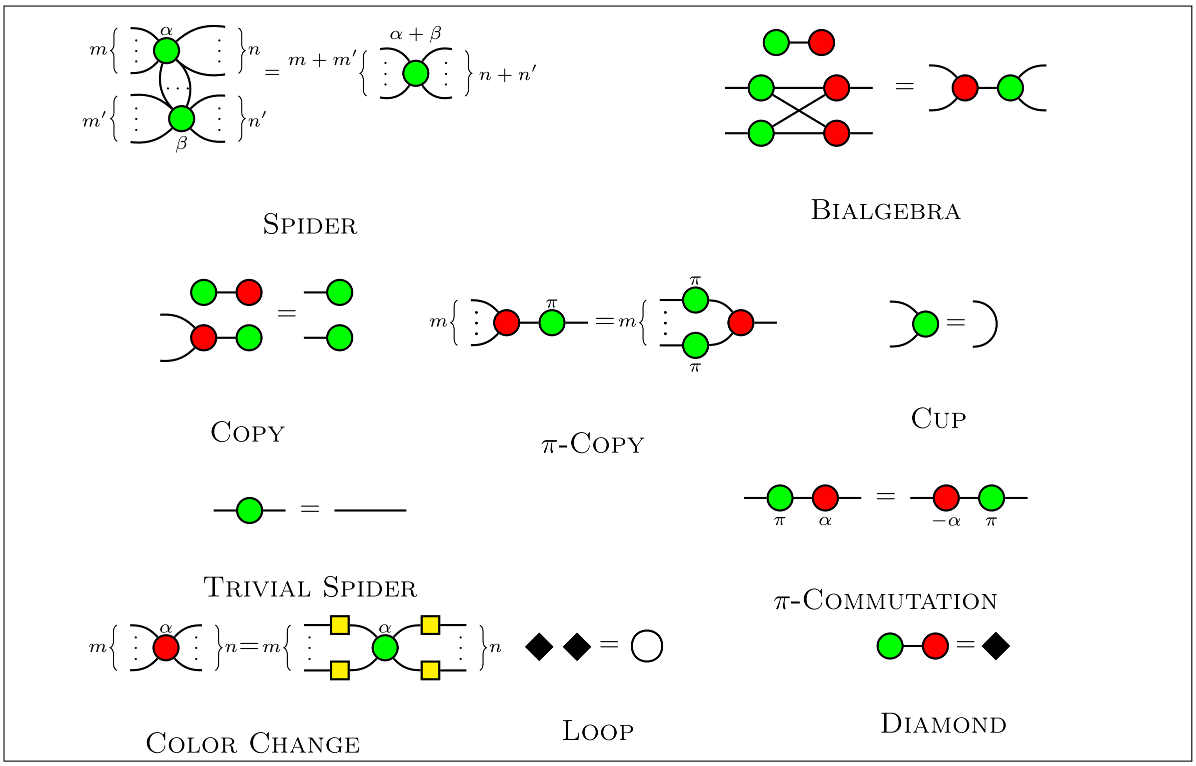
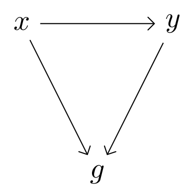


Figure 4.3: Relations in the category ZX

sets and functions. Let $\mathsf{X} := \mathsf{FinGraph} \downarrow \Gamma$ be the over-category where we chose a graph Γ to provide the objects of $\mathsf{X} := \mathsf{FinGraph}/\Gamma$ with the same type information as the ZX-diagrams. The functor L turns a finite set a into a certain discrete graph over Γ so that La can serve as inputs or outputs. To unpack what this all means, we start with the over-category.

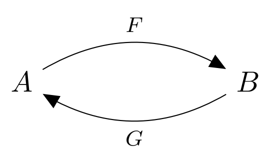
Definition 49. Let g be a graph. By a **graph over** g, we mean a graph morphism $x \to g$. A morphism between graphs over g is a graph morphism $x \to y$ such that



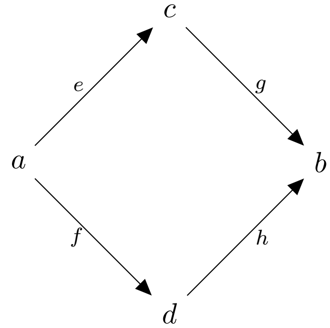
commutes.

One way to think of a graph over g is as a g-typed graph. Consider the following simple example.

Example 50. Let g be the graph



Let x be the graph



that lies over g via the map

$$a, b \mapsto A \qquad\qquad e, f \mapsto F$$

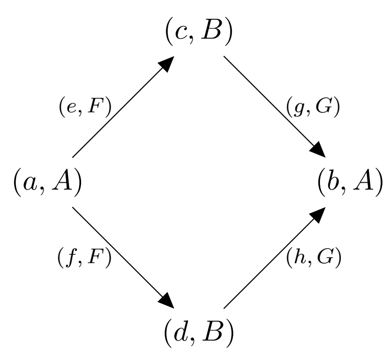
$$c, d \mapsto B \qquad\qquad g, h \mapsto G$$

If we think of the nodes and edges of g as types, then these types are transported to x along the fibers of this map. Thus x is a graph with the following type-assignment:

$$\begin{array}{lll} a : A & b : A & c : B \\ d : B & e : F & f : F \\ g : G & h : G & \end{array}$$

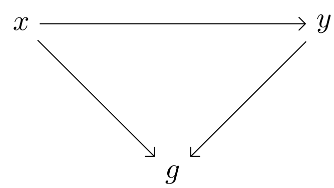
where ':' should be read 'is type'. Any graph over g can have two node types A, B and two edge types F, G. Edges can only go from an A-type node to a B-type node or vice versa. Edges cannot traverse nodes of the same type simply because there are no looped edges in g.

A compact way to draw a graph over g is to label its nodes and edges with their types. Thus, the over-graph $x \to g$ can be drawn as



One might recognize the class of graphs over g as something like a bipartite graph. The difference between graphs over g and bipartite graphs is that bipartite graphs are usually defined by graph theorists to satisfy *the property* that the nodes can be partitioned into two classes and the source and target of each edge must belong to different classes. On the other hand, graphs over g are graphs equipped with extra structure, namely the type information. This distinction does not appear in the graphs themselves, so we look at their morphisms.

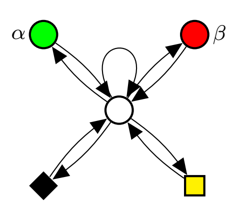
A morphism of graphs over g must respect the type information. So if $x \to g$ and $y \to g$ are graphs over g, then a morphism between them is a graph morphism $x \to y$ such that the diagram



commutes. Suppose that x is a single node typed A and y is a single node typed B. There is no morphism between them because the node in x must be sent to a node of type A. However, any two bipartite graphs with a single node and no edges are isomorphic. The moral of this example is by adding the type information, we added structure instead of imposing a property. We denote by $\mathsf{Graph} \downarrow g$ the category of graphs over g and their morphisms.

We exploit this method of defining 'typed graphs' to transform typical combinatorial graphs into ZX-diagrams. The types needed to make ZX-diagrams from graphs encoded into the graph Γ that we define now.

Definition 51. Let Γ be the graph



$\alpha, \beta \in [-\pi, \pi)$. (4.3)

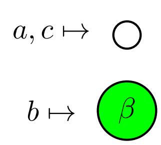
We have not drawn the entirety of Γ. In actuality, the green and red nodes run through $[-\pi, \pi)$ and each of them have a single arrow to and from the white node

Note that the graphs over Γ are completely determined by the function's behavior on the nodes. This is because there is at most one arrow between any two nodes. When comparing the Γ-types to the types appearing in the basic ZX-diagrams of Figure 4.2, there is a clear correlation except, perhaps, for the white node. To explain the white node, first observe that ZX-diagrams have dangling wires on either end. Dangling edges are not permitted in our definition of graphs, so the white node anchors them.

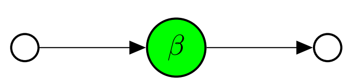
To draw graphs over Γ, we attach the type information to the nodes by rendering the nodes as red, greed, white, black, or yellow. This manner of drawing is more economical than drawing a graph and describing its map to Γ. For example, consider the graph

$$\bullet_a \longrightarrow \bullet_b \longrightarrow \bullet_c$$

with the map to Γ determined by



We draw this as



In our adjunction $L\colon \mathsf{A} \rightleftarrows \mathsf{X}\colon R$, we let X be $\mathsf{FinGraph} \downarrow \Gamma$. This is a topos by the fundamental theorem of topos theory, which we present in Theorem 113.

The most important objects in $\mathsf{FinGraph} \downarrow \Gamma$ are those corresponding to the basic ZX-diagrams. These are displaying in Figure 4.4. To choose a category A of interface types, we want to faithfully represent the fact that ZX-diagrams have a non-negative integer

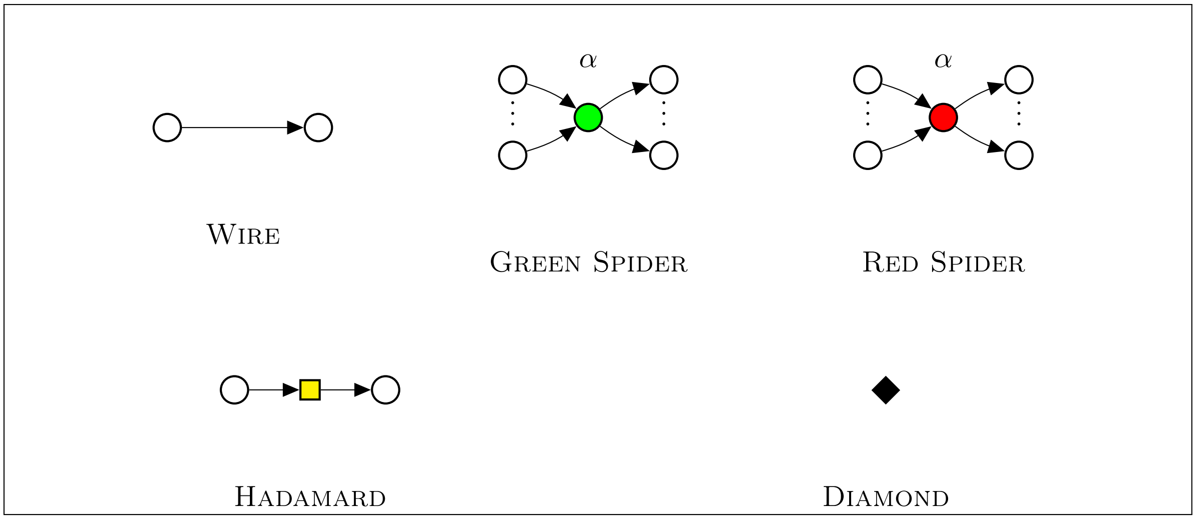


Figure 4.4: Basic ZX-diagrams as graphs over Γ

number of inputs and outputs. Therefore, we let A be the topos FinSet of finite sets and functions.

We still need to define L and R in the adjunction

$$\mathsf{FinSet} \underset{R}{\overset{L}{\rightleftarrows}} \mathsf{FinGraph} \downarrow \Gamma \qquad (L \dashv R)$$

Define

$$L\colon \mathsf{FinSet} \to \mathsf{FinGraph} \downarrow \Gamma$$

by letting La be the edgeless graph with node set a that is constant over the whites node in Γ. A function $f\colon a \to b$ of finite sets becomes of morphism Lf of graphs over Γ that simply reinterprets the action of f on elements of a set to white nodes in a graph. Define

$$R\colon \mathsf{FinGraph} \downarrow \Gamma \to \mathsf{FinSet}$$

by defining $R(x \to \Gamma)$ as the fiber in x of the white node. Given a morphism of graphs over Γ, R restricts it to the function on only the white nodes.

Lemma 52. The functor pair

$$\mathsf{FinSet} \underset{R}{\overset{L}{\rightleftarrows}} \mathsf{FinGraph} \downarrow \Gamma \qquad (L \dashv R)$$

forms an adjunction and L preserves pullbacks.

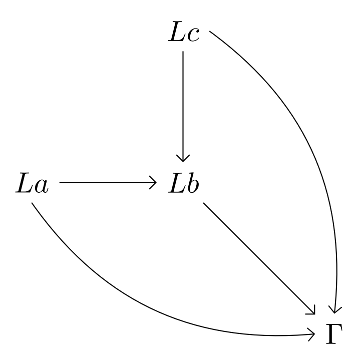
Proof. Observe that the composite RL is the identity functor. So the unit $\eta\colon a \to RLa$ is the identity which is natural in a. The counit $\varepsilon\colon LRx \to x$ is the inclusion of the white nodes of x into x. Given an arrow $f\colon x \to y$ in $\mathsf{FinGraph} \downarrow \Gamma$, the diagram

$$\begin{array}{ccc} LRx & \xrightarrow{\varepsilon_x} & x \\ {\scriptstyle LRf}\downarrow & & \downarrow{\scriptstyle f} \\ LRy & \xrightarrow[\varepsilon_y]{} & y \end{array}$$

commutes since LRf is a restriction of f. To show that L preserves pullbacks, take a cospan

$$a \to b \leftarrow c$$

in Set with pullback $a \times_b c$ and apply L to get the diagram



comprised of edgeless graphs La, Lb, and Lc that are constant over the white node in Γ. The pullback of this diagram is $La \times_{Lb} Lc \to \Gamma$ which is constant over the white node. This is isomorphic to $L(a \times_b c) \to \Gamma$ which is constant over the white node. ■

With our adjunction established, we can define structured cospans of graphs over Γ and therefore the symmetric monoidal double category of bold rewrites ${}_L\mathbb{B}\mathbf{old}\mathbb{R}\mathbf{ewrite}$ for the functor $L\colon \mathsf{FinSet} \to \mathsf{FinGraph} \downarrow \Gamma$ defined above. This double category has as objects the finite sets, as horizontal 1-arrows the structured cospans of graphs over Γ, as vertical 1-arrows the spans of finite sets with invertible legs, and as squares all possible bold rewrites of structured cospans. Clearly, ${}_L\mathbb{B}\mathbf{old}\mathbb{R}\mathbf{ewrite}$ is far bigger than the ZX-calculus because it contains graphs over Γ with no corresponding ZX-diagram. This does not mean, however, that ${}_L\mathbb{B}\mathbf{old}\mathbb{R}\mathbf{ewrite}$ serves no purpose. It plays the role of an ambient space in which we chisel out a sub-double category that *does* correspond to the ZX-calculus.

To begin the process of constructing this sub-double category of ${}_L\mathbb{B}\mathbf{old}\mathbb{R}\mathbf{ewrite}$, we identify structured cospans to capture the basic ZX-diagrams and identify bold rewrites of structured cospans for the basic ZX-relations. We also include some additional structured cospans to give the desired structure. Figure 4.5 depicts the basic ZX-diagrams as structured cospans.

Translating the relations between ZX-diagrams to structured cospans is quite straightforward. We provide several examples.

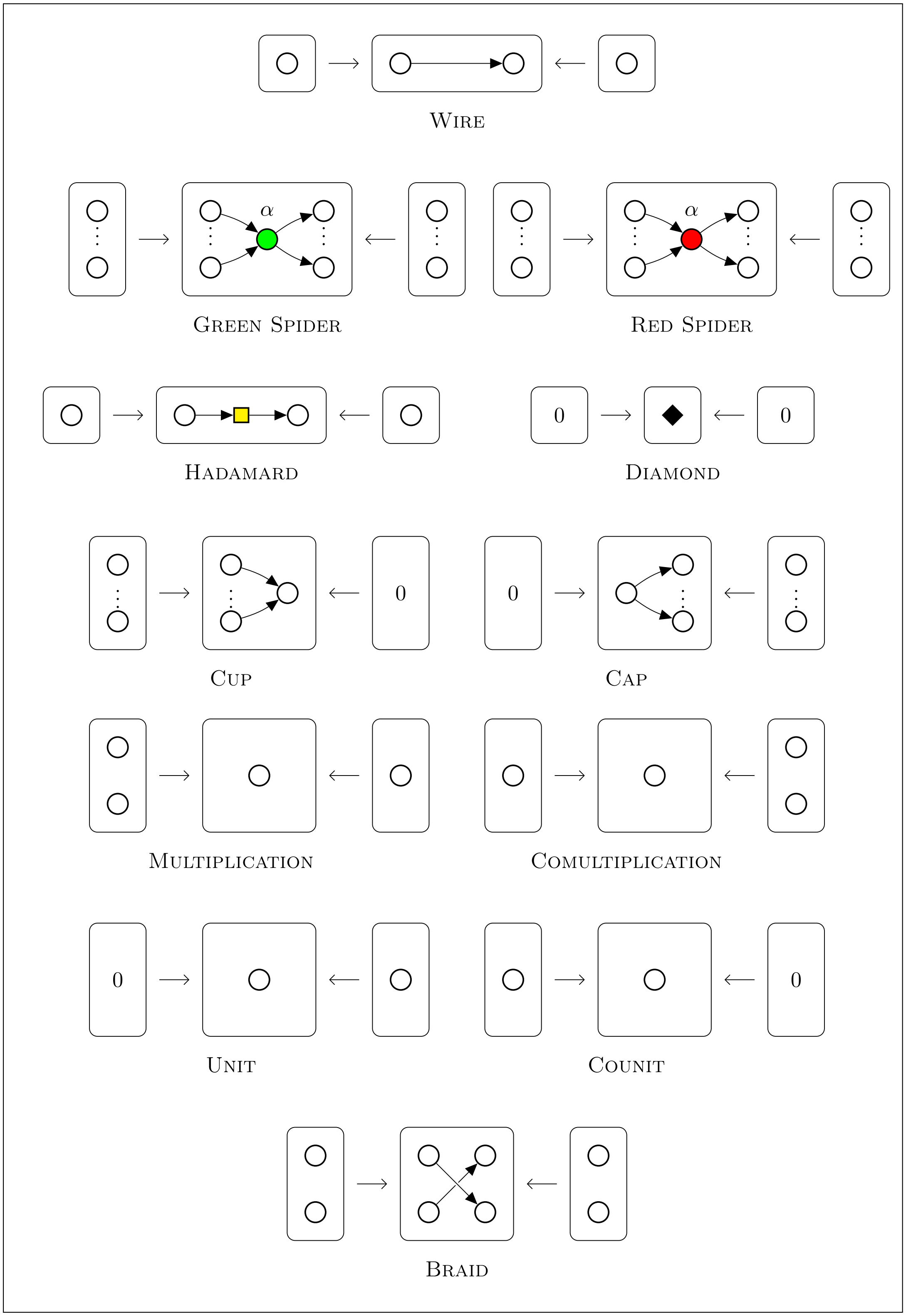
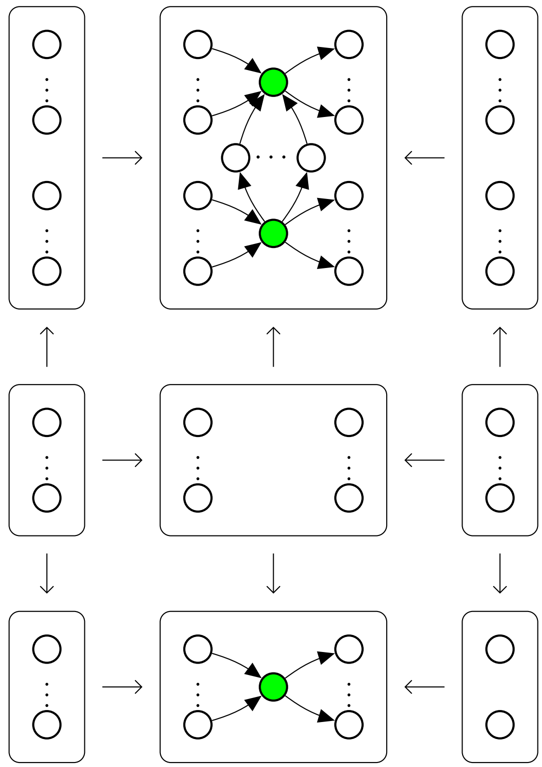
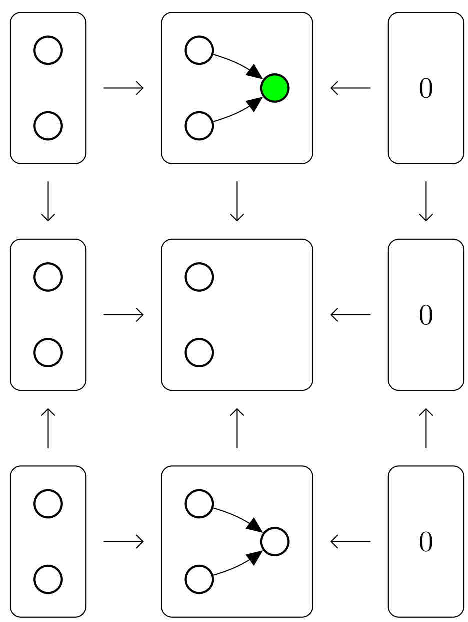


Figure 4.5: Basic ZX-diagrams as structured cospans

Spider Relations

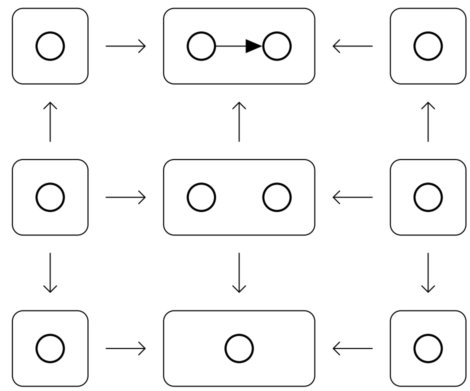


Cup Relation



The remaining relations from Figure 4.3 can be translated into spans of structured cospans in this way. We include an additional rewrite

Wire Relation



to account for the fact that the wire structured cospan in Figure 4.5 is, a priori, not an identity. This wire relation ensures that the wire structured cospan is an identity.

We are now ready to define the double category $\mathbb{ZX}$.

Definition 53. Let

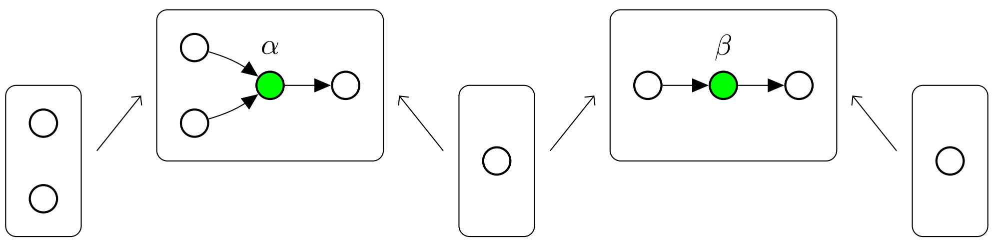
$$\mathsf{FinSet} \underset{R}{\overset{L}{\rightleftarrows}} \mathsf{FinGraph} \downarrow \Gamma \qquad (L \dashv R)$$

be the adjunction defined so that L assigns a set to the discrete graph that is constant over the white node on that set and where R returns the set of white nodes of a graph over Γ. Define $\mathbb{ZX}$ to be the isofibrant symmetric monoidal sub-double of ${}_L\mathbb{B}$**old**$\mathbb{R}$**ewrite** generated by the basic structured cospans and the basic rewrites for ZX-diagrams.

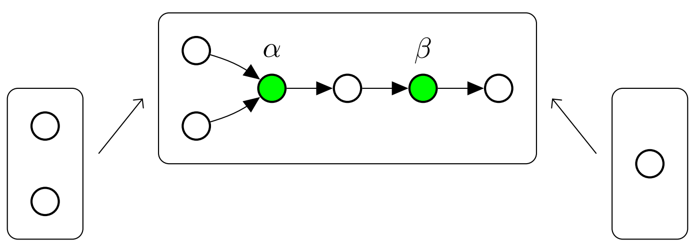
In this definition, using ${}_L\mathbb{B}$**old**$\mathbb{R}$**ewrite** as an ambient double category ensures that generating $\mathbb{ZX}$ is well-defined. All of the required structure and properties are in place and ${}_L\mathbb{B}$**old**$\mathbb{R}$**ewrite** bounds the generation. Now, because $\mathbb{ZX}$ is an isofibrant symmetric monoidal category—true by construction—we use Shulman's work [58] to provide the symmetric monoidal bicategory **ZX**.

Proposition 54. There is a symmetric monoidal bicategory **ZX** whose objects are finite sets, 1-arrows are generated by the basic L-structured cospans in Figure 4.5, and 2-arrows are bold rewrites generated by the basic rewrites of ZX-diagrams.

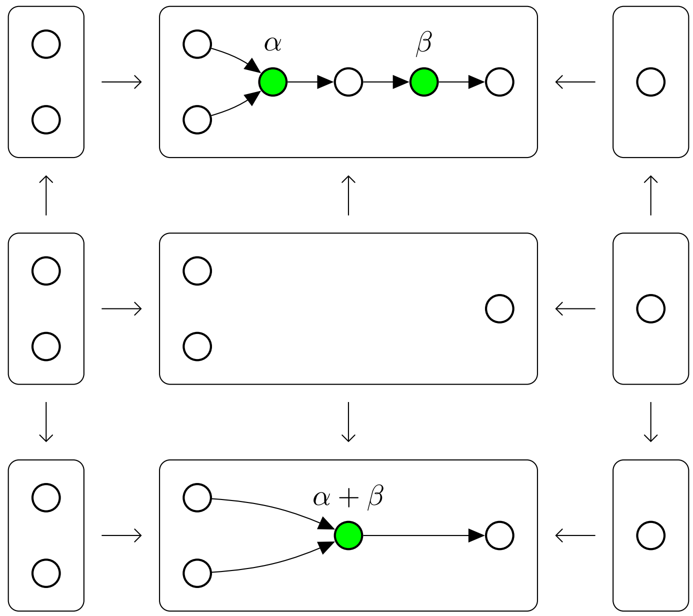
The ZX-diagrams appear in $\mathbb{ZX}$ as horizontal 1-arrows and in **ZX** as 1-arrows. Composing the ZX-diagrams works as it does in the original ZX-calculus; pushout formalizes the gluing of dangling edges. Indeed, composing basic diagrams provides 'compound' diagrams. For example, composing



gives



To this, we can apply the Spider Relation



Because the vertical 1-arrows are identities, this 2-arrow exists in both $\mathbb{ZX}$ and **ZX**. The spider relation simplifies the ZX-diagram in the top row to that in the bottom row.

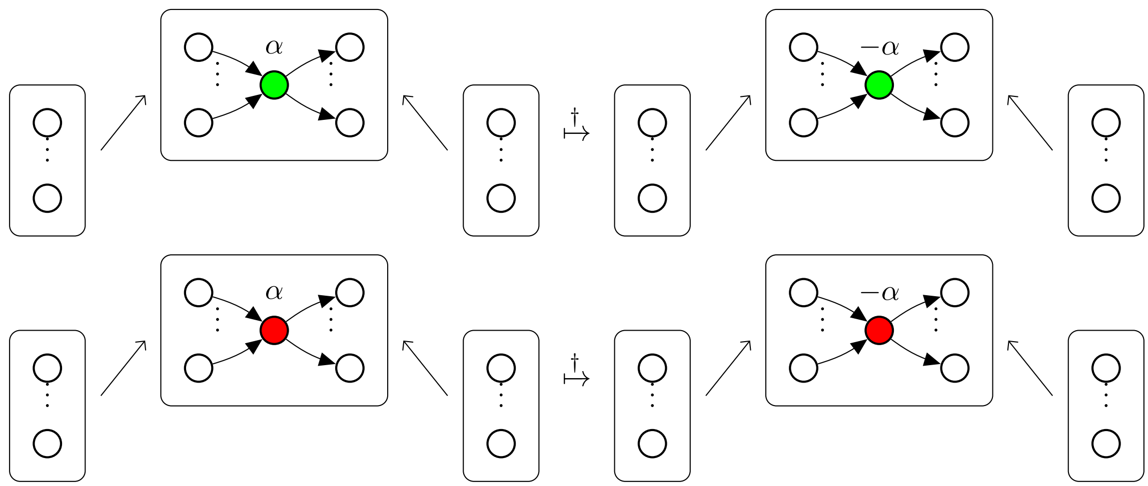
Theorem 55. The bicategory **ZX** is a bicategory of relations.

Proof. Because **ZX** includes the structure maps to give every object a Frobenius monoid structure, every requirement descends from ambient category $_L$**BoldRewrite** being a bicategory of relations (see Theorem 48). ■

This bicategory extends the original category ZX. To show this, we will show the 'decategorification' of **ZX** is ZX. The process of decategorification essentially turns an n-category into an $n-1$-category. For us, we turn a (weak) 2-category into a 1-category by identifying any 1-arrows connected by a zig-zag of 2-arrows.

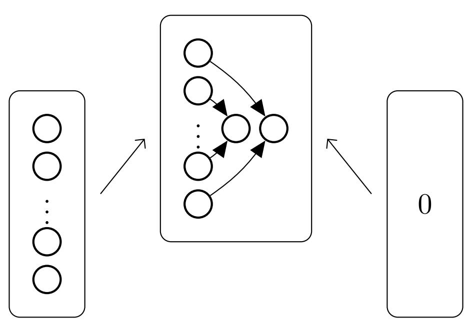
Definition 56. Define decat(**ZX**) to be the category whose objects are those of **ZX** and whose arrows the 1-arrows of **ZX** modulo the equivalence relation $\sim$ generated by $f \sim g$ if and only if there is a 2-arrow $f \Rightarrow g$ in $\mathbf{ZX}$.

Theorem 57. The category decat($\mathbf{ZX}$) is dagger compact via the identity on objects functor described by

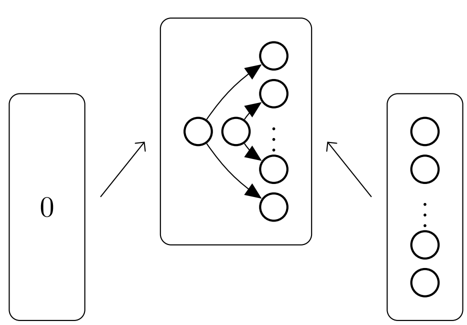


as well as by identity on the wire, Hadamard, and diamond morphisms.

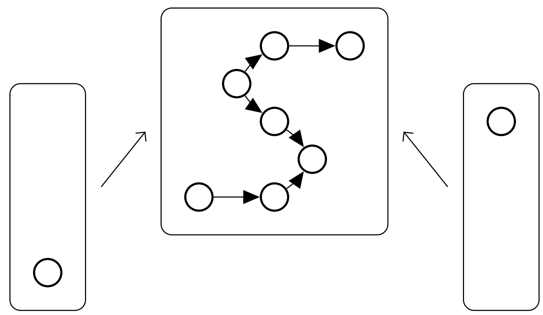
Proof. Compact closedness follows from the self duality of objects via the evaluation



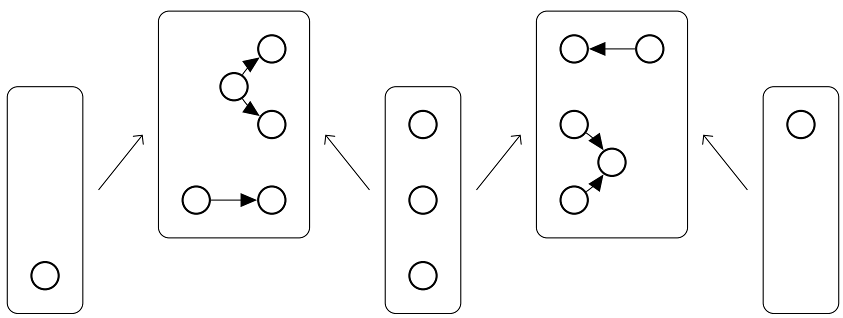
and coevaluation arrows



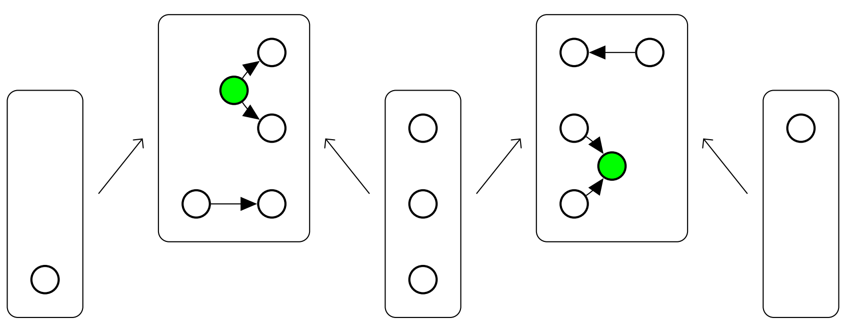
obtained by applying the braiding maps to the disjoint union of cups and caps. Moreover, we can derive the snake equation as follows. Decompose the arrow



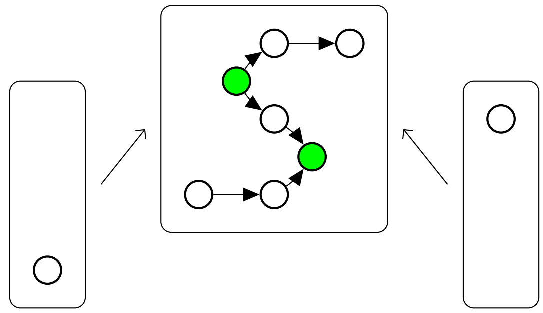
into



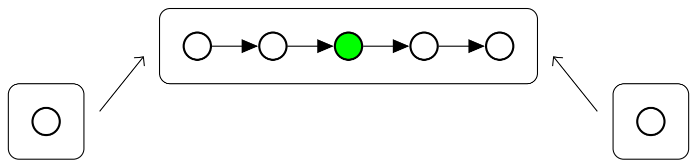
which by the cup relation, illustrated in Figure 4.3, equals



This can be composed to get



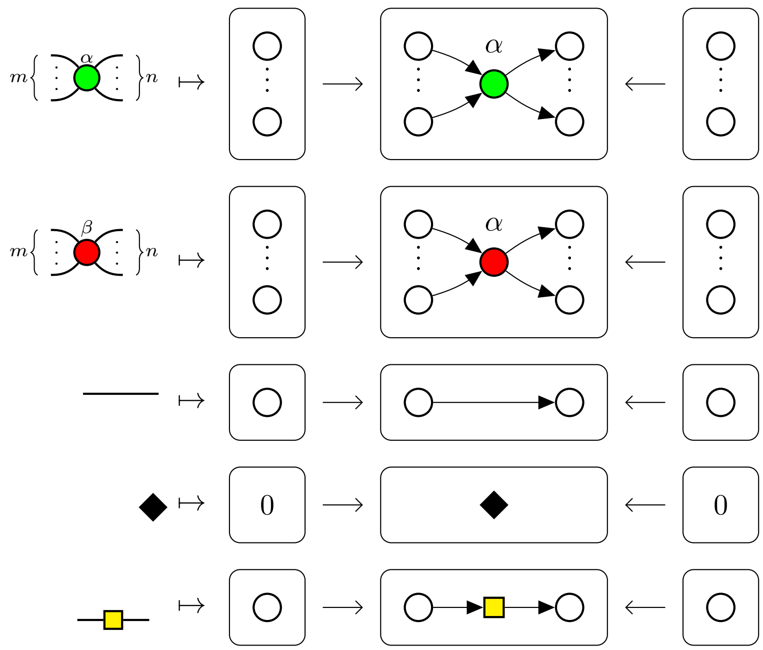
which equals



because of the spider relation. Finally, this equals the identity because of the trivial spider and wire relations. Showing that the described functor is a dagger functor is a matter of checking some easy to verify details. ■

We now show that $\mathbb{ZX}$ is an extension of ZX in the sense that the category decat(**ZX**) obtained from $\mathbb{ZX}$ is equivalent to ZX.

Theorem 58. The identity on objects, dagger compact functor $E\colon \mathsf{ZX} \to \mathrm{decat}(\mathbf{ZX})$ 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 decat(**ZX**) are all in the image of E.

Faithfulness is more involved. Let f, g be ZX-morphisms. Let $\widetilde{Ef}$, $\widetilde{Eg}$ be the representatives of Ef, Eg obtained by directly translating the graphical representation of f, g to structured cospans of graphs of Γ. For faithfulness, it suffices to show that the existence of a 2-arrow $\widetilde{Ef} \Rightarrow \widetilde{Eg}$ in **ZX** implies that $f = g$.

Observe that any 2-arrow α in **ZX** can be written, not necessarily uniquely, as sequence $\alpha_1 \Box \cdots \Box \alpha_n$ 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-arrow, 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 '+'. When we have a 2-arrow $\alpha_1 + \alpha_2 \colon Ef \Rightarrow Eg$ where α_1 is a basic 2-arrow 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. ■

Chapter 5

Decomposing systems

The idea of decomposing a whole into parts has long been useful. It exists across so many human disciplines, be it academic, artistic, or artisanal. A biologist decomposes life-forms into genuses and species. A literary critic decomposed a play into acts and scenes. A sommelier decomposes a wine into color, viscosity, aroma, and taste. In this chapter, as do the biologist, critic, and sommelier, we decompose. Though for us, we decompose a closed system into open sub-systems.

This may seem to conflict with the aim of this thesis, which is to advance a theory of open systems. However, we still recognize the value of closed systems. We just believe that our ideas on open systems are useful for closed systems.

As mathematicians, we must bring rigor to our decomposition. In this chapter, we do just that. We start by formalizing closed systems as structured cospans with an empty interface $0 \to x \leftarrow 0$. Then, using the fine rewriting paradigm from Chapter 3, we place structured cospans into the double category ${}_L\mathbb{F}\mathbf{ine}\mathbb{R}\mathbf{ewrite}$ as horizontal 1-arrows. To decompose a closed system

$$L0 \to x \leftarrow L0$$

is to write an arrow as a composite of arrows

$$L0 \to x_1 \leftarrow La_1 \to x_2 \leftarrow La_2 \cdots La_{n-1} \to x_n \leftarrow L0$$

We use such decompositions to prove our main result which states that two structured cospans

$$L0 \to x \leftarrow L0 \quad \text{and} \quad L0 \to x' \leftarrow L0$$

are equivalent precisely when there is a square between them. We interpret this result in three ways.

1. It shows that the rewriting relation for a closed system is functorial and is characterized using squares in a double category.

2. A closed system decomposes into open systems, and simplifying each open system simplifies the composite closed system.

3. Open systems provide a local perspective on the closed perspective via this decomposition.

There are two main thrusts to this proof. The first generalizes a classification of formal graph grammars given by Ehrig, et. al. [34]. This is Theorem 69. Gadducci and Heckel proved this in the case of graphs [35], but our result generalizes this to structured cospans. Our proof mirrors theirs.

5.1 Expressiveness of underlying discrete grammars

As mentioned above, we want to decompose closed systems into open systems. We did not yet mention which open systems are available to use. This depends on context. That is, whatever type of system one has, there is an appropriate grammar stipulated by a theory that describes that system. To illustrate, for an electrical system, a corresponding grammar would have rules for adding resistors in series, or adding the reciprocal of resistors in parallel. Therefore, our starting data is a grammar (X, P)—a topos X and a set of fine rewrite rules $P \coloneqq \{\ell_j \leftarrow k_j \rightarrow r_j\}$—plus a closed system x in X. Eventually entering the story is a topos A of input types and an adjunction between A and X. For now, however, we focus on the set of rewrite rules P.

We can prove the main result of this section, Theorem 69, by controlling the form of the rewrite rules. In particular, we want the intermediary of the rules, the k_j's, to be 'discrete'. In what follows, we discuss what we mean by 'discrete' and show that the grammar obtained by discretizing (X, P) is just as expressive as (X, P), by which we mean that the induced rewriting relations are equal. This result generalizes a characterization of discrete *graph grammars* given by Ehrig, et. al. [34, Prop. 3.3].

Our concept of 'discreteness' is borrowed from the flat modality on a local topos. However, we avoid the lengthy detour required to discuss the 'flat modality' and a 'local topos'. The background does not add to our story, so we point curious readers elsewhere [38, Ch. C3.6]. By avoiding that detour, we instead require the concept of a comonad, which we present in Definition 96.

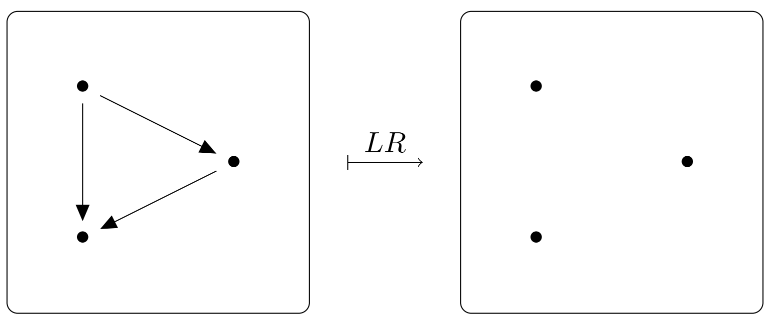
To start our discussion on discreteness, we define a 'discrete comonad'. The definition is straightforward enough, but its purpose may seem alien at first. After the definition, we explain its role in rewriting structured cospans.

Definition 59 (Discrete comonad)**.** A comonad on a topos is called **discrete** if its counit is monic. We use $\flat$ to denote a discrete comonad.

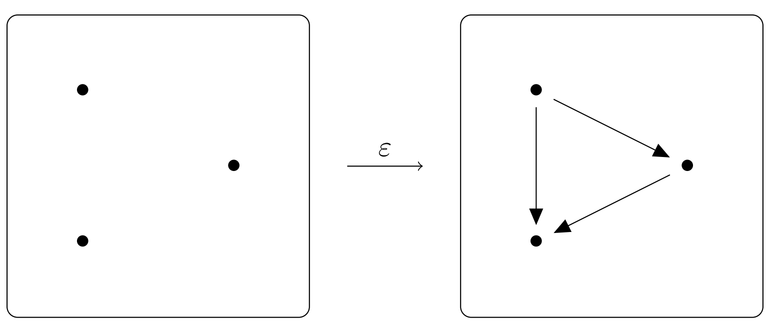
Secretly, we have been working with a discrete comonad all along. The adjunction

$$\mathsf{Set} \underset{R}{\overset{L}{\rightleftarrows}} \mathsf{RGraph} \quad (L \dashv R)$$

induces the comonad LR on RGraph. Applying LR to a graph x returns the edgeless graph underlying x, hence the term 'discrete'. For example



The counit $\varepsilon_x \colon LRx \to x$ of the comonad LR includes the underlying edgeless graph LRx into the original graph x. For example



Abstractly, this inclusion is why we ask for the counit to be monic. The property we capture with a discrete comonad comes from the systems interpretation of the adjunctions

$$\mathsf{A} \underset{R}{\overset{L}{\rightleftarrows}} \mathsf{X} \quad (L \dashv R)$$

between topoi. That is, R takes a system x, identifies the largest sub-system that can serve as an interface and turns that sub-system into an interface type Rx. Then L takes that interface type and turns it back into a system LRx. This process effectively strips away every part of a system leaving only those parts that can connect to the outside world. That means LRx is a part of x or, in the parlance of category theory, LRx is a subobject of x. Hence, we ask for a monic counit.

How do we plan to use discrete comonads? We use them to control the form of our grammars. In general, a rewrite rule has form

$$\ell \leftarrow k \rightarrow r$$

where there are no restrictions on what k can be. However, recall that k identifies the part of ℓ that is fixed throughout the rewrite. It does not direct how the rewrite is performed. Therefore, we can deform it a bit without changing the outcome of the applying the rewrite. In particular, we can discretize it by replacing k with $\flat k$. And because $\flat$ has a monic counit, we can insert $\flat k$ right into the middle of the fine rewrite rule.

Definition 60 (Discrete grammar)**.** Given a grammar (X, P), define the set $P_\flat$ as consisting of the rules

$$\ell \leftarrow k \leftarrow \flat k \rightarrow k \rightarrow r$$

for each rule $\ell \leftarrow k \rightarrow r$ in P. We call $(X, P_\flat)$ the **discrete grammar** underlying (X, P).

Discrete grammars are easier to work with than arbitrary grammars. So when given an opportunity to work with a discrete grammar instead of a non-discrete grammar, we should take it. Theorem 69 gives a sufficient condition that allows us to swap (X, P) for $(\mathsf{X}, P_\flat)$ without consequence. To prove this, however, we borrow from lattice theory which requires that we make a brief turn to fill in some required background.

Definition 61 (Lattice)**.** A lattice is a poset $(S, \leq)$ equipped with all finite joins $\bigvee$ and all finite meets $\bigwedge$. It follows that there is a minimal element and maximal element, realized as the empty meet and join respectively, which we denote by 0 and 1.

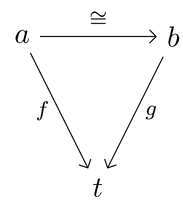
Joins and meets are also known as suprema and infima. We are using the definition of a lattice common in the category theory literature. This leaves out objects that some mathematicians might consider lattices. Below we give one counter-example and several examples of lattices, the last one being the most relevant.

Example 62 (Integer Lattice)**.** The integers with the usual ordering $\leq$ do not form a lattice because there is no minimal or maximal element.

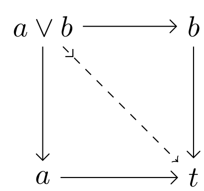
Example 63 (Lattice of power sets)**.** For any set S, its powerset $\mathcal{P}S$ is a poset via subset inclusion. The powerset becomes a lattice by taking join to be union $a \vee b \coloneqq a \cup b$, and meet to be intersection $a \wedge b \coloneqq a \cap b$. In general, union and intersection are defined over arbitrary sets, thus realizing arbitrary joins $\bigvee a_\alpha$ and arbitrary meets $\bigwedge a_\alpha$.

Those few examples provide intuition about lattices, but the next example is the most important lattice for us. It is the mechanism by which the power set is generalized into topos theory. It is called the subobject lattice.

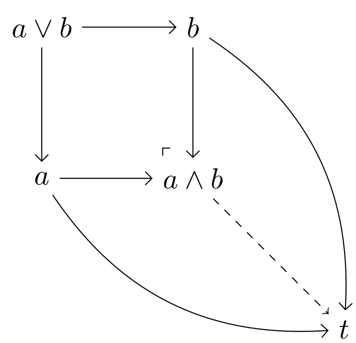
Example 64 (Subobject lattice)**.** Let T be a topos and t be an object. There is a lattice $\mathrm{Sub}(t)$ called the subobject lattice of t. The elements of $\mathrm{Sub}(t)$ are called subobjects. They are isomorphism classes of monomorphisms into t. Here, two monomorphisms f, g into t are isomorphic if there is a commuting diagram



The order on $\mathrm{Sub}(t)$ is given by $f \leq g$ if f factors through g, meaning there is an arrow $h \colon a \to b$ such that $f = gh$. Note that h is necessarily monic. The meet operation in $\mathrm{Sub}(t)$ is given by pullback



and join is given by pushout over the meet



We use subobject lattices to characterize which grammars are as expressive as their underlying discrete grammars. To do this, we require subobject lattices with arbitrary meets. The powerset lattice mentioned above has this property, but when do subobject lattices have this property? Here are several sufficient conditions, starting with a well-known result coming from the domain of order theory.

Proposition 65. Any lattice that has all joins also has all meets.

Proof. Consider a subset S of a lattice. Define the meet of S to be the join of the set of all lower bounds of S. ■

Proposition 66. Consider a topos T and object t. The subobject lattice $\mathrm{Sub}(t)$ has arbitrary meets when the over category $T \downarrow t$ has all products.

Proof. Because $T \downarrow t$ is a topos, it has equalizers. Thus giving it all products ensures the existence of all limits, hence meets. ■

Corollary 67. Consider a topos T and object t. The subobject lattice $\mathrm{Sub}(t)$ has arbitrary meets when the over category $T \downarrow t$ has all coproducts.

Proof. Combine Propositions 65 and 66. ■

Corollary 68. Consider a presheaf category $\mathsf{Set}^{\mathsf{C}^{\mathrm{op}}}$ on a small category C. For any presheaf x, $\mathrm{Sub}(x)$ has all meets.

Proof. The category $\mathsf{Set}^{\mathsf{C}^{\mathrm{op}}} \downarrow x$ of presheaves over x is again a presheaf category by Theorem 114 so has all products. ■

At last, we combine the discrete comonad, the discrete grammar, and the complete subobject lattice into a result on the expressiveness on discrete grammars.

Theorem 69. Let T be a topos and $\flat\colon \mathsf{T} \to \mathsf{T}$ be a discrete comonad. Let (T, P) be a grammar such that for every rule $\ell \leftarrow k \rightarrow r$ in P, the subobject lattice $\mathrm{Sub}(k)$ has all meets. Then the rewriting relation for (T, P) equals the rewriting relation for the underlying discrete grammar $(\mathsf{T}, P_\flat)$.

Proof. Suppose that (T, P) induces $g \rightsquigarrow h$. That means there exists a rule $\ell \leftarrow k \rightarrow r$ in P and a derivation

$$\begin{array}{ccccc} \ell & \longleftarrow & k & \longrightarrow & r \\ \big\downarrow & \lrcorner & \big\downarrow & \llcorner & \big\downarrow \\ g & \longleftarrow & d & \longrightarrow & h \end{array} \tag{5.1}$$

we can achieve that same derivation using rules in $P_\flat$. This requires we build a pushout complement w of the diagram

$$\begin{array}{ccc} k & \xleftarrow{\varepsilon} & \flat k \\ \big\downarrow & & \\ d & & \end{array}$$

Define

$$w := \bigwedge \{z \colon z \vee k = d\} \vee \flat k,$$

This comes with inclusions $\flat k \to w$ and $w \to d$. This w exists because $\mathrm{Sub}(k)$ has all meets. Note that $w \vee k = d$ and $w \wedge k = \flat k$ which means that

$$\begin{array}{ccc} k & \longleftarrow & \flat k \\ \big\downarrow & \lrcorner & \big\downarrow \\ d & \longleftarrow & w \end{array}$$

is a pushout. It follows that there is a derivation

$$\begin{array}{ccccccccc} \ell & \longleftarrow & k & \longleftarrow & \flat k & \longrightarrow & k & \longrightarrow & r \\ \big\downarrow & & \big\downarrow & & \big\downarrow & & \big\downarrow & & \big\downarrow \\ g & \longleftarrow & d & \longleftarrow & w & \longrightarrow & d & \longrightarrow & h \end{array} \tag{5.2}$$

with respect to $P_\flat$ because, the top row is a rule in $P_\flat$. Therefore, $g \rightsquigarrow h$ via P in Diagram (5.1) implies that $g \rightsquigarrow^* h$ via $P_\flat$ as shown in Diagram (5.2).

For the other direction, suppose $g \rightsquigarrow h$ via $P_\flat$, giving a derivation

$$\begin{array}{ccccc} \ell & \longleftarrow & \flat k & \longrightarrow & r \\ {\scriptstyle m}\big\downarrow & & {\scriptstyle \theta}\big\downarrow & & \big\downarrow{\scriptstyle m'} \\ g & \underset{\psi}{\longleftarrow} & d & \longrightarrow & h \end{array} \tag{5.3}$$

By construction of $P_\flat$, the rule $\ell \leftarrow \flat k \rightarrow r$ in $P_\flat$ was induced from a rule

$$\ell \xleftarrow{\tau} k \rightarrow r$$

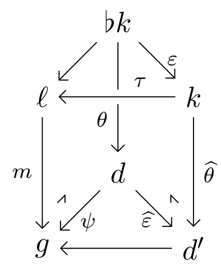
in P, meaning that the map $\flat k \rightarrow \ell$ factors through τ. Next, define d' to be the pushout of the diagram

$$\begin{array}{ccc} \flat k & \xrightarrow{\varepsilon} & k \\ {\scriptstyle \theta}\big\downarrow & & \big\downarrow{\scriptstyle \widehat{\theta}} \\ d & \underset{\widehat{\varepsilon}}{\longrightarrow} & d' \end{array}$$

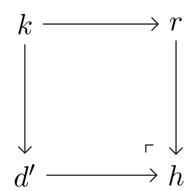
By invoking the universal property of this pushout with the maps

$$\psi \colon d \rightarrow g \quad \text{and} \quad m\tau \colon k \rightarrow \ell \rightarrow g,$$

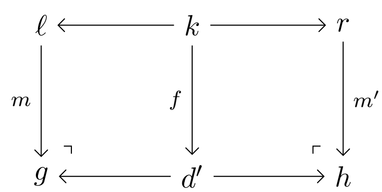
we get a canonical map $d' \rightarrow g$ that we can fit into a commuting diagram



whose back faces are pushouts. Using a standard diagram chasing argument, we can show that the front face is also a pushout. Similarly, the square



is a pushout. Sticking these two pushouts together



shows that $g \rightsquigarrow h$ arises from P.

Because the relation $\rightsquigarrow$ is the same for P and $P_\flat$, it follows that $\rightsquigarrow^*$ is also the same as claimed. ■

5.2 Rewriting structured cospans

Equipped with knowledge about when grammars and their underlying discrete grammars generate the same rewriting relation, we continue towards goal of decomposing closed systems. First, we revisit Section 2.2 to get some facts about grammars. We then obtain the language associated to a grammar in a functorial way. Finally, we show how to decompose into open subsystems a given system equipped with a grammar.

Recall the category Gram. The objects of Gram are pairs (T, P) where T is a topos and P is a set of rewrite rules in T. The arrows $(\mathsf{T}, P) \to (\mathsf{T}', P')$ of Gram are rule-preserving functors $\mathsf{T} \to \mathsf{T}'$. Our interest now lies in the full subcategory of structured cospan grammars $\mathsf{StrCspGram}$ whose objects are the grammars of form $({}_L\mathsf{StrCsp}, P)$ where P consists of fine rewrites of structured cospans, meaning they have the form

$$\begin{array}{ccccc} La & \longrightarrow & x & \longleftarrow & La' \\ \cong\uparrow & & \uparrow & & \uparrow\cong \\ Lb & \longrightarrow & y & \longleftarrow & Lb' \\ \cong\downarrow & & \downarrow & & \downarrow\cong \\ Lc & \longrightarrow & z & \longleftarrow & Lc' \end{array}$$

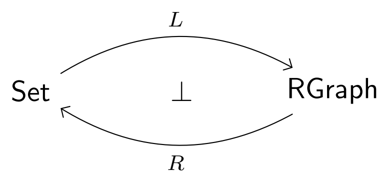
and the left adjoint L has a monic counit.

It is on this category $\mathsf{StrCspGram}$ that we define a functor encoding the rewrite relation to each grammar. We denote this functor

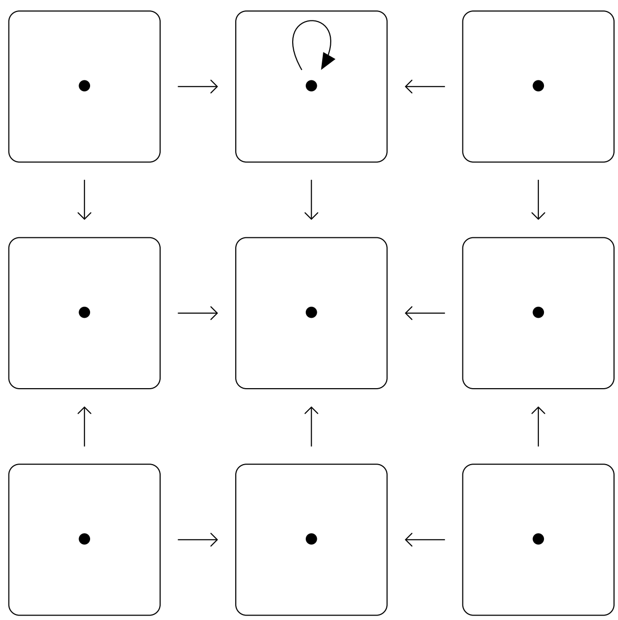
$$\mathrm{Lang} \colon \mathsf{StrCspGram} \to \mathsf{DblCat}$$

where Lang is short for 'language'. This is an appropriate term as this functor provides *(i)* the terms formed by connecting together open systems (instead of, in linguistics, concatenating units of syntax) and *(ii)* the rules governing how to interchange open systems (instead of parts of speech). To help visualize this, we sketch a simple example.

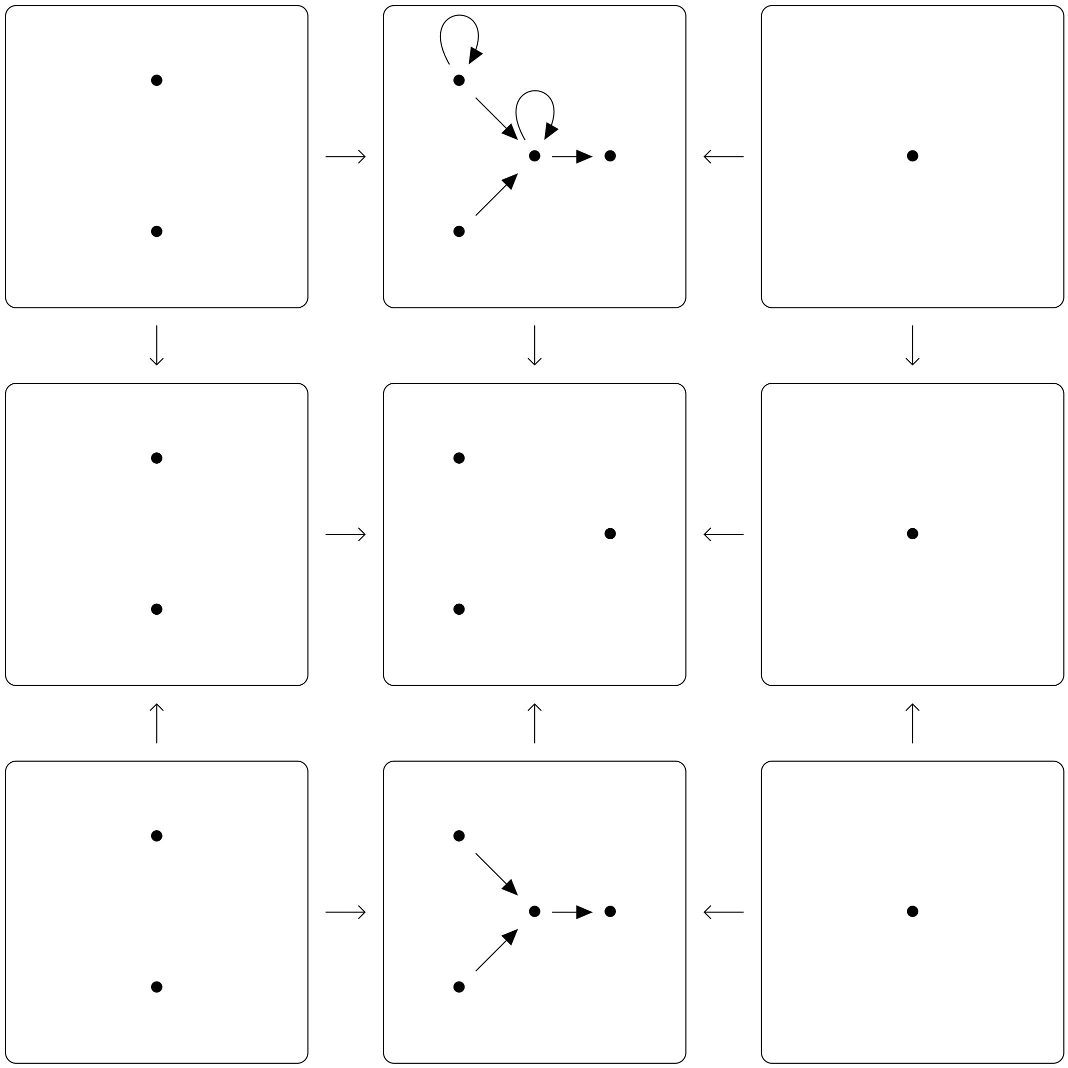
Example 70. Start with the, by now familiar, adjunction



For this L, ${}_L\mathsf{StrCsp}$ is the category of open graphs. Make a grammar from ${}_L\mathsf{StrCsp}$ by defining a P to have the single rule



The language associated to this grammar consists of all open graphs. The rewrite relation says $g \rightsquigarrow^* h$ if we obtain h be removing loops from g. We illustrate this with the following square in the double category $\mathrm{Lang}({}_L\mathsf{StrCsp}, P)$.



To actually construct Lang, we use functors $D\colon \mathsf{StrCspGram} \to \mathsf{StrCspGram}$ and $S\colon \mathsf{StrCspGram} \to \mathsf{DblCat}$. Roughly, D sends a grammar $({}_L\mathsf{StrCsp}, P)$ to all of the rewrite rules derived from P and S generates a double category on the squares obtained from the rewrite rules of a grammar $({}_L\mathsf{StrCsp}, P)$. In this way, we get the language of a grammar as a double category where the squares are the rewrite rules. The next lemma defines D and gives some of its properties.

Lemma 71. There is an idempotent functor $D\colon \mathsf{StrCspGram} \to \mathsf{StrCspGram}$ defined as follows. On objects define $D({}_L\mathsf{StrCsp}, P)$ to be the grammar $({}_L\mathsf{StrCsp}, P_D)$, where P_D consists of all rules $g \leftarrow h \rightarrow d$ witnessing the relation $g \rightsquigarrow h$ with respect to $({}_L\mathsf{StrCsp}, P)$. On arrows, define $DF\colon D({}_L\mathsf{StrCsp}, P) \rightarrow D({}_{L'}\mathsf{StrCsp}, Q)$ to be F. Moreover, the identity on $\mathsf{StrCspGram}$ is a subfunctor of D.

Proof. That $D({}_L\mathsf{StrCsp}, P)$ actually gives a grammar follows from the fact that pushouts respect monics in a topos [42, Lem. 12].

To show that D is idempotent, we show that for any grammar $({}_L\mathsf{StrCsp}, P)$, we have $D({}_L\mathsf{StrCsp}, P) = DD({}_L\mathsf{StrCsp}, P)$. Rules in $DD({}_L\mathsf{StrCsp}, P)$ appear in the bottom row of a double pushout diagram whose top row is a rule in $D({}_L\mathsf{StrCsp}, P)$, which in turn is the bottom row of a double pushout diagram whose top row is in $({}_L\mathsf{StrCsp}, P)$. Thus, a rule in $DD({}_L\mathsf{StrCsp}, P)$ is the bottom row of a double pushout diagram whose top row is in $({}_L\mathsf{StrCsp}, P)$. See Figure 5.1.

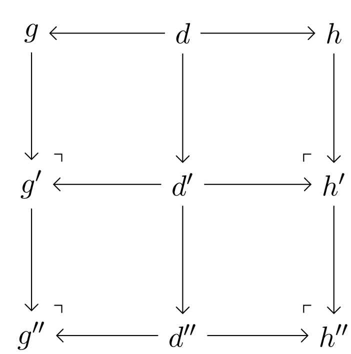


Figure 5.1: Stacked double pushout diagrams

The identity is a subfunctor of D because $\ell \rightsquigarrow r$ for any production $\ell \leftarrow k \rightarrow r$ in

$({}_L\mathsf{StrCsp}, P)$ via a triple of identity arrows. Hence there is a monomorphism

$$({}_L\mathsf{StrCsp}, P) \to D({}_L\mathsf{StrCsp}, P)$$

induced from the identity functor on ${}_L\mathsf{StrCsp}$. ■

In this lemma, we have created a functor D that sends a grammar to a new grammar consisting of all derived rules. That D is idempotent means that all rules derived from P can be derived directly; multiple applications of D are unnecessary. That the identity is a subfunctor of D means that set of the derived rules P_D contains the set of initial rules P.

The next stage in defining Lang is to define $S\colon \mathsf{StrCspGram} \to \mathsf{DblCat}$. On objects, let $S({}_L\mathsf{StrCsp}, P)$ be the sub-double category of ${}_L\mathbb{S}\mathbf{trCsp}$ generated by the rules in P considered as squares. On arrows, S sends

$$F\colon ({}_L\mathsf{StrCsp}, P) \to ({}_{L'}\mathsf{StrCsp}, P')$$

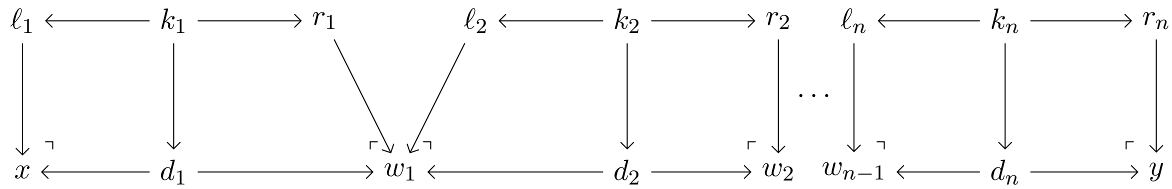
to the double functor defined that extends the mapping between the generators of $S({}_L\mathsf{StrCsp}, P)$ and $S({}_{L'}\mathsf{StrCsp}, P')$. This preserves composition because F preserves pullbacks and pushouts.

Definition 72. (Language of a grammar) The **language functor** is defined to be Lang $:=$ SD.

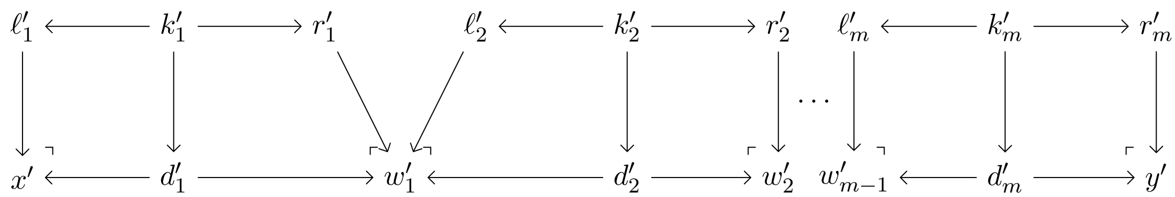
To witness the rewriting relation on a closed system as a square in a double category, we require this next lemma that formalizes the analogy between rewriting the disjoint union of systems and tensoring squares.

Lemma 73. If $x \rightsquigarrow^* y$ and $x' \rightsquigarrow^* y'$, then $x + x' \rightsquigarrow^* y + y'$

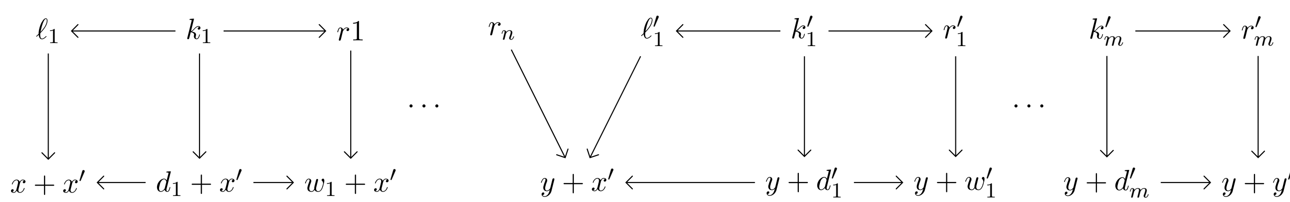
Proof. If the derivation $x \rightsquigarrow^* y$ comes from a string of double pushout diagrams



and the derivation $x' \rightsquigarrow^* y'$ comes from a string of double pushout diagrams

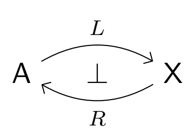


realize $x + x' \rightsquigarrow^* y + y'$ by



■

As promised, we can now decompose closed systems into open systems. For this, we need a topos of closed systems X equipped with a grammar (X, P). The closed systems need interfaces, meaning we need to introduce an adjunction



where L preserves pullbacks and has a monic counit. At this point, the material from the previous section returns. This adjunction gives a discrete comonad $\flat := LR$ from which we form the discrete grammar $(\mathsf{X}, P_\flat)$. Now define the structured cospan grammar $({}_L\mathsf{StrCsp}, \widehat{P_\flat})$ where $\widehat{P_\flat}$ contains the rule

$$\begin{array}{ccccc} L0 & \longrightarrow & \ell & \longleftarrow & LRk \\ \uparrow & & \uparrow & & \uparrow \\ L0 & \longrightarrow & LRk & \longleftarrow & LRk \\ \downarrow & & \downarrow & & \downarrow \\ L0 & \longrightarrow & r & \longleftarrow & LRk \end{array} \tag{5.4}$$

for each rule $\ell \leftarrow LRk \rightarrow r$ of $P_\flat$. We use $({}_L\mathsf{StrCsp}, \widehat{P_\flat})$ to prove our main theorem.

Before stating the theorem, we note that this theorem generalizes work by Gadducci and Heckel [35] whose domain of inquiry was graph rewriting. The arc of our proof follows theirs.

Theorem 74. Fix an adjunction $(L \dashv R)\colon \mathsf{X} \rightleftarrows \mathsf{A}$ with monic counit. Let (X, P) be a grammar such that for every X-object x in the apex of a production of P, the lattice $\mathrm{Sub}(x)$ has all meets. Given $g, h \in \mathsf{X}$, then $g \rightsquigarrow^* h$ in the rewriting relation for a grammar (X, P) if and only if there is a square

$$\begin{array}{ccccc} LR0 & \longrightarrow & g & \longleftarrow & LR0 \\ \uparrow & & \uparrow & & \uparrow \\ LR0 & \longrightarrow & d & \longleftarrow & LR0 \\ \downarrow & & \downarrow & & \downarrow \\ LR0 & \longrightarrow & h & \longleftarrow & LR0 \end{array}$$

in the double category $\mathrm{Lang}({}_L\mathsf{StrCsp}, \widehat{P_\flat})$.

Proof. We show sufficiency by inducting on the length of the derivation. If $g \rightsquigarrow^* h$

in a single step, meaning that there is a diagram

$$\begin{array}{ccccc}
\ell & \longleftarrow & LRk & \longrightarrow & r \\
\downarrow & & \downarrow & & \downarrow \\
g & \longleftarrow & d & \longrightarrow & h
\end{array}$$

then the desired square is the horizontal composition of

$$\begin{array}{ccccccccc}
L0 & \longrightarrow & \ell & \longleftarrow & LRk & \longrightarrow & d & \longleftarrow & L0 \\
\uparrow & & \uparrow & & \uparrow & & \uparrow & & \uparrow \\
L0 & \longrightarrow & LRk & \longleftarrow & LRk & \longrightarrow & d & \longleftarrow & L0 \\
\downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\
L0 & \longrightarrow & r & \longleftarrow & LRk & \longrightarrow & d & \longleftarrow & L0
\end{array}$$

The left square is a generator and the right square is the identity on the horizontal arrow $LRk \to d \leftarrow L0$. The square for a derivation $g \rightsquigarrow^* h \rightsquigarrow j$ is the vertical composition of

$$\begin{array}{ccccc}
L0 & \longrightarrow & g & \longleftarrow & L0 \\
\uparrow & & \uparrow & & \uparrow \\
L0 & \longrightarrow & d & \longleftarrow & L0 \\
\downarrow & & \downarrow & & \downarrow \\
L0 & \longrightarrow & h & \longleftarrow & L0 \\
\uparrow & & \uparrow & & \uparrow \\
L0 & \longrightarrow & e & \longleftarrow & L0 \\
\downarrow & & \downarrow & & \downarrow \\
L0 & \longrightarrow & j & \longleftarrow & L0
\end{array}$$

The top square is from $g \rightsquigarrow^* h$ and the second from $h \rightsquigarrow j$.

Conversely, proceed by structural induction on the generating squares of $\mathrm{Lang}({}_L\mathsf{StrCsp}, \widehat{P_\flat})$. It suffices to show that the rewrite relation is preserved by vertical and horizontal composition by generating squares. Suppose we have a square

$$\begin{array}{ccccc} L0 & \longleftarrow & w & \longrightarrow & L0 \\ \uparrow & & \uparrow & & \uparrow \\ L0 & \longleftarrow & x & \longrightarrow & L0 \\ \downarrow & & \downarrow & & \downarrow \\ L0 & \longleftarrow & y & \longrightarrow & L0 \end{array}$$

corresponding to a derivation $w \rightsquigarrow^* y$. Composing this vertically with a generating square, which must have form

$$\begin{array}{ccccc} L0 & \longleftarrow & y & \longrightarrow & L0 \\ \uparrow & & \uparrow & & \uparrow \\ L0 & \longleftarrow & L0 & \longrightarrow & L0 \\ \downarrow & & \downarrow & & \downarrow \\ L0 & \longleftarrow & z & \longrightarrow & L0 \end{array}$$

corresponding to a production $y \leftarrow L0 \rightarrow z$ gives

$$\begin{array}{ccccc} L0 & \longleftarrow & w & \longrightarrow & L0 \\ \uparrow & & \uparrow & & \uparrow \\ L0 & \longleftarrow & L0 & \longrightarrow & L0 \\ \downarrow & & \downarrow & & \downarrow \\ L0 & \longleftarrow & z & \longrightarrow & L0 \end{array}$$

which corresponds to a derivation $w \rightsquigarrow^* y \rightsquigarrow z$. Composing horizontally with a generating square

$$\begin{array}{ccccc} L0 & \longleftarrow & \ell & \longrightarrow & L0 \\ \uparrow & & \uparrow & & \uparrow \\ L0 & \longleftarrow & LRk & \longrightarrow & L0 \\ \downarrow & & \downarrow & & \downarrow \\ L0 & \longleftarrow & r & \longrightarrow & L0 \end{array}$$

corresponding with a production $\ell \leftarrow LRk \rightarrow r$ results in the square

$$\begin{array}{ccccc} L0 & \longleftarrow & w+\ell & \longrightarrow & L0 \\ \uparrow & & \uparrow & & \uparrow \\ L0 & \longleftarrow & x+LRk & \longrightarrow & L0 \\ \downarrow & & \downarrow & & \downarrow \\ L0 & \longleftarrow & y+r & \longrightarrow & L0 \end{array}$$

But $w + \ell \rightsquigarrow^* y + r$ as seen in Lemma 73. ■

With this result, we have completely described the rewrite relation for a grammar (X, P) with squares in $\mathrm{Lang}({}_L\mathsf{StrCsp}, \widehat{P}_\flat)$ framed by the initial object of X. These squares are rewrites of a closed system in the sense that the interface is empty. We can instead begin with a closed system x in X as represented by a horizontal arrow $L0 \rightarrow x \leftarrow L0$ in $\mathrm{Lang}({}_L\mathsf{StrCsp}, \widehat{P}_\flat)$ and decompose it into a composite of sub-systems, that is a sequence of composable horizontal arrows

$$\begin{array}{ccccccccccc} & x_1 & & & x_2 & & & & x_n & & \\ \nearrow & & \nwarrow & \nearrow & & \nwarrow & & \nearrow & & \nwarrow & \\ L0 & & & La_1 & & & La_2 \quad \cdots \quad La_{n-1} & & & & L0 \end{array}$$

Rewriting can be performed on each of these sub-systems

$$
\begin{array}{ccccccccccc}
L0 & \longrightarrow & x_1 & \longleftarrow & La_1 & & La_{n-1} & \longrightarrow & x_n & \longleftarrow & L0 \\
\cong\uparrow & & \uparrow & & \uparrow\cong & & \cong\uparrow & & \uparrow & & \uparrow\cong \\
L0 & \longrightarrow & x_1' & \longleftarrow & La_1' & \cdots & La_{n-1} & \longrightarrow & x_n' & \longleftarrow & L0 \\
\cong\downarrow & & \downarrow & & \downarrow\cong & & \cong\downarrow & & \downarrow & & \downarrow\cong \\
L0 & \longrightarrow & x_1'' & \longleftarrow & La_1'' & & La_{n-1} & \longrightarrow & x_n'' & \longleftarrow & L0 \\
 & & \vdots & & & & & & \vdots & & \\
L0 & \longrightarrow & y_1 & \longleftarrow & La_1 & & La_{n-1} & \longrightarrow & y_n & \longleftarrow & L0 \\
\cong\uparrow & & \uparrow & & \uparrow\cong & & \cong\uparrow & & \uparrow & & \uparrow\cong \\
L0 & \longrightarrow & y_1' & \longleftarrow & La_1 & \cdots & La_{n-1} & \longrightarrow & y_n' & \longleftarrow & L0 \\
\cong\downarrow & & \downarrow & & \downarrow\cong & & \cong\downarrow & & \downarrow & & \downarrow\cong \\
L0 & \longrightarrow & y_1'' & \longleftarrow & La_1 & & La_{n-1} & \longrightarrow & y_n'' & \longleftarrow & L0
\end{array}
$$

The composite of these squares is a rewriting of the original system.

Chapter 6

Conclusions

Our work here demarcates a starting line on the path towards a fully general mathematical theory of systems. We now have a syntax to reason with. Built into this syntax is a mechanism to identify when distinct systems behave similarly. That is, our syntax reflects semantics.

The semantics side requires attention. We can conjecture that the category Rel of sets and relations will be the most appropriate category to serve as our semantic universe. The naive idea behind this belief is that semantics should describe the relationship between inputs and outputs possible for a particular system. If not Rel, then something structurally similar such as the category Hilb of Hilbert spaces and linear maps. This would be appropriate semantics for the ZX-calculus.

Given a more robust theory of semantics to work with, we can fill in the larger picture of a general language for systems. To do this, Lawvere's 'functorial semantics' [45] is a promising area from which to pull. Functorial semantics has been successful in developing universal algebra, and the author believes that we can leverage Lawvere's thinking in the systems context. To what extent, however, remains an open question.

Appendix A

An account of some category theory topics

Category theory has been in mainstream mathematical discourse for decades now. This section does not seek to add to an already crowded literature on category theory. Instead, we give just enough background for those readers coming to this thesis without much knowledge about category theory. For a more in depth study of category theory, there are many excellent resources [2, 44, 46, 56].

As a baseline, we assume basic knowledge of category theory. This includes the definitions of categories, functors, natural transformations, limits, colimits, adjunctions, monoidal categories, and symmetric monoidal categories. But our needs extend beyond these basic concepts, so we provide the reader with a brief account of some more advanced topics.

A.1 Enrichment and bicategories

The most familiar examples of categories are built from mathematical widgets and their homomorphisms. For example, the category Vect_F whose objects are vector spaces over a fixed field F and arrows are linear maps. Yet, as a category, Vect_F does not truly capture everything we like about vector spaces. We are missing the fact that, for any two vector spaces V and W, the space of linear maps from V to W form a vector space by pointwise addition and scaling. Yet the hom-set $\mathsf{Vect}_F(V, W)$ is merely a collection of linear maps without additional structure. The theory of enriched categories fixes this drawback.

Many familiar categories are actually enriched. For example, the category Set of sets has that, for any two sets x, y, the collection of arrows $\mathsf{Set}(x, y)$ is actually a set. We say that Set is enriched over Set. Given the category Mod_R whose objects are modules over an arbitrary ring R and any two such modules x, y, the collections of arrows $\mathsf{Mod}_R(x, y)$ is actually a $\mathbb{Z}$-module. Thus we say that Mod_R is enriched over $\mathsf{Mod}_{\mathbb{Z}}$. However, to be an enriched category, it is not enough for the collections of arrows to simply have additional structure. Cohesion is needed.

Definition 75 (Enriched category)**.** Let $(\mathsf{M}, \otimes, I, \alpha, \lambda, \rho)$ be a monoidal category. A category C is **enriched** over M consists of

- a class $\mathrm{ob}(\mathsf{C})$ of objects,
- an object $\mathsf{C}(a, b)$ of M for each pair $a, b \in \mathrm{ob}(\mathsf{C})$ that collects the arrows of type $a \to b$
- an arrow $1_a \colon I \to \mathsf{C}(a, a)$ in M that chooses an identity arrow on a

- an arrow

$$\circ_{abc} \colon \mathsf{C}(b,c) \otimes \mathsf{C}(a,b) \to \mathsf{C}(a,c)$$

  for each triple of objects $a, b, c \in \mathrm{ob}(\mathsf{C})$ that defines the composition

together with a commuting diagram expressing associativity

$$\begin{array}{ccc}
(\mathsf{C}(c,d) \otimes \mathsf{C}(b,c)) \otimes \mathsf{C}(a,b) & \xrightarrow{\circ \otimes \mathrm{id}} & \mathsf{C}(b,d) \otimes \mathsf{C}(a,d) \\
 & & \downarrow \circ \\
\alpha \downarrow & & \mathsf{C}(a,d) \\
 & & \uparrow \circ \\
\mathsf{C}(c,d) \otimes (\mathsf{C}(b,c) \otimes \mathsf{C}(a,b)) & \xrightarrow[\mathrm{id} \otimes \circ]{} & \mathsf{C}(c,d) \otimes \mathsf{C}(a,c)
\end{array}$$

and commuting diagrams expressing left and right unity

$$\begin{array}{ccc}
I \otimes \mathsf{C}(a,b) & \xrightarrow{1 \otimes \mathrm{id}} & \mathsf{C}(b,b) \otimes \mathsf{C}(a,b) \\
 & \searrow^{\lambda} \quad \swarrow_{\circ} & \\
 & \mathsf{C}(a,b) &
\end{array}
\qquad
\begin{array}{ccc}
\mathsf{C}(a,b) \otimes I & \xrightarrow{\mathrm{id} \otimes 1} & \mathsf{C}(a,b) \otimes \mathsf{C}(a,a) \\
 & \searrow^{\rho} \quad \swarrow_{\circ} & \\
 & \mathsf{C}(a,b) &
\end{array}$$

When M is actually a 2-category and the above diagrams only commute up to natural isomorphism, then we say that C is **weakly enriched** over M.

In this thesis, the we are interested in one example of an weakly enriched category: a bicategory. In short, a bicategory is a category weakly enriched in the 2-category Cat. Thus a bicategory has a *category* of arrows between objects, not merely a collection of arrows.

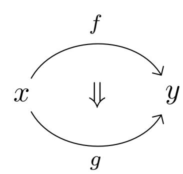
Defining a bicategory to be a category weakly enriched in Cat is elegant but hardly illuminating. Thus, the definition is worth unpacking but, for clarity's sake, we only approximate the definition by providing the important information to know and ignoring technical details.

Definition 76 (Bicategory)**.** A bicategory **C** consists of

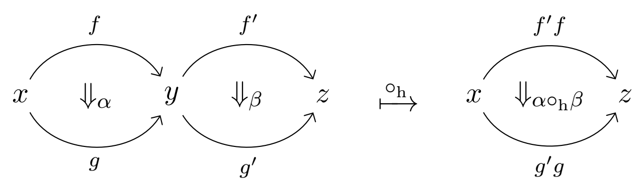
- a collection of objects $\mathrm{ob}(\mathbf{C})$
- for each pair of objects x, y, a collection of arrows of type $x \to y$ which compose, that is

$$(x \xrightarrow{f} y \xrightarrow{g} z) \mapsto (x \xrightarrow{gf} z)$$

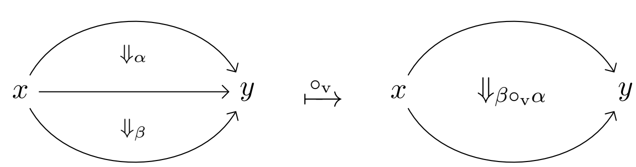
- for each pair of arrows $f, g \colon x \to y$ of the same type, a collection of 2-arrows



together with operations expressing a horizontal composition



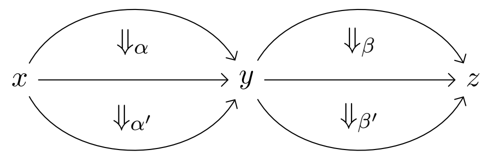
and vertical composition



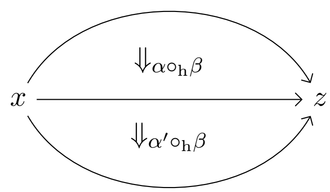
that satisfy the interchange law

$$(\alpha \circ_{\mathrm{h}} \beta) \circ_{\mathrm{v}} (\alpha' \circ_{\mathrm{h}} \beta') = (\alpha \circ_{\mathrm{v}} \alpha') \circ_{\mathrm{h}} (\beta \circ_{\mathrm{v}} \beta')$$

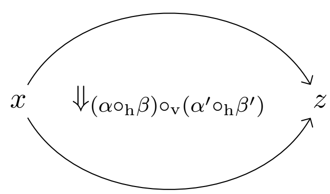
The interchange law states that given an array of 2-arrows



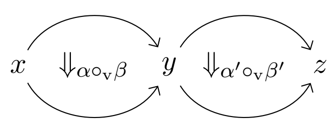
performing the two horizontal compositions



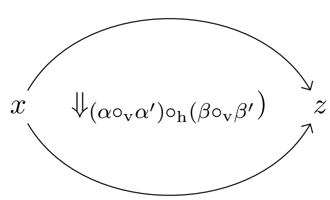
followed by the vertical composition



gives exactly the same 2-arrow as first performing the two vertical compositions



followed by the horizontal composition



That definition deconstructs a bicategory, laying out all of the components. Next, we give a definition in the spirit of enrichment.

Definition 77 (Bicategory)**.** Consider the monoidal 2-category $(\mathbf{Cat}, \times, \mathbf{1})$. A bicategory $\mathbf{C}$ has

- a collection of objects $\mathrm{ob}(\mathbf{C})$
- for each pair of objects $a, b \in \mathrm{ob}(\mathbf{C})$, a category $\mathbf{C}(a, b)$ of arrows
- for each object $a \in \mathrm{ob}(\mathbf{C})$, a functor $\mathrm{id}_a \colon \mathbf{1} \to \mathbf{C}(a, a)$ that chooses the identity element
- for each triple of objects $a, b, c \in \mathrm{ob}(\mathbf{C})$, a functor

  $$\circ_{a,b,c} \colon \mathbf{C}(b, c) \times \mathbf{C}(a, b) \to \mathbf{C}(a, c)$$

  expressing composition

such that, for all $a, b, c, d \in \mathrm{ob}(\mathbf{C})$, the associativity diagram

$$\begin{array}{ccccc}
(\mathbf{C}(c,d) \times \mathbf{C}(b,c)) \times \mathbf{C}(a,b) & & \xrightarrow{\alpha} & & \mathbf{C}(c,d) \times (\mathbf{C}(b,c) \times \mathbf{C}(a,b)) \\
\downarrow{\scriptstyle \circ \times \mathrm{id}} & & & & \downarrow{\scriptstyle \mathrm{id} \times \circ} \\
\mathbf{C}(b,d) \times \mathbf{C}(a,b) & & \Downarrow\cong & & \mathbf{C}(c,d) \times \mathbf{C}(a,c) \\
& \searrow{\scriptstyle \circ} & & \swarrow{\scriptstyle \circ} & \\
& & \mathbf{C}(a,d) & &
\end{array}$$

and the left and right unitor diagrams

$$\begin{array}{ccc}
\mathbf{1} \times \mathbf{C}(a,b) & \xrightarrow{i_b \times \mathrm{id}} & \mathbf{C}(b,b) \times \mathbf{C}(a,b) \\
& \searrow{\scriptstyle \lambda} \quad \Downarrow\cong & \downarrow{\scriptstyle \circ} \\
& & \mathbf{C}(a,b)
\end{array}
\qquad
\begin{array}{ccc}
\mathbf{C}(a,b) \times \mathbf{1} & \xrightarrow{i_a \times \mathrm{id}} & \mathbf{C}(a,b) \times \mathbf{C}(a,a) \\
& \searrow{\scriptstyle \rho} \quad \Downarrow\cong & \downarrow{\scriptstyle \circ} \\
& & \mathbf{C}(a,b)
\end{array}$$

commute up to a natural isomorphism.

We observe that the objects of the hom-category $\mathbf{C}(a,b)$ are arrows in $\mathbf{C}$ and the arrows of $\mathbf{C}(a,b)$ are 2-arrows in $\mathbf{C}$. Composition in $\mathbf{C}(a,b)$ is the vertical composition in $\mathbf{C}$. The composition of the arrows and horizontal composition of 2-arrows in $\mathbf{C}$ is given by the functor $\circ$ which, by light of it preserving composition, gives the interchange law.

A.2 Internalization and double categories

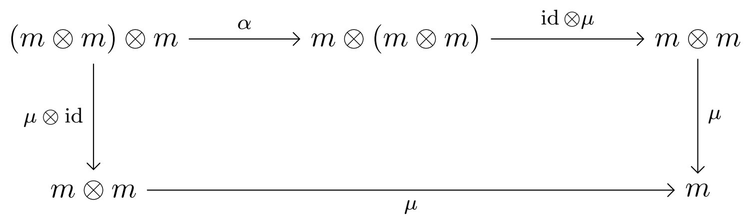
Most treatments of mathematics base definitions on set theory. The definitions for a monoid, topological space, poset, and so on all begin by establishing a set. An alternative viewpoint is to internalize such gadgets in a category.

For example, a monoid is traditionally defined to be a set M together with an identity element $e \in M$ equipped with a binary operation $M \times M \to M$ such that for all $x, y, z \in M$, we have $ex = x = xe$ and $(xy)z = x(yz)$. However, we can also define a monoid *internal* to a category.

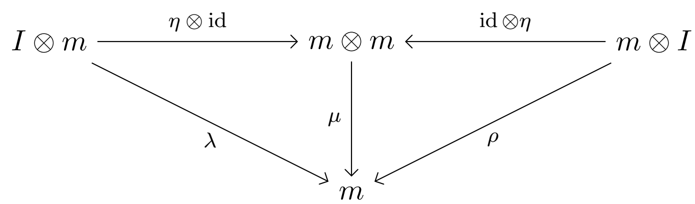
Definition 78 (Internal monoid)**.** Let $(\mathsf{C}, \otimes, I)$ be a monoidal category. A **monoid internal to** C consists of an object $m \in \mathrm{ob}(\mathsf{C})$ and two arrows in C

- (multiplication) $\mu \colon m \otimes m \to m$,
- (unit) $\eta \colon I \to m$

such that the associator diagram

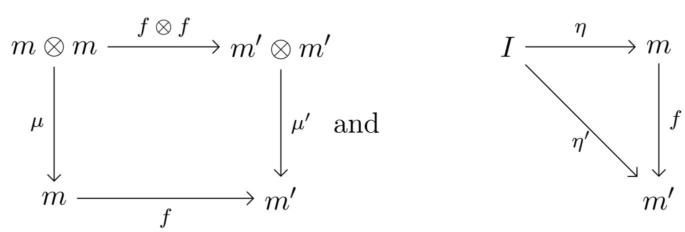


and unitor diagram



commute.

A **morphism of monoids** is an arrow $f\colon m \to m'$ in C between two monoid objects (m, μ, η) and (m', μ', η') that preserve multiplication and the unit as expressed by the following commuting diagrams



We can also provide an internal monoid with a commutative structure.

Definition 79 (Internal commutative monoid)**.** Given a symmetric monoidal category $(\mathsf{C}, \otimes, I, \tau)$ where τ is the twist map, a **commutative monoid internal to** C is, first, a monoid internal to C with the additional property that the diagram

$$\begin{array}{ccc} m \otimes m & \xrightarrow{\tau} & m \otimes m \\ & {\scriptstyle \mu}\searrow \quad \swarrow{\scriptstyle \mu} & \\ & m & \end{array}$$

commutes

Algebraic structures often have dual counterparts, and internal monoids are no exception.

Definition 80 (Internal comonoid)**.** Given a monoidal category $(\mathsf{C}, \otimes, I)$, a **comonoid internal to** C is a monoid internal to C^{op}. If $(\mathsf{C}, \otimes, I, \tau)$ is a symmetric monoidal category, then a **cocommutative comonoid internal to** C is a cocommutative comonoid internal to C^{op}.

In other words, we define comonoids exactly as we did monoids in Definitions 78 and 79 except we turn the arrows around. Many familiar algebraic objects can be exhibited as monoids internal to select categories.

Example 81. A monoid internal to Set is an ordinary monoid. A monoid internal to the category Ab of abelian groups is a ring. A monoid internal to a category $[\mathsf{C}, \mathsf{C}]$ of endofunctors is a monad on C.

As in algebra, objects can have multiple structures simultaneously. The most important for us is the Frobenius monoid.

Definition 82 (Frobenius monoid)**.** An object $(m, \mu, \eta, \delta, \varepsilon)$ in a monoidal category $(\mathsf{C}, \otimes, I)$ is called a **Frobenius monoid** if (m, μ, η) is a monoid object, (m, δ, ε) is a comonoid structure and the equation

$$(\mathrm{id} \otimes \mu)(\delta \otimes \mathrm{id}) = \delta\mu = (\mu \otimes \mathrm{id})(\mathrm{id} \otimes \delta).$$

holds.

Internalization can be extended to constructions beyond monoids and their variants. The most important construction for us is the internalization of a category.

Definition 83 (Internal category)**.** Let D be a category. A **category** $\mathbb{C}$ **internal to** D consists of the data

- an object $\mathbb{C}_0 \in \mathrm{ob}(\mathsf{D})$ of *objects* of $\mathbb{C}$
- an object $\mathbb{C}_1 \in \mathrm{ob}(\mathsf{D})$ of *arrows* of $\mathbb{C}$
- source and target arrows $s, t\colon \mathbb{C}_1 \to \mathbb{C}_0$ in D
- an identity arrow $e\colon \mathbb{C}_0 \to \mathbb{C}_1$ in D
- a composition arrow $\circ\colon \mathbb{C}_1 \times_{\mathbb{C}_0} \mathbb{C}_1 \to \mathbb{C}_1$

together with commuting diagrams

- that specify the source and target of the identity arrow

- that specify the source and target of composite arrows

$$\begin{CD} \mathbb{C}_1 \times_{\mathbb{C}_0} \mathbb{C}_1 @>{\circ}>> \mathbb{C}_1 \\ @V{p_1}VV @VV{s}V \\ \mathbb{C}_1 @>>{s}> \mathbb{C}_0 \end{CD} \qquad \begin{CD} \mathbb{C}_1 \times_{\mathbb{C}_0} \mathbb{C}_1 @>{\circ}>> \mathbb{C}_1 \\ @V{p_2}VV @VV{t}V \\ \mathbb{C}_1 @>>{t}> \mathbb{C}_0 \end{CD}$$

- that specify associativity

$$\begin{CD} \mathbb{C}_1 \times_{\mathbb{C}_0} \mathbb{C}_1 \times_{\mathbb{C}_0} \mathbb{C}_1 @>{\circ \times_{\mathbb{C}_0} \mathrm{id}}>> \mathbb{C}_1 \times_{\mathbb{C}_0} \mathbb{C}_1 \\ @V{\mathrm{id} \times_{\mathbb{C}_0} \circ}VV @VV{\circ}V \\ \mathbb{C}_1 \times_{\mathbb{C}_0} \mathbb{C}_1 @>>{\circ}> \mathbb{C}_1 \end{CD}$$

- that specify unit laws

$$\begin{array}{ccccc} \mathbb{C}_0 \times_{\mathbb{C}_0} \mathbb{C}_1 & \xrightarrow{e \times_{\mathbb{C}_0} \mathrm{id}} & \mathbb{C}_1 \times_{\mathbb{C}_0} \mathbb{C}_1 & \xleftarrow{\mathrm{id} \times_{\mathbb{C}_0} e} & \mathbb{C}_1 \times_{\mathbb{C}_0} \mathbb{C}_0 \\ & {\scriptstyle p_2} \searrow & & \swarrow {\scriptstyle p_1} & \\ & & \mathbb{C}_0 & & \end{array}$$

If we are instead working in an ambient 2-category D and the diagrams only commute up to natural isomorphism, we say that $\mathbb{C}$ is **weakly internal** to D.

The most important example of an internal category for us is a (pseudo) double category. A **(pseudo) double category** $\mathbb{C}$ is a category weakly internal to Cat. This can be unpacked.

Roughly, a double category consists of two categories $\mathbb{C}_0$ and $\mathbb{C}_1$ that we consider as follows.

- The $\mathbb{C}_0$-objects are called the objects of $\mathbb{C}$.

- The $\mathbb{C}_0$-arrows are called the vertical arrows in $\mathbb{C}$.

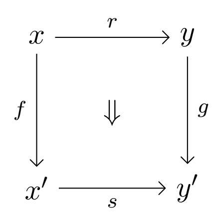
- The $\mathbb{C}_1$-objects are called the the horizontal arrows in $\mathbb{C}$.

- The $\mathbb{C}_1$-arrows are called the squares of $\mathbb{C}$.

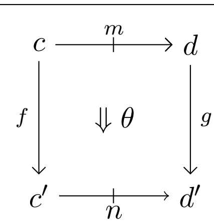
This data is depicted in Figure A.1. When the vertical arrows are both identities, we call the square **globular**.

Double categories often arise when a mathematical object has two different sorts of morphisms. One morphism type becomes the horizontal arrows, which we denote by $\nrightarrow$, and the other morphism type becomes the vertical arrows, which we denote by $\to$.

Example 84. There is a double category whose objects are sets, vertical arrows $f\colon x \to y$ are functions, horizontal arrows $r\colon x \nrightarrow y$ are relations $r \subseteq x \times y$, and squares



are inclusions of relations $gr \subseteq sf$.



$c, c', d, d' \in \mathrm{ob}(\mathsf{C}_0)$

$f, g \in \mathrm{arr}(\mathsf{C}_0)$

$m, n \in \mathrm{ob}(\mathsf{C}_1)$

$\theta \in \mathrm{arr}(\mathsf{C}_1)$

Figure A.1: A square in a double category

The first definition for a double category we gave—a category weakly internal to Cat—is too terse to provide much meaningful interpretation. So we unpack it.

Definition 85 (Double category)**.** A **pseudo double category** $\mathbb{C}$, or simply **double category**, consists of a category of objects $\mathbb{C}_0$ and a category of arrows $\mathbb{C}_1$ together with the following functors

$$U\colon \mathbb{C}_0 \to \mathbb{C}_1,$$

$$S, T\colon \mathbb{C}_1 \to \mathbb{C}_0,$$

$$\odot\colon \mathbb{C}_1 \times_{\mathbb{C}_0} \mathbb{C}_1 \to \mathbb{C}_1$$

where the pullback $\mathbb{C}_1 \times_{\mathbb{C}_0} \mathbb{C}_1$ is taken over S and T. These functors satisfy the equations

$$SUa = a = TUa \tag{A.1}$$

$$S(x \odot y) = Sy \tag{A.2}$$

$$T(x \odot y) = Tx. \tag{A.3}$$

This also comes equipped with natural isomorphisms

$$\alpha\colon (x \odot y) \odot z \to x \odot (y \odot z) \tag{A.4}$$

$$\lambda\colon Ua \odot x \to x \tag{A.5}$$

$$\rho\colon x \odot Ua \to x \tag{A.6}$$

such that $S(\alpha)$, $S(\lambda)$, $S(\rho)$, $T(\alpha)$, $T(\lambda)$, and $T(\rho)$ are each identities and that the coherence axioms of a monoidal category are satisfied.[1]

[1] Sometimes the term **horizontal 1-cell** is used for these [58], and for good reason. A $(n \times 1)$-category consists of categories $\mathbf{D_i}$ for $0 \leq i \leq n$ where the objects of $\mathbf{D_i}$ are i-cells and the morphisms of $\mathbf{D_i}$ are vertical $i+1$-morphisms. A double category is then just a (1×1)-category. From this perspective, 'cells' are always objects with morphisms going between them.

As for notation, we write vertical and horizontal morphisms with the arrows $\to$ and $\nrightarrow$, respectively, and 2-morphisms we draw as in Figure A.1.

One can define double functors and double transformations, but we refrain having no need of them in this thesis. Double categories, double functors, and double transformations form a 2-category DblCat.

Like categories, we can equip double categories with additional structure. We focus on adding a monoidal structure. As is typical in category theory, we can provide definitions at various levels of abstraction. As such, a symmetric monoidal double category is a monoid weakly internal to DblCat. This uses the same definition of a monoid internal to a category D as above, though the diagrams commute up to invertible transformation. It is worth unpacking this definition.

Definition 86 (Monoidal double category)**.** A **monoidal double category** $(\mathbb{C}, \otimes)$ is a double category $\mathbb{C}$ equipped with a functor $\otimes \colon \mathbb{C} \times \mathbb{C} \to \mathbb{C}$ such that

1. $\mathbb{C}_0$ and $\mathbb{C}_1$ are both monoidal categories.

2. If I is the monoidal unit of $\mathbb{C}_0$, then U_I is the monoidal unit of $\mathbb{C}_1$.

3. The functors S and T are strict monoidal and preserve the associativity and unit constraints.

4. There are globular 2-isomorphisms

$$\mathfrak{x} \colon (x \otimes y) \odot (x' \otimes y') \to (x \odot x') \otimes (y \odot y')$$

and

$$\mathfrak{u} \colon U(a \otimes b) \to Ua \otimes Ub$$

5. The following diagrams that express the constraint data for the double functor $\otimes$ commute

$$
\begin{array}{ccc}
((x \otimes y) \odot (x' \otimes y')) \odot (x'' \otimes y'') & \xrightarrow{\mathfrak{x} \odot 1} & ((x \odot x') \otimes (y \odot y')) \odot (x'' \otimes y'') \\
\alpha \downarrow & & \downarrow \mathfrak{x} \\
(x \otimes y) \odot ((x' \otimes y') \odot (x'' \otimes y'')) & & ((x \odot x') \odot x'') \otimes ((y \odot y') \odot y'') \\
1 \odot \mathfrak{x} \downarrow & & \downarrow \alpha \otimes \alpha \\
(x \otimes y) \odot ((x' \odot x'') \otimes (y' \odot y'')) & \xrightarrow{\mathfrak{x}} & (x \odot (x' \odot x'')) \otimes (y \odot (y' \odot y''))
\end{array}
$$

$$
\begin{array}{ccc}
(x \otimes y) \odot U(a \otimes b) & \xrightarrow{1 \odot \mathfrak{u}} & (x \otimes y) \odot (Ua \otimes Ub) \\
\rho \downarrow & & \downarrow \mathfrak{x} \\
x \otimes y & \xleftarrow{\rho \otimes \rho} & (x \odot Ua) \otimes (y \odot Ub)
\end{array}
\qquad
\begin{array}{ccc}
U(a \otimes b) \odot (x \otimes y) & \xrightarrow{\mathfrak{u} \odot 1} & (Ua \otimes Ub) \odot (x \otimes y) \\
\lambda \downarrow & & \downarrow \mathfrak{x} \\
x \otimes y & \xleftarrow{\lambda \otimes \lambda} & (Ua \odot x) \otimes (Ub \odot y)
\end{array}
$$

6. The following diagrams commute expressing the associativity isomorphism for $\otimes$ is a transformation of double categories.

$$
\begin{array}{ccc}
((x \otimes y) \otimes z) \odot ((x' \otimes y') \otimes z') & \xrightarrow{a \odot a} & (x \otimes (y \otimes z)) \odot (x' \otimes (y' \otimes z')) \\
\mathfrak{x} \downarrow & & \downarrow \mathfrak{x} \\
((x \otimes y) \odot (x' \otimes y')) \otimes (z \odot z') & & (x \odot x') \otimes ((y \otimes z) \odot (y' \otimes z')) \\
\mathfrak{x} \otimes 1 \downarrow & & \downarrow 1 \otimes \mathfrak{x} \\
((x \odot x') \otimes (y \odot y')) \otimes (z \odot z') & \xrightarrow{a} & (x \odot x') \otimes ((y \odot y') \otimes (z \odot z'))
\end{array}
$$

$$
\begin{array}{ccc}
U((a \otimes b) \otimes c) & \xrightarrow{Ua} & U(a \otimes (b \otimes c)) \\
\mathfrak{u} \downarrow & & \downarrow \mathfrak{u} \\
U(a \otimes b) \otimes Uc & & Ua \otimes U(b \otimes c) \\
\mathfrak{u} \otimes 1 \downarrow & & \downarrow \mathrm{id} \otimes \mathfrak{u} \\
(Ua \otimes Ub) \otimes Uc & \xrightarrow{a} & Ua \otimes (Ub \otimes Uc)
\end{array}
$$

7. The following diagrams commute expressing that the unit isomorphisms for $\otimes$ are

transformations of double categories.

$$
\begin{array}{ccc}
(x \otimes UI) \odot (y \otimes UI) & \xrightarrow{\mathfrak{x}} & (x \odot y) \otimes (UI \odot UI) \\
\Big\downarrow {\scriptstyle r \odot r} & & \Big\downarrow {\scriptstyle 1 \otimes \rho} \\
x \odot y & \xleftarrow{r} & (x \odot y) \otimes UI
\end{array}
\qquad
\begin{array}{ccc}
 & & Ua \otimes UI \\
 & \nearrow^{\mathfrak{u}} & \Big\downarrow {\scriptstyle r} \\
U(a \otimes I) & \xrightarrow{U_r} & Ua
\end{array}
$$

$$
\begin{array}{ccc}
(UI \otimes x) \odot (UI \otimes y) & \xrightarrow{\mathfrak{x}} & (UI \odot UI) \otimes (x \odot y) \\
\Big\downarrow {\scriptstyle \ell \odot \ell} & & \Big\downarrow {\scriptstyle \lambda \otimes 1} \\
x \odot y & \xleftarrow{\ell} & UI \otimes (x \odot y)
\end{array}
\qquad
\begin{array}{ccc}
 & & UI \otimes Ua \\
 & \nearrow^{\mathfrak{u}} & \Big\downarrow {\scriptstyle \ell} \\
U(I \otimes a) & \xrightarrow{U_\ell} & Ua
\end{array}
$$

A **braided monoidal double category** is a monoidal double category such that:

8. $\mathbb{C}_0$ and $\mathbb{C}_1$ are braided monoidal categories.

9. The functors S and T are strict braided monoidal functors.

10. The following diagrams commute expressing that the braiding is a transformation of double categories.

$$
\begin{array}{ccc}
(x \odot x') \otimes (y \odot y) & \xrightarrow{\mathfrak{s}} & (y \odot y') \otimes (x \odot x') \\
\Big\downarrow {\scriptstyle \mathfrak{x}} & & \Big\downarrow {\scriptstyle \mathfrak{x}} \\
(x \otimes y) \odot (x' \otimes y') & \xrightarrow{\mathfrak{s} \odot \mathfrak{s}} & (y \otimes x) \odot (y' \otimes x')
\end{array}
\qquad
\begin{array}{ccc}
Ua \otimes Ub & \xleftarrow{\mathfrak{u}} & U(a \otimes b) \\
\Big\downarrow {\scriptstyle \mathfrak{s}} & & \Big\downarrow {\scriptstyle U_{\mathfrak{s}}} \\
Ub \otimes Ua & \xleftarrow{\mathfrak{u}} & U(b \otimes a)
\end{array}
$$

Finally, a **symmetric monoidal double category** is a braided monoidal double category $\mathbb{C}$ such that

11. $\mathbb{C}_0$ and $\mathbb{C}_1$ are symmetric monoidal.

In Example 84, we saw a double category whose vertical arrows are functions and horizontal arrows are relations. But, functions are examples of relations. So in a sense, the vertical arrows are redundant because that information is contained in the horizontal arrows. The next definitions formalizes this observation.

Definition 87 (Companion and conjoint)**.** Let $\mathbb{C}$ be a double category and $f\colon a \to b$ a vertical arrow. A **companion** of f is a horizontal arrow $\widehat{f}\colon a \nrightarrow b$ together with squares

$$\begin{array}{ccc} a & \xrightarrow{\widehat{f}} & b \\ {\scriptstyle f}\big\downarrow & \Downarrow & \big\downarrow{\scriptstyle \mathrm{id}} \\ b & \xrightarrow[Ub]{} & b \end{array} \qquad \text{and} \qquad \begin{array}{ccc} a & \xrightarrow{Ua} & a \\ {\scriptstyle a}\big\downarrow & \Downarrow & \big\downarrow{\scriptstyle f} \\ a & \xrightarrow[\widehat{f}]{} & b \end{array}$$

such that the following equations hold:

$$\begin{array}{ccc} a & \xrightarrow{Ua} & a \\ {\scriptstyle \mathrm{id}}\big\downarrow & \Downarrow & \big\downarrow{\scriptstyle f} \\ a & \xrightarrow{\widehat{f}} & b \\ {\scriptstyle f}\big\downarrow & \Downarrow & \big\downarrow{\scriptstyle \mathrm{id}} \\ b & \xrightarrow[Ub]{} & b \end{array} \quad = \quad \begin{array}{ccc} a & \xrightarrow{Ua} & a \\ {\scriptstyle f}\big\downarrow & \Downarrow Uf & \big\downarrow{\scriptstyle f} \\ b & \xrightarrow[Ub]{} & b \end{array} \tag{A.7}$$

$$\begin{array}{ccccc} a & \xrightarrow{Ua} & a & \xrightarrow{\widehat{f}} & b \\ {\scriptstyle \mathrm{id}}\big\downarrow & \Downarrow & \big\downarrow{\scriptstyle f} & \Downarrow & \big\downarrow{\scriptstyle \mathrm{id}} \\ a & \xrightarrow[\widehat{f}]{} & b & \xrightarrow[Ub]{} & b \end{array} \quad = \quad \begin{array}{ccc} a & \xrightarrow{\widehat{f}} & b \\ {\scriptstyle \mathrm{id}}\big\downarrow & \Downarrow \mathrm{id}_{\widehat{f}} & \big\downarrow{\scriptstyle b} \\ a & \xrightarrow[\widehat{f}]{} & b \end{array} \tag{A.8}$$

A **conjoint** of f, denoted $\check{f}\colon b \nrightarrow a$, is a companion of f in the double category $\mathbb{C}^{h\cdot\mathrm{op}}$ obtained by reversing the horizontal 1-morphisms, but *not* the vertical 1-morphisms.

Definition 88 (Fibrant double category)**.** We say that a double category is **fibrant** if every vertical 1-morphism has both a companion and a conjoint. If every *invertible* vertical 1-morphism has both a companion and a conjoint, then we say the double category is **isofibrant**.

In some sense, a double category is more than a bicategory. One might believe that there is some way to extract a bicategory from a double category. In fact you can.

Definition 89 (Horizontal edge bicategory)**.** Given a double category $\mathbb{C}$, the **horizontal edge bicategory** $H(\mathbb{C})$ of $\mathbb{C}$ is the bicategory whose objects are those of $\mathbb{C}$, arrows are horizontal arrows of $\mathbb{C}$, and 2-arrows are the globular squares.

Even though we can turn any double category into a bicategory by throwing out the vertical arrows, what becomes of double categories with additional structure? The next theorem partially answers this puzzle.

Theorem 90 ([58, Theorem 5.1])**.** Let $\mathbb{C}$ be an isofibrant symmetric monoidal double category. Then $H(\mathbb{C})$ is a symmetric monoidal bicategory.

The wonderful thing about this theorem is that the axioms for the symmetric monoidal bicategory definition are typically much harder to check than the axioms for symmetric monoidal double category, and so it provides a streamlined way to construct a symmetric monoidal bicategory.

A.3 Bicategories of relations

In the early days of bicategory theory, when concerned mathematicians were exploring additional structures placed on bicategories, they discovered that the coherence involved tended to be convoluted. And so they did what mathematicians typically do, restrict their considerations to a more manageable case.

Looking at the definition of a monoidal bicategory, one is confronted with many diagrams commuting. By placing certain restrictions on the type of 2-arrows in your monoidal bicategory, this coherence is greatly simplified. The particular case we are interested in comes when the tensor $\otimes$ behaves like a product in the sense that there is a diagonal arrow $\Delta_x \colon x \to x \otimes x$ and a terminal object I (the empty product a.k.a. the unit for product). A motivating example comes from studying relations.

Relations are pervasive throughout mathematics. They play an central role in the theory of rewriting as evidenced through the importance of the rewriting relation. Classically, a relation is thought of as a subset of a product of sets $R \subseteq A \times B$. This set-theoretic perspective on relations has a category-theoretic counterpart. Given any category C, we can talk about relations *internal* to C. To foster our intuition, we first look at relations internal to Set.

Example 91. A relation internal to Set from x to y is a subobject $r \rightarrowtail x \times y$. Set-theoretically speaking, r is a subset of $x \times y$. This matches the classical notion of relation.

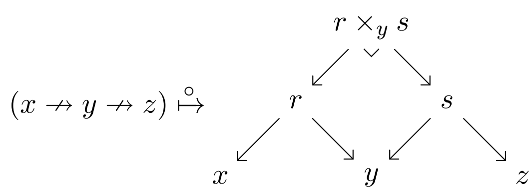
However, defining a relation internal to a category C as a subobject of a binary product is poor form. Not all categories have products. Hence the following definition is given.

Definition 92 (Internal relation)**.** A **relation internal to a category** C, denoted $x \nrightarrow y$ for $x, y \in \mathrm{ob}(\mathsf{C})$, is a jointly monic span

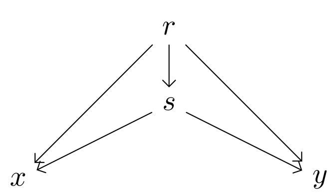
$$x \xleftarrow{f} r \xrightarrow{g} y.$$

That is, for any pair of arrows $f', g' \colon u \to r$ such that $ff' = fg'$ and $gf' = gg'$, then $f' = g'$. When C has binary products, this is equivalent to the pairing $\langle f, g \rangle \colon r \to x \times y$ being a monomorphism.

The categorical minded mathematician might see this and ask if we can construct category from the objects of C and its internal relations. If C is a topos, then the answer is yes. This is not the broadest class of categories for which this construction works, but the class of topoi is as broad as we can go without writing another section of this appendix. Given a topos C, there is a category $\mathsf{Rel}(\mathsf{C})$ called **the category of relations internal to** C. Its objects are those of C and arrows $\mathsf{Rel}(\mathsf{C})(x, y)$ are internal relations $x \leftarrow r \rightarrow y$. Composition is given by pullback

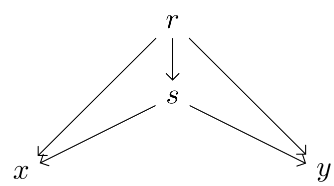


In fact, $\mathsf{Rel}(\mathsf{C})$ can be promoted to a bicategory $\mathbf{Rel}(\mathsf{C})$ by taking as 2-arrows maps of spans. Specifically, a 2-arrow between internal relations $x \leftarrow r \rightarrow y$ to $x \leftarrow s \rightarrow y$ is an arrow $f\colon r \rightarrow s$ of C fitting into the commuting diagram



It follows from the jointly monic condition that given any other arrow $g\colon r \rightarrow s$ fitting into the above diagram, it follows that $f = g$. The parallel between relations in Set is clear: a morphism of relations is like a subset inclusion.

Remark 93. There is a name to the property of $\mathbf{Rel}(\mathsf{C})$ that between parallel arrows, either a single 2-arrow exists or none does. It is called being **locally posetal**. Another way of saying this is that $\mathbf{Rel}(\mathsf{C})$ is a category enriched in Pos, the category of posets and order preserving functions. This means that for any objects x, y of $\mathbf{Rel}(\mathsf{C})$, there is a poset $\mathbf{Rel}(\mathsf{C})(x, y)$ whose elements are the relations from $x \nrightarrow y$ that are internal to C and the ordering is defined by setting $r \leq s$ whenever there is an arrow $r \to s$ in C such that the diagram



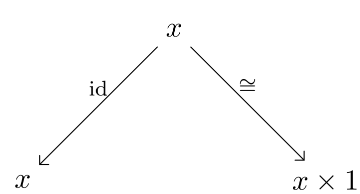
commutes. Because of this, we denote 2-arrows in locally posetal bicategories by $\leq$ instead of $\Rightarrow$. We explain enriched category theory basics in Appendix A.1.

Fix a cartesian category $(\mathsf{T}, \times, 1)$ with T a topos. This cartesian structure provides $\mathbf{Rel}(\mathsf{T})$ with some nice structure of its own. First, there is a tensor product in the form of a pseudo-functor

$$\otimes \colon \mathbf{Rel}(\mathsf{T}) \times \mathbf{Rel}(\mathsf{T}) \to \mathbf{Rel}(\mathsf{T})$$

defined by $(x, y) \mapsto x \times y$ where $\times$ is the product in T, and pointwise application of $\times$ on the jointly monic spans. We also have natural isomorphisms

- $x \to x \otimes 1$ given by the internal relation



- $x \otimes y \to y \otimes x$ given by the internal relation

$$x \times y \xleftarrow{\ \mathrm{id}\ } x \times y \xrightarrow{\ \cong\ } y \times x$$

- $(x \otimes y) \otimes z \to (x \otimes y) \otimes z$ given by the internal relation

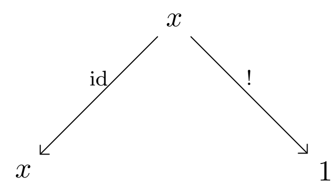
$$(x \times y) \times z \xleftarrow{\ \mathrm{id}\ } (x \times y) \times z \xrightarrow{\ \cong\ } (x \times y) \times z$$

that satisfy the required coherence conditions. Because $\mathbf{Rel}(\mathsf{T})$ is locally posetal, the 1-category coherence laws for unity, symmetry, and associativity suffice.

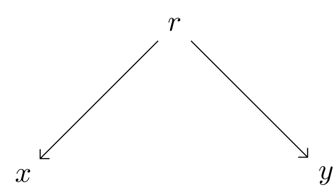
Because the definition of $\otimes$ uses the cartesian structure on T, there is a cartesian-like quality to $\otimes$ in $\mathbf{Rel}(\mathsf{T})$. However, 2-limits are difficult, so we characterize this quality via comonoids. Before talking about comonoids in $\mathbf{Rel}(\mathsf{T})$, we look at comonoids in T. Observe that by taking T to be cartesian, every object in T has a comonoid structure: the comultiplication $\Delta_x \colon x \to x \times x$ is given by the diagonal map and the counit $\varepsilon_x \colon x \to 1$ is the unique map to the terminal object 1. We lift this to define a comonoid structure on $\mathbf{Rel}(\mathsf{T})$ by setting the comultiplication $\Delta \colon x \to x \otimes x$ as the internal relation

$$x \xleftarrow{\ \mathrm{id}\ } x \xrightarrow{\ \langle \mathrm{id}, \mathrm{id} \rangle\ } x \times x$$

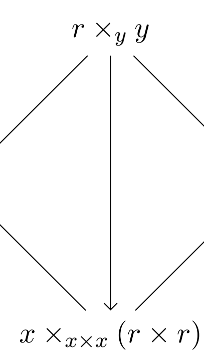
and the counit to be the internal relation



Every arrow in $\mathbf{Rel}(\mathsf{T})$ plays nicely with the comonoid structure. Suppose we have an arrow $r\colon x \nrightarrow y$, hence a jointly monic span



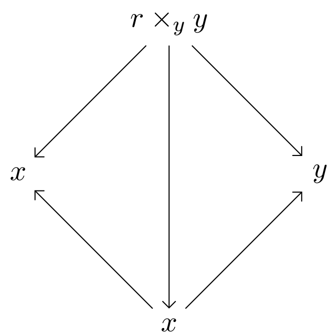
Then r is a lax comonoid homomorphism in that there are 2-arrows $\Delta_y r \leq (r \otimes r)\Delta_x$ and $\varepsilon_y r \leq t_x$. The lax preservation of comultiplication is the 2-arrow



where $r \times_y y \cong r$ and one can determine that $r \times r$ is a subobject of $x \times_{x \times x} (r \times r)$. The 2-arrow then is the composite

$$r \times_y y \xrightarrow{\cong} r \xrightarrow{\Delta} r \times r \rightarrowtail x \times_{x \times x} (r \times r).$$

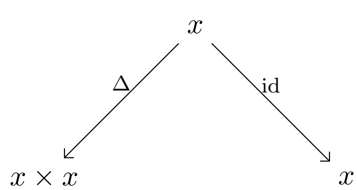
The lax preservation of unit is the 2-arrow

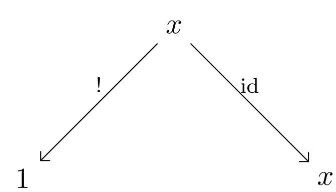


obtained as the composite

$$r \times_y y \xrightarrow{\cong} r \to x.$$

Also, because we are working with spans, we can turn them around to give a monoid structure $\Delta_x^*\colon x \otimes x \nrightarrow x$ and $\varepsilon_x^*\colon 1 \nrightarrow x$ given by the respective spans

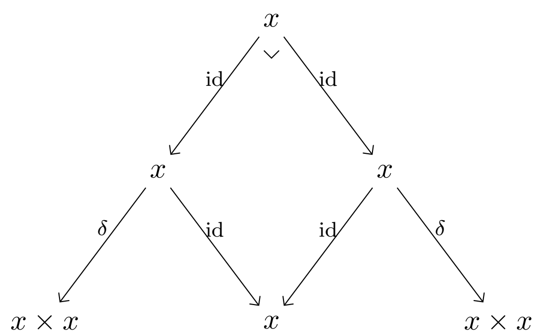




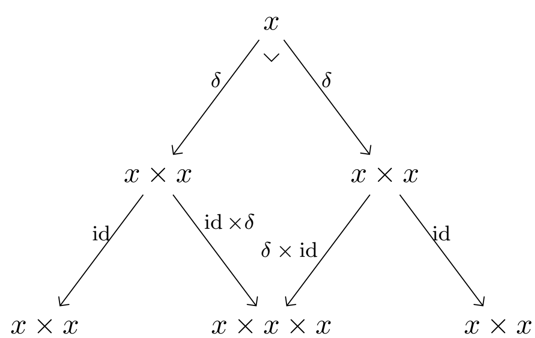
What Carboni and Walters did was to take this structure as primitive to define a Cartesian bicategory. Though they went farther by axiomatizing another important property of **Rel**(T). Namely that any object x of **Rel**(T) is a Frobenius monoid (see Definition 82) which, recall, requires the equation

$$(\mathrm{id} \otimes \mu)(\delta \otimes \mathrm{id}) = \delta\mu = (\mu \otimes \mathrm{id})(\mathrm{id} \otimes \delta).$$

to hold. The left hand side of this equation is given by the composite



and the right-hand side of the equation is given by the composite



Hence, the equality of the composite spans. In Section A.4, we axiomatize the structures and properties found in a category of relations internal to a topos.

Having though about $\mathbf{Rel}(\mathsf{T})$, we can now axiomatize some important structures. The first structure needed is a tensor product for a bicategory. In general, the coherence can be quite complicated but simplifies significantly when restricting our attention to locally posetal bicategories.

Definition 94. A tensor product $\otimes\colon \mathbf{B} \times \mathbf{B} \to \mathbf{B}$ on a locally posetal bicategory $\mathbf{B}$ is a pseudo-functor equipped with an unit object I and natural isomorphisms

$$\rho\colon x \to x \otimes I \qquad \lambda\colon x \to I \otimes x$$

$$\sigma\colon x \otimes y \to y \otimes x \qquad \alpha\colon (x \otimes y) \otimes z \to x \otimes (y \otimes z)$$

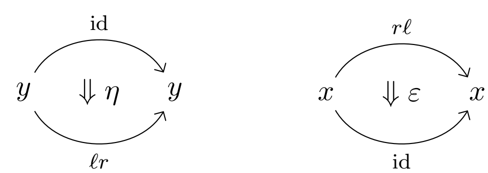
that satisfy the classical coherence conditions.

We also need to place the concept of adjoint functors into a general bicategory. The data of an adjoint pair—two functors and two natural transformations—are merely 1-arrows and 2-arrows in **Cat**. However, this structure can be supported by bicategories other than **Cat**.

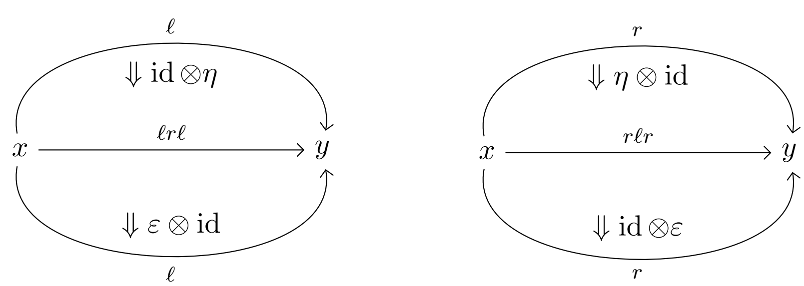
Definition 95 (Adjunction)**.** Let **B** be a bicategory. We say the 1-arrows

$$\ell\colon x \to y \quad \text{and} \quad r : y \to x$$

form an **adjunction**, with ℓ the left adjoint and r the right adjoint if there exist 2-arrows



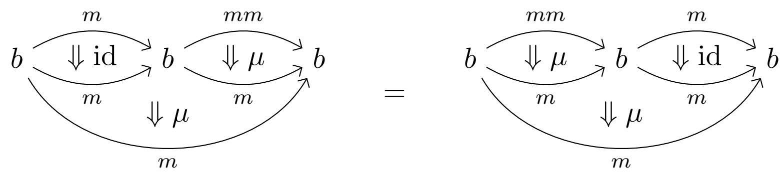
respectively named the unit and the counit such that each composite



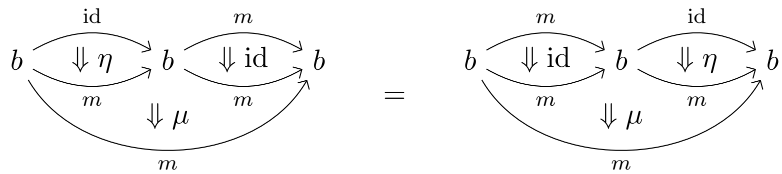
is an identity.

Closely related to adjoint arrows are the dual concepts of monad and comonad. Also like adjunctions, the most common monads and comonads are internal to the 2-category **Cat**. Comonads in particular are relevant for us in Section 5.1.

Definition 96 ((Co)monad)**.** In a bicategory $\mathbf{B}$, an arrow $m\colon b \to b$ is called a **monad** if there are 2-arrows $\mu\colon mm \to m$ and $\eta\colon \mathrm{id}_b \to m$ such that



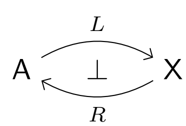
and also



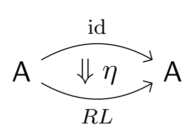
When the 2-arrows are reversed, we get a **comonad**

There is a close relationship between adjunctions, monads, and comonads. Instead of exploring this relationship in its full generality, we restrict our attention to adjunctions, monads, and comonads in Cat.

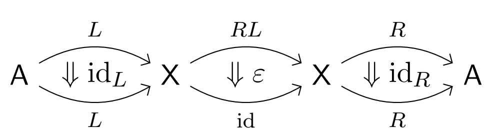
For any adjunction



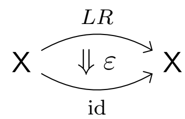
with unit η and counit ε, we define a monad $RL\colon \mathsf{A} \to \mathsf{A}$ with unit



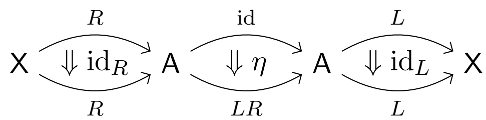
and multiplication $R\varepsilon L\colon RLRL \Rightarrow RL$ given by the horizontal composite



The adjunction also induces a comonad with counit



and comultiplication $L\eta R \colon LR \Rightarrow LRLR$ given by the composite



We use this latter fact in Section 5.1.

The opposite direction, from monads to adjunctions, is a more subtle issue because to each monad is associated a family of adjunctions. This is not used in this thesis, however, so we point the reader to a standard reference [46] to learn more.

We now have all of the background needed to define a cartesian bicategory.

Definition 97 (Cartesian bicategory)**.** A cartesian bicategory consists of the following data:

- a locally posetal bicategory $\mathbf{B}$
- a tensor product $\otimes \colon \mathbf{B} \times \mathbf{B} \to \mathbf{B}$
- for every object x of $\mathbf{B}$, a cocommutative monoid structure $\Delta_x \colon x \to x \otimes x$ and $\varepsilon_x \colon x \to x \otimes I$

such that

- every arrow $r \colon x \to r$ is a lax comonoid homomorphism, that is

$$\Delta_y r \leq (r \otimes r)\Delta_x \quad \text{and} \quad \varepsilon_y r \leq \varepsilon_x$$

- for each object x, comultiplication Δ_x and counit ε_x have right adjoints Δ_x^* and ε_x^* that give a commutative monoid structure to x.

Such a bicategory is called cartesian because of its similarities to a cartesian category.

Another nice feature we saw in our favorite cartesian bicategory $\mathbf{Rel}(\mathsf{C})$ is that each object x is a Frobenius monoid. When we append this axiom to those for a cartesian bicategory, we obtain a more complete axiomatization of $\mathbf{Rel}(\mathsf{C})$. Because of this we call such a gadget a **bicategory of relations**.

Definition 98 (Bicategory of relations)**.** A **bicategory of relations** is a cartesian bicategory $\mathbf{B}$ such that for all objects x, the structure maps $\Delta_x, \varepsilon_x, \Delta_x^*, \varepsilon_x^*$ satisfy the Frobenius law

$$\Delta_x \Delta_x^* = (\mathrm{id} \otimes \Delta_x)(\Delta_x^* \otimes \mathrm{id}).$$

It follows from the Frobenius law that in a bicategory of relations, every object is its own dual. This brings us to our next section on duality in bicategories.

A.4 Duality in bicategories

One's first encounter with the term 'dual' is typically in linear algebra. Recall that given a K-vector space V and its dual V^*, there is a linear map $V^* \otimes_K V \to K$. Also, K is the identity with respect to $\otimes_K$, that is $K \otimes_K V \cong V$. The fact every object in the monoidal category $(\mathsf{Vect}_K, \otimes_K, K)$ of K-vector spaces and K-linear maps has such a dual can be generalized to other monoidal categories. Such categories are called **compact closed**.

Briefly returning to the previous section, we left off saying that in a bicategory of relations every object is its own dual. And though the coherence is more complicated for bicategories in general, locally posetal bicategories, such as bicategories of relations, skirt this issue. Due to the restriction on 2-arrows, showing that a locally posetal bicategory is compact closed is exactly the same as showing a categories is compact closed. Hence our next theorem, that a bicategory of relations is necessarily compact closed, holds true and it is the Frobenius law that provides this structure.

Theorem 99. A bicategory of relations is compact closed.

Proof. See Theorem 2.4 in Carboni and Walters [13]. ■

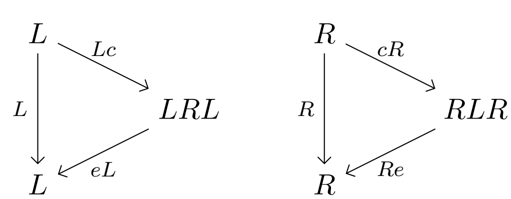
For the remainder of this section, we move beyond locally posetal bicategories to discuss compact closure for generic monoidal bicategories.

To define 'compact closed bicategories' as conceived by Stay [59], we discuss a notion of duality suitable for bicategories. We write LR for the tensor product of objects L and R and fg for the tensor product of morphisms f and g.

Definition 100 (Dual pair, category)**.** A **dual pair** in a symmetric monoidal category $(\mathsf{C}, \otimes, I)$ is a tuple (L, R, e, c) with objects L and R, called the **left** and **right** duals, and morphisms

$$e\colon LR \to I \qquad c\colon I \to RL,$$

called the **counit** and **unit**, respectively, such that the following diagrams commute.



A category such such that every object has a dual is called **compact closed**.

Definition 101 (Dual pair, bicategory)**.** Inside a monoidal bicategory, a **dual pair** is a tuple $(L, R, e, c, \alpha, \beta)$ with objects L and R, morphisms

$$e\colon LR \to I \qquad c\colon I \to RL,$$

and invertible 2-morphisms

$$\begin{array}{ccc} L & \xrightarrow{L} & L \\ \downarrow & & \uparrow \\ LI & \Downarrow \alpha & IL \\ {\scriptstyle Lc}\downarrow & & \uparrow{\scriptstyle eL} \\ L(RL) & \longrightarrow & (LR)L \end{array} \qquad \begin{array}{ccc} R & \xrightarrow{R} & R \\ \downarrow & & \uparrow \\ RI & \Downarrow \beta & RI \\ {\scriptstyle cR}\downarrow & & \uparrow{\scriptstyle Re} \\ (RL)R & \longrightarrow & R(LR) \end{array}$$

called **cusp isomorphisms**. If this data satisfies the swallowtail equations in the sense that the diagrams in Figure A.2 are identities, then we call the dual pair **coherent**.

Recall that a symmetric monoidal category is called **compact closed** if every object is part of a dual pair. We can generalize this idea to bicategories by introducing 2-morphisms and some coherence axioms. The following definition is due to Stay [59].

Definition 102 (Compact closed bicategory)**.** A **compact closed** bicategory is a symmetric monoidal bicategory for which every object R is part of a coherent dual pair.

The difference between showing compact closedness in categories versus bicategories might seem quite large because of the swallowtail equations. Looking at Figure A.2,

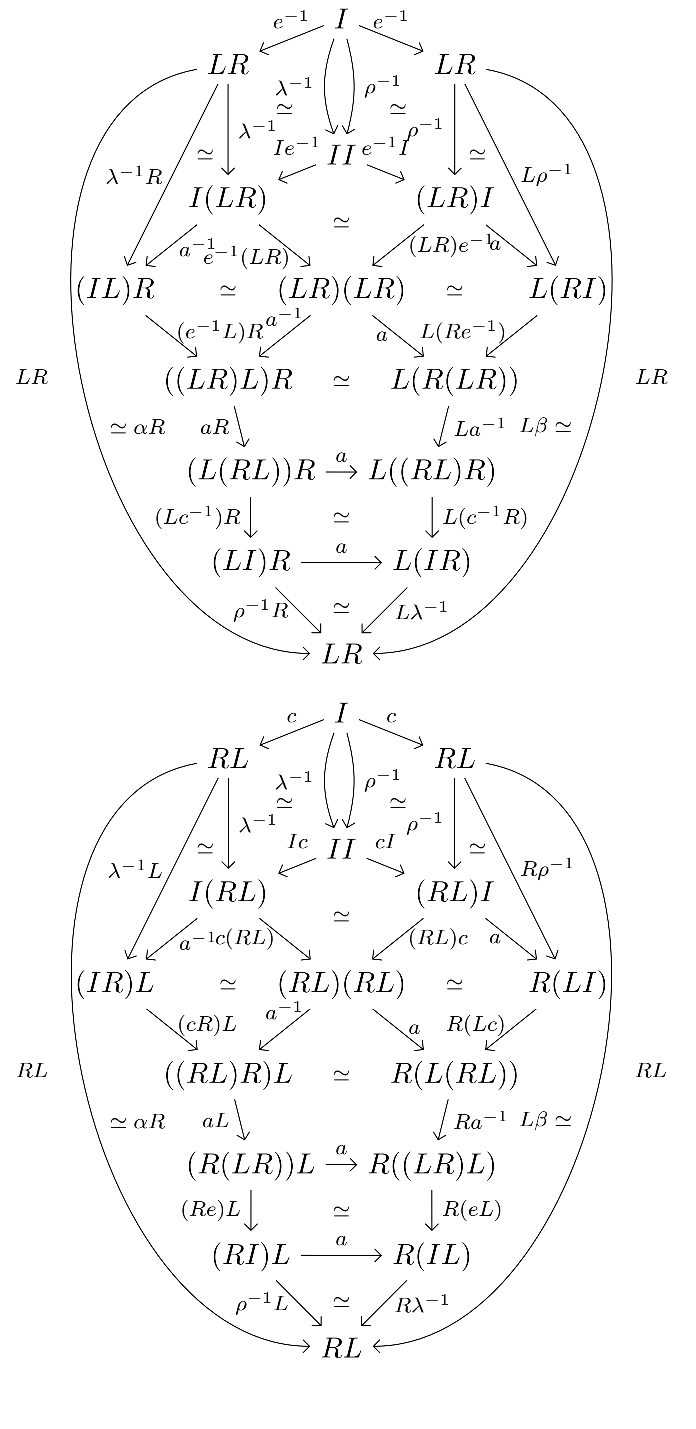


Figure A.2: The swallowtail diagrams for the unit and counit.

it is no surprise that these can be incredibly tedious to work with. Fortunately, Pstrágowski [55] proved a wonderful strictification theorem that effectively circumvents the need to consider the swallowtail equations.

Theorem 103 ([55, p. 22])**.** Given a dual pair $(L, R, e, c, \alpha, \beta)$, we can find a cusp isomorphism β' such that $(L, R, e, c, \alpha, \beta')$ is a coherent dual pair.

A.5 Adhesive categories

After Ehrig, et. al. introduced double pushout graph rewriting [34], there were several attempts at axiomatizing it. The first successful attempt is called High-Level Replacement Systems (HLRS) [32, 33]. To be thorough, we include the axioms of an HLRS.

Definition 104 (High level replacement system)**.** A category C is called a **High Level Replacement System** if

1. pushouts exist for all spans $a \leftarrow b \rightarrow c$ such that one arrow is monic;
2. pullbacks exist for all cospans $a \rightarrow b \leftarrow c$ where both arrows are monic;
3. pushouts and pullbacks respect monomorphisms;
4. for any diagram

$$\begin{array}{ccccc} a & \longrightarrow & b & \rightarrowtail & c \\ \downarrow & & \downarrow & & \downarrow \\ d & \longrightarrow & e & \rightarrowtail & f \end{array}$$

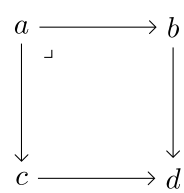
   such that the marked arrows are monic, the outside rectangle is a pushout, and the right square is a pullback, then the left square is a pushout;

5. binary coproducts exist;

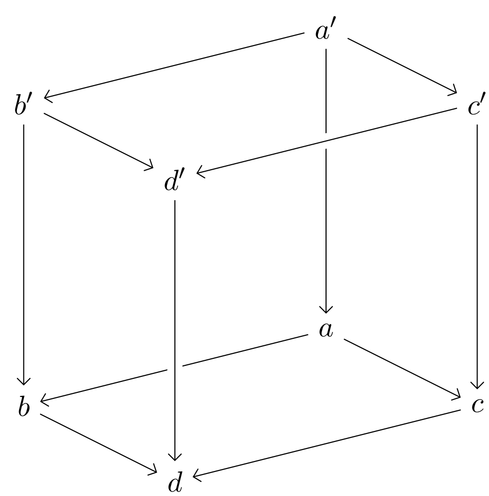
6. any pushout of a span with a monic arrow is also a pullback.

This collection of axioms was curated to prove theorems such as the local Church–Rosser and concurrency, the presence of which provide a rich rewriting theory. Lack and Sobociński later provided a more compact set of axioms that also allowed local Church–Rosser and concurrency theorems [42]. To earn the shorter list of axioms, they packed quite a bit of information into an axiom by using a 'Van Kampen square'.

A **Van Kampen square** is a pushout



that, when placed on the bottom of a cube



such that the back faces are pullbacks, then the front faces are pullbacks if and only if the top face is a pushout.

Definition 105 (Adhesive category)**.** An **adhesive category**

1. has pushouts along monomorphisms;

2. has pullbacks;

3. pushouts along monomorphisms are Van Kampen squares.

Roughly, the Van Kampen condition places adhesive categories in the company of distributive categories and extensive categories in the sense of a compatibility between certain finite limits and finite colimits. In the case of distributive categories, there is a compatibility between products and coproducts. For extensive categories, pullbacks and coproducts play nicely together. The Van Kampen condition stipulates the compatibility between pullback and pushout.

Certainly, the definition of an adhesive category is more elegant than that of an HLRS. The price of elegance is the dense Van Kampen condition. While adhesive categories are not exactly HRLS's, they are closely related as one might expect.

Proposition 106 ([42, Lem. 29])**.** An adhesive category with an initial object is an HLRS.

Though fewer in number, the axioms for an adhesive category are non-trivial. Also, adhesive categories are not so well-known outside of rewriting theory. Therefore, instead of working with adhesive category, we work with a much more well-known class of category: a topos. Fortunately, every elementary topos is adhesive. This result is the subject of a paper by Lack and Sobociński [43].

Theorem 107. Every elementary topos is adhesive.

Because topoi are our categories of choice for the present work and in light of Theorem 107, we leave our discussion of adhesive categories here. In the next section, we cover topos theory, but just enough for our needs. This includes facts that morally belong to adhesive category theory and also hold true for topoi.

A.6 Topoi

When searching the literature on topos theory, one finds myriad descriptions of what a topos is like. Suffice to say, any topos has a geometric aspect and a logical aspect. With regards to the geometric aspect, a topos is like a generalized space, where the objects are subspaces and the arrows describe how the various subspaces relate to one another. But to each topos, there is an internal logic from which we can recover various logics by using the arrows to and from the subobject classifier which we define now[2].

Definition 108 (Subobject classifier)**.** A **subobject classifier** is a monomorphism

$$\texttt{true}\colon 1 \to \Omega$$

from the terminal object with the property that, for every objects $t \in \mathsf{T}$ and subobject $s \to t$, there exists a unique arrow χ_s fitting into the pullback diagram

$$\begin{array}{ccc} s & \longrightarrow & 1 \\ \big\downarrow & \lrcorner & \big\downarrow \\ t & \xrightarrow[\chi_s]{} & \Omega \end{array}$$

In the category Set, any two element set is a subobject classifier. Take the set $\{0, 1\}$. Then any function into that set determines a subobject, here just a subset, by taking

[2] For a full account of logic via topos theory, see Part D of Johnstone's *Sketches of an Elephant* [38].

the fiber of 1. Similarly, any subobject $s \to t$ determines a map $\chi_s \colon t \to \{0,1\}$ by sending an element of t to 1 if it belongs to s and sending an element of t to 0 if it does not belong to s.

Definition 109 (Topos)**.** A **topos** T is a category with finite limits, is cartesian closed, and has a subobject classifier.

The examples we give below cover our needs.

Example 110.

1. The archetypal topos is the category Set. The subobject classifier is the two-element set $\{0,1\}$ where we interpret 0 as 'false' and 1 as 'true'.

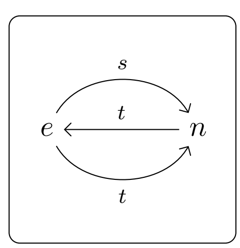
2. Presheaf categories $\mathsf{Set}^{\mathsf{C}^{\mathrm{op}}}$ are topoi when C is a small category. The subobject classifier is the functor $\mathsf{C}^{\mathrm{op}} \to \mathsf{Set}$ that sends any object c in C to the set of subfunctors of $\mathsf{C}(-,c)$. This is called a 'sieve' of c.

3. Finite presheaf categories are topoi. These are functor categories of the type $\mathsf{FinSet}^{\mathsf{C}^{\mathrm{op}}}$ for C finite.

Of these classes of examples, the presheaf topoi are the most pertinent. There is one specific presheaf topos that we particularly like.

Example 111. Our favorite example of a presheaf topos is RGraph, the category of reflexive directed multi-graphs. This is the category of presheaves on

$\mathsf{C}^{\mathrm{op}} :=$



such that all arrows $n \to n$ are the identity. A presheaf $g \colon \mathsf{C}^{\mathrm{op}} \to \mathsf{Set}$ then consists of two sets $g(e)$ and $g(n)$ considered as sets of edges and nodes. Then there are two arrows of type $g(e) \to g(n)$ assigning each edge its source and target and one arrow of type $g(n) \to g(e)$ assigning a reflexive edge to each node. This is exactly a reflexive graph. A natural transformation θ between presheaves $g, h \colon \mathsf{C}^{\mathrm{op}} \to \mathsf{Set}$ is a pair of functions $\theta_e \colon g(e) \to h(e)$ and $\theta_n \colon g(n) \to h(n)$ such that the squares

$$\begin{array}{ccc} g(e) & \xrightarrow{g(t)} & g(n) \\ {\scriptstyle \theta_e}\downarrow & & \downarrow{\scriptstyle \theta_n} \\ h(e) & \xrightarrow[h(t)]{} & h(n) \end{array} \qquad \begin{array}{ccc} g(e) & \xrightarrow{g(t)} & g(n) \\ {\scriptstyle \theta_e}\downarrow & & \downarrow{\scriptstyle \theta_n} \\ h(e) & \xrightarrow[h(t)]{} & h(n) \end{array} \qquad \begin{array}{ccc} g(e) & \xrightarrow{g(t)} & g(n) \\ {\scriptstyle \theta_e}\downarrow & & \downarrow{\scriptstyle \theta_n} \\ h(e) & \xrightarrow[h(t)]{} & h(n) \end{array}$$

commute. These squares assert that the natural transformations preserve source, targets, and reflexive nodes. Hence, this is precisely the data of a reflexive graph morphism.

Because topoi have both geometric and logical aspects, there are morphisms of topos for each.

Definition 112 (Geometric morphism)**.** A **geometric morphism** between topoi $\mathsf{X} \to \mathsf{A}$ is an adjunction

$$\mathsf{A} \underset{R}{\overset{L}{\rightleftarrows}} \mathsf{X} \quad (L \dashv R)$$

such that L preserves finite limits. We call L the inverse image functor and R the direct image functor.

Geometric morphisms abstract from continuous maps between spaces $f \colon S \to T$. Denote by $\mathcal{O}S$ and $\mathcal{O}T$ the open sets of S and T. Then f induces the direct image map

$f_*\colon \mathcal{O}S \to \mathcal{O}T$ that sends a set $A \subseteq S$ to its image $\{t \in T | \exists a \in A. fa = t\}$. But f also induces an inverse image map $f^*\colon \mathcal{O}T \to \mathcal{O}S$ that sends a set $B \subseteq T$ to its preimage $\{s \in S | \exists b \in B. fs = b\}$. Observe that f_* preserves finite intersections and f^* preserves finite intersection and unions. This mirrors the fact that, in a geometric morphism the right adjoint preserves finite limits and the left adjoint preserves finite limits and colimits.

Now that the basic definition of a topos are given, we provide just enough theory to develop the ideas in this thesis.

The first result we give is often called the fundamental theorem of topos theory [38, A.2.3.2].

Theorem 113. Given a topos T and an object t of T, then the over-category $\mathsf{T} \downarrow t$ is also a topos.

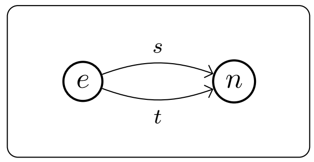
The operation of 'slicing over an object' is stable in presheaf topoi. This result uses a construction called the category of elements. Given a functor $f\colon \mathsf{C} \to \mathsf{Set}$, its category of elements, denoted $\int^f \mathsf{C}$, has for objects pairs (c, x) where c is an object of C and x is an element of the set fc. The arrows $(c, x) \to (d, y)$ are the set functions $fc \to fd$ such that $x \mapsto y$. The category of elements is a first foray into the much larger topic called 'the Grothendieck construction'. However, it is not useful for us to pursue this topic.

Theorem 114. Let C be a small category and $F\colon \mathsf{C}^{\mathrm{op}} \to \mathsf{Set}$ a presheaf. Then the over-category $\mathsf{Set}^{\mathsf{C}^{\mathrm{op}}} \downarrow F$ is equivalent to the topos of presheaves on the category of elements $\int^F \mathsf{C}$.

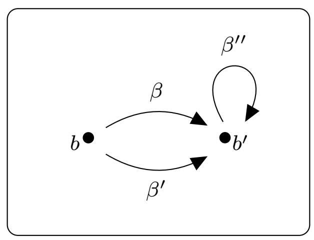
This result is used in Section 4.3. We illustrate it here with graphs.

Example 115. In this example, we illustrate the equivalence of Theorem 114 by translating an object from $\mathsf{Set}^{\mathsf{C}^{\mathrm{op}}} \downarrow F$ to a presheaf in the category $\mathsf{Set}^{\int^F \mathsf{C}}$ for a specific choice of F and C.

Let C^{op} be the walking graph category. That is,



We call this the walking graph category to suggest that the presheaves on C^{op} are exactly graphs and natural transformations between these functors are exactly the graph morphisms. Let F be the graph



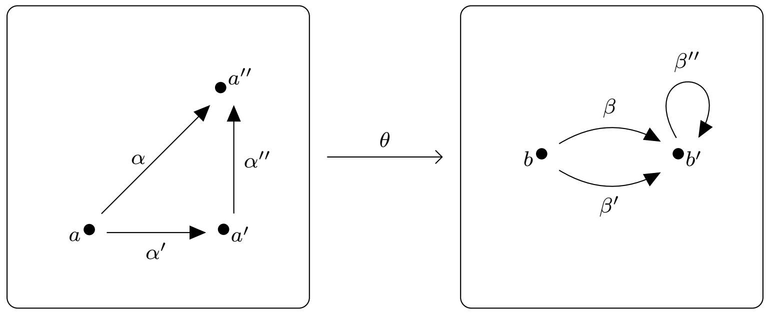
As a functor, $F\colon \mathsf{C}^{\mathrm{op}} \to \mathsf{Set}$ returns the edge set $Fe \coloneqq \{\beta, \beta', \beta''\}$, the node set $Fn \coloneqq \{b, b'\}$, the source map $Fs\colon Fe \to Fn$ defined by

$$Fs(\beta) \coloneqq b, \quad Fs(\beta') \coloneqq b, \quad Fs(\beta'') \coloneqq b'$$

and the target map $Ft\colon Fe \to Fn$ defined by

$$Ft(\beta) \coloneqq b', \quad Ft(\beta') \coloneqq b', \quad Ft(\beta'') \coloneqq b'.$$

The graph morphism $G \to F$, depicted by

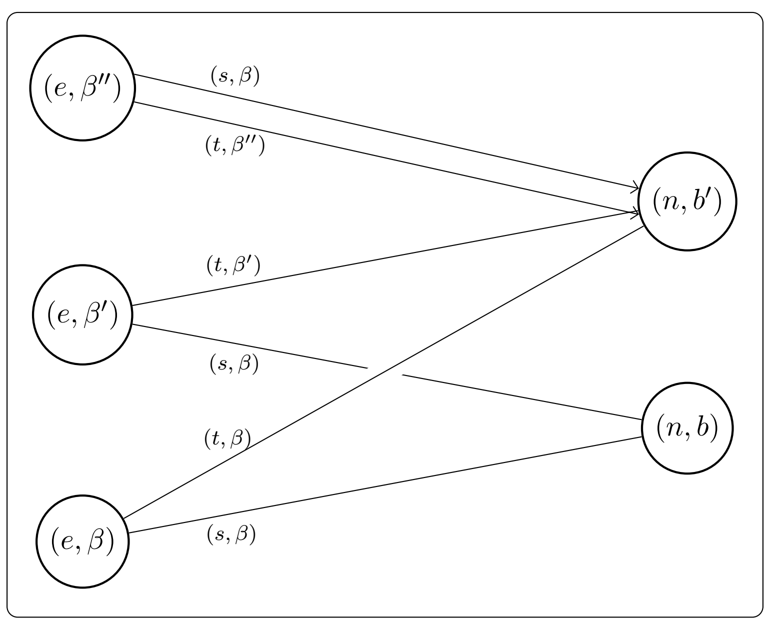


where θ is given by,

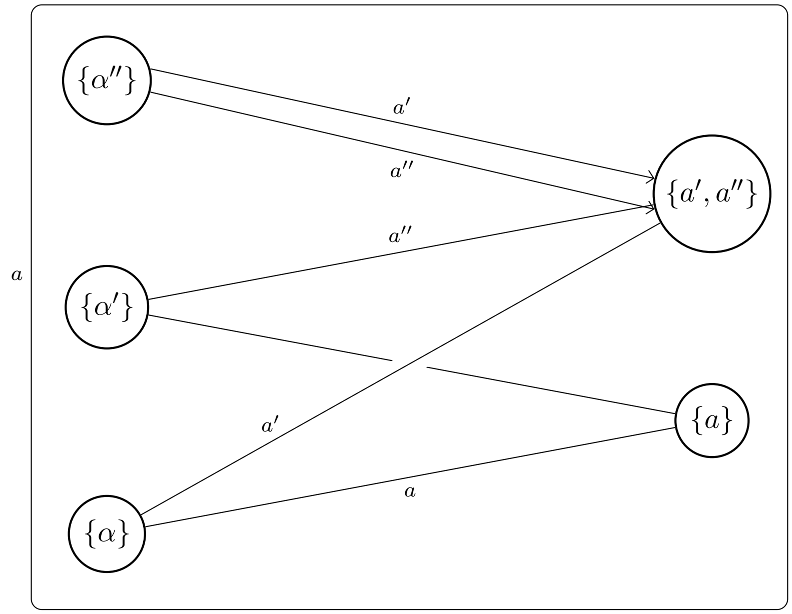
$$\theta(a) \coloneqq b \qquad \theta(a'), \theta(a'') \coloneqq b' \qquad \theta(\alpha) \coloneqq \beta \qquad \theta(\alpha') \coloneqq \beta' \qquad \theta(\alpha'') \coloneqq \beta''$$

is an object in $\mathsf{Set}^{\mathsf{C}^{\mathrm{op}}} \downarrow F$

According to Theorem 114, we can translate $G \to F$ to a presheaf on the category of elements $\int^F \mathsf{C}$, which we depict as



with the objects corresponding to the circles. The presheaf on this category that corresponds to $G \to F$ is given by the $\int^F \mathsf{C}$-shaped diagram in Set



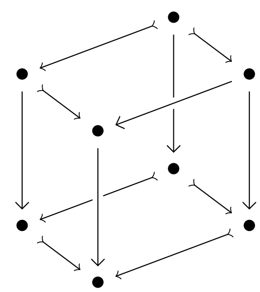
where the arrows are labeled to suggest the function they represent. The sets in this diagram are given by the fibers of θ. The edge and node functors determined by the arrows contain the information about where Gs and Gt send the elements in the fibers.

We have now finished the topos theory needed for this thesis. The remaining discussion morally belongs to the theory of rewriting and, in particular, adhesive category theory. However, because all topoi are adhesive and we restrict our attention to topoi, we place the discussion in here.

The following two lemmas are used.

Lemma 116 ([42, Lem. 4.2-3])**.** In a topos, monomorphisms are stable under pushout. Also, pushouts along monomorphisms are pullbacks.

Lemma 117 ([42, Lem. 6.3])**.** In a topos, consider a cube



whose top and bottom faces consist of only monomorphisms. If the top face is a pullback and the front faces are pushouts, then the bottom face is a pullback if and only if the back faces are pushouts.

Two properties that are desirable for rewriting systems are local Church–Rosser and concurrency. We do not use these results in this thesis, so we choose to not discuss them. Instead, we point the reader to the existing literature [25, 42]

Bibliography

[1] S. Abramsky, B. Coecke. A categorical semantics of quantum protocols. Logic in Computer Science. Proceedings of the 19th Annual IEEE Symposium. 2004. Also available at arXiv:quant-ph/0402130.

[2] S. Awody. Category Theory. Oxford University Press. 2010.

[3] F. Baader, T. Nipkow. Term Rewriting and All That. Cambridge University Press. 1999.

[4] M. Backens. Completeness and the ZX-Calculus. 2016. Available as arXiv:1602.08954.

[5] J. Baez, K. Courser. Structured cospans. *In preparation.*

[6] J. Baez, K. Courser. Course-graining Markov processes. 2017. Available as arXiv:1710.11343.

[7] J. Baez, B. Fong, B. Pollard. A compositional framework for Markov processes. J. Math. Phys. Vol. 57, No. 3. 2016. Also available as arXiv:1508.06448.

[8] J. Baez, B. Fong, B. Pollard. A compositional framework for passive linear networks. 2015. Available as arXiv:1504.05625.

[9] J. Baez, B. Pollard. A compositional framework for reaction networks. Rev. Math. Phys. Vol. 29, No. 09. 2017. Also available as arXiv:1504.02051.

[10] K. Bar, A. Kissinger, J. Vicary. Globular: an online proof assistant for higher-dimensional rewriting. 1st International Conference on Formal Structures for Computation and Deduction. LIPIcs. Leibniz Int. Proc. Inform. Vol. 52. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern. 2016. http://globular.science

[11] J. Bénabou. Introduction to bicategories. Reports of the Midwest category theory seminar. Springer, Berlin, Heidelberg. 1967.

[12] H. Bernstein, H. Byerly, F. Hopf, R. Michod, G.K. Vemulapalli. The Darwinian Dynamic. The Quarterly Review of Biology. Vol. 58, No. 2, Pp. 185–207. 1983.

[13] A. Carboni, R. Walters. Cartesian bicategories, I. J. Pure Appl. Algebra. Vol. 49, No. 1-2, Pp. 11-32. 1987.

[14] N. Chomsky. Syntactic Structures. Walter de Gruyter. 2002.

[15] D. Cicala. Spans of Cospans. Theory Appl. Categ. Vol. 33, No. 6, Pp. 131-147. 2018. Available at http://www.tac.mta.ca/tac/volumes/33/6/33-06.pdf.

[16] D. Cicala, K. Courser. Spans of cospans in a topos. Theory Appl. Categ. Vol. 33, No. 1, Pp. 1-22. 2018. Available at http://www.tac.mta.ca/tac/volumes/33/1/33-01.pdf

[17] D. Cicala. Rewriting structured cospans. *In preparation.*

[18] D. Cicala. Categorifying the ZX-calculus. 14th International Conference on Quantum Physics and Logic. Electron. Proc. Theor. Comput. Sci. Vol. 266, Pp. 294-314. 2018. Also available as arXiv:1704.07034.

[19] B. Coecke, R. Duncan. Interacting quantum observables. Automata, languages and programming. Part II. Lecture Notes in Comput. Sci. Vol. 5126. Springer, Berlin. 2008. Also available at https://ora.ox.ac.uk

[20] B. Coecke, B. Edwards, R. Spekkens. Phase groups and the origin of non-locality for qubits. Electron. Notes Theor. Comput. Sci. Vol. 270, No. 2. 2011. Also available as arXiv:1003.5005.

[21] B. Coecke, D. Pavlovic. Quantum measurements without sums. Mathematics of quantum computation and quantum technology. Chapman & Hall. Appl. Math. Nonlinear Sci. Ser. 2008. Also available at arXiv:0608035,

[22] B. Coecke, D. Pavlovic, J. Vicary. A new description of orthogonal bases. Math. Structures Comput. Sci. Vol. 23, No. 3. 2013. Also available at arXiv:0810.0812.

[23] B. Coecke, S. Perdrix. Environment and classical channels in categorical quantum mechanics. Log. Methods Comput. Sci. Vol. 4, No. 4. 2012. Also available as arXiv:1004.1598

[24] B. Coecke, R. Duncan. Interacting quantum observables: categorical algebra and diagrammatics. New J. Phys. Vol. 13. 2011. Also available as arXiv:0906.4725

[25] A. Corradini, H. Ehrig, R. Heckel, M. Loewe, U. Montanari. Algebraic Approaches to Graph Transformation, Part I: Basic Concepts and Double Pushout Approach. University of Pisa. 1996. Available at http://www.ncstrl.org:8900/ncstrl/servlet/search?formname=detail&id=oai%3Ancstrlh%3Aunipi_it%3Ancstrl.unipi_it%2F%2FTR-96-17.

[26] V. Danos, E. Kashefi, P. Panangaden. The measurement calculus. J. ACM. Vol. 54, No. 2. 2007. Also available as arXiv:0704.1263

[27] L. Dixon, R. Duncan, A. Kissinger. Open graphs and computational reasoning. 2010. Available as arXiv:1007.3794

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

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

[30] R. Duncan, S. Perdrix. Graph states and the necessity of Euler decomposition. Mathematical theory and computational practice. Lecture notes in Comput. Sci. Vol. 5635. Springer, Berlin. 2009. Also available as arXiv:0902.0500

[31] R. Duncan, S. Perdrix. Rewriting measurement-based quantum computations with generalised flow. Automata, Languages, and Programming. Springer. 2010. Also available at http://citeseer.ist.psu.edu

[32] H. Ehrig, A. Habel, H.J. Kreowski, F. Parisi-Presicce. From graph grammars to high level replacement systems. In *4th Int. Workshop on Graph Grammars and their Application to Computer Science. Lect. Notes Comput. Sc.* Vol. 532, Pp. 269–291. Springer Verlag. 1991.

[33] H. Ehrig, A. Habel, H.J. Kreowski, F. Parisi-Presicce. Parallelism and concurrency in high-level replacement systems. *Math. Struct. in Comp. Science.* Vol. 1. 1991.

[34] H. Ehrig, M. Pfender, H.J. Schneider. Graph-grammars: An algebraic approach. In *Switching and Automata Theory, 1973. SWAT'08. IEEE Conference Record of 14th Annual Symposium.* Pp. 167–180. IEEE. 1973.

[35] F. Gadducci, R. Heckel. An inductive view of graph transformation. *International Workshop on Algebraic Development Techniques.* Pp. 223–237. Springer. 1998. Also available at academia.edu

[36] R. Grandis, M. Paré. Intercategories. Theory Appl. Categ. Vol. 30, Pp. 1215-1255. 2015. Also available as arXiv:1412.0144.

[37] A. Habel, J. M uller, D. Plump. Double pushout graph transformations revisited. Math. Structures Comput. Sci. Vol. 11, No. 5, Pp. 637–688. 2001. Also available at researchgate.net

[38] P. Johnstone. Sketches of an Elephant: A Topos Theory Compendium. Oxford University Press. 2002.

[39] A. Joyal, R. Street. The geometry of tensor calculus. Adv. Math. Vol. 88, No. 1. 1991.

[40] A. Kissinger, V. Zamdzhiev. Quantomatic: a proof assistant for diagrammatic reasoning. Automated deduction—CADE 25. Lecture Notes in Comput. Sci. Vol. 9195. Springer. 2015. Also available as arXiv:1503.01034

[41] A. Kissinger. Pictures of processes: automated graph rewriting for monoidal categories and applications to quantum computing. Ph.D. Thesis. University of Oxford. Available as arXiv:1203.0202.

[42] S. Lack, P. Sobocinski. Adhesive categories. International Conference on Foundations of Software Science and Computation Structures. Pp. 273–288. Springer, Berlin. 2004. Also available at https://link.springer.com/

[43] S. Lack, P. Sobocinski. Toposes are adhesive. International Conference on Graph Transformations. Lecture Notes in Comput. Sci. Vol. 4178, Pp. 184–198. 2006. Also available at http://citeseerx.ist.psu.edu/

[44] W. Lawvere, S. Schanuel. Conceptual Mathematics: a First Introduction to Categories. Cambridge University Press. 2009.

[45] W. Lawvere. Functorial semantics of algebraic theories. Proc. Nat. Acad. Sci. U.S.A. Vol. 50, No. 3, Pp. 869–872. 1963. Available at https://www.ncbi.nlm.nih.gov

[46] S. MacLane. Categories for the Working Mathematician. Graduate Texts in Mathematics. Springer-Verlag. New York. 1998.

[47] S. MacLane, I. Moerdijk. Sheaves in geometry and logic: A first introduction to topos theory. Springer-Verlag. New York. 1994.

[48] T. Mailund. Functional Programming in R: Advanced Statistical Programming for Data Science, Analysis, and Finance. Apress. 2017.

[49] J. Master. Open Petri nets. Available as arXiv:1808.05415.

[50] S. Mukherjee, R. Nateghi. A Data-Driven Approach to Assessing Supply Inadequacy Risks Due to Climate-Induced Shifts in Electricity Demand. Risk Analysis. 2018. Available at https://onlinelibrary.wiley.com

[51] M. Nielson, I. Chuang. Quantum computations and quantum information. Cambridge University Press. 2000.

[52] H.T. Odum. Systems Ecology; An Introduction. John Wiley and Sons. New York. 1983.

[53] D. Pavlovic. Quantum and classical structures in non-deterministic computation. Quantum Interaction. Lecture Notes in Comput. Sci. Vol. 5494. Springer, Berlin. 2009. Also available as arxiv:0812.2266.

[54] R. Penrose. Applications of negative dimensional tensors. Combinatorial Mathematics and its Applications. Proc. Conf. Academic Press, London. 1971. Available at http://homepages.math.uic.edu/.

[55] P. Pstrągowski. On dualizable objects in monoidal bicategories, framed surfaces and the Cobordism Hypothesis. Available as arXiv:1411.6691.

[56] E. Riehl. Category Theory in Context. Courier Dover Publications. 2017. Available at http://math.rochester.edu/

[57] P. Selinger. A survey of graphical languages for monoidal categories. New structures for physics. Lecture Notes in Phys. Vol. 813. Springer, Heidelberg. 2011. Also available as arXiv:0908.3347.

[58] M. Shulman, Constructing symmetric monoidal bicategories. Available as arXiv:1004.0993.

[59] M. Stay, Compact closed bicategories. *Theory Appl. Categ*. Vol. 31, Pp. 755-798 2016. Available as arXiv:1301.1053.

[60] G. Wraith. Artin gluing. *J. Pure Appl. Algebra*. Vol. 4, Pp. 345–348. 1974.