Willoughby Seago

Theoretical Physics

Categories and Quantum Informatics

January 17, 2023

Course Notes

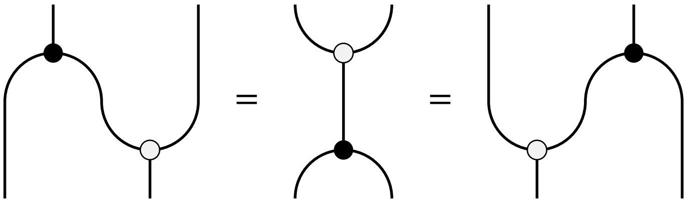
Categories and Quantum Informatics

Willoughby Seago

January 17, 2023

These are my notes from the course categories and quantum informatics. I took this course as a part of the theoretical physics degree at the University of Edinburgh.

These notes were last updated at 22:37 on April 22, 2023. For notes on other topics see https://github.com/WilloughbySeago/Uni-Notes.



Chapters

Page

Contents

Page

List of Figures

Page

One

Introduction

The material delivered in the lectures is included in these notes, but so is extra material pulled from the textbook [1], as well as other sources, such as [2]. I include various additional examples, some which require some mathematical background to understand. Some definitions are included in the appendices, but those unfamiliar with the material in these examples should feel free to just skip them. I've also included pointers to notes from other courses where appropriate, particularly the courses *Symmetries of Quantum Mechanics* and *Symmetries of Particles and Fields*, which both cover the areas of representation theory and Lie theory. The notes from these courses and others can be found at `https://github.com/WilloughbySeago/Uni-Notes`. Again, any unfamiliar material from these courses can be skipped. A fair few of the examples simply come from relevant Wikipedia pages, and I haven't performed detailed checks of all the facts in these cases.

There are a few notational things which don't align with the course. A big one is leaving out brackets for functors, writing FA and Ff instead of $F(A)$ and $F(f)$. The use of $-$ as a blank to be filled in with some object is also not used in the course.

Part I

Introduction

Two

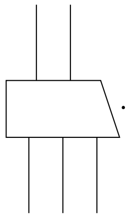
ZX Calculus

In this chapter we will introduce **ZX calculus**. This is a diagramatic notation for performing calculations. ZX calculus is mathematically rigorous, and developing the maths explaining this is a large part of this course. ZX calculus provides a higher level of abstraction that a quantum circuit, focusing less on implementation and more on what the circuit is doing. ZX calculus is built from a relatively small number of building blocks. It is the freedom we have in combining these that makes ZX calculus so powerful.

We'll introduce ZX calculus in a seemingly backwards manner, first introducing which sorts of diagrams we can have, then how to manipulate the diagrams then what the diagrams mean.

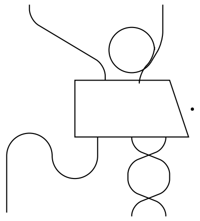
2.1 Types of Diagrams

A diagram in ZX calculus is somewhat like a flowchart. The playing field is the two-dimensional page. We imagine that time goes upwards and space extends to the left and right. This means that a process described by a ZX calculus starts by entering the bottom of the diagram and ends when we leave the top of the diagram. Qubits are represented by wires, which are just lines. Processes are represented by boxes, for now we won't focus on what the process might be. The following diagram represents a process which takes in three qubits and produces two qubits:



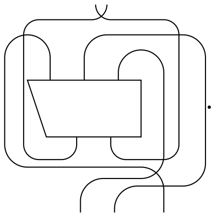
(2.1.1)

In diagrams it isn't important exactly how we draw the wires, so long as they are connected in the same way, so in the same order both on the box and along the top and bottom, the diagram corresponds to the same equation. For example, the following is equivalent to the previous diagram



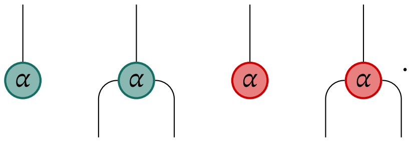
(2.1.2)

We are also free to change the orientation of the box, so long as the the connectivity stays the same. This is why we draw the box as a trapezium without rotational symmetry. For example, the following diagram is equivalent to both of the previous diagrams.



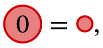
(2.1.3)

A sensible question to ask now is what process does this box represent. We'll get to this. For now we'll just say that the process can be built up of fundamental process. There are four processes which we use to build any diagram in ZX calculus. They are

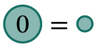


(2.1.4)

Actually, α can take any value in $[0, 2\pi)$, so there are really an uncountable number of these building blocks. For short if the phase is zero then we omit the label:



and



(2.1.5)

We call these **spiders**. In particular, the green is a Z spider and the red is an X spider.

Combining these pieces we can quickly build up fairly complex diagrams. For example, the diagram in Figure 2.1 is a process which takes in two qubits and outputs two qubits.

2.2 Simplifying Diagrams

There are two types of rules by which we might manipulate diagrams. The first, which we've already seen, are **graphical rules** which allow us to move different pieces around so long as we don't change the connectivity. More formally two diagrams are equivalent if the are isotopic as graphs, a concept we'll make precise later, but for now two diagrams are isotopic if fixing all of the inputs and outputs

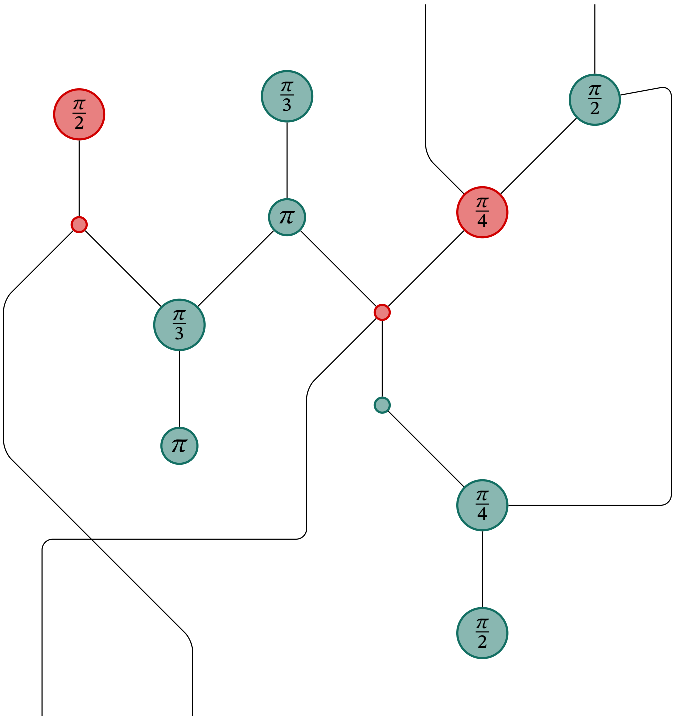
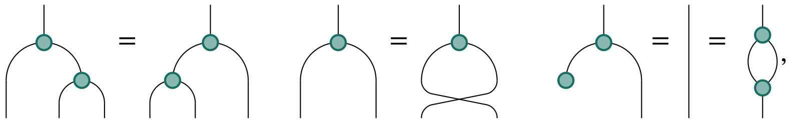


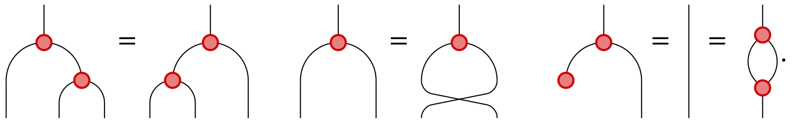
Figure 2.1: A diagram in ZX calculus taking two qubits to two qubits.

as well as the points at which they connect it is possible to continuously morph one into the other. We allow the wires to pass through each other in this process.

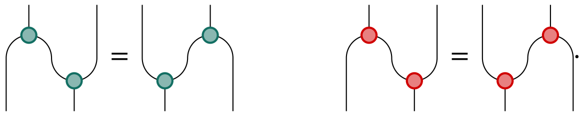
The second type of rule corresponds to specific properties of the basic building blocks. There are quite a few of these, and for now we'll just list them without much explanation. First we have the **monoid rules** which are



and the same for the other colour:

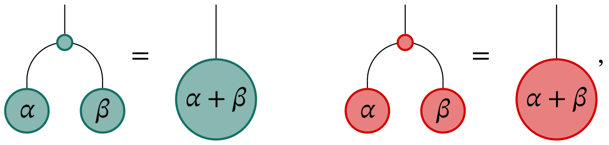


The next set of rules are called the **Frobenius rules**, they are



(2.2.1)

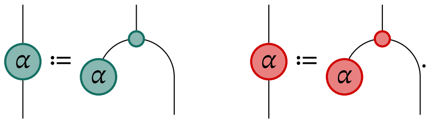
The **fusion rules** allows us to combine multiple nodes of the same colour in some circumstances:



(2.2.2)

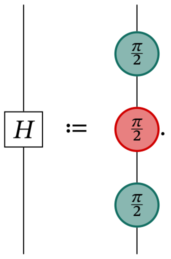
where addition is taken modulo 2π.

Before introducing the next set of rules we introduce some shorthand notation:



(2.2.3)

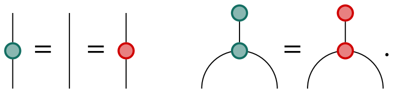
We also define the **Hadamard**:



(2.2.4)

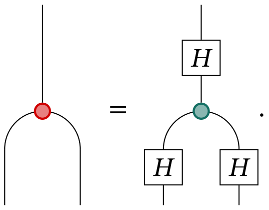
It's safe to give this a square symbol, with rotational symmetry, since we can see from the definition that it is rotationally symmetric.

We then have two **identity rules**, the first is just a repeat of one of the monoid rules, but now in this new shorthand, the second is new:



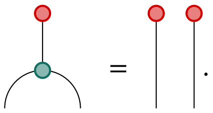
(2.2.5)

The next rule allows us to change the colour of a node, at the cost of some Hadamards, it is appropriately called the **colour change rule**:



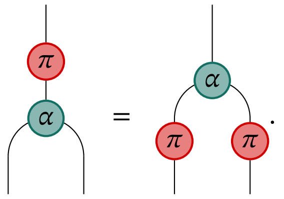
(2.2.6)

The next rule is called the **copy rule**, since it allows us to make two diagrams out of one:



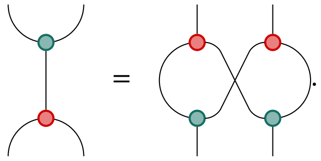
(2.2.7)

Our next rule allows for an X spider with a phase of π to be copied pulling it through a Z spider. It is called the **π-copy rule**:



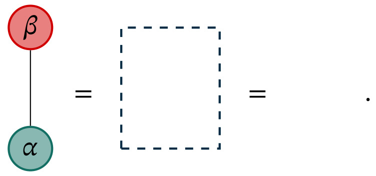
(2.2.8)

The next rule is called the **bialgebra rule**:



(2.2.9)

The final rule is rather simple, it's simply that we can ignore overall factors, called the **scalar rule**, it corresponds to the following:



(2.2.10)

Here the dashed box as well as the empty space both represent the empty diagram, which is simply the trivial identity process taking in no qubits, doing nothing, and outputting no qubits.

2.3 Interpretation

We'll see in more detail what these rules mean, where they come from, and why they have the names they do. For now it is enough to know that combined the monoid rules, Frobenius rules, fusion rules, and identity rules tell us that it doesn't matter how the dots of the same colour are connected, so long as the phases in the dots add to the same value modulo 2π.

We can represent a qubit as an element of $\mathbb{C}^2$. Then the rules about Z spiders tell us how to multiply matrices which are diagonal in the computational basis, $\{|0\rangle, |1\rangle\}$, with eigenvalues $\mathrm{e}^{i\alpha}$, and the rules about X spiders tell us how to multiply matrices which are diagonal in the Hadamard transformed basis formed from

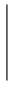
$$|+\rangle = H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \qquad \text{and} \qquad |-\rangle = H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{2.3.1}$$

The colour change rule tells us how to convert one basis to the other. The bialgebra rule tells us that these bases are complimentary, that they are at the maximal angle to each other. The copy and π-copy rules are just artefacts of nicely chosen bases.

Using this we can develop the **standard model** of ZX calculus, which represents each diagram as a matrix acting on the input qubits. More formally we can define a map

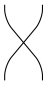
$$[\![-]\!] : (n\text{-to-}m \text{ qubit ZX diagram}) \to (2^n \times 2^m \text{ complex matrices}). \tag{2.3.2}$$

Then a process which takes a single qubit, does nothing to it, and immediately outputs it is represented as


$$\longmapsto \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}. \tag{2.3.3}$$

Unsurprisingly doing nothing to a qubit gives the identity.

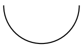
The following diagram takes in two qubits, swaps them, and then returns them:


$$\longmapsto \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}. \tag{2.3.4}$$

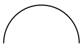
In terms of matrices this acts as follows:

$$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\left[\begin{pmatrix}a\\ b\end{pmatrix}\otimes\begin{pmatrix}c\\ d\end{pmatrix}\right] = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}ac\\ ad\\ bc\\ bd\end{pmatrix} = \begin{pmatrix}ac\\ bc\\ ad\\ bd\end{pmatrix} = \begin{pmatrix}c\\ d\end{pmatrix}\otimes\begin{pmatrix}a\\ b\end{pmatrix}. \tag{2.3.5}$$

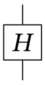
It is possible to have diagrams which create qubits from nothing. In this case we should take the input to simply be 1. The following diagram creates a pair of qubits:


$$\longmapsto \begin{pmatrix}1\\ 0\\ 0\\ 1\end{pmatrix}. \tag{2.3.6}$$

Similarly we can destroy qubits:

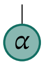

$$\longmapsto \begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix}. \tag{2.3.7}$$

The Hadamard gives the **Hadamard matrix**, note that we're ignoring an overall scalar, there's usually a factor of $1/\sqrt{2}$:

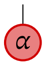


$$\longmapsto \begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}. \tag{2.3.8}$$

We can create a single qubit:

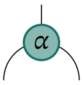


$\longmapsto \begin{pmatrix}1\\ e^{i\alpha}\end{pmatrix},$ and

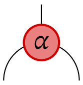


$$\longmapsto \begin{pmatrix}1+e^{i\alpha}\\ 1-e^{i\alpha}\end{pmatrix} \tag{2.3.9}$$

The three-arity spiders give the following matrices

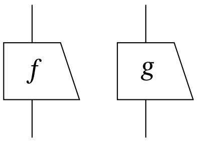


$$\longmapsto \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 0 & e^{i\alpha}\end{pmatrix}, \tag{2.3.10}$$



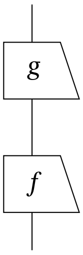
$$\longmapsto \begin{pmatrix}1+e^{i\alpha} & 1-e^{i\alpha} & 1-e^{i\alpha} & 1+e^{i\alpha}\\ 1-e^{i\alpha} & 1+e^{i\alpha} & 1+e^{i\alpha} & 1-e^{i\alpha}\end{pmatrix} \tag{2.3.11}$$

Writing two processes next to each other gives their tensor product:



$$\longmapsto f \otimes g \qquad (2.3.12)$$

Writing two processes one after the other connected up represents doing them in the order they are connected from bottom to top, which is composition:



$$\longmapsto g \circ f \qquad (2.3.13)$$

Note that for processes represented by matrices composition is just matrix multiplication.

This mapping makes precise what any one ZX diagram represents. What is important is that this isn't changed when we apply the rules of ZX calculus. This is the crux of the following theorem.

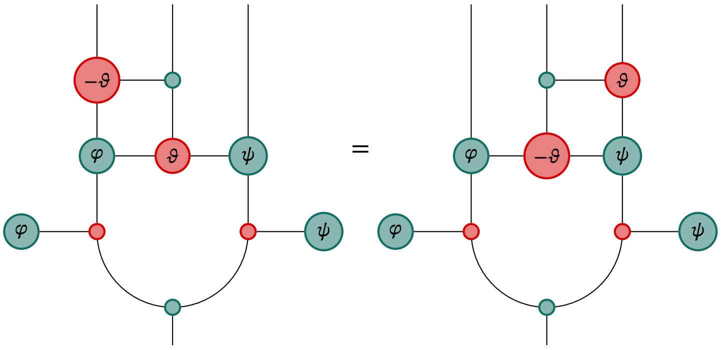
Theorem 2.3.14 — ZX Calculus is Sound. Let D_1 and D_2 be diagrams in ZX calculus. If $D_1 = D_2$ according to the rules of ZX calculus then $[\![D_1]\!] = [\![D_2]\!]$.

As well as being a rigorous way to manipulate objects ZX calculus can also approximate any process from m qubits to n qubits to arbitrary precision.

Theorem 2.3.15 — ZX Calculus is Approximately Universal. For any $2^m \times 2^n$ matrix, f, and any error margin, $\varepsilon > 0$, there exists a diagram, D, in ZX calculus built only from terms with phases an integer multiple of $\pi/4$ such that $\|[\![D]\!] - f\| < \varepsilon$ for some appropriate matrix norm $\|-\|$.

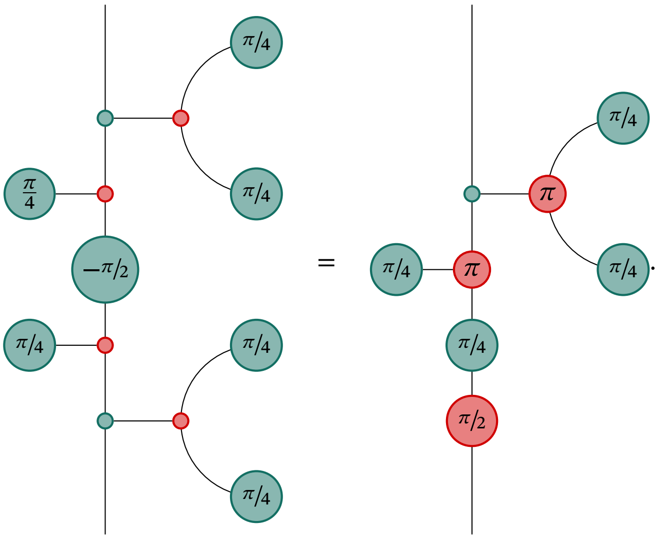
This is one of the reasons that ZX calculus is so powerful.

Another desirable quality for a notation like ZX calculus is that it be complete. By this we mean that if two matrices are equal and both given by some ZX diagram then there should be a graphical proof of this using only the rules of ZX calculus. This is the case if we assume the following two axioms, which are sound under the standard interpretation:



(2.3.16)

for any phases φ, ψ, and ϑ which are integer multiples of $\pi/4$, and the second axiom



(2.3.17)

Call ZX calculus with these rules added **π/4-ZX calculus**.

Theorem 2.3.18 — $\pi/4$-ZX Calculus is Complete. Let D_1 and D_2 be diagrams in $\pi/4$-ZX calculus. If $[\![D_1]\!] = [\![D_2]\!]$ then $D_1 = D_2$ under the axioms of $\pi/4$-ZX calculus.

The third thing making ZX calculus powerful is how well it can be automated. All a ZX calculation is is a finite labelled graph. Once you've implemented a way of applying the rules this can then be done very efficiently on a computer.

A common use of ZX calculus is in quantum circuit optimisation. Given some quantum algorithm as a quantum circuit it is often possible to optimise the circuit. For example, the T gate,

$$T = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{-i\pi/4} \end{pmatrix}, \tag{2.3.19}$$

is typically expensive to implement, so reducing the number of T gates in a circuit is usually desirable. Given some circuit making use of T gates we can use the universality of ZX calculus to convert the circuit into a ZX diagram, then manipulate the ZX diagram and then convert it back to a, hopefully, more optimised circuit with fewer T gates.

Three

Semantics

3.1 Types of Semantics

Consider the two following pseudocode fragments:

$$P = \text{if } 1 = 1 \text{ then F else G}, \tag{3.1.1}$$

$$Q = \text{if } 1 = 0 \text{ then F else F}. \tag{3.1.2}$$

Are P and Q the same program? There are two schools of thought:

- No. Clearly looking at them both programs are implemented differently, P makes reference to G, Q makes reference to 0.
- Yes. Both programs take no input and output F.

Whether or not we count these as the same program depends on what we are interested in.

To aid in our analysis we assign the code fragments their meanings, encoded in some appropriate mathematical object. This gives a mapping

$$[\![-]\!] : \text{Programs} \to \text{Mathematical Objects}. \tag{3.1.3}$$

For now we'll leave the details of exactly what mathematical objects alone. If we are interested in implementation details then we assign P and Q to objects encoding these details. This is called **operational semantics**. If we aren't interested in implementation details then we assign P and Q to objects which treat them as black boxes with inputs and outputs. This is called **denotational semantics**. If this is what we do then we find that $[\![P]\!] = [\![Q]\!] = [\![\text{F}]\!]$.

We want this mapping to preserve the structure of our programs. For example, suppose that we have two processes, F and G, which can be composed by running them one after another. We might write this as F; G in a language using semicolons to terminate a line. In order to reason about our program, regardless of which type of semantics we are interested in, we want the result to be the same if we compose the programs and then look at the semantics or look at the semantics and then compose the programs. That is we want

$$[\![\text{F; G}]\!] = [\![\text{F}]\!] \circ [\![\text{G}]\!]. \tag{3.1.4}$$

Here $\circ$ is some method of composing the semantics of two programs. Another structure which we may want to preserve is the ability to compute things in parallel, for example

$$[\![\text{paralell (F, G)}]\!] = [\![\text{F}]\!] \otimes [\![\text{G}]\!]. \tag{3.1.5}$$

3.2 Motivation

Why might we care about this sort of analysis? This reasoning can be used to ground assumptions and correct erroneous assumptions. We can also use semantics to justify transformations of programs, for example, to demonstrate that a compiled program does the same thing as the original program. It is also often the case that it is easier to reason about the mathematical objects encoding the programs, rather than the programs themselves, in fact sometimes it isn't possible to reason directly about the programs, such as in quantum computing where many operations are like black boxes, even if we can't look at the operational semantics we can still consider the denotational semantics and the flow of information through the program to compare programs.

Choosing different semantics also allows us to focus on different details. Operational semantics focus on implementation details, such as memory usage and running time, which is useful if we want to improve the efficiency of our programs. Denotational semantics focus on results, which is useful if we want to check that our program does what we want.

3.3 Mathematical Objects

There are many possible mathematical objects to encode programs. Simple programs can be modelled as set theoretical functions from some set of possible inputs to some set of possible outputs. More specifically we can use something like λ-calculus to represent programs. In this way we can represent both operational semantics, by writing our functions as expressions in the input variables, and denotational semantics, by focussing on the input and output values.

In quantum computing operational semantics are often not an option. So we will mostly stick to denotational semantics. The objects we choose to represent programs are categories. Category theory supports combing programs as we saw in the examples above and gives us a powerful graphical language for computations.

Part II

Categories

Four

Categories

4.1 Motivation

We want an abstract mathematical formalism in which the meaning of a program lives, allowing us to abstract away implementation details and reason about computations. Such a formalism provides a map

$$[\![-]\!] : \text{programs} \to \text{mathematical objects}. \tag{4.1.1}$$

There are several features which are desirable of such a formalism, including but not limited to,

- a notion of composition, if F and G are programs and $F;\ G$ is running F then G then we should have $[\![F;\ G]\!] = [\![G]\!] \circ [\![F]\!]$ where $\circ$ represents composition in this mathematical formalism;
- a notion of concurrency, if $F\ \mathrm{par}\ G$ corresponds to running F and G at the same time then we should have $[\![F\ \mathrm{par}\ G]\!] = [\![F]\!] \otimes [\![G]\!]$;
- a notion of calling programs recursively, if X is some code calling F then we should be able to compute $F(X)$, and this requires our mathematical formalism to have structures of the form $[\![F(X)]\!] = [\![F]\!]([\![X]\!])$.

More concisely, our formalism should preserve composition, concurrency, and function application.

The question we have to ask is what sort of mathematical objects we're considering. There are multiple options, each with their own advantages and disadvantages. Some common options are

- λ-calculus is an algebraic notation for functions. It is similar in syntax to *Haskell*. It works with anonymous functions, or lambda functions, such as the function $\lambda\ x.\ *\ 2\ x$ which takes an argument, x, and multiplies it by 2, using prefix notation. Compare this to the *Haskell* function

  ```
  timesThree x = (*) 2 x
  ```

  This choice is good for analysing implementation details and is simply a precise notation for applying functions, a common mathematical operation.
- Partially ordered sets, or posets, where the elements of the poset are partially completed calculations, any two elements are comparable if they are the

same calculation at, potentially, different levels of completion, and the more complete calculation is greater. This choice is good for analysing computations step by step without worrying about how each step is implemented.

- Categories, which is what we'll use. This option subsumes both λ-calculus, as a method of defining functions, and posets, which can be regarded as a special case of a category.

Our goal will be to work in a general category to develop theory, imposing only the required restrictions for things to work out. Then we can pick a particular category to analyse our work in, with the interpretation depending on the category we pick. Often we can work in a generic category and then specialise the result to a category to perform either classical or quantum computations.

4.2 Categories: The Idea

A category consists of two pieces of data:

- Objects, $A, B, C, \ldots$;
- Morphisms, $f : A \to B$, between objects.

Given some specific category there are various ways to think of computations occurring in this category. Some examples are given here:

- We can think of the objects as physical systems and the morphisms as processes. For example,
  - Two objects may be a full cup and an empty cup and a process may be drinking the drink or making a new drink.
  - Two objects might be a plate and pieces of broken pottery and a process may be dropping the plate on the floor.
- We can think of objects as data types, and morphisms as functions between these types. Borrowing *Haskell* notation some examples are
  - One object might be `Int`, and a morphism `f :: Int →Int` defined by `f n = 2 ∗ n`.
  - Another object might be `String`, and a morphism `len :: String −> Int` defined by

    ```
    len [] = 0
    len (x : xs) = 1 + len xs
    ```

  - Another object might be `Num a ⇒[a]`, and a morphism

    ```
    mySum :: (Num a) ⇒ [a] → a
    mySum [] = 0
    mySum (x : xs) = x + mySum xs
    ```

- Objects are algebraic structures and morphisms are structure preserving maps. For example,
  - Objects are sets and morphisms are functions.

  - Objects are groups and morphisms are homomorphisms.
  - Objects are topological spaces and morphisms are continuous functions.
  - Objects are vector spaces and morphisms are linear maps.

- Objects are logical propositions and morphisms are implications between them. For example, we could take the objects "it rains" and "I get wet" and then we might have a morphism "it rains" $\implies$ "I get wet". But what if we're indoors? We might introduce another object, "I am outside", and then we may have a morphism "it rains" $\wedge$ "I am outside" $\implies$ "I get wet". Here we've implicitly defined another object "it rains" $\wedge$ "I am outside" using logical conjunction, $\wedge$. This is actually an example of a product in this category, we'll see what this means later.

It turns out that the second and fourth examples, programs/algorithms and propositions/implications are actually the same! This is known as the Curry–Howard isomorphism, and allows us to write proofs as programs and vice versa.

The mindset that one should have when doing category theory is

> Morphisms are more important than objects.

This might seem backwards at first, for example we spend a lot of time thinking about groups, and homomorphisms are only one aspect that we consider, but it turns out that we can learn a lot about groups by studying how they relate to other groups, and this is done through homomorphisms. This mindset is particularly useful for cases such as quantum computing where we *can't* look at internal structure, and can only look at how systems relate to each other.

4.3 Categories: The Definition

Categories are objects and morphisms. The definition of a category simply states what we mean by this and the properties that morphisms are expected to have.

Definition 4.3.1 — Category A **category**, C, consists of the following data

- a collection of **objects**, $\mathrm{Ob}(\mathsf{C})$ (often denoted $\mathrm{Obj}(\mathsf{C})$ or simply C);
- for every pair of objects, $A, B \in \mathrm{Ob}(\mathsf{C})$ a collection of **morphisms** (also known as **maps** or **arrows**), $\mathsf{C}(A, B)$ (often denoted $\hom_{\mathsf{C}}(A, B)$, $\mathrm{Mor}_{\mathsf{C}}(A, B)$, possibly without the subscript C when the category is clear, this collection is often called a **hom set**), where for $f \in \mathsf{C}(A, B)$ we write $f : A \to B$ or $A \xrightarrow{f} B$;
- a map $\circ : \mathsf{C}(B, C) \times \mathsf{C}(A, B) \to \mathsf{C}(A, C)$ which assigns to each $f : A \to B$ and $g : B \to C$ some **composite** $(g \circ f) : A \to C$;
- for every object $A \in \mathrm{Ob}(\mathsf{C})$ a morphism $\mathrm{id}_A : A \to A$, that is $\mathrm{id}_A \in \mathsf{C}(A, A)$, called the **identity morphism**.

This data is subject to the following conditions:

- **associativity** of $\circ$: for all objects $A, B, C, D \in \mathrm{Ob}(\mathbf{C})$ and for all morphisms $f : A \to B$, $g : B \to C$, and $h : C \to D$ we have

$$h \circ (g \circ f) = (h \circ g) \circ f, \tag{4.3.2}$$

so we can unambiguously write $h \circ g \circ f$ for both of these;

- **identity** law: for all objects $A, B \in \mathrm{Ob}(\mathbf{C})$ and for all morphisms $f : A \to B$ we have

$$f \circ \mathrm{id}_A = f = \mathrm{id}_B \circ f. \tag{4.3.3}$$

4.3.1 Technicality

Notice that in the definition we use the word "collection". It is tempting to replace this with "set", but this can cause issues. For example, the set of all sets is not a set, due to Russell's paradox. However, we will shortly see that we have categories where the objects are all sets, and so $\mathrm{Ob}(\mathbf{C})$ cannot be a set in this case. We aren't going to worry too much about these types of issues, and may erroneously refer to these collections as sets. A category is **small** if both $\mathrm{Ob}(\mathbf{C})$ and the collection of all morphisms between any two objects are sets. A category is **locally small** if for all objects A and B $\mathbf{C}(A, B)$ is a set. Many statements we make throughout will apply only to small, or more likely locally small categories.

4.4 Categories: The Examples

4.4.1 Set

Definition 4.4.1 — Set The category **Set** has sets as objects. A morphism $A \to B$ is simply a function from A to B. Composition of morphisms is composition of functions, defined for $f : A \to B$ and $g : B \to C$ by $(g \circ f) : A \to C$ given by $(g \circ f)(a) = g(f(a))$ for all $a \in A$. The identity morphism is the identity function, $\mathrm{id}_A : A \to A$, given by $\mathrm{id}_A(a) = a$ for all $a \in A$.

Lemma 4.4.2 **Set** is a category.

Proof. The collection $\mathbf{Set}(A, B)$ of functions $A \to B$ exists for all sets A and B. It will be empty if $B = \varnothing$ and $A \neq \varnothing$ which is allowed, if $A = \varnothing$ and $B = \varnothing$ then there is a unique function $f : \varnothing \to \varnothing$ which is also the identity on the empty set. Function composition is associative. Take sets A, B, C, and D, and morphisms $f : A \to B$, $g : B \to C$, and $h : C \to D$. Then

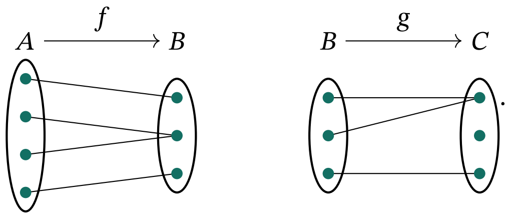
$$(h \circ (g \circ f))(x) = h((g \circ f)(x)) = h(g(f(x))) = (h \circ g)(f(x)) = ((h \circ g) \circ f)(x) \tag{4.4.3}$$

for all $x \in A$, so $h \circ (g \circ f) = (h \circ g) \circ f$. The identity function is then such that

$$(f \circ \mathrm{id}_A)(x) = f(\mathrm{id}_A(x)) = f(x) = \mathrm{id}_B(f(x)) = (\mathrm{id}_B \circ f)(x) \qquad (4.4.4)$$

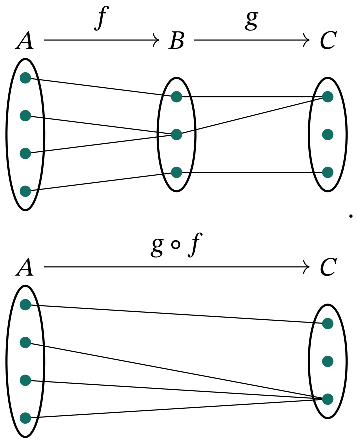
and so $f \circ \mathrm{id}_A = f = \mathrm{id}_B \circ f$. □

We can think of a function $f : A \to B$ as dynamically indicating how elements of A transform into elements of B. Pictorially, we can represent $f : A \to B$ and $g : B \to C$ as



(4.4.5)

Then composition is just following the lines between sets:



$A \xrightarrow{f} B \xrightarrow{g} C$
$A \xrightarrow{g \circ f} C$

(4.4.6)

As a process composition in set is just doing one thing and then the other.

Set is the prototypical category. It is often tempting to think of other examples of categories, which we'll meet shortly, as simply sets with extra structure, and indeed this is often, but not always, the case. A **concrete category** is a category in which all objects can be mapped to sets and morphisms can be mapped to functions between these sets. This mapping is done with something called a functor, which we'll define later (Definition 4.8.1).

We'll list some properties of **Set** here, some of which won't make sense until later.

- **Set** is a concrete category.
- **Set** has the empty set as an initial object, and the singleton as a terminal object.
- **Set** is complete and co-complete, in particular the product is the Cartesian product and the coproduct is the disjoint union.

- **Set** is a monoidal category with the monoidal product given by the Cartesian product.

4.4.2 Rel

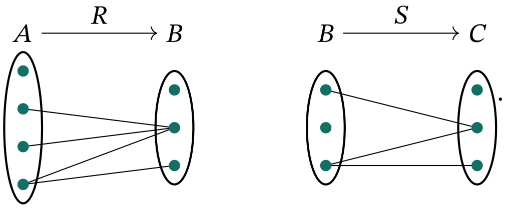
> **Definition 4.4.7 — Relation** Let A and B be sets. A **relation**, R, is a subset of $A \times B$. If $(a, b) \in R$ we write aRb.

Relations are morphisms in a category we will defined shortly (Definition 4.4.15), so for $R \subset A \times B$ we write $R \colon A \to B$. In a similar manner to functions between sets being dynamic transformations we can think of relations as being non-deterministic transformations, where each element can transform into multiple objects, or possibly don't map across at all. For example, if we take $A = \{a, b, c, d\}$, $B = \{1, 2, 3\}$, and $C = \{\alpha, \beta, \gamma\}$ then the relations

$$R = \{(b, 2), (c, 2), (d, 2), (d, 3)\} \subseteq A \times B, \qquad \text{and} \tag{4.4.8}$$

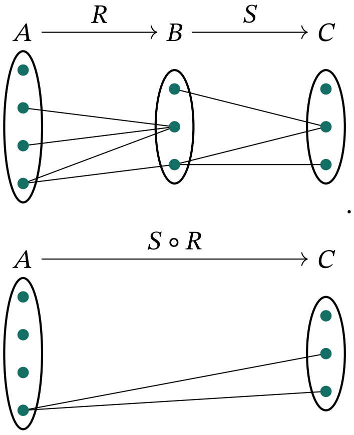
$$S = \{(1, \beta), (3, \beta), (3, \gamma)\} \subseteq B \times C \tag{4.4.9}$$

can be represented pictorially as



(4.4.10)

As with **Set** we then compose relations by joining up these lines:



(4.4.11)

That is,

$$S \circ R = \{(d, \beta), (d, \gamma)\} \subseteq A \times C. \tag{4.4.12}$$

This leads to the following definition.

Definition 4.4.13 — Relation Composition Let A, B, and C be sets with relations $R \subseteq A \times B$ and $S \subseteq B \times C$. Then the **composite relation** $S \circ R$ is the set

$$S \circ R := \{(a, c) \in A \times C \mid \exists b \in B \text{ such that } (a, b) \in R \text{ and } (b, c) \in S\}.$$

Now that we have composition we just need an identity, and after some playing around with the definition of composition one quickly comes to the identity

$$\mathrm{id}_A := \{(a, a) \in A \times A \mid a \in A\} \subseteq A \times A. \tag{4.4.14}$$

Now we can define a category.

Definition 4.4.15 — Rel The category **Rel** has sets as objects. A morphism $R\colon A \to B$ is a relation $R \subseteq A \times B$. Composition of morphisms is composition of relations, defined for $R\colon A \to B$ and $g\colon B \to C$ by

$$S \circ R = \{(a, c) \mid \exists b \in B : aRb \wedge bSc\} \subseteq A \times C. \tag{4.4.16}$$

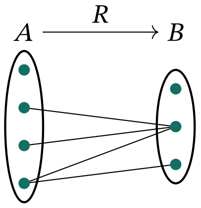
The identity morphism is the identity relation, $\mathrm{id}_A : A \to A$, given by

$$\mathrm{id}_A = \{(a, a) \mid a \in A\} \subseteq A \times A. \tag{4.4.17}$$

Lemma 4.4.18 **Rel** is a category.

Proof. The collection $\mathbf{Rel}(A, B)$ is simply the power set of $A \times B$. Importantly the empty set is a relation on any pair of sets, including the empty set, and the empty set is the identity relation on the empty set. So consider nonempty sets.
Composition of relations is associative. Let $R\colon A \to B$, $S\colon B \to C$, and $T\colon C \to D$ be relations. We want to show that $T \circ (S \circ R) = (T \circ S) \circ R$. To do so we will prove that each pair $(a, b) \in T \circ (S \circ R)$ is also an element of $(T \circ S) \circ R$. The converse, that each pair $(a, b) \in (T \circ S) \circ R$ is an element of $T \circ (S \circ R)$, follows by the same logic in reverse. So take some $(a, b) \in T \circ (S \circ R)$. By definition there exists some x such that $(x, b) \in T$ and $(a, x) \in S \circ R$. Thus, there exists some y such that $(y, x) \in S$ and $(a, y) \in S$. Since $(y, x) \in S$ and $(x, b) \in T$ we have that $(y, b) \in T \circ S$. Since $(a, y) \in R$ we have $(a, b) \in (T \circ S) \circ R$.
Let $R\colon A \to B$ be a relation and $\mathrm{id}_A = \{(a, a) \mid a \in A\}$ the identity relation. Then for all $(a, b) \in R$ we have $(a, a) \in \mathrm{id}_A$, meaning that $(a, b) \in R \circ \mathrm{id}_A$. Similarly for all $(a, b) \in R \circ \mathrm{id}_A$ we must have x such that $(x, b) \in R$, and $(x, a) \in \mathrm{id}_A$, which means that $x = a$ and so $(a, b) \in R$. Thus $R \circ \mathrm{id}_A = R$. Similarly $\mathrm{id}_B \circ R = R$. Hence the identity law is satisfied. □

We can represent relations as binary $|B| \times |A|$ matrices with a 1 in the (i, j) slot if $(a, b) \in R$, where a is the jth element of A and b the ith element of B in some arbitrary fixed ordering of A and B. Further this mapping is one-to-one, meaning each binary matrix also defines a representation. For example, indexing top to bottom we have



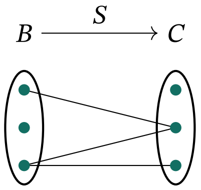
$$\leftrightsquigarrow M(R) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.4.19}$$

Composition of relations is then given by matrix multiplication taking everything mod 2. More formally we have a map

$$M\colon \mathcal{P}(A \times B) \to \mathcal{M}_{|B|\times|A|}(\mathbb{Z}_2) \tag{4.4.20}$$

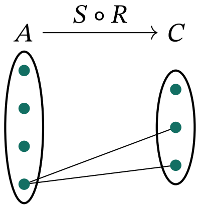
where $\mathbb{Z}_2 = \{0, 1\}$ has addition and multiplication defined mod 2.

As an example notice that we have



$$\leftrightsquigarrow M(S) = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.4.21}$$

and



$$\leftrightsquigarrow M(S \circ R) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.4.22}$$

The composite can then be calculated as

$$M(S \circ R) = M(S)M(R) = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.4.23}$$

This map, M, actually defines a functor (Definition 4.8.1) $M\colon$ **Rel** → $\mathbf{Mat}_{\mathbb{Z}_2}$, where $\mathbf{Mat}_{\mathbb{Z}_2}$ is the category of binary matrices, whose objects are natural numbers and a morphism $n \to m$ is an $n \times m$ matrix. The functor M then maps sets to their size, $A \mapsto M(A) = |A|$, and relations to matrices as described above.

This is actually an example of a more general idea of a representation[1], where we replace objects with matrices preserving the structure of the original space. These ideas can all be understood as functors $\mathbf{C} \to \mathbf{Mat}_{\Bbbk}$ for some appropriate category $\mathbf{C}$ and ring $\Bbbk$. This in turn is more commonly thought of as a map $\mathbf{C} \to \mathbf{FVect}_{\Bbbk}$ identifying matrices with linear maps. This allows for generalisations to infinite dimensional representations, $\mathbf{C} \to \mathbf{Vect}_{\Bbbk}$. We'll define $\mathbf{FVect}_{\Bbbk}$ and $\mathbf{Vect}_{\Bbbk}$ shortly (Definitions 4.4.28 and 4.4.35).

[1] see *Symmetries of Quantum Mechanics* or *Symmetries of Particles and Fields*

On the surface **Rel** seems quite similar to **Set**, after all the objects of both are the same. However, it's really the morphisms which are important, and these are quite different. The ability to replace relations with matrices means that **Rel** is actually quite similar to another category, **Hilb** (Definition 4.4.47). This makes **Rel** a nice in between for **Set** and **Hilb**, which turn out to be the categories in which we think of most classical and quantum computing as occurring in respectively.

We now list some properties of **Rel**:

- **Rel** is a concrete category.
- **Rel** is a dagger category.
- **Rel** has both products and coproducts given by disjoint union.
- **Rel** is a monoidal category with the monoidal product given by the Cartesian product.

4.4.3 $\mathbf{Vect}_{\Bbbk}$, $\mathbf{FVect}_{\Bbbk}$, **Hilb**, and **FHilb**

4.4.3.1 $\mathbf{Vect}_{\Bbbk}$ and $\mathbf{FVect}_{\Bbbk}$

Definition 4.4.24 — Vector Space A **vector space**, $(V, \Bbbk, +, \cdot)$, is a set, V, a field[a], $\Bbbk$, and two operations, **vector addition**, $+ : V \times V \to V$, and **scalar multiplication**, $\cdot : \Bbbk \times V \to V$, such that

- $(V, +)$ is an Abelian group, that is
  - vector addition is associative: $u + (v + w) = (u + v) + w$ for all $u, v, w \in V$;
  - vector addition is commutative: $u + v = v + u$ for all $u, v \in V$;
  - additive identity: there exists $0 \in V$ such that $0 + v = v$ for all $v \in V$;
  - additive inverse: for every $v \in V$ there exists $-v \in V$ such that $v + (-v) = v - v = 0$;
- scalar multiplication distributes over vector addition: $\alpha \cdot (u + v) = (\alpha \cdot v) + (\alpha \cdot v)$ for all $\alpha \in \Bbbk$ and $u, v \in V$;
- field identity acts as the identity: $1 \cdot v = v$ for all $v \in V$ where 1 is the multiplicative identity in $\Bbbk$;
- field addition distributes over scalar multiplication: $(\alpha + \beta) \cdot v = (\alpha \cdot v) + (\beta \cdot v)$;
- compatibility of field and scalar multiplication: $\alpha \cdot (b \cdot v) = (a \cdot_{\Bbbk} b) \cdot v$ for all $\alpha, \beta \in \Bbbk$ and $v \in V$ where $\cdot_{\Bbbk}$ is multiplication in the field.

[a] if you don't know what this is just replace it with $\mathbb{R}$ or $\mathbb{C}$ and don't worry about it

Definition 4.4.25 — Linear Map Let $(V, \Bbbk, +_V, \cdot_V)$ and $(W, \Bbbk, +_W, \cdot_W)$ be vector spaces over the same field, $\Bbbk$. A **linear map** between these vector spaces is a function $T\colon V \to W$ such that

$$T(u +_V v) = T(u) +_W T(v), \qquad \text{and} \qquad T(\alpha \cdot_V v) = \alpha \cdot_W T(v) \tag{4.4.26}$$

for all $\alpha \in \Bbbk$ and $u, v \in V$.

From now on we drop the explicit symbol for scalar multiplication, writing αv for $\alpha \cdot v$, as well as dropping labels differentiating which vector space an operation is defined on. So linearity is expressed as

$$T(u + v) = T(u) + T(v), \qquad \text{and} \qquad T(\alpha v) = \alpha T(v). \tag{4.4.27}$$

We also refer to V and W as vector spaces (over $\Bbbk$) leaving the operations (and potentially the field) implicit.

Definition 4.4.28 — $\mathbf{Vect}_\Bbbk$ Fix some field $\Bbbk$. The category $\mathbf{Vect}_\Bbbk$ has vector spaces over $\Bbbk$ as objects and linear maps as morphisms. Composition of morphisms is composition of linear maps, which is composition of the underlying functions. The identity morphisms are the identity linear maps, which are the underlying identity functions.

Lemma 4.4.29 $\mathbf{Vect}_\Bbbk$ is a category.

Proof. Composition of linear maps inherits associativity from the underlying functions and similarly the identity laws follow from the identity laws of the underlying functions. We therefore only need to show that the composite of two linear maps is again linear. Let $T\colon U \to V$ and $S\colon V \to W$ be linear maps between vector spaces U, V, and W over some field $\Bbbk$. Then for all $u, v \in V$ and $\alpha \in \Bbbk$ we have

$$\begin{aligned}(S \circ T)(u + v) &= S(T(u + v)) = S(T(u) + T(v)) \\ &= S(T(u)) + S(T(v)) = (S \circ T)(u) + (S \circ T)(v)\end{aligned} \tag{4.4.30}$$

using the linearity of T and then S, and

$$(S \circ T)(\alpha v) = S(T(\alpha v)) = S(\alpha T(v)) = \alpha S(T(v)) = \alpha(S \circ T)(v) \tag{4.4.31}$$

again using linearity of T and then S. Hence the composite of two linear maps is again a linear map. □

Definition 4.4.32 — Basis Let V be a vector space. A subset $\mathcal{B} \subset V$ is **linearly independent** if for every finite subset of $\{e_1, \dots, e_n\} \subseteq \mathcal{B}$ satisfies

$$\alpha_1 e_1 + \cdots + \alpha_n e_n = 0 \tag{4.4.33}$$

only for the trivial case of $\alpha_i = 0$.

A subset $\mathcal{B} \subset V$ spans V if every $v \in V$ can be written as

$$v = \alpha_1 e_1 + \cdots + \alpha_n e_n \tag{4.4.34}$$

for some finite subset $\{e_1, \ldots, e_n\} \subseteq \mathcal{B}$. We call α_i the **components** of v. If a subset $\mathcal{B} \subset V$ is linearly independent and spans V then we call it a **basis**.

Every vector space has a basis (assuming the axiom of choice). It is a fact that any two bases have the same cardinality, which we call the **dimension** of the space. A vector space is **finite-dimensional** if it's dimension is finite.

Definition 4.4.35 — $\mathbf{FVect}_{\Bbbk}$ The category $\mathbf{FVect}_{\Bbbk}$ has finite-dimensional vector spaces as objects and linear maps as morphisms.

Lemma 4.4.36 $\mathbf{FVect}_{\Bbbk}$ is a category.

Proof. This follows from $\mathbf{Vect}_{\Bbbk}$ being a category. □

Linear maps are often thought of as matrices, although this only works in the finite-dimensional case. Take two vector spaces V and W with bases $\{v_i\}$ and $\{w_i\}$ respectively. Then a linear map $T : V \to W$ defines a matrix with components T_{ij} given by $T(v_i)_j$, where $T(v_i)_j$ is the jth component of $T(v_i)$, which is what we get evaluating the linear map at the ith basis vector, v_i. A matrix, T_{ij}, defines a linear map $T : V \to W$ similarly, by defining $T(v_i)$ to be formed from components $T(v_i)_j = T_{ij}$, and then using linearity to extend this definition to any vector in V.

Definition 4.4.37 — $\mathbf{Mat}_{\Bbbk}$ The category $\mathbf{Mat}_{\Bbbk}$ has natural numbers as objects with a morphism $n \to m$ being an $m \times n$ matrix with entries in $\Bbbk$. Composition of morphisms is matrix multiplication and the identity matrix is the identity morphism.

Lemma 4.4.38 $\mathbf{Mat}_{\Bbbk}$ is a category.

Proof. The product of an $m \times n$ matrix and a $n \times \ell$ matrix is an $m \times \ell$ matrix, reflecting the fact we can compose morphisms $\ell \to n$ and $n \to m$ to get a morphism $\ell \to m$. Associativity follows from associativity of matrix multiplication and the identity matrix is clearly the identity. □

There is an equivalence (Definition 4.8.5) $\mathbf{Mat}_{\Bbbk} \to \mathbf{FVect}_{\Bbbk}$ given by sending $n \to \Bbbk^n$ and sending a matrix to a linear map as described above. This formalises the idea that linear maps and matrices are equivalent ways of doing linear algebra in finite dimensions.

4.4.3.2 Hilb and FHilb

Definition 4.4.39 — Inner Product Space An **inner product space**, $(V, \langle - | - \rangle)$, is a vector space, V, over the field $\Bbbk = \mathbb{R}, \mathbb{C}$ equipped with an **inner product** $\langle - | - \rangle : V \times V \to \Bbbk$ such that the inner product is

- conjugate symmetric: $\langle u|v\rangle = \langle v|u\rangle^*$ for all $u, v \in V$;
- linear in the *second* argument: $\langle u|\alpha v\rangle = \alpha\langle u|v\rangle$ for all $\alpha \in \Bbbk$ and $u, v \in V$, this implies antilinearity in the first argument: $\langle \alpha u|v\rangle = \alpha^*\langle u|v\rangle$;
- positive-definite: $\langle v|v\rangle > 0$ for all $v \in V$ with $v \neq 0$ and $\langle 0|0\rangle = 0$, note that conjugate symmetry implies $\langle v|v\rangle = \langle v|v\rangle^*$ so $\langle v|v\rangle$ is real.

Note that for the $\Bbbk = \mathbb{R}$ case we can simply ignore the complex conjugates.

Definition 4.4.40 — Hilbert Space A **Hilbert space** is an inner product space $(H, \langle - | - \rangle)$ such that H is complete with respect to the norm $\| - \| : H \to \mathbb{R}$ defined by $\|v\| := \sqrt{\langle v|v\rangle}$, we say that this is the norm induced by the inner product. Being complete means that if the sequence $\{v_i\} \subseteq H$ is such that

$$\sum_{i=1}^{\infty} \|v_i\| \tag{4.4.41}$$

converges (as a series in $\mathbb{R}$) then

$$\sum_{i=1}^{\infty} v_i \tag{4.4.42}$$

converges to some $v \in H$, in the sense that for all $\varepsilon > 0$ there exists some $N \in \mathbb{N}$ such that for all $n > N$ we have

$$\left\| v - \sum_{i=1}^{n} v_i \right\| < \varepsilon, \tag{4.4.43}$$

or equivalently,

$$\lim_{n} \left\| v - \sum_{i=1}^{n} v_i \right\| = 0. \tag{4.4.44}$$

We will assume that Hilbert spaces are inner product spaces over $\mathbb{C}$ unless stated otherwise, although most facts will hold for real Hilbert spaces as well. Complex Hilbert spaces are where all quantum mechanics takes place, so we don't lose much by making this assumption.

This requirement of completeness is really just a technical requirement to make things well defined in certain circumstances, and we won't ever have reason to make use of it explicitly. For our purposes inner product space and Hilbert space are basically synonyms, but we're slightly safer working in Hilbert spaces due to this extra condition.

Definition 4.4.45 — Bounded Linear Map Let H and K be Hilbert spaces with norms $\|-\|_H$ and $\|-\|_K$ respectively, both induced by the inner product. A **bounded linear map** is a map $T: H \to K$ such that

$$\|T(v)\|_K \leq M\|v\|_H \tag{4.4.46}$$

for some $M \in \mathbb{R}$ and all $v \in H$.

Definition 4.4.47 — Hilb and FHilb Fix some field $\Bbbk = \mathbb{R}, \mathbb{C}$. The category **Hilb** has Hilbert spaces as objects and bounded linear maps as morphisms. The category **FHilb** has finite-dimensional Hilbert spaces as objects and bounded linear maps as morphisms.

Lemma 4.4.48 **Hilb** and **FHilb** are categories.

Proof. Associativity and identities follow from the underlying functions. We need only show that the composite of two bounded linear maps is again a bounded linear map. Let $T: H \to K$ and $S: K \to J$ be bounded linear maps between the Hilbert spaces H, K, and J. Then there exist some $M, N \in \mathbb{R}$ such that $\|T(v)\|_K \leq M\|v\|_H$ and $\|S(u)\|_J \leq N\|u\|$ for all $v \in H$ and $u \in K$. Then we have $\|(S \circ T)(v)\|_J = \|S(T(v))\|_J \leq N\|T(v)\|_K \leq NM\|v\|_H$ for all $v \in V$, and $NM \in \mathbb{R}$, so the map is bounded again. The composite of two linear maps is linear, as shown in Lemma 4.4.29, so the composite of two bounded linear maps is a bounded linear map. □

We now make a few basic definitions.

Definition 4.4.49 — Orthonormal Let H be a Hilbert space and $\{e_i\}$ a basis of H, then we say that this basis is **orthogonal** if $\langle e_i | e_j \rangle = 0$ for $i \neq j$. If $\langle e_i | e_j \rangle = \delta_{ij}$ we say the basis is **orthonormal**.

Definition 4.4.50 — Adjoint If H and K are Hilbert spaces and $T: H \to K$ is a linear map then we define the **adjoint** to be the linear map $T^\dagger : K \to H$ such that $\langle T(v) | w \rangle = \langle v | T^\dagger(w) \rangle$.

In terms of matrices representing linear maps the adjoint corresponds to the conjugate transpose matrix.

Definition 4.4.51 — Bras and Kets Let H be a Hilbert space. Given some $v \in H$ its **ket** is the map $|v\rangle : \mathbb{C} \to H$ defined by $z \mapsto zv$ and its bra is the map $\langle v| : H \to \mathbb{C}$ defined by $w \mapsto \langle v | w \rangle$.

This definition is rather formal and doesn't correspond to how we usually think about bras and kets. Typically we think about elements of H as being kets. This works since each linear map $\mathbb{C} \to H$ is completely defined by where it sends 1, so really linear maps of this form just pick out elements of H, and $|v\rangle$ is such that $1 \mapsto 1v = v$. So we often write $|v\rangle$ when we might actually mean v by this definition. We then think of bras similarly as being their own vectors in some other space, which we'll define in the next definition, and then we think of the mapping $w \mapsto \langle v|w\rangle$ as simply performing multiplication $\langle v||w\rangle = \langle v|w\rangle$.

Definition 4.4.52 — Dual Space Let V be a vector space. Then $V^* := \mathbf{Vect}_{\Bbbk}(V, \Bbbk)$ is the **dual space**.

This definition means that V^* is the space of linear maps $V \to \Bbbk$, which in the case of a Hilbert space we can recognise as the space of bras. Note that we are equipping $\mathbf{Vect}_{\Bbbk}(V, \Bbbk)$ with the obvious vector space (or Hilbert space) structure given by adding and scaling linear maps pointwise.

4.4.4 More Examples

Example 4.4.53 — Sets with Structure Many categories can be described as having objects formed from sets with structure, and morphisms being structure preserving functions. These are all concrete categories, in the sense that we can forget the structure and just think of them as sets and functions. Some examples of these sets with structure are

- The category **Mon** has monoids as objects and monoid homomorphisms as morphisms.
- The category **Grp** has groups as objects and group homomorphisms as morphisms.
- The category **Ring** has rings as objects and ring homomorphisms as morphisms.
- The category **CRing** has commutative rings as objects and ring homomorphisms as morphisms.
- The category **Field** has fields as objects and field homomorphisms as morphisms.
- The category R–**Mod** for a fixed ring R has left modules over R as objects and module homomorphisms as morphisms. Note that the special case where R is a field is $\mathbf{Vect}_R$.
- The category **Top** has topological spaces as objects and continuos maps as morphisms.
- The category $\mathbf{Top}_\bullet$ has pointed topological spaces, $(X, \bullet)$, as objects, that is topological spaces with some special point, $\bullet$, and based maps as morphisms, that is continuous maps preserving this special point, that is $f : (X, \bullet) \to (Y, *)$ is continuous and $f(\bullet) = *$.

- The category **Pos** has posets as objects and monotone functions as morphisms.

See Chapter A for relevant definitions.

Example 4.4.54 — Posets Given a poset, $(P, \leq)$, we can define a category whose objects are elements of P and there is a unique morphism $a \to b$ if $a \leq b$. Since this morphism is unique and $a \leq a$ the identity is simply the unique morphism $a \to a$. Since $a \leq b$ and $b \leq c$ implies $a \leq c$ there is a unique morphism $a \to c$ in this case, which must then be the morphism formed by composing the morphisms $a \to b$ and $b \to c$.

Example 4.4.55 — Single Object Categories Given a monoid, M, we can interpret it as a category with a single object, which we call $\bullet$, and then the elements of M are morphisms $\bullet \to \bullet$ with composition of morphisms given by the monoid product. The identity of the monoid is the identity morphism. In a sense categories just generalise monoids to have more objects. The process of defining a quantity, then mapping the definition to a particular category with a single object, and then allowing there to be multiple objects is called **oidification**, and the resulting object is suffixed with -oid. So a monoidoid is a category.
Given a group, G, we can interpret it as a single object category where all morphisms are isomorphisms (Definition 4.6.1). This extra condition simply reflects the condition that a group has all inverses. A **groupoid** is then a category in which all morphisms are isomorphisms.
Note that we can proceed in the opposite direction, given a category with a single object (with all morphisms being isomorphisms) this can be interpreted as a monoid (group).

4.5 Diagrams

Composition of morphisms can be quite confusing. There is lots of data to specify, which objects are involved, what are the morphisms called, do the morphisms have any other properties we might be interested in and so on. It doesn't help that the order in which we write composition of functions is the reverse of the order in which the functions are applied. The solution to this is to draw pictures with all of the objects as nodes and the morphisms as arrows between them. Composition of morphisms is then given by reading the arrows backwards. A simple example is

$$A \xrightarrow{f} B \xrightarrow{g} C \tag{4.5.1}$$

which represents two morphisms, $f : A \to B$ and $g : B \to C$, which can be composed to give $(g \circ f) : A \to C$. We might add this morphism to our diagram,

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[rr, bend right, "g\circ f"'] & B \arrow[r, "g"] & C \end{tikzcd} . \tag{4.5.2}$$

although we don't have to since it's existence is ensured by the definition of a category. Both paths taken in this diagram give the same result.

In general if all paths in a diagram between two fixed objects give the same result we say that the diagram **commutes**, or is a **commutative diagram**. This can be useful to specify a lot of algebraic relations in a single commutative diagram. For example, the diagram

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "h"'] & B \arrow[r, "g"] \arrow[d, "i"'] & C \\ D \arrow[r, "k"'] & E \arrow[ur, "j"'] & \end{tikzcd} \tag{4.5.3}$$

commuting is equivalent to the following requirements:

$$g \circ f = j \circ k \circ h, \qquad i \circ f = k \circ h, \qquad \text{and} \qquad g = j \circ i, \tag{4.5.4}$$

which are given by requiring that the outer parallelogram commutes, the square commutes, and the triangle commutes respectively.

Since the definition of a category means every object has an identity morphism and that these identity morphisms don't really affect composition we leave the identity morphisms implicit in the diagram. We could write them in, for example

$$\begin{tikzcd} A \arrow[loop left, "\mathrm{id}_A"] \arrow[r, "f"] & B \arrow[loop right, "\mathrm{id}_B"] \end{tikzcd} \tag{4.5.5}$$

commuting is simply the identity law, telling us that, among other things, $f \circ \mathrm{id}_A = f = \mathrm{id}_B \circ f$. We can also state the associativity law as the commutativity of the following:

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[rr, bend left, "g\circ f"] & B \arrow[r, "g"] \arrow[rr, bend right, "h\circ g"'] & C \arrow[r, "h"] & D. \end{tikzcd} \tag{4.5.6}$$

4.6 Terminology

We now introduce some terminology which will be helpful.

Definition 4.6.1 For a morphisms $f : A \to B$

- we call A the **domain**, or **source**, of f;
- we call B the **codomain**, or **target**, of f;
- we call f an **endomorphism** if $A = B$;
- we call f an **isomorphism** if there exists $f^{-1} : B \to A$ such that $f^{-1} \circ f = \mathrm{id}_A$ and $f \circ f^{-1} = \mathrm{id}_B$;
- we call f an **automorphism** if it is an isomorphism and endomorphism;

- we call A **isomorphic** to B if f is an isomorphism, and write $A \cong B$;
- we call f an **epimorphism**, or **epic**, if $g \circ f = h \circ f$ implies $g = h$ for all $g, h \colon B \to C$;
- we call f a **monomorphism**, or **monic**, if $f \circ g = f \circ h$ implies $g = h$ for all $g, h \colon C \to A$.

The most important definition here is isomorphisms. Often equality is too strict a requirement for comparing two objects. Instead isomorphism is usually the correct level of similarity. In particular in a concrete category objects are isomorphic if there is an invertible structure preserving map between them, giving a one-to-one pairing of elements. This allows for trivial differences between objects, like renaming of elements or operations, to be ignored. For example, the groups $(\{0, 1\}, +_2)$ and $(\{1, -1\}, \cdot)$ are not equal, but they are isomorphic.

Lemma 4.6.2 Let $\mathbf{C}$ be a category with objects A and B. If $f \colon A \to B$ is an isomorphism then the inverse is unique.

Proof. Suppose this wasn't the case, so $f \colon A \to B$ has two inverses, $g, g' \colon B \to A$, which are both such that $g \circ f = g' \circ f = \mathrm{id}_A$ and $f \circ g = f \circ g' = \mathrm{id}_B$. Then we have

$$g = g \circ \mathrm{id}_B = g \circ (f \circ g') = (g \circ f) \circ g' = \mathrm{id}_A \circ g' = g'. \tag{4.6.3}$$

□

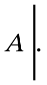
The condition that f and f^{-1} are inverses is equivalent to requiring that

$$\begin{tikzcd} A \arrow[r, shift left, "f"] & B \arrow[l, shift left, "f^{-1}"] \end{tikzcd} \tag{4.6.4}$$

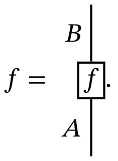
commutes.

4.7 Graphical Notation

In this section we will introduce a graphical notation which can be used to reason about categories. This notation treats categories as processes, taking some input and producing some output. We read the notation from bottom to top, which we can imagine is the progression of "time" through the process. This notation obeys the rules, and spirit, of category theory. In particular, we don't write objects. Instead we represent each object through it's identity morphism, so A is represented by id_A. An identity morphism is then represented by a line, labelled by the object, representing a process in which nothing happens, the input at the bottom of the page, is just passed directly to the output at the top of the page:

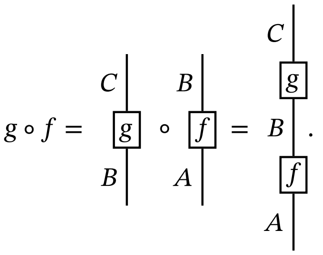
$\mathrm{id}_A =$  . (4.7.1)

A general morphism, $f : A \to B$, is represented in this graphical notation by placing a box on the wire, representing some process occurring, and labelling the wire either side with the appropriate object:



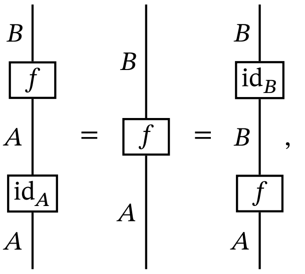
(4.7.2)

Composition of morphisms is represented by doing one process after the other, which in this notation just means writing one box after the other and joining them by a wire of the appropriate type. If we introduce a second morphism $g : B \to C$ then we have



(4.7.3)

This graphical notation makes the axioms of a category implicit. For the identity law since identity morphisms are just drawn as wires clearly the following holds:



(4.7.4)

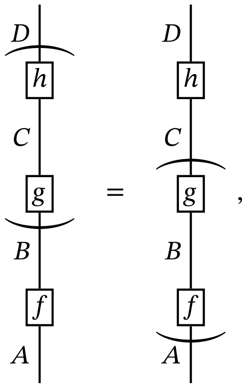
and this is just the identity law

$$f \circ \mathrm{id}_A = f = \mathrm{id}_B \circ f. \qquad (4.7.5)$$

Similarly we don't bother drawing brackets around composites, since associativity makes them unnecessary. If we did draw brackets then considering the objects and morphisms

$$A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D \qquad (4.7.6)$$

we can write the associativity law as



(4.7.7)

which is just

$$(h \circ g) \circ f = h \circ (g \circ f). \tag{4.7.8}$$

So by manipulating this graphical notation naturally we apply the category axioms. This makes this notation very powerful. While this notation looks new if we were to instead draw it horizontally and replace wires with $\circ$ and id_X then it would just be the usual algebraic notation for morphism composition. The real power of this notation comes when we introduce new concepts, such as monoidal products, which extend the graphical notation into a two-dimensional notation which works in much the same way, while the usual algebraic notation quickly becomes cluttered and hard to use.

Note that while wires don't need to be vertical, sometimes it's helpful to have them not be to fit more things in, wires are not, in general, allowed to be horizontal, this is so that we can't turn a morphism "upside down", although we'll see later that this is sometimes possible.

4.8 Functors

The spirit of category theory is that it is not objects that are important but morphisms between them. This suggests that whenever we see a mathematical object we should look for maps between them preserving the relevant structure. Well, categories are objects, so we should look for maps between them. Such maps should preserve the category's structure, which is

- the collection of objects;
- the collections of morphisms;
- the composition of morphisms;
- the identity morphisms.

For example, if we have a morphism $f : A \to B$ in one category then our map should send this to another morphism in the new category, and the domain and codomain of that morphism should somehow be related to A and B. It shouldn't matter whether we do composition of morphisms and then map, or map and the compose morphisms. Identities should map to identities. To this end we make the following definition of a map between categories preserving this structure.

Definition 4.8.1 — Functor Let $\mathbf{C}$ and $\mathbf{D}$ be categories. A **functor**, $F\colon \mathbf{C} \to \mathbf{D}$, is composed of two mappings:

- each object $A \in \mathrm{Ob}(\mathbf{C})$ is mapped to some object $F(A) \in \mathrm{Ob}(\mathbf{D})$;
- each morphism $f \in \mathbf{C}(A, B)$ is mapped to some morphism $F(f) \in \mathbf{D}(F(A), F(B))$;

such that

- composition is preserved: $F(g \circ f) = F(g) \circ F(f)$ for $f\colon A \to B$ and $g\colon B \to C$ morphisms in $\mathbf{C}$;
- identities are preserved: $F(\mathrm{id}_A) = \mathrm{id}_{F(A)}$ for $A \in \mathrm{Ob}(\mathbf{C})$.

Notation 4.8.2 It is common to drop the brackets when applying a functor, a bit like we may apply a linear map, T, to some vector, v, by writing Tv and thinking of it as matrix multiplication. In this case a functor $F\colon \mathbf{C} \to \mathbf{D}$ maps each $A \in \mathrm{Ob}(\mathbf{C})$ to some object $FA \in \mathrm{Ob}(\mathbf{D})$, each morphism $f \in \mathbf{C}(A, B)$ to some morphism $Ff \in \mathbf{D}(FA, FB)$, such that $F(g \circ f) = Fg \circ Ff$ and $F\mathrm{id}_A = \mathrm{id}_{FA}$.

Note that what we have defined above is a **covariant functor**. It is also possible to define a **contravariant functor** which reverses the direction of morphisms. The difference in the definition is that each $f \in \mathbf{C}(A, B)$ is assigned to some $Ff \in \mathbf{D}(FB, FA)$ and $F(g \circ f) = Ff \circ Fg$. We'll assume that all functors are covariant unless stated otherwise.

Example 4.8.3 — Functors

- The **constant functor**, $\mathrm{const}_X\colon \mathbf{C} \to \mathbf{D}$, maps every object of $\mathbf{C}$ to $X \in \mathrm{Ob}(\mathbf{D})$ and every morphism $f\colon A \to B$ in $\mathbf{C}$ to the identity morphism $\mathrm{id}_X\colon X \to X$ in $\mathbf{D}$.
- The **identity functor**, $\mathrm{id}_{\mathbf{C}}\colon \mathbf{C} \to \mathbf{C}$ maps every object and morphism in $\mathbf{C}$ to itself.
- If $\mathbf{C}$ has products we can define the **diagonal functor**, $\Delta\colon \mathbf{C} \to \mathbf{C} \times \mathbf{C}$, such that $\Delta(A) = A \times A$ for all objects A and $\Delta(f) = f \times f$ for all morphisms f.
- The power set can be regarded as a functor $\mathcal{P}\colon \mathbf{Set} \to \mathbf{Set}$, sending each set to its power set and each function $f\colon A \to B$ to the map sending $U \in \mathcal{P}(A)$ to its image
$$f(U) \coloneqq \{f(u) \mid u \in U\}. \tag{4.8.4}$$
- The map $V \mapsto V^*$ defines a functor $\mathbf{Vect}_{\Bbbk} \to \mathbf{Vect}_{\Bbbk}$.

- If G is a group viewed as a one object category then a functor $G \to$ **Set** defines a group action, sending the single object of G to the set, S, upon which G acts, and sending each morphism, g (recall morphisms are just elements of the group) to the mapping on that set $s \mapsto g\,.\,s$.

- If G is a group viewed as a one object category then a functor $G \to$ **Vect**$_{\Bbbk}$ defines a group action on a vector space, which is to say this functor defines a representation.

- If G and H are groups viewed as one object categories then a functor $G \to H$ is simply a group homomorphism, sending the single object of G to the single object of H and preserving composition, which is the group operation.

- The map sending each complex Lie group to its Lie algebra is a functor **LieGrp** → **LieAlg**[a].

- A **forgetful functor**, **C** → **Set** maps a category to **Set** by "forgetting" the structure of the objects in **C**. A category, **C**, is called **concrete** if there exists a forgetful functor **C** → **Set**.

  - The forgetful functor $U\colon$ **Grp** → **Set** sends each group to its underlying set and each group homomorphism to its underlying function.
  - The forgetful functor $U\colon$ **Vect**$_{\Bbbk}$ → **Set** sends each vector space to its underlying set and each linear transformation to its underlying function.
  - The forgetful functor $U\colon$ **Top** → **Set** sends each topological space to its underlying set and each continuous function to its underlying function.

- A partially forgetful functor **C** → **D** generalises forgetful functors by forgetting some, but not necessarily all, structure of the objects in **C**.

  - The partially forgetful functor **Grp** → **Mon** sends each group to itself, but now viewed as a monoid, essentially forgetting that inverses exist.
  - The partially forgetful functor **Hilb** → **Vect**$_{\Bbbk}$ sends each Hilbert space to itself, but forgets the inner product structure.
  - The partially forgetful functor **CRing** → **Ring** sends each commutative ring to itself, forgetting that multiplication is commutative.
  - The partially forgetful functor **Ring** → **Ab** sends each ring to its additive group, forgetting how to multiply.

- A **free functor** is, in a sense[b] the reverse of a forgetful functor, sending a set to the most general object of the appropriate type.

- The free functor **Set** → **Grp** sends each set the free group it generates, which is the set of words generated by concatenating elements of the set and elements declared to be their inverses, with the group operation being concatenation and the identity the empty string. No other relations between elements of the set exist.
- The free functor **Set** → $\mathbf{Vect}_{\mathbb{k}}$ takes a set and declares that all the elements in it are linearly independent and form a basis for a vector space whose elements are formal linear combinations of these elements.

[a] See *Symmetries of Quantum Mechanics* or *Symmetries of Particles and Fields*

[b] This sense can be made formal through the notion of an adjoint, an important concept in category theory but one we won't explore.

The next definition introduces some terminology used to talk about functors.

Definition 4.8.5 A functor $F\colon \mathbf{C} \to \mathbf{D}$ is

- **full** when the mapping $f \mapsto Ff$ defines a surjection $\mathbf{C}(A,B) \to \mathbf{D}(FA,FB)$;
- **faithful** when the mapping $f \mapsto Ff$ defines an injection $\mathbf{C}(A,B) \to \mathbf{D}(FA,FB)$;
- **essentially surjective on objects** when for each $B \in \mathrm{Ob}(\mathbf{D})$ there is some $A \in \mathrm{Ob}(\mathbf{C})$ such that $FA \cong B$, essentially when the mapping $A \mapsto FA$ is surjective, but replacing equality with isomorphism in the definition of surjectivity;
- an **equivalence** when it is full, faithful, and essentially surjective on objects;
- an **endofunctor** if $\mathbf{C} = \mathbf{D}$.

The most important definition here is that of an equivalence. Equality of categories is far too strong a condition. We can also define a notion of isomorphism between categories, but this is also too strong. Relaxing the requirements for isomorphism, by changing surjective on objects to essentially surjective on objects, we get the notion of equivalence, which is the sweet spot where a functor preserves enough structure for the two categories to be the same for all intents and purposes, without being too restrictive.

It should be noted that there are multiple, equivalent, definitions of equivalence. The one we've given here is probably the easiest to understand, but not always the easiest to apply.

One example we've already seen of an equivalence is the functor $\mathbf{Mat}_{\mathbb{k}} \to \mathbf{FVect}_{\mathbb{k}}$ sending $n \mapsto \mathbb{k}^n$ and a matrix to the associated linear map on the vector space with some fixed basis. This is an equivalence since all finite dimensional vector spaces of the same dimension are isomorphic, which means that this functor is essentially surjective on objects.

Now that we have categories and maps between them it is natural to ask if this forms a category, and indeed it does!

Definition 4.8.6 — Cat The category **Cat** has (small) categories as its objects and functors as its morphisms. Composition of functors is composition of the two mappings $\mathrm{Ob}(\mathbf{C}) \to \mathrm{Ob}(\mathbf{D})$ and $\mathbf{C}(A, B) \to \mathbf{D}(FA, FB)$, and the identity is the identity functor.

To be clear, if $F\colon \mathbf{C} \to \mathbf{D}$ and $G\colon \mathbf{D} \to \mathbf{E}$ are functors then the functor $(G \circ F)\colon \mathbf{C} \to \mathbf{E}$ sends $A \in \mathrm{Ob}(\mathbf{C})$ to $(G \circ F)(A) = (G \circ F)A = G(F(A)) = GFA$, and sends $f \in \mathbf{C}(A, B)$ to $(G \circ F)(f) = (G \circ F)f = G(F(f)) = GFf \in \mathbf{E}(GFA, GFB)$. Associativity is guaranteed by associativity of the underlying mappings and the identity functor is clearly an identity morphism.

4.9 Natural Transformations

We have now defined functors as mathematical objects. Following the spirit of category theory we should look for maps between functors preserving their structure. A map between functors should preserve the functor structure. This means that it shouldn't matter if we map to a different functor and then apply the functor, or apply a functor and then map to the other functor. This leads to the following definition.

Definition 4.9.1 — Natural Transformation Let **C** and **D** be categories and $F, G\colon \mathbf{C} \to \mathbf{D}$ functors. For every object $A \in \mathrm{Ob}(\mathbf{C})$ a **natural transformation**, $\zeta\colon F \Rightarrow G$, assigns a morphism $\zeta_A : FA \to GA$ in **D** such that for every morphism $f\colon A \to B$ in **C** the following diagram commutes:

$$\begin{tikzcd} FA \arrow[r, "\zeta_A"] \arrow[d, "Ff"'] & GA \arrow[d, "Gf"] \\ FB \arrow[r, "\zeta_B"'] & GB. \end{tikzcd} \tag{4.9.2}$$

If every component, ζ_A, is an isomorphism then ζ is called a **natural isomorphism** and F and G are said to be **naturally isomorphic**, written $F \cong G$.

This leads to an alternative, equivalent, definition of an equivalence. A functor $F\colon \mathbf{C} \to \mathbf{D}$ is an equivalence if and only if there is a functor $G\colon \mathbf{D} \to \mathbf{C}$ and natural isomorphisms $G \circ F \Rightarrow \mathrm{id}_{\mathbf{C}}$ and $F \circ G \Rightarrow \mathrm{id}_{\mathbf{D}}$. Intuitively, an equivalence is a function which is invertible up to natural isomorphism.

Note that it is common to write all of the data needed to define a natural transformation as

$$\begin{tikzcd} \mathbf{C} \arrow[r, bend left, "F"{name=U}] \arrow[r, bend right, "G"'{name=L}] & \mathbf{D}. \arrow[Rightarrow, from=U, to=L] \end{tikzcd} \tag{4.9.3}$$

Example 4.9.4 — Natural Transformations These examples are taken from [2] and make use of the notation used there representing a natural transformation as a collection of morphisms indexed by objects, $(\zeta_A)_{A\in\mathrm{Ob}(\mathbf{C})}$.

- A **discrete category** is a category in which the only maps are the identity maps. If $\mathbf{C}$ is a discrete category then a functor $F\colon \mathbf{C}\to\mathbf{D}$ is simply a family of objects $(FA)_{A\in\mathrm{Ob}(\mathbf{C})}$. In this case a natural transformation $\zeta\colon F\Rightarrow G$ is just a family of maps $(\zeta_A : FA \to GA)_{A\in\mathrm{Ob}(\mathbf{C})}$. The naturality axiom is automatically fulfilled since it holds trivially for identities.

- Fix some natural number n. For any commutative ring, R, the $n\times n$ matrices with entries in R form a monoid, $M_n(R)$, under matrix multiplication. Any ring homomorphism $R\to S$ induces a monoid homomorphism $M_n(R)\to M_n(S)$ by acting elementwise on the entries of the matrix with the homomorphism. This defines a functor $M_n : \mathbf{CRing}\to\mathbf{Mon}$.

  The elements of any ring, R, form a monoid, $U(R)$, under multiplication, this is an example of a forgetful functor $U\colon \mathbf{CRing}\to\mathbf{Mon}$.

  Every $n\times n$ matrix, X, over a commutative ring R has a determinant $\det_R(X)\in R$. The properties

  $$\det_R(XY)=\det_R(X)\det_R(Y),\qquad\text{and}\qquad \det_R(I)=1 \quad (4.9.5)$$

  tell us that for each commutative ring R the function $\det_R : M_n(R)\to U(R)$ is a monoid homomorphism. Thus we have a family of maps

  $$(\det_R : M_n(R)\to U(R))_{R\in\mathrm{Ob}(\mathbf{CRing})}. \qquad (4.9.6)$$

  This family of maps defines a natural transformation $\det : M_n\Rightarrow U$.

- Consider the category $\mathbf{FVect}_{\Bbbk}$. The map $(-)^* : \mathbf{FVect}_{\Bbbk}\to\mathbf{FVect}_{\Bbbk}$ sending each vector space to its dual is a contravariant functor. Thus the map $(-)^{**} : \mathbf{FVect}_{\Bbbk}\to\mathbf{FVect}_{\Bbbk}$ sending each vector space to its double dual is also a covariant functor. For each $V\in\mathrm{Ob}(\mathbf{FVect}_{\Bbbk})$ we have a canonical isomorphism $\zeta_V\colon V\to V^{**}$. Given $v\in V$ the element $\zeta_V(v)\in V^{**}$ is evaluation at v, that is $\zeta_V(v)\colon V^*\to\Bbbk$ maps $\varphi\in V^*$ to $\varphi(v)\in\Bbbk$. This defines a natural transformation $\zeta\colon 1_{\mathbf{FVect}_{\Bbbk}}\Rightarrow(-)^{**}$ meaning that each vector space is naturally isomorphic to its double dual. Note that given $F,G\colon \mathbf{C}\to\mathbf{D}$ we say that $FA\cong GA$ naturally in A if F and G are naturally isomorphic. In this case $V\cong V^{**}$ naturally in V.

Functors are objects, and natural transformations are maps between them, so we should try to define a category.

Definition 4.9.7 — Functor Category Let $\mathbf{C}$ and $\mathbf{D}$ be categories. The **functor category** $[\mathbf{C}, \mathbf{D}]$, also written $\mathbf{D}^{\mathbf{C}}$, has functors $F, G : \mathbf{C} \to \mathbf{D}$ as objects and natural transformations $F \Rightarrow G$ as morphisms. Composition of natural transformations is done component wise and the identity is the identity natural transformation which assigns the to each $A \in \mathrm{Ob}(\mathbf{C})$ the identity morphism id_{FA} in $\mathbf{D}$.

An example of a functor category is the category of G-sets[2] for some group G, which is $[G, \mathbf{Set}]$, where G is viewed as a one object category. Similarly $[G, \mathbf{Vect}_{\Bbbk}]$ is the category of $\Bbbk$-linear representations of G. In both of these cases morphisms are so called equivariant maps. Given sets (vector spaces) X and Y upon which we have a G action (representation) defined an **equivariant map** is a function (linear map) $f : X \to Y$ such that $f(g \,.\, x) = g \,.\, f(x)$, that is a map which commutes with the group action, so

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "g."'] & Y \arrow[d, "g."] \\
X \arrow[r, "f"'] & Y
\end{tikzcd} \tag{4.9.8}$$

commutes. This diagram is exactly what we get if we specialise the diagram defining a natural transformation to functors $G \to \mathbf{Set}$ ($G \to \mathbf{Vect}_{\Bbbk}$).

4.10 Initial and Terminal Objects

A comathematician is a comachine for turning cotheorems into ffee.

Definition 4.10.1 — Initial and Terminal Objects Let $\mathbf{C}$ be a category. An **initial object**, $I \in \mathrm{Ob}(\mathbf{C})$ is an object which has a unique morphism into every other object. That is for all $A \in \mathrm{Ob}(\mathbf{C})$ there is a unique morphism $I \to A$.
A **terminal object**, $T \in \mathrm{Ob}(\mathbf{C})$ is an object which has a unique morphism out of every other object. That is for all $A \in \mathrm{Ob}(\mathbf{C})$ there is a unique morphism $A \to T$.

Notice that these two concepts are related by turning the arrows around. We say that the two concepts are opposite or dual. This is common in category theory, that we have two such definitions, and as such it motivates the following definition.

Definition 4.10.2 — Opposite Category Given a category, $\mathbf{C}$, the **opposite category**, $\mathbf{C}^{\mathrm{op}}$, has the same objects, that is $\mathrm{Ob}(\mathbf{C}) = \mathrm{Ob}(\mathbf{C}^{\mathrm{op}})$, but all morphisms are reversed, that is if we have a morphism $f : A \to B$ in $\mathbf{C}$ then there is a morphism $f^{\mathrm{op}} : B \to A$ in $\mathbf{C}^{\mathrm{op}}$, so $\mathbf{C}(A, B) = \mathbf{C}^{\mathrm{op}}(B, A)$.

[2] that is sets upon which G acts

With this definition we can reformulate the definition of initial and terminal objects into an initial object is an object with a unique morphism into every other object, and a terminal object is an initial object in the opposite category.

Another concept that we can reformulate with opposite categories is contravariant functors. A contravariant functor, $F\colon \mathbf{C} \to \mathbf{D}$, which is such that $F(g \circ f) = Ff \circ Fg$, is exactly a covariant functor from the opposite category, $F\colon \mathbf{C}^{\mathrm{op}} \to \mathbf{D}$, or equivalently, into the opposite category, $F\colon \mathbf{C} \to \mathbf{D}^{\mathrm{op}}$.

In general if we define some property, x, in a category we can often find the same property in the opposite category, and we prefix this property, viewed in the original category, with "co". For example, in the next section we will define products, and a product in the opposite category is a coproduct. Given a property the coproperty is defined by reversing all of the arrows, so an initial object is a coterminal object, and a terminal object is a coinitial object, and so on.

4.11 Products

We want to generalise the Cartesian product of sets, which is defined as

$$A \times B \coloneqq \{(a, b) \mid a \in A \text{ and } b \in B\}. \tag{4.11.1}$$

To do so we need to remove reference to elements, since this isn't a concept that we have in an arbitrary category. Clearly this definition defines a new set, so we have a new object in **Set**. In order to do away with reference to elements notice that we can define two functions $p_A : A \times B \to A$ and $p_B : A \times B \to B$ which project out the elements of a pair. That is $p_A(a, b) = a$ and $p_B(a, b) = b$. We can summarise this as

$$A \xleftarrow{p_A} A \times B \xrightarrow{p_B} B. \tag{4.11.2}$$

The key insight comes in requiring that $A \times B$ is, in a sense, the most general[3] set satisfying this property, since the Cartesian product doesn't add in any unnecessary restrictions. Thus, if we have another set, X, which is a candidate for defining the product then there must be morphisms $f : X \to A$ and $g : X \to B$ which are candidates for p_A and p_B respectively. We can now make "most general" precise. Given $A \times B$ with X, f, and g we should be able to recreate f and g by first mapping to $A \times B$ and then projecting out. This map to $A \times B$ is what we call the product of f and g, written $f \times g$, (f, g), or $\binom{f}{g}$. We can summarise this definition by requiring that the following diagram commutes:

$$\begin{tikzcd}
& X \arrow[ld, "f"'] \arrow[rd, "g"] \arrow[d, dashed, "!"', "f\times g"] & \\
A & A \times B \arrow[l, "p_A"] \arrow[r, "p_B"'] & B.
\end{tikzcd} \tag{4.11.3}$$

Here we use a dashed arrow to denote that this diagram is telling us that this arrow exists, and the ! tells us that this arrow is unique.

This is an example of a more general concept in category theory, called a **limit**. The general idea is that we can define some prototype, here $A \times B$ with the maps p_A and p_B. Then we posit a candidate object which factors through the prototype, then we look for an object satisfying this property.

[3] this concept, both for products and more general (co)limits, is explained well in [3], which is where I'm replicating this argument from.

Notice that we now have done away with all references to elements of $A \times B$, so we can generalise this definition to arbitrary categories.

Definition 4.11.4 — Product Let **C** be a category with objects A and B. The product of A and B, if it exists, is the object $A \times B$ and the morphisms $p_A : A \times B \to A$ and $p_B : A \times B \to B$ such that

$$\begin{tikzcd}
 & X \arrow[ld, "f"'] \arrow[d, dashed, "!"', "f\times g"] \arrow[rd, "g"] & \\
A & A \times B \arrow[l, "p_A"] \arrow[r, "p_B"'] & B
\end{tikzcd} \tag{4.11.5}$$

commutes for all X with morphisms to $f : X \to A$ and $g : X \to B$.
The **coproduct** is the product in the opposite category.

Example 4.11.6 — Products

- In **Set** the product is the Cartesian product, and the coproduct is the disjoint union.
- In **Top** the product of two topological spaces is the topological space formed from the Cartesian product of the two topological spaces equipped with the product topology, which is the coarsest topology for which all projections are continuous.
- In R—**Mod** the product is the Cartesian product of the modules with addition defined componentwise and multiplication defined to be distributive.
- In **Grp** the product is the direct product of groups, that is the Cartesian product of the underlying sets and then componentwise multiplication.
- In **Rel** products are disjoint unions.
- In a poset viewed as a single object category the product of two elements is the greatest lower bound.
- In **Hask**[a], the category of *Haskell* types with *Haskell* functions as morphisms, a product type is one defined as follows:

  ```
  data Product a b = Product a b
  ```

  This can be thought of as the Cartesian product, so here we have pairs, or tuples, of type `(a, b)`. On the other hand a sum type, or coproduct type, is defined as:

  ```
  data Sum a b = A a | B b
  ```

  This can be thought of as the disjoint union, which is can defined for sets by “tagging” each element with the set it comes from, then taking the usual union, here we "tag" the element by using one of the constructors, A or B, and then our final type allows for either of these constructors to be used, taking their union.

Notice that we need something of both type a and b to define a product, corresponding to the "and" nature of the product, and only something of type a or type b to define the sum type, corresponding to the "or" nature of the sum.

[a] **Hask** can be thought of as a subcategory of **Set**, where the only objects are the *Haskell* types, viewed as sets with all values of that type as their elements. There are some tricky things about *Haskell*, such as the existence of bottom, $\perp$, which represents an error. This also causes problems when trying to take products and coproducts, but typically we just ignore these issues.

Definition 4.11.7 — Product Category Let $\mathbf{C}$ and $\mathbf{D}$ be categories. The **product category $\mathbf{C} \times \mathbf{D}$** has

- as objects pairs of objects (A, B) with $A \in \mathrm{Ob}(\mathbf{C})$ and $B \in \mathrm{Ob}(\mathbf{D})$;
- as morphisms $(A_1, B_1) \to (A_2, B_2)$ pairs of morphisms (f, g) with $f : A_1 \to A_2$ and $g : B_1 \to B_2$ that is, $f \in \mathbf{C}(A_1, A_2)$ and $g \in \mathbf{D}(B_1, B_2)$;
- as composition componentwise composition from the two categories, that is $(f_2, g_2) \circ (f_1, g_1) = (f_2 \circ f_1, g_2 \circ g_1)$;
- as identities pairs of identities from the two categories, that is $\mathrm{id}_{(A,B)} = (\mathrm{id}_A, \mathrm{id}_B)$.

For the case where $\mathbf{C}$ and $\mathbf{D}$ are small the product category $\mathbf{C} \times \mathbf{D}$ is exactly the categorical product of $\mathbf{C}$ and $\mathbf{D}$ in **Cat**.

The main use of product categories is to define bifunctors, which are the generalisation of functors to two variables. A **bifunctor** is exactly a functor $\mathbf{C} \times \mathbf{D} \to \mathbf{E}$, but it's usually easier to think of $\mathbf{C}$ and $\mathbf{D}$ as being separate, just like how we might treat the horizontal and vertical axes as being independent in a function $\mathbb{R}^2 \to \mathbb{R}$.

Five

Monoidal Categories

5.1 Tensor Products

Definition 5.1.1 — Bilinear Let U, V and W be vector spaces over $\Bbbk$. A **bilinear map** is a function $T\colon U \times V \to W$ which is linear in each variable. That is, the maps $T(-, v)\colon U \to W$ defined by $u \mapsto T(u, v)$ and $T(u, -)\colon V \to W$ defined by $v \mapsto T(u, v)$ are linear.

The tensor product provides a way to combine two vector spaces into a new vector space. The simplest definition is that given vector spaces U and V over some field $\Bbbk$ the tensor product $U \otimes V$ consists of all linear combinations of elements of the form $u \otimes v$ with $u \in U$ and $v \in V$. This definition makes direct reference to elements of the vector spaces, so isn't compatible with the spirit of category theory. The useful thing about the tensor product is it combines two vector spaces into one in such a way that we can define linear maps on the new space in terms of linear maps on the original spaces. For example, the linear map sending $u \in U$ to $2u$ automatically extends to a linear map on $U \otimes V$ sending $u \otimes v$ to $2u \otimes v = (2u) \otimes v = 2(u \otimes v)$. This inspires the following definition.

Definition 5.1.2 — Tensor Product Let U, V, and W be vector spaces[a] over $\Bbbk$. The **tensor product** of U and V is a vector space, which we call $U \otimes V$, equipped with a bilinear map $f\colon U \times V \to U \otimes V$ such that for all bilinear maps $g\colon U \times V \to W$ there exists a unique linear map $h\colon U \otimes V \to W$ such that $g = h \circ f$. That is, such that

$$\begin{tikzcd}
U \times V \arrow[r, "f\text{ (bilinear)}"] \arrow[dr, "g\text{ (bilinear)}"'] & U \otimes V \arrow[d, dashed, "h\text{ (linear)}"] \\
 & W
\end{tikzcd} \tag{5.1.3}$$

commutes.

[a] this definition also holds for R-modules replacing linear with R-linear.

The important idea here is that a linear map $U \otimes V \to W$ is just as good as a bilinear map $U \times V \to W$, and since linear maps are much nicer to work with, and since $U \times V$ is not a vector space, we usually prefer to work with $U \otimes V$.

Note that not all elements of $U \otimes V$ are of the form $u \otimes v$. For example, there is no way to write $e_1 \otimes e_1 + e_2 \otimes e_2 \in V \otimes V$ in this form. This will be important later.

The tensor product also gives us a natural way to combine two linear maps, we simply act on the relevant part of the tensor product with the relevant function.

Definition 5.1.4 — Tensor Product of Linear Maps Let U, U', V and V' be vector spaces over $\Bbbk$. Let $f : U \to U'$ and $g : V \to V'$. Then the **tensor product** of f and g is defined to be the map $f \otimes g : U \otimes V \to U' \otimes V'$ given by defining $(f \otimes g)(u \otimes v) = f(u) \otimes g(v)$ and extending this to all of $U \otimes V$ through linearity.

The tensor product of vector spaces can be extended to any inner product space in a straight forward manner, the inner product simply factorises.

Definition 5.1.5 — Tensor Product of Inner Product Spaces Let $(U, \langle - | - \rangle_U)$ and $(V, \langle - | - \rangle_W)$ be inner product spaces (Hilbert spaces) over $\Bbbk$. Then $(U \otimes V, \langle - | - \rangle_{U \otimes V})$ is an inner product space (Hilbert space) with the inner product

$$\langle u \otimes v | u' \otimes v' \rangle_{U \otimes V} = \langle u | u' \rangle_U \langle v | v' \rangle_V. \tag{5.1.6}$$

5.2 Monoidal Categories: The Idea

Monoidal categories generalise the tensor product to other categories. The idea being that tensor products allow us to work with multiple objects at the same time, or in parallel. Recall that we've seen four ways to consider categories:

- physical systems and the processes occurring;
- data types and the algorithms manipulating them;
- algebraic structures and structure preserving functions;
- logical propositions and implications between them.

We can consider the idea of a monoidal category in each of these frameworks:

- independent systems evolving separately;
- running algorithms in parallel;
- products or sums of geometric structures;
- using separate proofs of P and Q to prove $P \wedge Q$.

5.3 Monoidal Categories: The Need for Specificity

Consider processes A, B, and C with some notion of parallel composition, $\otimes$. So, $A \otimes B$ is what we get by doing both processes A and B at the same time. After playing around with this idea for a while one may come upon the question of what the relationship between

$$(A \otimes B) \otimes C \qquad \text{and} \qquad A \otimes (B \otimes C) \tag{5.3.1}$$

should be. It's not right for them to be equal, since this isn't the case for vector spaces with the tensor product or for sets with the Cartesian product, which is the equivalent in **Set** as we'll see later.

For example, viewed as sets we have $((a,b),c)$ as an element of $(A\times B)\times C$, but $(a,(b,c))$ as an element of $A\times(B\times C)$. Clearly there is a map $(A\times B)\times C \to A\times(B\times C)$ given by $((a,b),c) \mapsto (a,(b,c))$. We say that this map associates these distinct objects. This map is also clearly invertible, the inverse being $(a,(b,c)) \mapsto ((a,b),c)$. So, we have $(A \times B) \times C \cong A \times (B \times C)$.

More questions arise when we think about what the processes A, B, and C are. For example there is often a concept of a trivial system where we don't do anything. We want it to be such that running this trivial system in parallel with another system is as if we weren't doing the trivial thing at all. However this is clearly not the case in, for example, a computer running a trivial algorithm, it still has to compile and run this trivial algorithm. So again we want the computation of A and the trivial system in parallel to be isomorphic to the computation of A alone.

Another question, which we won't see an answer too until later, is does order matter? That is, should we treat $A \otimes B$ and $B \otimes A$ as the equal? What about isomorphic? It is also possible that they aren't even isomorphic, although in many of the cases we're interested in they will be.

To answer these questions we have to be careful when defining a monoidal category, the task of the next section.

5.4 Monoidal Categories: The Definition

Definition 5.4.1 — Monoidal Category A **monoidal category** is formed from the following data:

1. a category, **C**;
2. a **tensor product functor**
$$- \otimes - \colon \mathbf{C} \times \mathbf{C} \to \mathbf{C}, \tag{5.4.2}$$
also called the **monoidal product**;
3. a **unit object** $I \in \mathrm{Ob}(\mathbf{C})$;
4. a natural isomorphism
$$\alpha \colon (- \otimes -) \otimes - \Rightarrow - \otimes (- \otimes -) \tag{5.4.3}$$

called the **associator** which has components

$$\alpha_{A,B,C} : (A \otimes B) \otimes C \to A \otimes (B \otimes C) \tag{5.4.4}$$

for $A, B, C \in \mathrm{Ob}(\mathbf{C})$;

5. a natural isomorphism

$$\lambda : I \otimes - \Rightarrow - \tag{5.4.5}$$

called the **left unitor** which has components

$$\lambda_A : I \otimes A \to A \tag{5.4.6}$$

for $A \in \mathrm{Ob}(\mathbf{C})$;

6. a natural isomorphism

$$\rho : - \otimes I \Rightarrow - \tag{5.4.7}$$

called the **right unitor** which has components

$$\rho_A : A \otimes A \to A \tag{5.4.8}$$

for $A \in \mathrm{Ob}(\mathbf{C})$.

This data must be such that the **triangle equation** holds, which is that

$$\begin{tikzcd}
(A \otimes I) \otimes B \arrow[rr, "\alpha_{A,I,B}"] \arrow[dr, "\rho_A \otimes \mathrm{id}_B"'] & & A \otimes (I \otimes B) \arrow[dl, "\mathrm{id}_A \otimes \lambda_B"] \\
& A \otimes B &
\end{tikzcd} \tag{5.4.9}$$

commutes for all $A, B \in \mathrm{Ob}(\mathbf{C})$, and the **pentagon equation** holds, which is that

$$\begin{tikzcd}
& (A \otimes (B \otimes C)) \otimes D \arrow[rr, "\alpha_{A,B\otimes C,D}"] & & A \otimes ((B \otimes C) \otimes D) \arrow[dr, "\mathrm{id}_A \otimes \alpha_{B,C,D}"] & \\
((A \otimes B) \otimes C) \otimes D \arrow[ur, "\alpha_{A,B,C} \otimes \mathrm{id}_D"] \arrow[drr, "\alpha_{A\otimes B,C,D}"'] & & & & A \otimes (B \otimes (C \otimes D)) \\
& & (A \otimes B) \otimes (C \otimes D) \arrow[urr, "\alpha_{A,B,C\otimes D}"'] & &
\end{tikzcd} \tag{5.4.10}$$

commutes for all $A, B, C, D \in \mathrm{Ob}(\mathbf{C})$.

This is quite a large definition so we'll break it down and hopefully this will make it less daunting. First lets look at what the data of a monoidal category tells us:

1. the category tells us the type of objects and morphisms we are considering;

2. the tensor product functor tells us how to combine two objects to get a new object, and two morphisms to get a new morphism;

3. the unit object represents a trivial process in which nothing happens;

4. the associator allows us to move brackets around and only change things by an isomorphism, which means no important change occurs;

5. the unitors allow us to remove the unit object when it is involved in a product, again only changing things by an isomorphism.

The naturality condition for α gives the commuting diagram

$$\begin{tikzcd}
(A \otimes B) \otimes C \arrow[r, "\alpha_{A,B,C}"] \arrow[d, "(f\otimes g)\otimes h"'] & A \otimes (B \otimes C) \arrow[d, "f\otimes(g\otimes h)"] \\
(A' \otimes B') \otimes C' \arrow[r, "\alpha_{A',B',C'}"'] & A' \otimes (B' \otimes C')
\end{tikzcd} \tag{5.4.11}$$

for all $f : A \to A'$, $g : B \to B'$, and $h : C \to C'$. In other words, it doesn't matter if we apply maps and then move brackets, or move brackets and then apply maps. The naturality conditions for λ and ρ can be combined into the following commuting diagram

$$\begin{tikzcd}
I \otimes A \arrow[r, "\lambda_A"] \arrow[d, "\mathrm{id}_I\otimes f"'] & A \arrow[d, "f"'] & A \otimes I \arrow[l, "\rho_A"'] \arrow[d, "f\otimes \mathrm{id}_I"] \\
I \otimes B \arrow[r, "\lambda_B"] & B & B \otimes I \arrow[l, "\rho_B"]
\end{tikzcd} \tag{5.4.12}$$

for all $f : A \to B$. In other words, we can remove the identity and then apply f, or we can do nothing to the identity while applying f and then remove the identity. Note that it is common to write I for id_I, there should be no confusion as $I \otimes f$ can only mean $\mathrm{id}_I \otimes f$, there is no way to take the tensor product of an object and a morphism. Thus, we could write the naturality condition for λ, the left square, as either $f \circ \lambda_A = \lambda_B \circ (\mathrm{id}_I \otimes f)$ or the slightly more succinct $f \circ \lambda_A = \lambda_B \circ (I \otimes f)$. In fact, this is sometimes done for all objects, not just the unit.

Now lets look at the triangle equation. It tells us how the unitors and associator combine. Starting at the top left, with $(A \otimes I) \otimes B$ there are two ways to get to $A \otimes B$. We can either remove I by applying ρ_A to the $A \otimes I$ bit, which means applying $\rho_A \otimes \mathrm{id}_B$ to $(A \otimes I) \otimes B$, or we can reassociate, by applying $\alpha_{A,I,B}$ to get $A \otimes (I \otimes B)$ then remove the unit by applying λ_B to the $I \otimes B$ bit, which means applying $\mathrm{id}_A \otimes \lambda_B$ to $A \otimes (I \otimes B)$. The triangle equality says that both of these are actually the same, that is

$$\rho_A \otimes \mathrm{id}_B = (\mathrm{id}_A \otimes \lambda_A) \circ \alpha_{A,I,B}. \tag{5.4.13}$$

Finally, lets look at the pentagon equation. This tells us how the associator can be applied to reassociate products of four objects, and it turns out that this is enough to completely specify how the associator reassociates products of any number of elements. Starting on the left we have the completely left associated $((A \otimes B) \otimes C) \otimes D$. One path we can take to reassociate is to ignore D and reassociate $(A \otimes B) \otimes C$ to $(A \otimes (B \otimes C))$, this is done using $\alpha_{A,B,C} \otimes \mathrm{id}_D$. Next we can ignore the fact that $B \otimes C$ is the product of two objects and think of it as a single object, X, so we have $(A \otimes X) \otimes D$, which we can reassociate using $\alpha_{A,X,D} = \alpha_{A,B\otimes C,D}$ to get $A \otimes ((B \otimes C) \otimes D)$. Finally we can ignore A and reassociate $(B \otimes C) \otimes D$ using $\mathrm{id}_A \otimes \alpha_{B,C,D}$ to get $A \otimes (B \otimes (C \otimes D))$. Alternatively, if we start with $((A \otimes B) \otimes C) \otimes D$ we can treat $A \otimes B$ as a single object, Y, and reassociate $(Y \otimes C) \otimes D$ with $\alpha_{Y,C,D} = \alpha_{A\otimes B,C,D}$ to get $(A \otimes B) \otimes (C \otimes D)$. Then we treat $C \otimes D$ as a single object, Z, and reassociate $(A \otimes B) \otimes Z$ using $\alpha_{A,B,Z} = \alpha_{A,B,C\otimes D}$ to get $A \otimes (B \otimes (C \otimes D))$. The pentagon equation tells us that both of these ways of reassociating from left to right give are the same, that is

$$(\mathrm{id}_A \otimes \alpha_{B,C,D}) \circ (\alpha_{A,B\otimes C,D}) \circ (\alpha_{A,B,C} \otimes \mathrm{id}_D) = \alpha_{A,B,C\otimes D} \circ \alpha_{A\otimes B,C,D}. \tag{5.4.14}$$

To summarise, the triangle equation says that all ways of removing I from $(A \otimes I) \otimes B$ to get $A \otimes B$ are the same, and the pentagon equation says that all ways of reassociating $((A \otimes B) \otimes C) \otimes D$ to get $A \otimes (B \otimes (C \otimes D))$ are the same.

Another way of viewing the definition of a monoidal category is as the (vertical[1]) **categorification** of a monoid. By this we mean that we generalise a structure applied to sets, here a monoid, to a structure on categories. Typically this is done by replacing elements with objects, functions with functors, and equality with isomorphism. Compare the definitions of a monoid, M, and a monoidal category, C:

• Elements, $a, b, c \in M$	• Objects, $A, B, C \in \mathrm{Ob}(\mathsf{C})$
• A binary function $\cdot : M \times M \to M$	• A bifunctor $\otimes : \mathsf{C} \times \mathsf{C} \to \mathsf{C}$
• $(a \cdot b) \cdot c = a \cdot (b \cdot c)$	• $(A \otimes B) \otimes C \cong A \otimes (B \otimes C)$
• Identity $e \in M$ such that $e \cdot a = a = a \cdot e$	• Unit $I \in \mathrm{Ob}(\mathsf{C})$ such that $I \otimes A \cong A \cong A \otimes I$

While these requirements seem quite complex they are incredibly powerful, and fairly easy to work with in practice, because they are so restrictive. It turns out that the monoidal product works exactly how our intuition says it does, and this is the crux of the next theorem, which we shan't prove.

Theorem 5.4.15 — Coherence Theorem for Monoidal Categories. In a monoidal category any well-typed equation built solely from associators, unitors, and their inverses holds.

By well-typed we simply mean that both sides of the equation must be morphisms with matching domain and codomain, and that any compositions are defined, so morphisms in the composition match domain to codomain. This means that we don't have to actually use the triangle and pentagon equations directly, as long as they hold we can write down pretty much anything we like as long as it makes sense and it will be true.

[1] cf. horizontal categorification, or oidification, in which we realise a particular structure within a single object category and then generalise to multi-object categories, e.g. a group is a one object category in which all morphisms are isomorphisms, and a groupoid is a category in which all morphisms are isomorphisms. Categories are already a generalisation of monoids in this sense.

5.5 Monoidal Categories: The Examples

5.5.1 Set

The obvious choice for a monoidal product on **Set** is the Cartesian product. We then need to look for a unit, something which we can take a Cartesian product with but not really change anything. One way we might come to an answer is to recognise that the unit behaves just like 1 in a product, hang on, 1 is something that we can represent as a set, a singleton, and there's our answer. Given some set A and some singleton[2] $\{\bullet\}$ the Cartesian product $A \times \{\bullet\}$ consists of elements $(a, \bullet)$ with $a \in A$, and $\{\bullet\} \times A$ consists of elements $(\bullet, a)$. From both of these we can easily recover A by just ignoring the $\bullet$. This gives us our unitors. We've already seen in Section 5.3 how $((a, b), c)) \mapsto (a, (b, c))$ defines the associator for the Cartesian product. Thus we can make the following definition.

Definition 5.5.1 — Set as a Monoidal Category The category **Set** can be promoted to a monoidal category by defining

1. the monoidal product to be the Cartesian product;
2. the unit to be some singleton, $I = \{\bullet\}$;
3. the associator to have components $\alpha_{A,B,C} : (A \times B) \times C \to A \times (B \times C)$ defined by $((a, b), c) \mapsto (a, (b, c))$;
4. the left unitor at A to be the function $\lambda_A : \{\bullet\} \times A \to A$ defined by $(\bullet, a) \mapsto a$;
5. the right unitor at A to be the function $\rho_A : A \times \{\bullet\} \to A$ defined by $(a, \bullet) \mapsto a$.

! This is not the only way we can make **Set** into a monoidal category, but it is the only one that we'll use and it is usually what people are talking about when they say **Set** is a monoidal category. Another ways of making **Set** into a monoidal category involve taking $A \otimes B = A \sqcup B$ to be the disjoint union with $I = \varnothing$. Yet another choice is $A \otimes B = A \sqcup B \sqcup (A \times B)$ with $I = \varnothing$.

This common definition of **Set** as a monoidal category is an example of a more general idea. Recall that in a category **C** a terminal object, $T \in \mathrm{Ob}(\mathbf{C})$, is an object such that for any object $A \in \mathrm{Ob}(\mathbf{C})$ there exists exactly one morphism $A \to T$. Similarly an initial object, $I \in \mathrm{Ob}(\mathbf{C})$, is an object such that for any object $A \in \mathrm{Ob}(\mathbf{C})$ there exists exactly one morphism $I \to A$. Any category, **C**, with terminal objects and products can be made into a monoidal category by taking the monoidal product to be the categorical product and the unit to be the terminal object. Similarly any category with initial objects and coproducts[3] can be made into a monoidal category by taking the tensor product to be the coproduct and the unit to be the initial object. An example of a coproduct/initial object monoidal category is **Set** with disjoint union as a monoidal product.

Lemma 5.5.2 **Set** equipped with the Cartesian product is a monoidal category.

[2] Since an isomorphism in **Set** is a bijection all sets of the same size are isomorphic, so all singletons are the same for the purpose of category theory.

[3] A **coproduct** is defined similarly to a categorical product, but with the arrows reversed.

Proof. We first demonstrate that the triangle equation holds. To do so we modify the diagram defining the triangle equation to show the elements being mapped, instead of the objects, and commutativity should be clear from this

$$\begin{tikzcd}
((a,\bullet),b) \arrow[rr, mapsto, "\alpha_{A,I,B}"] \arrow[dr, mapsto, "\rho_A\times\mathrm{id}_B"'] & & (a,(\bullet,b) \arrow[dl, mapsto, "\mathrm{id}_A\times\lambda_B"] \\
& (\rho_A(a,\bullet),\mathrm{id}_B(b)) = (a,b) = (\mathrm{id}_A(a),\lambda_B(\bullet,b)). &
\end{tikzcd} \tag{5.5.3}$$

Similarly, we can demonstrate the commutativity of the pentagon:

$$\begin{tikzcd}
\begin{array}{c}(\alpha_{A,B,C}((a,b),c),\mathrm{id}_D(D)) \\ = ((a,(b,c)),d)\end{array} \arrow[r, mapsto, "\alpha_{A,B\times C,D}"] & \begin{array}{c}\alpha_{A,B\times C,D}((a,(b,c)),d) \\ = (a,((b,c),d))\end{array} \arrow[dd, mapsto, "\mathrm{id}_A\times\alpha_{B,C,D}"] \\
(((a,b),c),d) \arrow[u, mapsto, "\alpha_{A,B,C}\times\mathrm{id}_D"] \arrow[d, mapsto, "\alpha_{A\times B,C,D}"'] & \\
\begin{array}{c}\alpha_{A\times B,C,D}(((a,b),c),d) \\ = ((a,b),(c,d))\end{array} \arrow[r, mapsto, "\alpha_{A,B,C\times D}"'] & \begin{array}{c}(\mathrm{id}_A(a),\alpha_{B,C,D}((b,c),d)) \\ = (a,(b,(c,d))) = \\ \alpha_{A,B,C\times D}((a,b),(c,d)).\end{array}
\end{tikzcd} \tag{5.5.4}$$

Finally, we need to show that the unitors and associator satisfy the naturality conditions. First consider the left unitor λ. We can see from the following diagram that the relevant naturality square commutes for all $f : A \to B$:

$$\begin{tikzcd}
(\bullet,a) \arrow[r, mapsto, "\lambda_A"] \arrow[d, mapsto, "\mathrm{id}_I\times f"'] & \lambda_A(\bullet,a) = a \arrow[d, mapsto, "f"] \\
(\mathrm{id}_I(\bullet),f(a)) = (\bullet,f(a)) \arrow[r, mapsto, "\lambda_B"'] & \lambda_B(\bullet,f(a)) = f(a).
\end{tikzcd} \tag{5.5.5}$$

Naturality of ρ is demonstrated in the same way. Naturality of α follows from

$$\begin{tikzcd}
((a,b),c) \arrow[r, mapsto, "\alpha_{A,B,C}"] \arrow[d, mapsto, "(f\times g)\times h"'] & (a,(b,c)) \arrow[d, mapsto, "f\times(g\times h)"] \\
((f(a),g(b)),h(c)) \arrow[r, mapsto, "\alpha_{A',B',C'}"'] & (f(a),(g(b),h(c)))
\end{tikzcd} \tag{5.5.6}$$

commuting for all $f : A \to A'$, $g : B \to B'$, and $h : C \to C'$. □

5.5.2 Rel

Having had success with the Cartesian product as the monoidal product we'll try the same in **Rel**. If $R \subseteq A \times B$ and $S \subset C \times D$ are relations then $R \times S \subseteq (A \times B) \times (C \times D)$ is a relation on $A \times B$ and $C \times D$ such that $(a, c)(R \times S)(b, d)$ if and only if aRb and cSd. As before the singleton, $\{\bullet\}$, acts as the unit, since as with **Set** we can easily recover A from either $\{\bullet\} \times A$ or $A \times \{\bullet\}$. The only difference is that in **Rel** our morphisms are relations, so instead of a function $\{\bullet\} \times A \to A$ giving us A back we instead define a relation on $(\{\bullet\} \times A) \times A$ where $(\bullet, a) \sim a$, we use $\sim$ here since this is an isomorphism in **Rel**, it is not an equivalence relation, since the two sets on either side of the relation are different. Similarly, the associators are given by changing the function in **Set** into a relation.

> **Definition 5.5.7 — Rel as a Monoidal Category** The category **Rel** can be promoted to a monoidal category by defining
>
> - the monoidal product to be the Cartesian product;
> - the unit to be some singleton, $I = \{\bullet\}$;
> - the associator to have components given by the relation $\sim \subseteq [(A \times B) \times C] \times [A \times (B \times C)]$ defined by $((a, b), c) \sim (a, (b, c))$;
> - the left unitor to have components given by the relation $\sim \subseteq [I \times A] \times A$ defined by $(\bullet, a) \sim a$;
> - the right unitor to have components given by the relation $\sim \subseteq [A \times I] \times A$ defined by $(a, \bullet) \sim a$.

The proof that this makes **Rel** a monoidal category is almost identical to the proof for **Set**, just replacing functions and equality with relations, so we won't repeat it.

This is one of the first examples where we see that **Rel** differs from **Set**, despite both having the Cartesian product as the monoidal product. The Cartesian product is *not* a categorical product (in the sense of Definition 4.11.4) in **Rel**, and $\varnothing$ is the terminal object of **Rel**, not $\{\bullet\}$. We'll see shortly that **Hilb** as a monoidal category is also formed using a non-categorical product, making **Rel** more like **Hilb** in this sense.

5.5.3 Hilb

It should not be surprising that $\mathbf{Vect}_{\Bbbk}$ and **Hilb**, and their finite-dimensional subcategories, can be made into monoidal categories, after all these were the models for the definition of monoidal categories we used at the start of the chapter. We'll define **Hilb** as a monoidal category of complex Hilbert spaces, but the definition is exactly the same for $\mathbf{Vect}_{\Bbbk}$ as a monoidal category, just replace $\mathbb{C}$ with $\Bbbk$ and ignore inner products.

> **Definition 5.5.8 — Hilb as a Monoidal Category** The category **Hilb** can promoted to a monoidal category by defining
>
> - the monoidal product to be the tensor product;

- the unit to be the one-dimensional Hilbert space[a] $\mathbb{C}$;
- the associator to have components $\alpha_{H,J,K} : (H \otimes J) \otimes K \to H \otimes (J \otimes K)$ defined on vectors of the form $(u \otimes v) \otimes w$ by $(u \otimes v) \otimes w \mapsto u \otimes (v \otimes w)$ and extended to all other vectors linearly;
- the left unitor to have components $\lambda_A : I \otimes A \to A$ defined on vectors of the form $1 \otimes v$ by $1 \otimes v \mapsto v$ and extended to all other vectors linearly;
- the right unitor to have components $\rho_A : A \otimes I \to A$ defined on vectors of the form $v \otimes 1$ by $v \otimes 1 \mapsto v$ and extended to all other vectors linearly.

[a] $\mathbb{C}$ is a vector space over itself, and a Hilbert space over itself with the inner product $\langle z|w \rangle = z^* w$.

As with **Set** there are other ways to make **Hilb** into a monoidal category, but this is the one that is most useful and is what we will mean when we say **Hilb** is a monoidal category.

It's worth taking a minute to discus how the associator and unitors are defined. Recall that vectors in $H \otimes J$ are of the form $\sum_i u_i \otimes v_i$ with $u_i \in H$ and $v_i \in J$. We can define a linear map $T : H \otimes J \to K$ by defining $T(u \otimes v)$ for any $u \in H$ and $v \in J$, requiring that this map is linear then completely defines T since we must then have

$$T\left(\sum_i u_i \otimes v_i\right) = \sum_i T(u_i \otimes v_i). \tag{5.5.9}$$

In the specific case of the associator we have that

$$(u_1 \otimes v_1) \otimes w_1 + (u_2 \otimes v_2) \otimes w_2 + \cdots \xmapsto{\alpha_{H,J,K}} u_1 \otimes (v_1 \otimes w_1) + u_2 \otimes (v_2 \otimes w_2) + \cdots , \tag{5.5.10}$$

We further have that if $T(u \otimes v)$ is defined for some specific $u \in H$ and any $v \in J$ then we can extend this linearly to be defined for any vector parallel to u, which we might write as λu, by $T(\lambda u \otimes v) = \lambda T(u \otimes v)$. Using this the left unitor is given by

$$\lambda_A(z \otimes u) = \lambda_A(z1 \otimes u) = z\lambda_A(1 \otimes u) = zu \tag{5.5.11}$$

where we first view $z \in \mathbb{C}$ as a vector, and then as a scalar scaling the unit vector $1 \in \mathbb{C}$. The right unitor is defined analogously.

5.5.4 More Examples

Example 5.5.12 — Monoidal Categories

- The category of (small) categories, **Cat**, is a monoidal category when equipped with the product of categories as the monoidal product and **1**, the category with a single object and only its identity morphism, as the unit. This is another example of a categorical product/terminal object monoidal category.

- The tensor product generalises to R−**Mod** and makes R−**Mod** a monoidal category with the tensor product over R, $\otimes_R$, serving as the monoidal product and R, viewed as a module over itself, serving as the unit. This also extends to algebras over R.

- The category of Abelian groups with group homomorphisms, **Ab**, is a monoidal category equipped with the monoidal product $\otimes_{\mathbb{Z}}$, which is the product of Abelian groups viewed as $\mathbb{Z}$-modules, where na for $n \in \mathbb{Z}$ and $a \in G$ is $a + \cdots + a$ if $n > 0$, e if $n = 0$, and $-a - \cdots - a$ if $n < 0$, with G viewed as an additive group and where each sum here has n terms.

- Any monoid can be viewed as a monoidal category by taking the set of objects to be the set of elements, the only morphisms to be identities, and the monoid product as the tensor product of objects.

- The category, $[\mathbf{C}, \mathbf{C}]$ of endofunctors on some category, $\mathbf{C}$, is a monoidal category with composition of functors as the monoidal product and the identity functor as the unit. This example is important in defining monads.

- The category of pointed topological spaces, $\mathbf{Top}_\bullet$, is a monoidal category with the smash product as the monoidal product and the pointed 0-sphere (that is the two point space $(\{-1, 1\}, \{\varnothing, \{-1\}, \{1\}, \{-1, 1\}\})$ with one point chosen as the base point) serving as the unit. The smash product, $\wedge$, is defined between two pointed spaces, $(X, x_\bullet)$ and $(Y, y_\bullet)$, by forming $X \times Y$ then identifying $(x, y_\bullet) \sim (x_\bullet, y)$ for all $x \in X$ and $y \in Y$, and then equipping the space $(X \times Y)/\sim$ with the quotient topology.

- Given any category, $\mathbf{C}$, we can define a free monoidal category in an analogous way to defining a free monoid, simply take the objects to be finite sequences, $A_1 \otimes \cdots \otimes A_n$, of objects in $\mathbf{C}$, then we have a morphism $A_1 \otimes \cdots \otimes A_n \to B_1 \otimes \cdots \otimes B_m$ only if $m = n$, in which case the morphism is a finite sequence of morphisms $f_1 \otimes \cdots \otimes f_n$, with $f_i : A_i \to B_i$ a morphism in $\mathbf{C}$. The monoidal product is then the concatenation of these lists, giving $(A_1 \otimes \cdots \otimes A_n) \otimes (B_1 \otimes \cdots \otimes B_m) = A_1 \otimes \cdots \otimes A_n \otimes B_1 \otimes \cdots \otimes B_m$, where now m and n may differ. The unit is the empty sequence of objects.

5.6 Interchange Law

The interchange law allows us to swap composition and the monoidal product, and is automatically satisfied by any monoidal category.

Theorem 5.6.1 — Interchange Law. Let **C** be a monoidal category with monoidal product $\otimes$. Consider the following objects morphisms in **C**:

$$A \xrightarrow{f} B \xrightarrow{g} C, \qquad \text{and} \qquad D \xrightarrow{h} E \xrightarrow{j} F. \tag{5.6.2}$$

We have

$$(g \circ f) \otimes (j \circ h) = (g \otimes j) \circ (f \otimes h). \tag{5.6.3}$$

Proof. For the proof we use the notation $\otimes(-,-) = - \otimes -$ to emphasise the functoriality of $\otimes$. This allows us to write

$$(g \circ f) \otimes (j \circ h) = \otimes(g \circ f, j \circ h). \tag{5.6.4}$$

Now consider $\mathbf{C} \times \mathbf{C}$, the domain of $\otimes$. We can identify that $(g \circ f, j \circ h) = (g, j) \circ (f, h)$, where composition on the left is in **C** and composition on the right is in $\mathbf{C} \times \mathbf{C}$. Thus in $\mathbf{C} \times \mathbf{C}$ we have

$$(A, D) \xrightarrow{(f,h)} (B, E) \xrightarrow{(g,j)} (C, F) \tag{5.6.5}$$

which we can write as

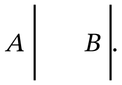
$$X \xrightarrow{p} Y \xrightarrow{q} Z. \tag{5.6.6}$$

Then we have $\otimes(q \circ p) = (\otimes(q)) \circ (\otimes(p))$ by the functoriality of $\otimes$. That is, we have $(\otimes(g, j)) \circ (\otimes(f, h))$, which we can then write as $(g \otimes j) \circ (f \otimes h)$. Putting this together we have

$$\begin{aligned}(g \circ f) \otimes (j \circ h) &= \otimes(g \circ f, j \circ h) && (5.6.7)\\ &= \otimes((g, j) \circ (f, h)) && \text{composition in } \mathbf{C} \times \mathbf{C}\\ &= (\otimes(g, j)) \circ (\otimes(f, h)) && \text{functoriality of } \otimes\\ &= (g \otimes j) \circ (f \otimes h). \quad \square\end{aligned}$$

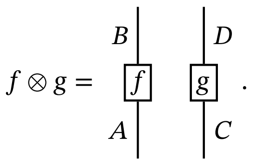
5.7 Graphical Notation

We can extend the graphical notation for categories to monoidal categories. To do so we make use of the idea that the monoidal product combines processes in parallel, and so we write the monoidal product by simply writing the two processes next to each other. For example, if we have objects A and B then, representing objects as identity morphisms, we write

$A \otimes B =$ 

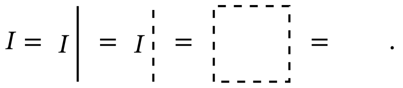
. (5.7.1)

For morphisms $f : A \to B$ and $g : C \to D$ we draw $(f \otimes g) : A \otimes C \to B \otimes D$ as



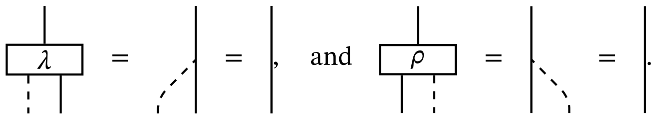
(5.7.2)

The unit, I, is drawn as the empty diagram, or as a dotted line if we want to remember that it's there:



(5.7.3)

This means we don't have to draw left or right unitors, since their action is to simply remove the unit, and we don't draw the unit any way. If we are drawing units as dashed lines then the left and right unitors can be draw as



(5.7.4)

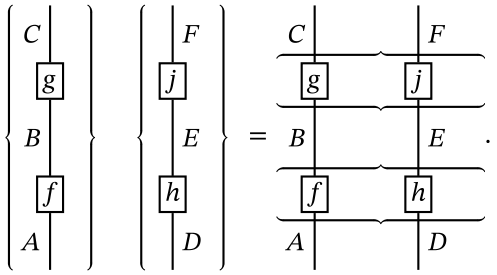
respectively. Here the wires joining is a morphism, λ or ρ, we're just not labelling it.

Similarly, we don't have to draw the associator since we're not drawing any brackets anyway. This only works because of the coherence theorem, which means that we can only build a single morphism of a given type, and so the unitors and associators don't give us any information that the domain and codomain don't.

In the graphical notation the interchange law,

$$(g \circ f) \otimes (j \circ h) = (g \otimes j) \circ (f \otimes h), \tag{5.7.5}$$

becomes



(5.7.6)

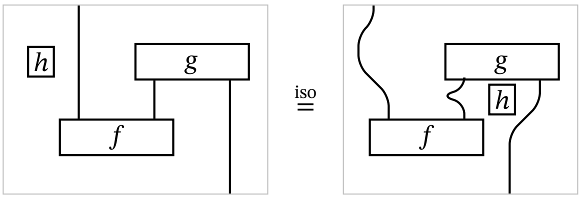
In order to proceed with calculations we need to define what it means for two diagrams to be "the same". Exact equality is too strict, since it's impossible to actually draw two identical diagrams. Besides, we want to be able to move bits of the diagrams around, such as sliding morphisms along the wires, or drawing the wires in different ways. This leads to the following definition.

Definition 5.7.7 — Planar Isotopy Two diagrams are **planar isotopic** if one can be deformed continuously into the other such that

- the diagrams remain confined to a rectangular region of the plane;
- input and output wires terminate at the bottom and top of this region, and the order they reach the edge cannot change;
- components never intersect.

This can be made more rigorous through the idea of an isotopy, which is a homotopy between the two diagrams such that at every point the diagrams remain embedded in the rectangular region of the plane in a non-intersecting way and the inputs and outputs are fixed.

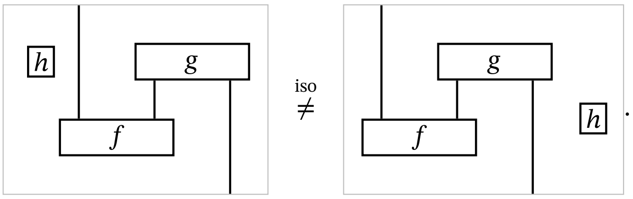
For example, consider the following diagrams involving the morphisms $f : I \to A \otimes B$, $g : B \otimes C \to I$, and $h : I \to I$. This example is a planar isotopy



(5.7.8)

since all that happened is the wires were bent and the h box moved under the f box and between the two wires, all of which can happen without ever crossing any wires or morphism boxes (which we treat as point like for the purposes of isotopies). Note that the light grey box represents the bounding region to which the diagram is confined.

This example is *not* a planar isotopy, since it is not possible to do this deformation without the h box either leaving the bounded region or crossing over one of the wires:



(5.7.9)

5.7.1 Correctness Theorem

In order for the graphical notation to be useful we need the legal moves in the graphical notation, planar isotopies, to correspond to legal moves in monoidal categories, the axioms. Fortunately this is the case, leading to the following theorem, which we won't prove.

Theorem 5.7.10 — Correctness of the Graphical Notation for Monoidal Categories. Any well-formed equation of the form $f = g$ for morphisms f and g in a monoidal category follows from the axioms of monoidal categories if and only if it holds in the graphical notation up to planar isotopy.

Note that by "well-formed" we mean that f and g have the same (co)domains.

There are two parts to this theorem, known as soundness and completeness. For morphisms f and g in some monoidal category define

- $P(f,g)$ to be the proposition "under the axioms of a monoidal category $f = g$";
- $Q(f,g)$ to be the proposition "f and g are planar isotopic when expressed in the graphical notation".

Then **soundness** is the assertion that $P(f,g) \implies Q(f,g)$ for all such f and g. This is relatively easy to check, we simply check that every axiom of a monoidal category when translated into the graphical notation becomes a planar isotopy, and this is pretty trivial since most aspects of a monoidal category, units, unitors, and associators, simply aren't drawn in the graphical notation. The converse is **completeness** which is the assertion that $Q(f,g) \implies P(f,g)$ for all such f and g. This is harder to prove, to do so we must show that any planar isotopy can be generated by a finite set of moves, each of which obeys the axioms of a monoidal category, and that combining these moves also obeys the axioms of a monoidal category.

5.8 States

5.8.1 States

To follow the spirit of category theory we shouldn't talk about elements of sets, or particular vectors in some vector space. However, often we will have reason to want to refer to these things. Fortunately it is possible to do so using morphisms.

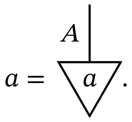
Consider some set, A, from which we want to pick an element, a. We can identify this selection of a single element with a function $f : \{\bullet\} \to A$ such that $f(\bullet) = a$. Similarly, consider some vector space, V, over $\Bbbk$ from which we want to pick a vector, v. We can identify this selection of a single element with a linear map $T : \Bbbk \to V$ such that $T(1) = v$, and then $T(k) = T(k1) = kT(v) = kv$ for all $k \in \Bbbk$. This way of getting around talking about the substructure of objects leads to the following definition.

> **Definition 5.8.1 — State** Consider a monoidal category with unit I and object A. A **state** of A is a morphism $I \to A$.

So the function f and the linear map T are states of A and V respectively. Another example is from **Rel**, where a state is a relation $R \subseteq \{\bullet\} \times A$. Since this will be of the form $R = \{(\bullet, a), (\bullet, b), \dots\}$ for $a, b, \dots, \in A$ we see that a state picks out a subset, $\{a, b, \dots\} \subseteq A$.

Now consider a Hilbert space, H. A state is a linear map $\mathbb{C} \to H$. We can define a state, $T_v : \mathbb{C} \to H$ for each $v \in H$ through $T_v(1) = v$, and so $T_v(z) = T_v(z1) = zT_v(1) = zv$, but this is exactly the definition of a ket (Definition 4.4.51), $T_v = |v\rangle$, so a state of a Hilbert space is exactly a ket. This is, after all, why we chose the word state, since we ultimately want to consider states of some quantum system, which are usually considered to be kets in some Hilbert space.

Graphically a state is a morphism $I \to A$, so since we don't draw I it looks like the morphism takes no input. For this reason we change the shape of the box to be a triangle. For example, the state $a : I \to A$ is represented as



(5.8.2)

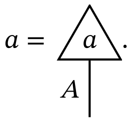
5.8.2 Effects

Often in category theory after making a definition we should ask what happens if we reverse the arrows in the definition. In the case of states this leads to the following definition.

> **Definition 5.8.3 — Effect** Consider a monoidal category with unit I and object A. An **effect** on A is a morphism $A \to I$.

Consider a Hilbert space, H. Then an effect is a linear map $H \to \mathbb{C}$. We can define one such map for each $v \in H$, $T_v : H \to \mathbb{C}$ and $T_v(w) = \langle v|w\rangle$, but this is exactly the definition of the bra (Definition 4.4.51), so $T_v = \langle v|$. Thus we can interpret an effect as an observation of a quantum system.

An effect is drawn similarly to a state, the effect $a : A \to I$ is drawn as



(5.8.4)

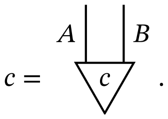
5.8.3 Joint States

Next we have to ask how the notion of a state combines with the monoidal product. This leads to the following definition.

> **Definition 5.8.5 — Joint State** Consider a monoidal category with unit I and objects A and B. A **joint state** of A and B is a morphism $I \to A \otimes B$.

This definition generalises to any number of objects.

We might draw a joint effect $c : I \to A \times B$ as



(5.8.6)

After playing around with this definition for a while one might recognise two types of joint effect, those which factor and those which don't.

> **Definition 5.8.7 — Product and Entangled State** Consider a monoidal category with unit I and objects A and B. A joint state, $c : I \to A \otimes B$, is a

product state if it is of the form

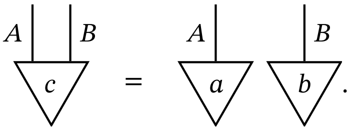
$$I \xrightarrow{\lambda_I^{-1}} I \otimes I \xrightarrow{a \otimes b} A \otimes B. \tag{5.8.8}$$

That is, we can write $c = (a \otimes b) \circ \lambda_I^{-1}$ as a product of two states, $a : I \to A$ and $b : I \to B$.
A joint state which cannot be written in this form is called an **entangled state**.

Note that $\lambda_I = \rho_I$ in any monoidal category, so we could equally well have defined a product state to be of the form

$$I \xrightarrow{\rho_I^{-1}} I \otimes I \xrightarrow{a \otimes b} A \otimes B \tag{5.8.9}$$

Graphically if c is a product state then



(5.8.10)

Example 5.8.11 In **Set** states are elements, so

- joint states of A and B are elements of $A \times B$;
- product states are elements $(a, b) \in A \times B$;
- entangled states don't exist.

In **Rel** states are subsets, so

- joint states of A and B are subsets of $A \times B$;
- product states are "square" subsets of the form $U \times V \subseteq A \times B$ where $U \subseteq A$ and $V \subseteq B$, for more details see Figure 5.1;
- entangled states are any subsets of $A \times B$ not of this form.

In **Hilb** states are vectors, so

- joint states of H and K are vectors in $H \otimes K$;
- product states are factorisable states, i.e. states which can be written as $u \otimes v$ for $u \in H$ and $v \in K$;
- entangled states are states which aren't factorisable, which are exactly the entangled states of quantum mechanics, such as[a] $|01\rangle + |10\rangle$.

[a] here we identify a vector with its ket, and we write implicit tensor products, meaning $|ab\rangle = |a\rangle \otimes |b\rangle \leftrightarrow a \otimes b$, the states $|0\rangle$ and $|1\rangle$ can be taken as two orthonormal basis vectors in $\mathbb{C}^2$.

This demonstrates another way in which **Rel** is more like **Hilb** than **Set**, both **Rel** and **Hilb** have entangled states.

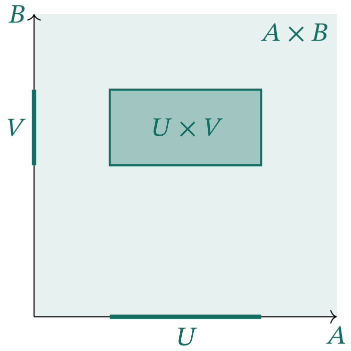


Figure 5.1: An example of a “square” subset of $\mathbb{R} \times \mathbb{R}$, identified with the plane, given $U, V \subseteq \mathbb{R}$ we get a rectangle $U \times V$ in the plane.

Six

Scalars

6.1 Motivation

Each vector space comes equipped, by definition, with a field of scalars. Most of the time this field is either $\mathbb{R}$ or $\mathbb{C}$, and we'll consider the complex case here. The scalars and the vector space come with a map, called scalar multiplication, which takes a scalar and a vector and produces a vector, $\cdot : \Bbbk \times V \to V$, $(a, v) \mapsto av$. We can similarly define a map between (bounded) linear maps, given by[1] $\cdot : \Bbbk \times \hom(V, W) \to \hom(V, W), (a, f) \mapsto \lambda \cdot f$ where if $f : V \to W$ is a (bounded) linear map then we define $a \cdot f$ to be the map $(a \cdot f) : V \to W$ defined by $(a \cdot f)(v) = af(v)$ where on the right hand side multiplication is just scalar multiplication.

This structure is very important, and indeed is part of the definition of a vector space. However, as stated here this requires us to talk of individual vectors, which is not in the spirit of category theory. In the next section we'll see how the field of scalars can be talked about in a categorical way, and in the section after that we'll generalise our observation to any monoidal category.

6.2 Scalars in **Hilb**

Lets work in **Hilb** and try to discus scalars, which we'll take as elements of $\mathbb{C}$, in a categorical way. We don't want to talk of individual elements of the field, or of individual vectors. Instead notice that given some scalar, $a \in \mathbb{C}$, can be identified with a linear map $\mathbb{C} \to \mathbb{C}$ defined by $1 \mapsto a$, which full defines the map through linearity. Let's call this map f_a.

Now consider composing these two maps of this type, if $a, b \in \mathbb{C}$, we have $(f_b \circ f_a)(1) = f_b(f_a(1)) = f_b(a1) = af_b(1) = ab$, so composition is multiplication in $\mathbb{C}$.

What about multiplication of bounded linear maps? Consider the bounded linear map $T : H \to K$. Given some $a \in \mathbb{C}$ we can consider

$$\lambda_K \circ (f_a \otimes T) \circ \lambda_H^{-1} : H \to K, \qquad H \xrightarrow{\lambda_H^{-1}} \mathbb{C} \otimes H \xrightarrow{f_a \otimes T} \mathbb{C} \otimes K \xrightarrow{\lambda_K} K, \quad (6.2.1)$$

this takes in some object in H, maps it to an object in $\mathbb{C} \otimes H$, applies $f_a \otimes T$, mapping it to an object in $\mathbb{C} \otimes K$, then maps this to an object in K. Let's consider an example, we have

$$\begin{aligned} v &\xmapsto{\lambda_H^{-1}} 1 \otimes v \xmapsto{f_a \otimes T} f_a(1) \otimes T(v) = a \otimes T(v) \\ &= a(1 \otimes T(v)) \xmapsto{\lambda_K} \lambda_K(a(1 \otimes T(v))) = a\lambda_K(1 \otimes T(v)) = aT(v). \quad (6.2.2) \end{aligned}$$

[1] We use hom here to avoid specifying exactly which category of vector spaces we are working in, it could be $\mathbf{Vect}_{\Bbbk}$, $\mathbf{FVect}_{\Bbbk}$, **Hilb**, or **FHilb**.

This shows that $\lambda_K \circ (f_a \otimes T) \circ \lambda_H^{-1}$ is $a \cdot T$ where $a \cdot T$ is the product of a scalar and a linear map in the normal sense. Putting this all together in a diagram we can define $a \cdot T$ to be the unique map such that the following diagram commutes

$$\begin{tikzcd}
H \arrow[r, "a \cdot T"] \arrow[d, "\lambda_A^{-1}"'] & K \\
\mathbb{C} \otimes H \arrow[r, "f_a \otimes T"'] & \mathbb{C} \otimes K. \arrow[u, "\lambda_B"']
\end{tikzcd} \tag{6.2.3}$$

We now drop the notation f_a for the map $f_a : \mathbb{C} \to \mathbb{C}$ defined by $f_a(1) = a$ and instead just write $a : \mathbb{C} \to \mathbb{C}$ so $a(1) = a$, which seems like a very natural notation. Then the product of a scalar and a map, $a \cdot T$, is the unique map such that the following diagram commutes

$$\begin{tikzcd}
H \arrow[r, "a \cdot T"] \arrow[d, "\lambda_A^{-1}"'] & K \\
\mathbb{C} \otimes H \arrow[r, "a \otimes T"'] & \mathbb{C} \otimes K. \arrow[u, "\lambda_B"']
\end{tikzcd} \tag{6.2.4}$$

From this we can see that the tensor product with an endomorphism on $\mathbb{C}$ is pretty much the same as multiplying a scalar and a map. The only difference is that for domains to match up we need to insert an extra object, and the easiest way to do this is with the unitors.

In the next section we'll take our work here in **Hilb** and generalise it to an arbitrary monoidal category.

6.3 Scalars

Definition 6.3.1 — Scalar Let **C** be a monoidal category with monoidal product $\otimes$ and unit I. Then a **scalar** is an endomorphism of the unit. That is, a scalar is a morphism in $\mathbf{C}(I, I)$, that is a morphism $I \to I$.

We know that $wz = zw$ for $w, z \in \mathbb{C}$, and it would be good to check if this generalises to our definition of scalars, and it does.

Lemma 6.3.2 — Scalars Commute Let **C** be a monoidal category with unit I. Then given any two scalars $a, b : I \to I$ we have $a \circ b = b \circ a$, that is scalars commute.

Proof. We want to show that the following diagram commutes

$$\begin{tikzcd}
I \arrow[r, "a"] \arrow[d, "b"'] & I \arrow[d, "b"] \\
I \arrow[r, "a"'] & I.
\end{tikzcd} \tag{6.3.3}$$

The naturality square for the left unitor, λ, is

$$\begin{tikzcd} I \otimes A \arrow[r, "\lambda_A"] \arrow[d, "\mathrm{id}_I \otimes f"'] & A \arrow[d, "f"] \\ I \otimes B \arrow[r, "\lambda_B"'] & B. \end{tikzcd} \tag{6.3.4}$$

This commutes for all $f \in \mathbf{C}(A, B)$ by the definition of naturality. Consider this square specialised to the case where $A = B = I$:

$$\begin{tikzcd} I \otimes I \arrow[r, "\lambda_I"] \arrow[d, "\mathrm{id}_I \otimes f"'] & I \arrow[d, "f"] \\ I \otimes I \arrow[r, "\lambda_I"'] & I. \end{tikzcd} \tag{6.3.5}$$

Now we have $f \in \mathbf{C}(I, I)$, so f is some scalar. The definition of a monoidal category is that λ is a natural isomorphism, so λ_I^{-1} exists. We can the write this diagram as

$$\begin{tikzcd} I \otimes I \arrow[d, "\mathrm{id}_I \otimes f"'] & I \arrow[l, "\lambda_I^{-1}"'] \arrow[d, "f"] \\ I \otimes I \arrow[r, "\lambda_I"'] & I. \end{tikzcd} \tag{6.3.6}$$

For later use it is useful to have the rotated version of this diagram:

$$\begin{tikzcd} I \arrow[r, "f"] \arrow[d, "\lambda_I^{-1}"'] & I \\ I \otimes I \arrow[r, "\mathrm{id}_I \otimes f"'] & I \otimes I. \arrow[u, "\lambda_I"'] \end{tikzcd} \tag{6.3.7}$$

Also note that by the coherence theorem we have $\lambda_I = \rho_I$. The naturality square for ρ_I, rotated, gives

$$\begin{tikzcd} I \arrow[r, "f"] & I \\ I \otimes I \arrow[u, "\rho_I"] \arrow[r, "f \otimes \mathrm{id}_I"'] & I \otimes I, \arrow[u, "\rho_I"'] \end{tikzcd} \tag{6.3.8}$$

and the same logic as we applied to the left unitor case gives

$$\begin{tikzcd} I \arrow[r, "f"] \arrow[d, "\rho_I^{-1}"'] & I \arrow[d, "\rho_I^{-1}"] \\ I \otimes I \arrow[r, "f \otimes \mathrm{id}_I"'] & I \otimes I. \end{tikzcd} \tag{6.3.9}$$

Consider the following objects and morphisms in $\mathbf{C}$:

$$A \xrightarrow{f} B \xrightarrow{g} C, \qquad \text{and} \qquad D \xrightarrow{h} E \xrightarrow{j} F. \tag{6.3.10}$$

Now consider the interchange law (Theorem 5.6.1),

$$(g \circ f) \otimes (j \circ h) = (g \otimes j) \circ (f \otimes h). \tag{6.3.11}$$

We have here

$$A \xrightarrow{g \circ f} C, \quad D \xrightarrow{j \circ h} F, \tag{6.3.12}$$

$$A \otimes D \xrightarrow{(j \circ h) \otimes (g \circ f)} C \otimes F, \tag{6.3.13}$$

$$A \otimes D \xrightarrow{f \otimes h} B \otimes E \xrightarrow{g \otimes j} C \otimes F. \tag{6.3.14}$$

$$\tag{6.3.15}$$

This implies commutativity of the following diagram

$$\begin{tikzcd}
A \otimes D \arrow[r, "f \otimes h"] \arrow[d, "\mathrm{id}_A \otimes (j \circ h)"'] & B \otimes E \arrow[d, "g \otimes j"] \\
A \otimes F \arrow[r, "(g \circ f) \otimes \mathrm{id}_F"'] & C \otimes F.
\end{tikzcd} \tag{6.3.16}$$

Now specialise this to the case where $A = B = C = D = E = F = I$, $h = g = \mathrm{id}_I$, then we have

$$\begin{tikzcd}
I \otimes I \arrow[r, "f \otimes \mathrm{id}_I"] \arrow[d, "\mathrm{id}_I \otimes j"'] & I \otimes I \arrow[d, "\mathrm{id}_I \otimes j"] \\
I \otimes I \arrow[r, "f \otimes \mathrm{id}_I"'] & I \otimes I.
\end{tikzcd} \tag{6.3.17}$$

Now $f, j \colon I \to I$ are scalars.
We can now take the commutative diagrams we have constructed and paste them together setting $f = a$ and $j = b$ to get the following cube:

$$\begin{tikzcd}
I \arrow[rr, "a"] \arrow[dr, "b"] \arrow[dd, "\lambda_I^{-1} \mid \rho_I^{-1}"'] & & I \arrow[dr, "b"] \arrow[dd, "\lambda_I^{-1} \mid \rho_I^{-1}"'] & \\
& I \arrow[rr, "a"] & & I \arrow[dd, "\lambda_I \mid \rho_I"'] \\
I \otimes I \arrow[rr, "a \otimes \mathrm{id}_I"] \arrow[dr, "\mathrm{id}_I \otimes b"'] & & I \otimes I \arrow[dr, "\mathrm{id}_I \otimes b"'] & \\
& I \otimes I \arrow[uu, "\lambda_I \mid \rho_I"'] \arrow[rr, "a \otimes \mathrm{id}_I"'] & & I \otimes I.
\end{tikzcd} \tag{6.3.18}$$

The sides of this cube all commute, since they are one of the modified naturality squares for either λ_I or ρ_I from the start of the proof. The bottom of the cube commutes as it is the diagram we derived from the interchange law. Thus, the top of the cube must commute, since any path along the top can be replaced with a path through the rest of the cube, which must necessarily all be the same. The top of the cube is exactly the diagram which states

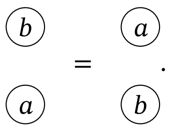
$$a \circ b = b \circ a \tag{6.3.19}$$

for all $a, b \in \mathbf{C}(I, I)$. □

Since a scalar is a map $I \to I$ graphically it is represented by a box with no wires. Because of this we'll represent scalars by a circle:

$$a = \textcircled{a}. \tag{6.3.20}$$

Commutativity of scalar multiplication becomes



(6.3.21)

Clearly these two diagrams are isotopic, so the correctness of the graphical notation proves commutativity of scalars, and does so much quicker than the proof above.

6.4 Scalar Multiplication

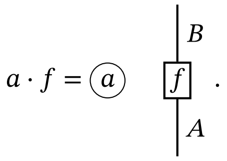
Definition 6.4.1 — Scalar Multiplication Let $\mathbf{C}$ be a monoidal category with unit I, left unitor λ, right unitor ρ, and monoidal product $\otimes$. Given a morphism $f : A \to B$, and a scalar $a : I \to I$ both in $\mathbf{C}$ then we define the **left scalar multiplication** of f by a, denoted $a \cdot f : A \to B$, to be the morphism

$$\lambda_B \circ (a \otimes f) \circ \lambda_A^{-1} : A \to B. \tag{6.4.2}$$

Put another way, the left scalar multiplication of f by a is the unique morphism $a \cdot f$ such that the following commutes

$$\begin{tikzcd}
A \arrow[r, "a \cdot f"] \arrow[d, "\lambda_A^{-1}"'] & B \\
I \otimes A \arrow[r, "f \otimes a"'] & I \otimes B. \arrow[u, "\lambda_B"']
\end{tikzcd} \tag{6.4.3}$$

Graphically, we have



(6.4.4)

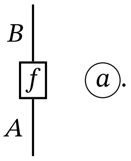
We can similarly define right scalar multiplication, $f \cdot a$, as

$$f \cdot a = \rho_B \circ (f \otimes a) \circ \rho_A^{-1}, \tag{6.4.5}$$

so that the following diagram commutes

$$\begin{tikzcd} A \arrow[r, "f \cdot a"] \arrow[d, "\rho_A^{-1}"'] & B \\ A \otimes I \arrow[r, "a \otimes f"'] & B \otimes I. \arrow[u, "\rho_B"'] \end{tikzcd} \tag{6.4.6}$$

This is then drawn as



(6.4.7)

Note that, in general, left and right scalar multiplication are not the same. In particular, the two diagrams are not isotopic as to move a to the other side of f we either have to cross a wire or leave the bounding box which the wires are attached to. We will consider only left scalar multiplication, which we'll simply call scalar multiplication. The statements we make will all transfer to right scalar multiplication in an obvious way.

What properties should scalar multiplication have? We look to **Hilb** as an example. In **Hilb** we have $1 \in \mathbb{C}$, corresponding to the map $1 \mapsto 1$, which is just the identity $\mathrm{id}_{\mathbb{C}}$. So the properties of 1 in a field, i.e. being a multiplicative identity, should extend to id_I in some arbitrary monoidal category.

Lemma 6.4.8 Let **C** be a monoidal category with unit I and $f : A \to B$ a morphism of **C**. Then $\mathrm{id}_I \cdot f = f$.

Proof. The definition of scalar multiplication, $\mathrm{id}_I \cdot f$, is that it makes

$$\begin{tikzcd} A \arrow[r, "\mathrm{id}_I \cdot f"] \arrow[d, "\lambda_A^{-1}"'] & B \\ I \otimes A \arrow[r, "f \otimes a"'] & I \otimes B. \arrow[u, "\lambda_B"'] \end{tikzcd} \tag{6.4.9}$$

commute. The hypothesis, that $\mathrm{id}_I \cdot f = f$ is then that the

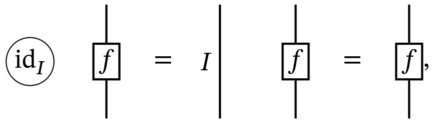
$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "\lambda_A^{-1}"'] & B \\ I \otimes A \arrow[r, "f \otimes a"'] & I \otimes B. \arrow[u, "\lambda_B"'] \end{tikzcd} \tag{6.4.10}$$

commutes. Replacing λ_A^{-1} with λ_A and reversing the arrow doesn't change the commutativity of the diagram. Hence, the diagram above commutes exactly when

$$\begin{tikzcd}
A \arrow[r, "a\cdot f"] & B \\
I \otimes A \arrow[u, "\lambda_A"] \arrow[r, "f\otimes a"'] & I \otimes B. \arrow[u, "\lambda_B"']
\end{tikzcd} \tag{6.4.11}$$

commutes. This is just the naturality square of λ, so commutes by definition, so the fist diagram commutes, so $\mathrm{id}_I \cdot f = f$. □

A proof of this same fact using the graphical notation is simply



(6.4.12)

since we draw the unit object, which corresponds to id_I in the graphical notation, as the empty diagram.

An obvious question we can ask is how do we multiply two scalars, since we should have some notion of multiplication in the field of scalars. Well it turns out that this is quite simple.

Lemma 6.4.13 Let $\mathbf{C}$ be a monoidal category with scalars a and b. Then $a \cdot b = a \circ b$.

Proof. This follows by commutativity of Equation (6.3.18), which starting at the top back left, moving forward then right gives $a \circ b$, but going down, forward, right, and up gives

$$\lambda_I \circ (a \otimes \mathrm{id}_I) \circ (\mathrm{id}_I \otimes b) \circ \lambda_I^{-1}. \tag{6.4.14}$$

Applying the interchange law to the middle of this morphism we have

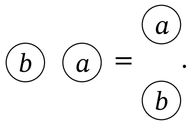
$$(a \otimes \mathrm{id}_I) \circ (\mathrm{id}_I \otimes b) = (a \circ \mathrm{id}_I) \otimes (\mathrm{id}_I \circ b) = a \otimes b. \tag{6.4.15}$$

Hence, we have

$$a \circ b = \lambda_I \circ (a \otimes b) \circ \lambda_I^{-1} = a \cdot b. \tag{6.4.16}$$

□

A proof of this same fact using the graphical notation is simply



(6.4.17)

An important property of scalar multiplication is a compatibility between scalar multiplication and field multiplication. By this we mean if we want to multiply a map by two scalars we can either multiply the map individually by both scalars, or we can compute the product of the two scalars and then multiply this by the map.

Lemma 6.4.18 Let $\mathbf{C}$ be a monoidal category with scalars a and b and a morphism f. Then $a \cdot (b \cdot f) = (a \cdot b) \cdot f = (a \circ b) \cdot f$.

Proof. Consider $a \cdot (b \cdot f)$. Expanding the definition of $\cdot$ we have

$$b \cdot f = \lambda_B \circ (b \otimes f) \circ \lambda_A^{-1}, \tag{6.4.19}$$

$$a \cdot (b \cdot f) = \lambda_B \circ (a \otimes (b \cdot f)) \circ \lambda_A^{-1}. \tag{6.4.20}$$

Hence,

$$a \cdot (b \cdot f) = \lambda_B \circ (a \otimes (\lambda_B \circ (b \otimes f) \circ \lambda_A^{-1})) \circ \lambda_A^{-1}. \tag{6.4.21}$$

Similarly, we have

$$a \cdot b = \lambda_I \circ (a \otimes b) \circ \lambda_I^{-1}, \tag{6.4.22}$$

$$(a \cdot b) \cdot f = \lambda_B \circ ((\lambda_I \circ (a \otimes b) \circ \lambda_I^{-1}) \otimes f) \circ \lambda_A^{-1}. \tag{6.4.23}$$

Now, consider the following diagram, which commutes by the coherence theorem:

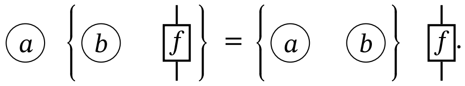
$$\begin{tikzcd}
A \arrow[r, "\mathrm{id}_A"] \arrow[d, "\lambda_A^{-1}"'] & A \arrow[r, "a\cdot(b\cdot f)"] \arrow[d, "\lambda_A^{-1}"] & B \arrow[r, "\mathrm{id}_B"] & B \\
I \otimes A \arrow[r, "\mathrm{id}_{I\otimes A}"] \arrow[drr, phantom] \arrow[ddr, "\lambda_I^{-1}\otimes \mathrm{id}_A"'] & I \otimes A \arrow[r, "a\otimes(b\cdot f)"] \arrow[d, "\mathrm{id}_I\otimes\lambda_A^{-1}"] & I \otimes B \arrow[r, "\mathrm{id}_{I\otimes B}"] \arrow[u, "\lambda_B"] & I \otimes B \arrow[u, "\lambda_B"'] \\
 & I \otimes (I \otimes A) \arrow[r, "a\otimes(b\otimes f)"] \arrow[d, "\alpha_{I,I,A}^{-1}"] & I \otimes (I \otimes B) \arrow[u, "\mathrm{id}_I\otimes\lambda_B"] & \\
 & (I \otimes I) \otimes A \arrow[r, "(a\otimes b)\otimes f"'] & (I \otimes I) \otimes B \arrow[u, "\alpha_{I,I,B}"] \arrow[uur, "\lambda_I\otimes\mathrm{id}_B"'] &
\end{tikzcd} \tag{6.4.24}$$

The expanded form of $a \cdot (b \cdot f)$ corresponds to starting at the top left, then applying id_A immediately, which does nothing, going right we then apply λ_A^{-1}, next we want to apply $a \otimes (b \cdot f)$, which can be done be going down, right, then up using the definition of scalar multiplication, finally going up again is the last step and takes us to B, to which we can apply id_B for free to move to the far right B.

On the other hand, the expanded form of $(a \cdot b) \cdot f$ corresponds to starting at the top left, going down applying λ_A^{-1}, then we want to apply $(\lambda_I \circ (a \otimes b) \circ \lambda_I^{-1}) \otimes f$. This can be done by first applying $\lambda_I^{-1} \otimes \mathrm{id}_A$ to apply the λ_I^{-1} part, then we can apply the $a \otimes b$ and f parts by applying $(a \otimes b) \otimes f$, next we can apply the λ_I part by applying $\lambda_I \otimes \mathrm{id}_B$. Finally we apply λ_B to get to the top right B.

Since both of these paths start and end at the same point they must correspond to the same morphism. □

The following is a proof in the graphical calculus:



(6.4.25)

Lemma 6.4.26 Let $\mathbf{C}$ be a (locally small) monoidal category with unit I. Then $\mathbf{C}(I, I)$ forms a monoid under scalar multiplication.

Proof. The product of two scalars is again a scalar. The monoid identity is $\mathrm{id}_I \in \mathbf{C}(I, I)$, since $\mathrm{id}_I \cdot a = a$ for all $a \in \mathbf{C}(I, I)$ by Lemma 6.4.8. Lemma 6.4.18 proves that $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all scalars $a, b, c \in \mathbf{C}(I, I)$, so scalar multiplication is associative. □

The final property we'll prove for now is a version of the interchange law.

Lemma 6.4.27 Let $\mathbf{C}$ be a monoidal category with scalars a and b and morphisms $f : A \to B$ and $g : B \to C$. Then $(b \cdot g) \circ (a \cdot f) = (b \circ a) \cdot (g \circ f)$.

Proof. The left hand side is

$$(\lambda_C \circ (b \otimes g) \circ \lambda_B^{-1}) \circ (\lambda_B \circ (a \otimes f) \circ \lambda_A^{-1}). \tag{6.4.28}$$

Using associativity of composition to change around the brackets and $\lambda_B^{-1} \circ \lambda_B = \mathrm{id}_B$, which we don't need to include in a chain of compositions, this becomes

$$\lambda_C \circ (b \otimes g) \circ (a \otimes f) \circ \lambda_A^{-1}. \tag{6.4.29}$$

The right hand side is

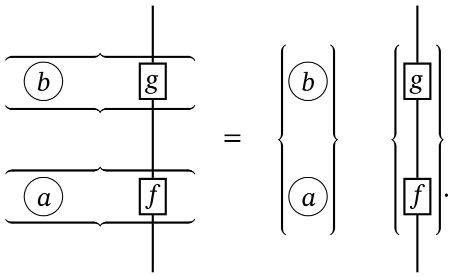
$$\lambda_C \circ ((b \circ a) \otimes (g \circ f)) \circ \lambda_A^{-1}. \tag{6.4.30}$$

Applying the interchange law to this we get

$$\lambda_C \circ (b \otimes g) \circ (a \otimes f) \circ \lambda_A^{-1}. \tag{6.4.31}$$

Clearly this is the same result as we got from the left hand side, so the two are equal. □

The following is a proof in the graphical calculus:



(6.4.32)

6.5 Examples

In **Set** scalars are functions $\{\bullet\} \to \{\bullet\}$. There is only one such function, $\bullet \mapsto \bullet$, which is simply $\mathrm{id}_{\{\bullet\}}$. Thus there is a single scalar in **Set**, and it is the trivial $\mathrm{id}_{\{\bullet\}}$ which is such that $\mathrm{id}_{\{\bullet\}} \cdot f = f$ for all $f \in \mathbf{Set}(A, B)$. This makes sense since sets don't come equipped with a scalar structure.

In **Rel** scalars are relations $R \subseteq \{\bullet\} \times \{\bullet\}$. There are two scalars, `true` $= \{(\bullet, \bullet)\} = \mathrm{id}_{\{\bullet\}}$ and `false` $= \varnothing$. Computing all possible products of these two scalars we see that clearly any product involving `false` must necessarily just be `false`, since if one of the sets in the definition of relation composition is empty then the result is also empty. We also have $\{(\bullet, \bullet)\} \circ \{(\bullet, \bullet)\} = \{(\bullet, \bullet)\}$. Hence, we have

$$\begin{array}{c|cc} \cdot & \texttt{true} & \texttt{false} \\ \hline \texttt{true} & \texttt{true} & \texttt{false} \\ \texttt{false} & \texttt{false} & \texttt{false} \end{array} \; . \tag{6.5.1}$$

This is exactly the result we would get if `true` and `false` were booleans and $\cdot$ was logical conjunction (and). This is why we called the `true` and `false` in the first place. Further, for any relation $R \subseteq A \times B$ we have `true` $\cdot\, R = R$ and `false` $\cdot\, R = \varnothing =$ `false`, so `true` and `false` behave a bit like 1 and 0 for scalar multiplication.

In **Hilb** scalars are linear maps $\mathbb{C} \to \mathbb{C}$. These are uniquely determined by where they send 1, so we can identify $z \in \mathbb{C}$ with the map $1 \mapsto z$, and then scalar multiplication behaves exactly as expected, after all we did base scalars on **Hilb** to start with.

Seven

Braided and Symmetric Monoidal Categories

7.1 Braided Monoidal Categories: The Idea

Recall that we've seen four ways to consider monoidal categories categories:

- physical systems and the processes occurring with independent systems evolving separately;
- data types and the algorithms with algorithms running in parallel;
- algebraic structures and structure preserving functions with products or sums;
- logical propositions and implications between them with separate proofs of P and Q proving $P \wedge Q$.

In all of these there is a sense in which order doesn't matter, that $A \otimes B$ and $B \otimes A$ are, if not the same, at least similar:

- it doesn't matter what order we place the independent systems, they will evolve to the states so the only difference is the order we place them;
- it doesn't matter if we run algorithm 1 on core 1 and algorithm 2 on core 2 or algorithm 1 on core 2 and algorithm 2 on core 1, the only difference is in how the algorithms run, the results are the same;
- it often doesn't matter what order we take a product, for example, if V and W are vector spaces then $V \otimes W$ and $W \otimes V$ are not the same, but we can turn one into the other through the mapping $v \otimes w \mapsto w \otimes v$;
- if we prove P and Q separately then we can prove $P \wedge Q$ or $Q \wedge P$.

The most important example here is that of the vector spaces, while, as is often the case, equality between $A \otimes B$ and $B \otimes A$ is too strict it is often enough to just have a map between them. The only question then is what properties should this map poses? Well, as usual a map replacing equality should be an isomorphism. We also want the isomorphism relating $A \otimes B$ and $B \otimes A$ to be related to the isomorphism relating $C \otimes D$ and $D \otimes C$. We want a family of such isomorphisms all related in some way, hang on, we have families of related isomorphisms, they're called natural isomorphisms.

7.2 Braided Monoidal Category: The Definition

Definition 7.2.1 — Braided Monoidal Category A **braided monoidal category** is a monoidal category, **C** equipped with a natural isomorphism, called the **braiding**,

$$\sigma : \; - \otimes - \Rightarrow - \otimes - \tag{7.2.2}$$

which for objects A and B has components

$$\sigma_{A,B} : A \otimes B \to B \otimes A. \tag{7.2.3}$$

These must satisfy the **hexagon equations**, that is the braiding should make the following diagrams commute:

$$\begin{tikzcd}
& A \otimes (B \otimes C) \arrow[r, "\sigma_{A,B\otimes C}"] \arrow[dl, "\alpha^{-1}_{A,B,C}"'] & (B \otimes C) \otimes A & \\
(A \otimes B) \otimes C \arrow[dr, "\sigma_{A,B}\otimes \mathrm{id}_C"'] & & & B \otimes (C \otimes A) \arrow[ul, "\alpha^{-1}_{B,C,A}"'] \\
& (B \otimes A) \otimes C \arrow[r, "\alpha_{B,A,C}"'] & B \otimes (A \otimes C), \arrow[ur, "\mathrm{id}_B\otimes \sigma_{A,C}"'] &
\end{tikzcd} \tag{7.2.4}$$

and

$$\begin{tikzcd}
& (A \otimes B) \otimes C \arrow[r, "\sigma_{A\otimes B,C}"] \arrow[dl, "\alpha_{A,B,C}"'] & B \otimes (C \otimes A) & \\
A \otimes (B \otimes C) \arrow[dr, "\mathrm{id}_A\otimes \sigma_{B,C}"'] & & & (C \otimes A) \otimes B \arrow[ul, "\alpha_{C,A,B}"'] \\
& A \otimes (C \otimes B) \arrow[r, "\alpha^{-1}_{A,C,B}"'] & (A \otimes C) \otimes B. \arrow[ur, "\sigma_{A,C}\otimes \mathrm{id}_B"'] &
\end{tikzcd} \tag{7.2.5}$$

The naturality condition for σ simply means that the following diagram commutes:

$$\begin{tikzcd}
A \otimes B \arrow[r, "\sigma_{A,B}"] \arrow[d, "f\otimes g"'] & B \otimes A \arrow[d, "g\otimes f"] \\
A' \otimes B' \arrow[r, "\sigma_{A',B'}"'] & B' \otimes A'
\end{tikzcd} \tag{7.2.6}$$

for all $f : A \to A'$ and $g : B \to B'$. In other words, it doesn't matter if we swap then apply a pair of maps or apply a pair of maps and then swap.

Now consider the first hexagon equation. Starting at the top left, with $A \otimes (B \otimes C)$ we can treat $B \otimes C$ as a single object and swap it with A to get $(B \otimes C) \otimes A$. The first hexagon equation tells us that this is the same as moving the brackets, swapping A and B while doing nothing to C, moving the brackets, swapping A and C while doing nothing to B, and then moving the brackets. More succinctly swapping $B \otimes C$ with A is the same as swapping A with B and C individually.

7.3 Braided Monoidal Category: The Examples

7.3.1 Set

The braiding in **Set** (as a monoidal category with the Cartesian product) is pretty obvious. It's simply swapping the elements in a pair.

Definition 7.3.1 — Set as a Braided Monoidal Category The monoidal category **Set**, with the Cartesian product as the monoidal product, can be promoted to a braided monoidal category by defining $\sigma_{A,B}$ to be the map $(a,b) \mapsto (b,a)$.

Theorem 7.3.2. **Set** is a braided monoidal category.

Proof. First we should show that σ is natural, this follows from commutativity of the following for all $f\colon A \to A'$ and $g\colon B \to B'$:

$$\begin{tikzcd}
(a,b) \arrow[r, mapsto, "\sigma_{A,B}"] \arrow[d, mapsto, "f\times g"] & (b,a) \arrow[d, mapsto, "g\times f"] \\
(f(a),g(b)) \arrow[r, mapsto, "\sigma_{A',B'}"] & (g(b),f(a)).
\end{tikzcd} \tag{7.3.3}$$

Next we need to prove the hexagon equations. We'll prove only the first, which follows from commutativity of the following diagram:

$$\begin{tikzcd}
 & (a,(b,c)) \arrow[r, mapsto, "\sigma_{A,B\otimes C}"] \arrow[dl, mapsto, "\alpha^{-1}_{A,B,C}"'] & ((b,c),a) & \\
((a,b),c) \arrow[dr, mapsto, "\sigma_{A,B}\times \mathrm{id}_C"'] & & & \begin{array}{c}(b,(c,a)) = \\ (\mathrm{id}_B(b),\sigma_{A,C}(a,c))\end{array} \arrow[ul, mapsto, "\alpha^{-1}_{B,C,A}"'] \\
 & \begin{array}{c}(\sigma_{A,B}(a,b),\mathrm{id}_C(c)) \\ = ((b,a),c)\end{array} \arrow[r, mapsto, "\alpha_{B,A,C}"'] & (b,(a,c)). \arrow[ur, mapsto, "\mathrm{id}_B\times\sigma_{A,C}"'] &
\end{tikzcd}$$

□

7.3.2 Rel

The braiding in **Rel** is similar to that in **Set**, we just turn a function into a relation.

Definition 7.3.4 — Rel as a Braided Monoidal Category The monoidal category **Rel** can be promoted to a braided monoidal category by defining $\sigma_{A,B}$ to be the relation $(a,b) \sim (b,a)$.

A proof that this is a valid braiding is very similar to the proof for **Set**.

7.3.3 Hilb

Definition 7.3.5 — Hilb as a Braided Monoidal Category The monoidal category **Hilb** can be promoted to a braided monoidal category by defining $\sigma_{H,K}$ to be the map $v \otimes w \mapsto w \otimes v$, and extended linearly to all of $H \otimes K$.

Again, the proof that this is a valid braiding is very similar to the proof for **Set**.

7.3.4 More Examples

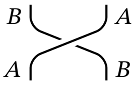
Example 7.3.6

- The category of bimodules with bimodule homomorphisms with the usual tensor product of modules is braided symmetric.

7.4 Graphical Notation

We can extend the graphical notation for monoidal categories to braided monoidal categories. To do so we make use of the idea that the braiding allows us to swap elements. For objects A and B in a braided monoidal category we denote the braiding $\sigma_{A,B} : A \otimes B \to B \otimes A$ as follows

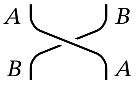
$$\sigma_{A,B} =$$



. (7.4.1)

Note that the order in which the wires cross, the starting on the bottom left wire going over the top, is important. Since σ is a natural isomorphism $\sigma^{-1}_{A,B} : B \otimes A \to A \otimes B$ exists, and we denote it by

$$\sigma^{-1}_{A,B} =$$



. (7.4.2)

Notice that now the bottom left wire goes under.

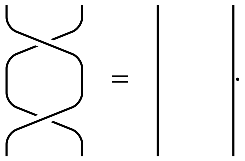
Now that we have wires passing over each other we have clearly entered the third dimension. This changes what it means for two diagrams to be the same.

Definition 7.4.3 — Spatial Isotopy Two diagrams are **spatial isotopic** if one can be deformed continuously into the other such that

- the diagrams remain confined in a cuboidal volume of three-dimensional space;
- input and output wires terminate at the bottom and top of this region;
- components never intersect.

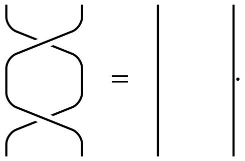
Note that we now drop the requirement on wires entering in the same order, since we can always cross the wires over, it doesn't really make sense as a definition anyway as we don't have an order defined on the plane, and wires now enter in the rectangle at the bottom of the cuboidal region.

That $\sigma_{A,B}$ and $\sigma^{-1}_{A,B}$ are inverses, tells us that $\sigma^{-1}_{A,B} \circ \sigma_{A,B} = \mathrm{id}_{A\otimes B}$, graphically[1],



(7.4.4)

Similarly, $\sigma_{A,B} \circ \sigma^{-1}_{A,B} = \mathrm{id}_{B\otimes A}$, or graphically,



(7.4.5)

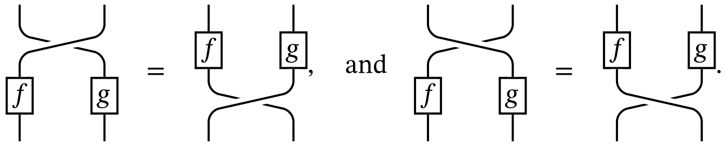
In both of these it looks like if the wires were strings we could just pull them taught, this is the type of thing that is allowed in a spatial isotopy.

We can define $\sigma^{-1} : \; -\otimes- \Rightarrow - \otimes -$ to be the natural transformation whose components $(\sigma^{-1})_{A,B}$ are just the inverse components of σ, so $(\sigma^{-1})_{A,B} = (\sigma_{A,B})^{-1} = \sigma^{-1}_{A,B}$, which is necessarily also natural. The naturality conditions

$$\sigma_{A',B'} \circ (f \otimes g) = (g \otimes f) \circ \sigma_{A,B}, \tag{7.4.6}$$

$$\sigma^{-1}_{A',B'} \circ (f \otimes g) = (g \otimes f) \circ \sigma^{-1}_{A,B}, \tag{7.4.7}$$

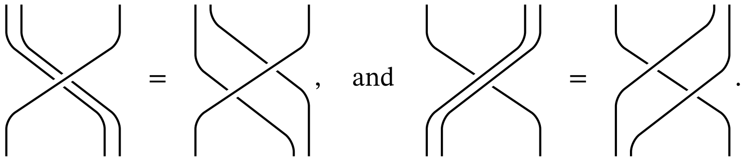
for all $f : A \to A'$ and $g : B \to B'$, can be expressed in the graphical notation as



(7.4.8)

So the naturality of σ and σ^{-1} means that graphically we can slide morphisms through crossings.

The hexagon equations become trivial in the graphical notation, they just tell us that crossing two objects at once or crossing them one at a time is the same:



(7.4.9)

7.4.1 Correctness Theorem

The graphical calculus for braided monoidal categories comes with a correctness theorem analogous to the correctness theorem for monoidal categories.

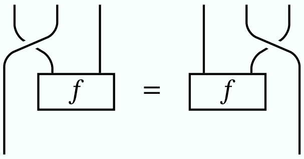
[1] note that the right hand side here can be read as $\mathrm{id}_A \otimes \mathrm{id}_B$, but functoriality of $\otimes$ means that $\mathrm{id}_A \otimes \mathrm{id}_B = \mathrm{id}_{A\otimes B}$, which is perhaps clearer with function notation: $\otimes(\mathrm{id}_A, \mathrm{id}_B) = \mathrm{id}_{\otimes(A,B)}$.

Theorem 7.4.10 — Correctness of the Graphical Notation for Braided Monoidal Categories. Any well-formed equation of the form $f = g$ for morphisms f and g in a braided monoidal category follows from the axioms of braided monoidal categories if and only if it holds in the graphical notation up to spatial isotopy.

Example 7.4.11 Consider the following equation of morphisms $B \to A \otimes B \otimes C$:

$$(\sigma_{A,B} \otimes \mathrm{id}_B) \circ (\mathrm{id}_A \otimes f) = (\mathrm{id}_A \otimes \sigma^{-1}_{B,C}) \circ (f \otimes \mathrm{id}_B) \tag{7.4.12}$$

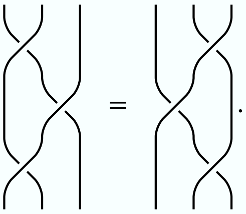
where $f : I \to A \otimes C$. This holds in a genera braided monoidal category as we can express this equation as the following equation of diagrams



(7.4.13)

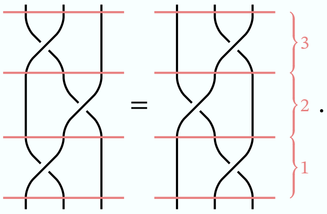
and we can see that these are spatially isotopic, we can just slide the B wire over the f morphism to the other side.

Example 7.4.14 Consider the following spatial isotopy of diagrams



(7.4.15)

We can convert this into an algebraic equation between morphisms. The first step is to work out what the source and target of these morphisms is. To do this we label the wires A, B, and C along the bottom, so the source is $A \otimes B \otimes C$. The wires must have the same labels all the way along, so if we chase these labels up the wires we see that the target is $C \otimes B \otimes A$. Next split the diagrams horizontally into sections so that only one thing is occurring in each section, this gives



(7.4.16)

We can then just read from the bottom up. Starting on the left region 1 is swapping A and B and leaving C alone, so it corresponds to $\sigma_{A,B} \otimes \mathrm{id}_C$, region 2 then leaves B alone (which is now the left most wire) and swaps A and C, so it is $\mathrm{id}_B \otimes \sigma_{A,C}$, and region 3 swaps B and C leaving A alone, so it is $\sigma_{B,C} \otimes \mathrm{id}_A$. Doing the same for the diagram on the right we get the equation

$$\begin{aligned}(\sigma_{B,C} \otimes \mathrm{id}_A) \circ (\mathrm{id}_B \otimes \sigma_{A,C}) \circ (\sigma_{A,B} \otimes \mathrm{id}_C)\\ = (\mathrm{id}_C \otimes \sigma_{A,C}) \circ (\sigma_{A,C} \otimes \mathrm{id}_B) \circ (\mathrm{id}_A \otimes \sigma_{B,C}).\end{aligned} \quad (7.4.17)$$

7.5 Symmetric Monoidal Category

It is tempting to think that swapping two objects and swapping them back should do nothing. However, this is not always the case, in general $A \otimes B$ and the result of applying $\sigma_{A,B}$ then $\sigma_{B,A}$ will only be isomorphic. Often though this isomorphism turns out to be the identity, and this leads to the following definition.

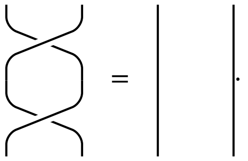
Definition 7.5.1 — Symmetric Monoidal Category A **symmetric monoidal category** is a braided monoidal category with braiding σ such that

$$\sigma_{B,A} \circ \sigma_{A,B} = \mathrm{id}_{A\otimes B} \quad (7.5.2)$$

for all objects A and B.

That is, a symmetric monoidal category is exactly a braided monoidal category in which we can swap and swap back without changing anything. So, in a symmetric monoidal category we have $\sigma^{-1}_{A,B} = \sigma_{B,A}$.

In the graphical notation $\sigma_{B,A} \circ \sigma_{A,B} = \mathrm{id}_{A\otimes B}$ is



(7.5.3)

Note the subtle difference to the identity $\sigma^{-1}_{A,B} \circ \sigma_{A,B}$ in Equation (7.4.5), here the wires swap which one is on top. One can imagine the wires being strings, pulling on them, and them passing through each other to become taught. This is the type of thing that is allowed in a four-dimensional isotopy.

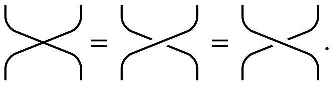
Definition 7.5.4 — Four-Dimensional Isotopy Two diagrams are related by a four-dimensional isotopy if one can be deformed continuously into the other such that

- the diagrams remain confined in a cuboidal volume of three-dimensional space;

- input and output wires terminate at the bottom and top of this region.

The extra dimension gives us room to move wires which seem linked in three dimensions past each other in the fourth dimension, unlinking them in three dimensions. This is part of a deep mathematical fact that there can be no (nontrivial) knots in four dimensions.

Since wires can pass through each other for a symmetric monoidal category we don't need to keep track of which wire is on top, so we write



(7.5.5)

Like (braided) monoidal categories there is a correctness theorem to go with the graphical notation.

Theorem 7.5.6 — Correctness of the Graphical Notation for Symmetric Monoidal Categories. Any well-formed equation of the form $f = g$ in a symmetric monoidal category follows from the axioms of symmetric monoidal categories if and only if it holds in the graphical notation up to graphical equivalence.

Note that unlike the monoidal and braided monoidal cases we don't have isotopy here, this is because although graphical equivalence (a deliberately vague term essentially defined by being the thing required to make this theorem true) is almost certainly four-dimensional isotopy it has not yet been proven.

Example 7.5.7

- **Set** equipped with the Cartesian product is a symmetric monoidal category;
- **Rel** equipped with the Cartesian product is a symmetric monoidal category;
- **Hilb** equipped with the tensor product is a symmetric monoidal category;
- **Grp** equipped with the Cartesian product with the trivial group as unit is a symmetric monoidal category;
- the category of representations equipped with the tensor product of representations, which is a tensor product of the representation spaces and factor-wise group action on the product, so $g\,.\,(v \otimes u) = (g\,.\,v) \otimes (g\,.\,u)$, is a symmetric monoidal category;
- **Cat** equipped with the product of categories is a symmetric monoidal category;

- every Cartesian category equipped with its categorical product as a monoidal product with the terminal object as the unit is a symmetric monoidal category by uniqueness of the categorical product;
- A $\mathbb{Z}_2$-graded vector space[a] is a vector space, V, along with a decomposition $V = V_0 \oplus V_1$. A $\mathbb{Z}_2$-graded map is a map $f : V \to W$ with $V = V_0 \oplus V_1$ and $W = W_0 \oplus W_1$ such that if $v \in V_0$ then $f(v) \in W_0$ and if $v \in V_1$ then $f(v) \in W_1$. The tensor product of vector spaces makes the category of $\mathbb{Z}_2$-graded vector spaces with grading preserving maps into a monoidal category. There are two ways to make this into a symmetric monoidal category. The first is to forget the grading, in which case the braiding is $u \otimes v \mapsto v \otimes u$, the second is that if $u, v \in V_1$ then we define the braiding to be $u \otimes v \mapsto -v \otimes u$ and if either $u \in V_0$ or $v \in V_0$ (or both) then the braiding is $u \otimes v \mapsto v \otimes u$.

[a] See also my notes from the *Quantum Field Theory* course, where a $\mathbb{Z}_2$-graded vector space equipped with a product is called a Grassmann algebra.

Eight

Dagger Categories

O happy dagger!

Juliet in Romeo and Juliet

8.1 Dagger Categories: The Idea

In the definition of **Hilb**, the category of Hilbert spaces, we didn't make use of the inner product, except for requiring its existence as part of the definition of a Hilbert space. This leaves a hole in our categorical approach to quantum computing, since the inner product is incredibly important in quantum mechanics. So, in this chapter we will demonstrate how we can introduce the important parts of an inner product to our categories.

Rather than the inner product itself we use the fact that the inner product can define an adjoint.

Definition 8.1.1 — Adjoint Let H be a Hilbert spaces and consider a bounded linear map $f : H \to H$. We define the **adjoint**[a] of this map, $f^\dagger : H \to H$, to be the unique map such that $\langle u|f(v)\rangle = \langle f^\dagger(u)|v\rangle$ for all $u, v \in H$.

[a] Not to be confused with adjoint functors.

The adjoint has the following properties:

$$(g \circ f)^\dagger = f^\dagger \circ g^\dagger, \qquad \mathrm{id}_H^\dagger = \mathrm{id}_H, \qquad \text{and} \qquad (f^\dagger)^\dagger. \tag{8.1.2}$$

These may be more familiar if we consider the finite dimensional case and represent the maps as matrices, in which case † is the Hermitian conjugate, that is transpose and take the complex conjugate, then we have

$$(AB)^\dagger = B^\dagger A^\dagger, \qquad I^\dagger = I, \qquad \text{and} \qquad (A^\dagger)^\dagger = A. \tag{8.1.3}$$

We can encode this information into a functor, which is what the next definition does.

Definition 8.1.4 — Dagger Functor The **dagger functor** on **Hilb** is the contravariant endofunctor $(-)^\dagger : \mathbf{Hilb} \to \mathbf{Hilb}$ which takes objects to themselves and morphisms to their adjoints as bounded linear maps.

Contravariance means that $F(g \circ f) = Ff \circ Fg$, which in this case means $(g \circ f)^\dagger = f^\dagger \circ g^\dagger$, which is certainly true. Since the dagger acts as an identity on objects and $\mathrm{id}_H^\dagger = \mathrm{id}_H = \mathrm{id}_{H^\dagger}$, satisfying the requirement that $F\mathrm{id}_A = \mathrm{id}_{FA}$, so we see that this is indeed a functor. The dagger functor has the further property of being an **involution**, meaning that $(f^\dagger)^\dagger = f$.

This functor captures all of the information in an inner product. To see this we demonstrate how we can recover the inner product just using the dagger. Consider some Hilbert space, H, with states $v, w : \mathbb{C} \to H$, that is kets $|v\rangle$ and $|w\rangle$. We can turn v into an effect $v^\dagger : H \to \mathbb{C}$. Then consider the composite $v^\dagger \circ w$. As a (bounded) linear map this is completely determined by where it sends 1, so consider this map evaluated at 1, we have

$$(v^\dagger \circ w)(1) = v^\dagger(w(1)) = \langle 1|v^\dagger(w(1))\rangle \tag{8.1.5}$$

where $\langle -|-\rangle$ is the inner product on $\mathbb{C}$, that is $\langle x|y\rangle = x^* y$, so $\langle 1|v^\dagger(w(1))\rangle = 1^* v^\dagger(w(1)) = v^\dagger(w(1))$. Then using the definition of the adjoint we have

$$(v^\dagger \circ w)(1) = \langle 1|v^\dagger(w(1))\rangle = \langle (v^\dagger)^\dagger(1)|w(1)\rangle = \langle v(1)|w(1)\rangle = \langle v|w\rangle \tag{8.1.6}$$

where the last step is to identify the state w evaluated at 1 as the ket $|w\rangle$ and the effect v evaluated at 1 as the bra $\langle w|$. This calculation shows that using only the dagger, and the most basic inner product on $\mathbb{C}$, we can reconstruct the inner product. This suggests that we can use the dagger, a functor, to generalise the inner product to other categories.

8.2 Dagger Categories: The Definition

Definition 8.2.1 Let **C** be a category. A **dagger functor** on **C** is an involutive contravariant endofunctor $(-)^\dagger : \mathbf{C} \to \mathbf{C}$ which acts as the identity on objects.
A **dagger category** is a category equipped with a dagger.

8.3 Dagger Categories: The Examples

> *Is this a dagger I see before me?*
>
> Macbeth in Macbeth

8.3.1 Set

Lemma 8.3.1 It is not possible to make **Set** into a dagger category.

Proof. Suppose that **Set** is a dagger category, so we have a involutive contravariant functor $(-)^\dagger : \mathbf{Set} \to \mathbf{Set}$. Write $|A|$ for the cardinality of a set A. It is known that the homset $\mathbf{Set}(A,B)$ contains $|B|^{|A|}$ elements. Similarly, $\mathbf{Set}(B,A)$ has $|A|^{|B|}$ elements. For every $f \in \mathbf{Set}(A,B)$ the dagger gives us some $f^\dagger \in \mathbf{Set}(B,A)$. Further, the dagger is involutive, so is its own inverse on morphisms. This means that the mapping of sets $(-)^\dagger : \mathbf{Set}(A,B) \to \mathbf{Set}(B,A)$ is a bijection. However, if $A = \{1\}$ and $B = \{1,2\}$ then we have $|\mathbf{Set}(A,B)| = 2^1 = 2$ and $|\mathbf{Set}(B,A)| = 1^2 = 1$, so clearly these sets are not in bijection, a contradiction. □

8.3.2 Rel

Definition 8.3.2 — Rel as a Dagger Category The category **Rel** can be promoted to a dagger category by defining the dagger to be the **converse relation**. If we have a relation $R : A \to B$ then the converse $R^\dagger : B \to A$ is

$$R^\dagger = \{(b,a) \mid (a,b) \in R\} \subseteq B \times A. \tag{8.3.3}$$

Lemma 8.3.4 **Rel** along with the converse relation is a dagger category.

Proof. Clearly the converse relation is an involution, we need only check then that it is a contravariant functor. Consider first the identity relation, $\mathrm{id}_A = \{(a,a) \mid a \in A\}$. We have

$$\mathrm{id}_A^\dagger = \{(a,a) \mid (a,a) \in \mathrm{id}_A\} = \mathrm{id}_A = \mathrm{id}_{A^\dagger} \tag{8.3.5}$$

where the last step uses that the dagger acts as the identity on objects. Now suppose we have two relations $R : A \to B$ and $S : B \to C$. Their composite is the relation

$$S \circ R = \{(a,c) \mid \exists b \in B \text{ such that } aRb \text{ and } bSc\}. \tag{8.3.6}$$

The converse of this is

$$\begin{aligned}
(S \circ R)^\dagger &= \{(c,a) \mid (a,c) \in (S \circ R)\} && (8.3.7)\\
&= \{(c,a) \mid \exists b \in B \text{ such that } aRb \text{ and } bSc\} && (8.3.8)\\
&= \{(c,a) \mid \exists b \in B \text{ such that } bR^\dagger a \text{ and } cS^\dagger b\} && (8.3.9)\\
&= \{(c,a) \mid \exists b \in B \text{ such that } cS^\dagger b \text{ and } bR^\dagger a\} && (8.3.10)\\
&= R^\dagger \circ S^\dagger. && (8.3.11)
\end{aligned}$$

So the converse is indeed a contravariant functor. □

This is another example of **Rel** being more like **Hilb** than **Set**.

8.3.3 Hilb

Definition 8.3.12 — Hilb as a Dagger Category The category **Hilb** can be promoted to a dagger category by defining the dagger to be the adjoint. That is, if $f : H \to K$ is a bounded linear map between Hilbert spaces H and K with inner products $\langle - | - \rangle_H$ and $\langle - | - \rangle_K$ respectively then $f^\dagger$ is the unique bounded linear operator such that

$$\langle f^\dagger(k) | h \rangle_H = \langle k | f(h) \rangle_K \tag{8.3.13}$$

for all $h \in H$ and $k \in K$.

We won't prove that this is a dagger, since this would require us to get into details like proving the uniqueness part of the definition and that the adjoint of a bounded linear map is again a bounded linear map. Since **Hilb** is our model category for defining the dagger in the first place the important properties of the dagger, like being involutive and contravariant, should be familiar already, although perhaps not in this language.

Notice that the definition requires the inner product. It turns out that this is one of the first times where we see something done with **Hilb** which can't be done with $\mathbf{Vect}_{\Bbbk}$. To see this we first note that for $\mathbf{Vect}_{\Bbbk}$ and two given vector spaces V and W the set $\mathbf{Vect}_{\Bbbk}(V, W)$ can be made into a vector space by defining pointwise addition, that is given $\varphi, \psi \in \mathbf{Vect}_{\Bbbk}(V, W)$ define $\varphi + \psi \in \mathbf{Vect}_{\Bbbk}(V, W)$ to be the linear map

$$(\varphi + \psi)(v) = \varphi(v) + \psi(v). \tag{8.3.14}$$

Similarly, scalar multiplication is given pointwise, given $z \in \Bbbk$ we define $z\varphi \in \mathbf{Vect}_{\Bbbk}(V, W)$ to be the linear map

$$(z\varphi)(v) = z\varphi(v). \tag{8.3.15}$$

It is easy to check that these definitions make $\mathbf{Vect}_{\Bbbk}(V, W)$ a vector space.

Now, suppose there was a dagger functor $(-)^\dagger : \mathbf{Vect}_{\Bbbk} \to \mathbf{Vect}_{\Bbbk}$. This gives a mapping of morphisms $\mathbf{Vect}_{\Bbbk}(V, W) \to \mathbf{Vect}_{\Bbbk}(W, V)$. Specifically, choosing $W = \Bbbk$ we have a mapping $\mathbf{Vect}_{\Bbbk}(V, \Bbbk) \to \mathbf{Vect}_{\Bbbk}(\Bbbk, V)$. However, if V is infinite dimensional then $\mathbf{Vect}_{\Bbbk}(\mathbb{C}, V)$ has strictly smaller dimension than $\mathbf{Vect}_{\Bbbk}(V, \Bbbk)$, intuitively there are more ways to squash V down into $\Bbbk$ than there are to expand $\Bbbk$ to fill the infinite dimensions of V. However, this means that we cannot have a bijection $\mathbf{Vect}_{\Bbbk}(V, W) \to \mathbf{Vect}_{\Bbbk}(W, V)$ for all vector spaces V and W, as since $\mathbf{Vect}_{\Bbbk}(V, W)$ and $\mathbf{Vect}_{\Bbbk}(W, V)$ are vector spaces themselves such a bijection, which can also be shown to be linear, would be an isomorphism of vector spaces, and isomorphic vector spaces must have the same dimension[1].

Finite dimensional vector spaces, $\mathbf{FVect}_{\Bbbk}$, don't have this problem, so we *can* define a dagger $(-)^\dagger : \mathbf{FVect}_{\Bbbk} \to \mathbf{FVect}_{\Bbbk}$, by choosing an inner product, which can be done once we've chosen a basis, and then constructing adjoints. However, there is no "canonical" way to do this, so it's not helpful. A **canonical** way to do something is, intuitively a way to do something such that we don't make any arbitrary choices, this can often be made rigorous by defining something to be the "canonical" choice if all ways of doing it end up being naturally isomorphic.

[1] note that it is possible infinite dimensional vector spaces of the same dimension are not isomorphic, but being isomorphic necessitates having the same dimension. For finite dimensional vector spaces being isomorphic and having the same dimension are equivalent statements (for a fixed field).

8.3.4 More Examples

Example 8.3.16

- Any monoid, M, with an involutive map $f : M \to M$ can be considered as a one-object dagger category by taking the dagger to be $m^\dagger = f(m)$ for all $m \in M$, recall that when viewing a monoid as a one-object category the elements of the monoid are the morphisms of the category.
- Any groupoid (category where all morphisms are isomorphisms) is a dagger category with the dagger being the inverse.
- Any discrete category is trivially a dagger category.
- Let $\mathbf{C}$ and $\mathbf{D}$ be dagger categories and $F, G : \mathbf{C} \to \mathbf{D}$ functors. Then given a natural transformation $\zeta : F \Rightarrow G$ we can define a dagger componentwise by defining $\zeta^\dagger : G \Rightarrow F$ to be the natural transformation whose component at $A \in \mathrm{Ob}(\mathbf{D})$ is $\zeta_A^\dagger$. This makes the functor category $[\mathbf{C}, \mathbf{D}]$ into a dagger category.

8.4 Terminology

We now introduce some terminology useful for talking about special properties of morphisms under daggers.

Definition 8.4.1 A morphism $f : A \to B$ in a dagger category is

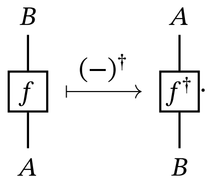
- the **adjoint** of $g : B \to A$ if $g = f^\dagger$;
- **self-adjoint** if $f = f^\dagger$ (requiring $A = B$);
- a **projection** if[a] $f = f^\dagger$ and $f \circ f = f$;
- **unitary** when $f^\dagger \circ f = \mathrm{id}_A$ and $f \circ f^\dagger = \mathrm{id}_B$;
- an **isometry** when $f^\dagger \circ f = \mathrm{id}_A$;
- a **partial isometry** when $f^\dagger \circ f$ is a projection;
- **positive** when $f = g^\dagger \circ g$ for some morphism $g : A \to A$ (requiring $A = B$).

[a] sometimes the requirement of being self-adjoint is left out of this definition

Many of these should be familiar from **Hilb**. Note that the definition of positive is inspired by $\mathbb{C}$ with the dagger being complex conjugate where $z^* z = |z|^2$ is non-negative.

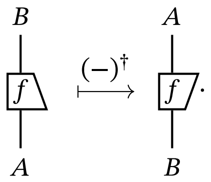
8.5 Graphical Notation

Since daggers swap the direction of functions we can depict them in the graphical notation as a rotation in the horizontal axis:



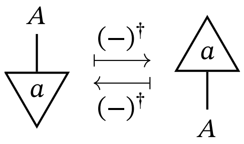
(8.5.1)

Rather than writing daggers on every morphism that we dagger we instead use a shape which breaks the horizontal mirror symmetry, so that we can reflect it and it is clearly different without having to add in a dagger:



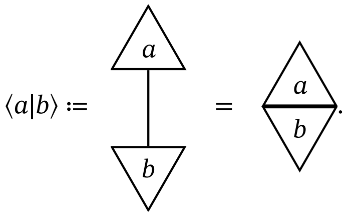
(8.5.2)

Notice that if $a : I \to A$ is a state in a dagger category then $a^\dagger : A \to I$ is an effect in the same category. So daggers relate states and effects:



(8.5.3)

We can use this to generalise the inner product and bra-ket notation to any dagger category by defining



(8.5.4)

That is, for $a : A \to I$ and $b : I \to A$

$$\langle a|b\rangle \coloneqq a \circ b. \tag{8.5.5}$$

8.6 Monoidal Dagger Categories

In order to use both monoidal categories and dagger categories at the same time we want the two functors $- \otimes -$ and $(-)^\dagger$ to play nicely together, we get this through the following definition.

Definition 8.6.1 — Monoidal Dagger Category A **monoidal dagger category** is a dagger category which is also monoidal such that

- $(f \otimes g)^\dagger = f^\dagger \otimes g^\dagger$ for all morphisms f and g;
- the associator and unitors are unitary at every object.

A **braided monoidal dagger category** is a monoidal dagger category equipped with a unitary braiding.
A **symmetric monoidal dagger category** is a braided monoidal dagger category for which the braiding is symmetric.

More compactly, a (braided or symmetric) monoidal dagger category is exactly what you would expect with the requirements that all of the natural isomorphisms involved in the definition of the (braided or symmetric) monoidal part are unitary and the dagger on the tensor product reverses the order of the morphisms: $(f \otimes g)^\dagger = f^\dagger \otimes g^\dagger$.

Lemma 8.6.2 **Hilb** is a symmetric monoidal dagger category.

Proof. Let H, J, K, L be Hilbert spaces. Suppose we have two morphisms $f : H \to K$ and $g : J \to L$. Then we have a morphism $f \otimes g : H \otimes J \to K \otimes L$. With $H \otimes J$ and $K \otimes L$ Hilbert spaces with inner products $\langle - | - \rangle_{H \otimes J}$ and $\langle - | - \rangle_{K \otimes L}$. The adjoint to $f \otimes g$ is defined to be the unique linear map $(f \otimes g)^\dagger : K \otimes L \to H \otimes J$ satisfying

$$\langle (f \otimes g)^\dagger (v' \otimes w') | v \otimes w \rangle_{H \otimes J} = \langle v' \otimes w' | (f \otimes g)(v \otimes w) \rangle_{K \otimes L} \tag{8.6.3}$$

for all $v \in H$, $w \in J$, $v' \in K$, and $w' \in L$. We then have

$$\begin{aligned} \langle v' \otimes w' | (f \otimes g)(v \otimes w) \rangle_{K \otimes L} &= \langle v' \otimes w' | f(v) \otimes g(w) \rangle_{K \otimes L} && (8.6.4) \\ &= \langle v' | f(v) \rangle_K \langle w' | g(w) \rangle_L && (8.6.5) \\ &= \langle f^\dagger(v') | v \rangle_H \langle g^\dagger(w') | w \rangle_J && (8.6.6) \\ &= \langle (f^\dagger \otimes g^\dagger)(v' \otimes w') | v \otimes w \rangle_{H \otimes J}, \end{aligned}$$

so we have $(f \otimes g)^\dagger = f^\dagger \otimes g^\dagger$.
For unitarity first consider the left unitor, λ. Fixing some Hilbert space, H, the left unitor gives an isomorphism $\lambda_H : I \otimes H \to H$. The dagger of this is defined as the unique morphisms such that

$$\langle \lambda_H^\dagger (v) | 1 \otimes w \rangle_{I \otimes H} = \langle v | \lambda_H (1 \otimes w) \rangle_H = \langle v | w \rangle_H. \tag{8.6.7}$$

Clearly, this holds if $\lambda_H^\dagger = \lambda_H^{-1}$ since

$$\langle \lambda_H^{-1}(v) | 1 \otimes w \rangle_{I \otimes H} = \langle 1 \otimes v | 1 \otimes w \rangle_{I \otimes H} = \langle 1 | 1 \rangle_I \langle v | w \rangle_H = \langle v | w \rangle_H. \tag{8.6.8}$$

So by uniqueness we have $\lambda_H^\dagger = \lambda_H^{-1}$, hence λ_H is unitary. Unitarity of the right unitor follows similarly.

Now consider the associator, α. Fixing some Hilbert spaces H, J, and K, we get an isomorphism $\alpha_{H,J,K} : (H \otimes J) \otimes K \to H \otimes (J \otimes K)$. Then $\alpha^{\dagger}_{H,J,K}$ is defined as the unique morphism such that

$$\begin{aligned} &\langle \alpha^{\dagger}_{H,J,K}(u \otimes (v \otimes w)) | (u' \otimes v') \otimes w' \rangle_{(H\otimes J)\otimes K} && (8.6.9)\\ &= \langle u \otimes (v \otimes w) | \alpha_{H,J,K}((u' \otimes v') \otimes w') \rangle_{H\otimes(J\otimes K)} && (8.6.10)\\ &= \langle u \otimes (v \otimes w) | (u' \otimes (v' \otimes w')) \rangle_{H\otimes(J\otimes K)}. && (8.6.11) \end{aligned}$$

Clearly this holds if $\alpha^{\dagger}_{H,J,K} = \alpha^{-1}_{H,J,K}$:

$$\begin{aligned} &\langle \alpha^{-1}_{H,J,K}(u \otimes (v \otimes w)) | (u' \otimes v') \otimes w' \rangle_{(H\otimes J)\otimes K} && (8.6.12)\\ &= \langle (u \otimes v) \otimes w | (u' \otimes v') \otimes w' \rangle_{(H\otimes J)\otimes K} && (8.6.13)\\ &= \langle (u \otimes v) | u' \otimes v' \rangle_{H\otimes J} \langle w | w' \rangle_K && (8.6.14)\\ &= \langle u | u' \rangle_H \langle v | v' \rangle_J \langle w | w' \rangle_K && (8.6.15)\\ &= \langle u | u' \rangle_H \langle v \otimes w | v' \otimes w' \rangle_{J\otimes K} && (8.6.16)\\ &= \langle u \otimes (v \otimes w) | u' \otimes (v' \otimes w') \rangle_{H\otimes(J\otimes K)}. && (8.6.17) \end{aligned}$$

So by uniqueness $\alpha^{\dagger}_{H,J,K} = \alpha^{-1}_{H,J,K}$, and so $\alpha_{H,J,K}$ is unitary. □

Rel is also a symmetric monoidal dagger category.

Part III

Duals

Nine

Duals

9.1 Duals: The Idea

Consider some Hilbert space, H. We can construct its dual space, $H^* \coloneqq \mathbf{Hilb}(H, \mathbb{C})$. This is also a Hilbert space with vector addition given by pointwise addition, so for $\varphi, \psi \in \mathbf{Hilb}(H, \mathbb{C})$ we define $\varphi + \psi \in \mathbf{Hilb}(H, \mathbb{C})$ to be the bounded linear map

$$(\varphi + \psi)(v) = \varphi(v) + \psi(v) \tag{9.1.1}$$

and for $z \in \mathbb{C}$ we define $z\varphi \in \mathbf{Hilb}(H, \mathbb{C})$ to be the bounded linear map

$$(z\varphi)(v) = z\varphi(v). \tag{9.1.2}$$

The inner product on H^* follows from the Riesz representation theorem, which states that for all $\varphi \in H^*$ there exists a unique $f_\varphi \in H$ such that

$$\varphi(v) = \langle f_\varphi | v \rangle \tag{9.1.3}$$

for all $v \in H$. Using this we define an inner product on H^* according to

$$\langle \varphi | \psi \rangle_{H^*} = \langle f_\varphi | f_\psi \rangle_H. \tag{9.1.4}$$

This idea generalises to monoidal categories, although it is not particularly obvious from the definition.

9.2 Duals: The Definition

> **Definition 9.2.1 — Dual** Let $(\mathsf{C}, \otimes, I, \lambda, \rho, \alpha)$ be a monoidal category. An object $L \in \operatorname{Ob}(\mathsf{C})$ is **left dual** to an object $R \in \operatorname{Ob}(\mathsf{C})$, and R is **right dual** to L, which we write as $L \dashv R$, when there exist morphisms
>
> - $\eta\colon I \to R \otimes L$, called the **unit morphism**; and
> - $\varepsilon\colon L \otimes R \to I$, called the **counit morphism**

such that the following diagrams commute

$$\begin{tikzcd}
L \arrow[r, "\rho_L^{-1}"] \arrow[d, "\mathrm{id}_L"'] & L \otimes I \arrow[r, "\mathrm{id}_L \otimes \eta"] & L \otimes (R \otimes L) \arrow[d, "\alpha^{-1}_{L,R,L}"] \\
L & I \otimes L \arrow[l, "\lambda_L"] & (L \otimes R) \otimes R, \arrow[l, "\varepsilon \otimes \mathrm{id}_L"]
\end{tikzcd} \tag{9.2.2}$$

$$\begin{tikzcd}
R \arrow[r, "\lambda_R^{-1}"] \arrow[d, "\mathrm{id}_R"'] & I \otimes R \arrow[r, "\eta \otimes \mathrm{id}_R"] & (R \otimes L) \otimes R \arrow[d, "\alpha_{R,L,R}"] \\
R & R \otimes I \arrow[l, "\rho_R"] & R \otimes (L \otimes R). \arrow[l, "\mathrm{id}_R \otimes \varepsilon"]
\end{tikzcd} \tag{9.2.3}$$

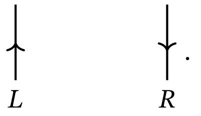
If L is both left and right-dual to R we call L the **dual** of R.

Consider the first diagram here. It says that we can add $R \otimes L$ on the right of L and then remove $L \otimes R$ from the left and this is the same as doing nothing. The second diagram is similar but swapping R and L.

Notice that η is a state of $R \otimes L$. We will later show that it is an entangled state, which is useful for quantum mechanics stuff.

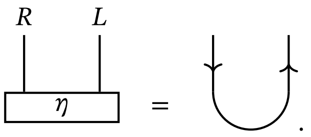
9.3 Graphical Notation

Suppose we have a category and two objects L and R with $L \dashv R$. In the graphical notation we draw L as a wire with an arrow pointing upwards, and we draw R as a wire with an arrow pointing down:



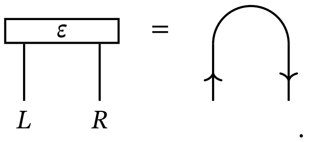
(9.3.1)

We draw the unit morphism, $\eta : I \to R \otimes L$, as a bent wire:



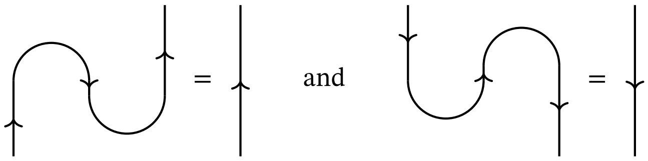
(9.3.2)

Similarly, the counit morphism, $\varepsilon : L \otimes R \to I$ is drawn as



(9.3.3)

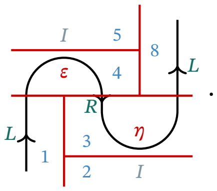
The duality conditions then becomes



(9.3.4)

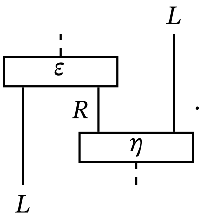
which are known as the **snake equations**.

It can be a bit difficult to see how this is the same as the duality condition at first. Consider the first snake equation. At the bottom we have a single wire with an arrow pointing up, so L, the space next to this can represent the unit, so we have $L \otimes I$. Then we're applying η to this identity, giving $L \otimes (R \otimes L)$. Next we reassociate, giving $(L \otimes R) \otimes L$, then apply ε, giving $I \otimes L$, and finally we remove the unit, leaving us with just L, and the line on the right hand side is just id_L. It can be helpful to break the diagram up into the following sections:



(9.3.5)

We can also draw this as



(9.3.6)

9.4 Duals: The Examples

9.4.1 FHilb

The monoidal category **FHilb** has all duals. Every finite-dimensional Hilbert space is both left and right dual to its dual space, H^* in a canonical way. The counit of the duality $H \dashv H^*$ is given by the map

$$\varepsilon : H \otimes H^* \to \mathbb{C} \tag{9.4.1}$$

$$|\varphi\rangle \otimes \langle\psi| \mapsto \langle\varphi|\psi\rangle. \tag{9.4.2}$$

Given an orthonormal basis, $\{|i\rangle\}$, the unit is given by the map

$$\eta : \mathbb{C} \to H^* \otimes H \tag{9.4.3}$$

$$1 \mapsto \sum_i \langle i| \otimes |i\rangle \tag{9.4.4}$$

extended by linearity to all of $\mathbb{C}$. We'll show later that despite using a basis in this definition the unit morphism is basis independent.

We now need to show that the snake equations hold. We'll do the first one here, using the version split into different regions above. Region 1 consists of just L, which in this case is H. We can show that the snake equation holds on the basis vectors, and then by linearity it holds for all vectors, so consider some specific basis vector $|j\rangle$ for fixed j. The first thing we do is affix the unit on the right, using ρ_H^{-1}, which gives us $|j\rangle \otimes 1$, having now accounted for region 2. Region 3 says to apply

$\mathrm{id}_H \otimes \eta$, which gives

$$|j\rangle \otimes \left(\sum_i \langle i| \otimes |i\rangle\right) = \sum_i |j\rangle \otimes (\langle i| \otimes |i\rangle) \cong \sum_i (|j\rangle \otimes \langle i|) \otimes |i\rangle \tag{9.4.5}$$

having used linearity of the tensor product to pull the sum out, and then the associator to move the brackets. Now region 5 tells us to apply $\varepsilon \otimes \mathrm{id}_H$, which gives

$$\sum_i \langle i|j\rangle \otimes |i\rangle = \sum_i \delta_{ij} \otimes |i\rangle = 1 \otimes |j\rangle \tag{9.4.6}$$

having used the orthonormality of the basis and then the fact that $0 \otimes |i\rangle$ gives zero when we act on it with the left unitor, and so the final step, region 6, of applying the left unitor gives us $|j\rangle$. So we see that this is indeed the identity.

Infinite dimensional Hilbert spaces don't have duals. The issue is with the sum

$$\sum_i \langle i| \otimes |i\rangle, \tag{9.4.7}$$

which is not defined in the infinite dimensional case.

9.4.2 $\mathbf{Mat}_{\Bbbk}$

Recall that the category $\mathbf{Mat}_{\Bbbk}$ has natural numbers as objects and $\mathbf{Mat}_{\Bbbk}(n, m)$ consists of all $m \times n$ matrices with entries in $\Bbbk$. This is a monoidal category with the monoidal product on morphisms given by the Kronecker product of matrices and on objects by the normal product of natural numbers. The unit object is the natural number 1.

In this category every object is self-dual with the canonical choice of unit and counit being

$$\eta : 1 \mapsto \sum_i |i\rangle \otimes |i\rangle, \qquad \text{and} \qquad \varepsilon : |i\rangle \otimes |j\rangle \mapsto \delta_{ij} 1 \tag{9.4.8}$$

where we give linear maps to be interpreted as matrices.

9.4.3 Rel

In **Rel** every object is self-dual. The unit morphism, : $1 \mapsto S \times S$, and counit morphism, $S \times S \to 1$, with $1 = \{\bullet\}$ being the singleton set, and S an arbitrary set, are defined to be the relations

- $\bullet \sim_\eta (s, s)$ for all $s \in S$; and
- $(s, s) \sim_\varepsilon \bullet$ for all $s \in S$

respectively.

We should check that the snake equations hold, again we'll check the first here. We start with $s \in S$, acting with the right unitor we have that $s \sim_{\rho_S} (s, s)$. Acting with $\mathrm{id}_S \otimes \eta$ we get $(s, s) \sim_{\mathrm{id}_S \otimes \eta} (s, (t, t))$ for all $t \in S$. Reassociating we have $(s, (t, t)) \sim_\alpha ((s, t), t)$. Acting with $\varepsilon \otimes \mathrm{id}_S$ we have $((s, t), t) \sim_{\varepsilon \otimes \mathrm{id}_S} (\bullet, t)$ where we now fix $t = s$, since if this is not the case then there is no relation here, compare this with the use of δ_{ij} in the case of **Hilb**. Finally, using the left unitor we have $(\bullet, t) \sim_{\lambda_S} t$, with $t = s$, so this is indeed the identity relation as chaining these relations together we get $s \sim s$.

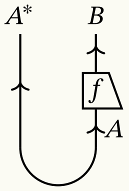
9.4.4 Set

The category of sets is quite boring when it comes to duals, only singletons have duals, and up to isomorphism there's only one singleton, and it is self dual. To show this we make the following definitions.

Definition 9.4.9 — Names and Conames Consider some monoidal category with dualities $A \dashv A^*$ and $B \dashv B^*$. Given a morphism $f\colon A \to B$ its

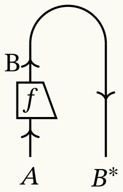
- **name**, $\ulcorner f \urcorner\colon I \to A^* \otimes B$; and
- **coname**, $\llcorner f \lrcorner\colon A \otimes B^* \to I$,

are the morphisms



(9.4.10)

and

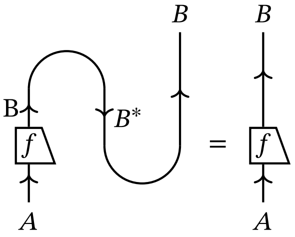


(9.4.11)

respectively.

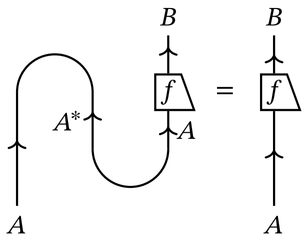
Note that the name is defined using the unit of the duality $A \dashv A^*$, and the coname is defined using the counit of the duality $B \dashv B^*$.

The useful thing about names/conames is that we can recover the morphism from them. To recover the morphism from the coname we just have to add in a unit (and an associator) forming the morphism



(9.4.12)

having applied one of the snake equations of $B \dashv B^*$ to straighten out the wire. Similarly to recover the morphism from the name we just have to add in a counit (and an associator) forming the morphism



(9.4.13)

having applied one of the snake equations of $A \dashv A^*$ to straighten out the wire.

In **Set** the unit object, the singleton, 1, is terminal, meaning all morphisms into 1 from the same object are equal. This means that all conames $\llcorner f \lrcorner : A \otimes B^* \to 1$ must be equal. Thus by the above construction all morphisms $f : A \to B$ are equal. This can only be the case if $A \cong B \cong 1$, or if $B = \varnothing$, but $\varnothing$ does not have a dual since there is no function $1 \to \varnothing \times \varnothing^*$ for any set $\varnothing^*$, since $\varnothing \times X = \varnothing$ for any set X, and there are no functions into $\varnothing$, apart from the identity $\varnothing \to \varnothing$, which isn't useful here as the domain we're interested in is 1.

Ten

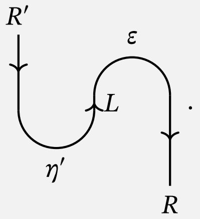
Properties of Duals

10.1 Duals are Unique up to Isomorphism

Often in maths after making a definition one of the first things we ask is about the uniqueness of the thing that we are defining. Strict uniqueness is too strong a condition when it comes to uniqueness of many categorical definitions. Instead we look at the weaker condition of uniqueness up to isomorphism. That is, if some property defines an object up to isomorphism and both A and B have this property then we must have $A \cong B$. Duals are unique up to isomorphism.

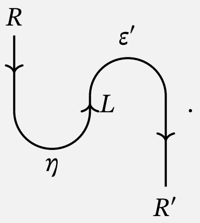
Lemma 10.1.1 — Duals are Unique up to Isomorphism Let $\mathbf{C}$ be a monoidal category with a duality $L \dashv R$, then, $L \dashv R'$ if and only if $R \cong R'$. Similarly $L' \dashv R$ if and only if $L \cong L'$.

Proof. We start with the forwards implication. Suppose that we have the dualities $(L, R, \eta, \varepsilon)$ and $(L, R', \eta', \varepsilon')$. We then want to define a map $R \to R'$ and show that this map is an isomorphism. It is possible to construct a map $R \to R'$ using the unit η' of $L \dashv R'$, and the counit ε of $L \dashv R$:



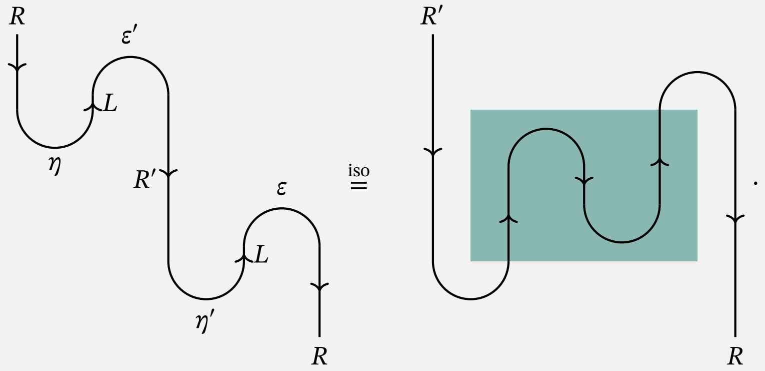
(10.1.2)

Similarly, we can define a map $R' \to R$ using η and ε':

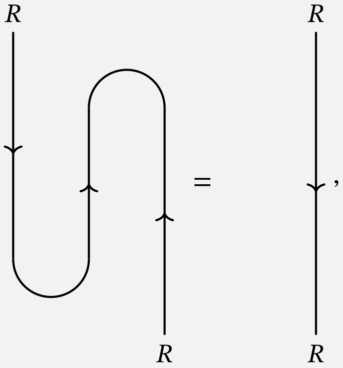


(10.1.3)

Now computing the composite we have

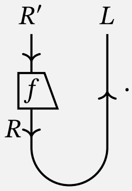


We can identify the shaded region as the snake equation for the duality $L \dashv R$, and so we can replace it with id_L. This leaves us with



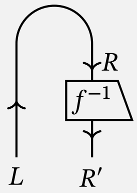
(10.1.4)

which is the snake equation for the duality $L \dashv R$. The composite the other way around can be worked out similarly, and is $\mathrm{id}_{R'}$. This proves the forward direction, that dualities $L \dashv R$ and $L \dashv R'$ can only exist if $R \cong R'$. Now suppose that we have a duality $L \dashv R$ and an isomorphism $f : R \to R'$. We can construct a duality $L \dashv R'$. In order to define a unit for this duality, $\eta' : I \to R' \otimes L$, we need a way to produce $R' \otimes L$. With the given data there is only one way to do this, we first produce $R \otimes L$ with η, then produce $R' \otimes L$ by acting on this with $f \otimes \mathrm{id}_L$. Thus, we define the unit of the duality $L \dashv R'$ to be



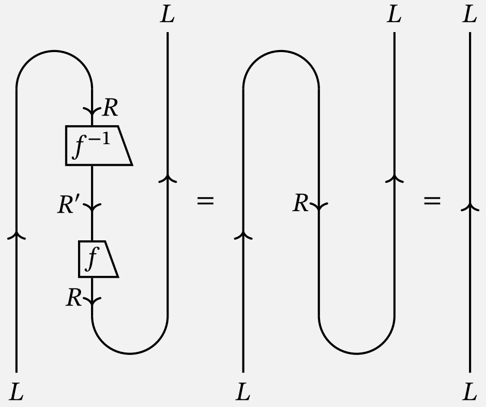
(10.1.5)

Similarly, we can define the counit of the dual $L \dashv R'$ to be



(10.1.6)

Computing the composite of this we have



(10.1.7)

having used $f^{-1} \circ f = \mathrm{id}_R$ and then the snake equation. Similarly, the composite in the opposite order is $\mathrm{id}_{R'}$, we just use the snake equation and then inverses.

The proof that $L \dashv R$ and $L' \dashv R$ if and only $L \cong L'$ follows the same steps. □

10.2 Unit Determines Counit

Everything in this section holds if we swap unit and counit around, an example of duality in the broader sense of reversing morphisms.

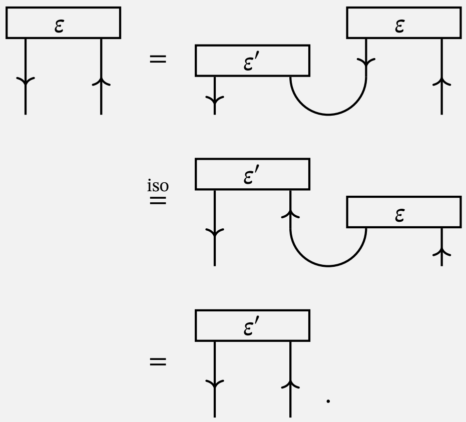
The definition of a dual is actually somewhat overdetermined, if we know the unit of a duality then this uniquely determines the counit. However, finding the counit is not always trivial, so we specify both the unit and counit when we specify a duality.

The proof that the unit uniquely defines the counit is reminiscent of the proof of uniqueness of the inverse in a group, just using the snake equations in place of the group laws. This proof is as follows. Take a group, G, and some element $g \in G$. Then suppose that both h and h' are inverses of g. In particular, we then have $gh' = 1 = hg$. Hence,

$$h = h1 = hgh' = 1h' = h'. \tag{10.2.1}$$

Lemma 10.2.2 — Unit Determines Counit Let **C** be a monoidal category with dualities $(L, R, \eta, \varepsilon)$ and $(L, R, \eta, \varepsilon')$. Then $\varepsilon = \varepsilon'$.
Similarly if we have dualities $(L, R, \eta, \varepsilon)$ and $(L, R, \eta', \varepsilon)$ then $\eta = \eta'$.

Proof. First consider the counit ε, we can take one leg of this and treat it as the identity, which we can then use the snake equation on. Rearranging the diagram allows us to apply the snake equation again, leaving us with just the counit ε'. That is,



(10.2.3)

□

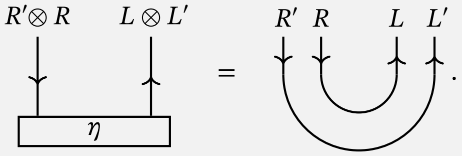
Note that if we have dualities $(L, R, \eta, \varepsilon)$ and $(L, R, \eta', \varepsilon')$ we cannot say whether $\eta = \eta'$ or $\varepsilon = \varepsilon'$.

10.3 Duals Respect Tensor Products

In a monoidal category we can take tensor products and sometimes find dualities. We should check if these are compatible, that is, is a duality of tensor products a tensor product of dualities? The answer is yes.

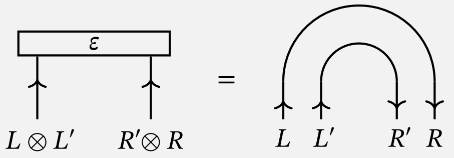
Lemma 10.3.1 — Duals Respect Tensor Products Let **C** be a monoidal category with dualities $L \dashv R$ and $L' \dashv R'$, then $L \otimes L' \dashv R' \otimes R$.

Proof. Suppose $L \dashv R$ and $L' \dashv R'$. We can define a duality $L \otimes L' \dashv R' \otimes R$ by defining the unit using the two units of $L \dashv R$ and $L' \dashv R'$:



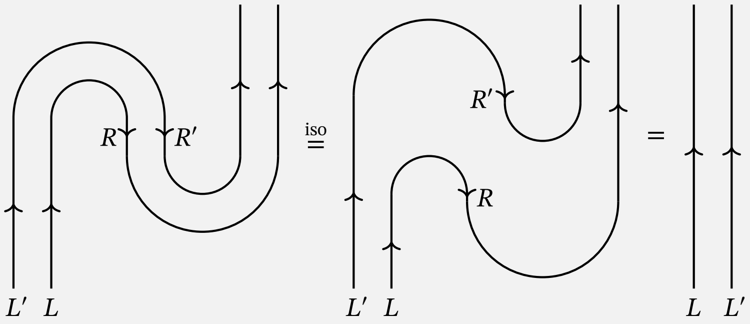
(10.3.2)

Similarly, we an define the counit using the two units of $L \dashv R$ and $L' \dashv R'$:



(10.3.3)

We can prove the snake equations using the graphical notation and the snake equations for the two individual dualities:



(10.3.4)

The other snake equation can be proved in the same way. □

Along with the tensor product we also get the unit object, this is always self dual.

Lemma 10.3.5 — The Unit Object is Self Dual Let **C** be a monoidal category and I the monoidal unit. Then $I \dashv I$.

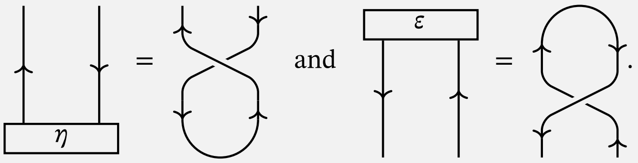
Proof. First, notice that we have morphisms $\lambda_I^{-1} : I \to I \otimes I$ and $\lambda_I : I \otimes I \to I$, which have the correct domain and codomain to be candidates for the unit, η, and counit, ε, respectively. If we consider the commuting diagrams defining duals then taking the unit to be $\eta = \lambda_I^{-1}$ and the counit to be $\varepsilon = \lambda_I$ these diagrams are formed entirely from unitors, identities, and associators, and so commute by the coherence theorem. Hence, $(I, I, \lambda_I^{-1}, \lambda_I)$ is a duality. □

10.4 Duals Respect Braidings

In a braided monoidal category as well as the monoidal structure of the last section we also have a braiding. We should check if this is compatible with dualities. Since braidings are about swapping the orders of objects in a tensor product duality respecting braiding would mean that if we have a duality and we swap left and right using a braiding we should still have a duality.

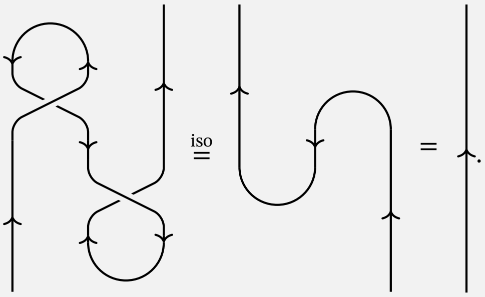
Lemma 10.4.1 — Duals Respect Braidings Let $\mathbf{C}$ be a braided monoidal category with duality $L \dashv R$ then there is a duality $R \dashv L$.

Proof. Given a duality $L \dashv R$ and a braiding, σ, we can construct a new duality $R \dashv L$ by defining the new unit and counit as



(10.4.2)

We can prove the snake equations using the graphical notation and the snake equations for the original duality:



(10.4.3)

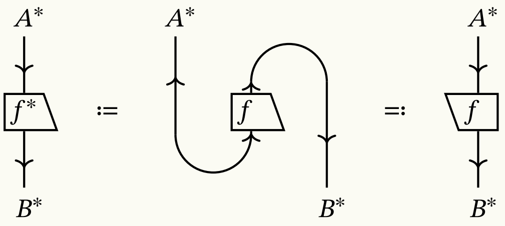
The other snake equation can be proved in the same way. □

This allows us to be a bit less careful with what is left and right in a duality in any braided monoidal category, which is most of the categories we're interested in.

10.5 Dual Morphisms

If we have a morphism, say $f : A \to B$, in which A and B happen to have duals, $A \dashv A^*$ and $B \dashv B^*$, then it makes sense to ask if we can find a morphism between the right duals. We might initially look for a morphism $A^* \to B^*$, but we won't find one in general that works nicely with the duality. Instead we find a morphism $B^* \to A^*$ instead.

Definition 10.5.1 — Dual Morphism Given a morphism $f : A \to B$ with dualities $A \dashv A^*$ and $B \dashv B^*$ the **right dual**, $f^* : B^* \to A^*$, is defined as

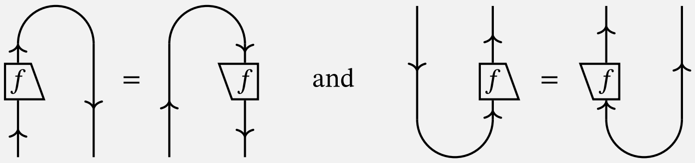


(10.5.2)

where the first equality defines f^* and the second defines the graphical notation for f^*, we rotate the box, as opposed to $f^\dagger$, where we reflect the box.

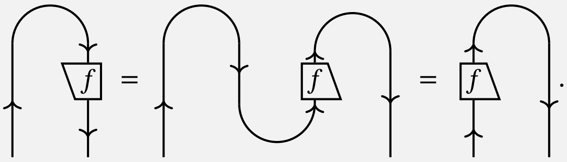
The idea behind duals is that as well as sliding morphisms along wires we can also slide them through units and counits, and in doing so they are rotated. This is the next lemma.

Lemma 10.5.3 — Sliding For all morphisms $f : A \to B$ in a monoidal category with chosen duals $A \dashv A^*$ and $B \dashv B^*$ we have



(10.5.4)

Proof. We'll prove the first of these. It follows immediately in the graphical notation by expanding the definition of f^* and applying the snake equations:



(10.5.5)

The other equation can be proved similarly. □

Now we have defined duals of objects and morphisms it makes sense to ask if taking duals defines a functor, and indeed it does.

Definition 10.5.6 — Dual Functor In a monoidal category, C, with chosen right duals, that is where we have made a choice for every object, A, of some A^* such that $A \dashv A^*$, there is a contravariant functor

$$(-)^* : \mathsf{C} \to \mathsf{C} \tag{10.5.7}$$

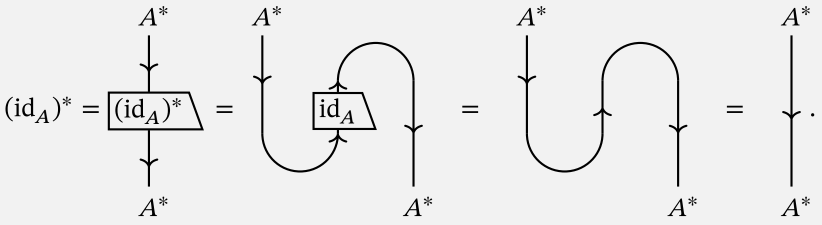
$$A \mapsto A^* \tag{10.5.8}$$

$$f \mapsto f^* \tag{10.5.9}$$

where $A \dashv A^*$ is the chosen duality, and $f^* : B^* \to A^*$ is the right dual of the morphism $f : A \to B$. This is the **dual functor**.

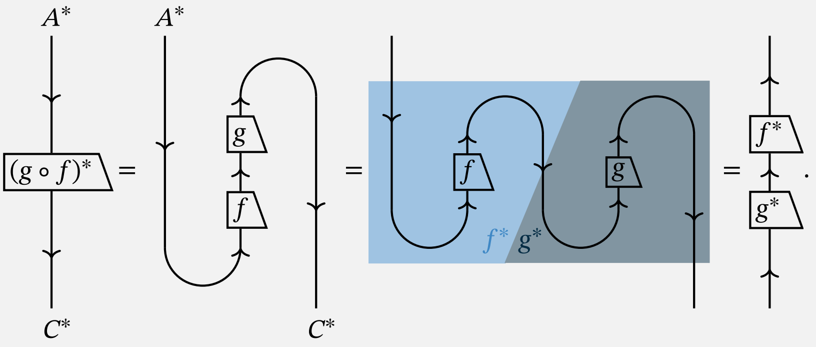
Lemma 10.5.10 The dual functor is a functor.

Proof. We start by proving that $(\mathrm{id}_A)^* = \mathrm{id}_{A^*}$ for all objects A. This is immediately clear from the definition of id_{A^*} in graphical notation and the equations:



(10.5.11)

This proves $(\mathrm{id}_A)^* = \mathrm{id}_{A^*}$, as required for a functor.
Given morphisms $f : A \to B$ and $g : B \to C$ we can form $(g \circ f) : A \to C$ and consider $(g \circ f)^* : C^* \to A^*$. We can then prove that $(g \circ f)^* = f^* \circ g^*$, as required for a contravariant functor. To do so we insert the identity, using the snake equation, between f and g:



(10.5.12)

This proves that $(g \circ f)^* = f^* \circ g^*$. Hence, $(-)^*$ is a contravariant functor.
□

10.5.1 Examples

In $\mathbf{FVect}_{\Bbbk}$ and **FHilb** a morphism is a linear map $f : V \to W$, and the dual spaces are spaces of linear maps into $\Bbbk$ or $\mathbb{C}$ respectively. The dual of f is the map of linear maps $f : W^* \to V^*$ defined by $f^*(e) = e \circ f$ where $e : W \to \Bbbk$ or $e : W \to \mathbb{C}$ respectively is an arbitrary linear map. This is sometimes called the **pullback**.

In $\mathbf{Mat}_{\Bbbk}$ the morphisms are matrices, and the dual is just the transpose.

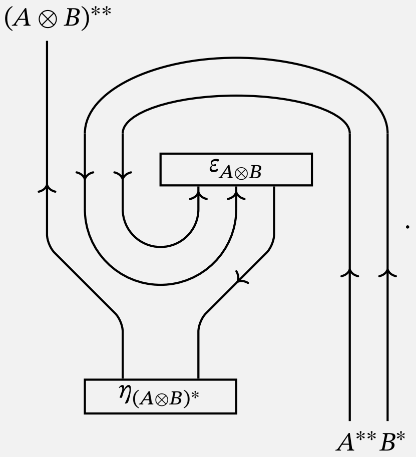
In **Rel** the dual of a relation is its converse. This means the right duals functor, $(-)^*$, and the dagger functor, $(-)^\dagger$, are the same, since both act as identities on objects since all objects are self dual in **Rel** and the dagger acts as an identity on objects by definition.

10.6 Double Duals

We've already seen that every vector space is canonically isomorphic to its double dual, that is $V \cong V^{**} = (V^*)^*$. In fact this holds in an arbitrary category with chosen right duals. Further, the action of taking double duals is compatible with the monoidal product.

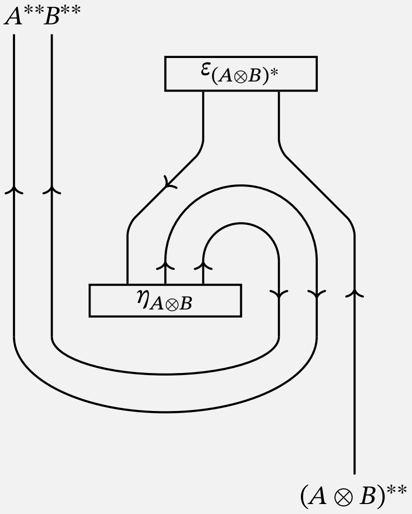
Lemma 10.6.1 In a monoidal category with chosen right duals $A^{**} \otimes B^{**} \cong (A \otimes B)^{**}$.

Proof. To prove this we need to witness an isomorphism $A^{**} \otimes B^{**} \to (A \otimes B)^{**}$. One such isomorphism is



(10.6.2)

The inverse to this morphism, proving it is an isomorphism, is



(10.6.3)

where the cups and caps are the units and counits of the relevant dualities. To show these are inverses compose them and then apply the appropriate snake equations for each unit-counit pair which appears. □

Eleven

Teleportation

11.1 Quantum Teleportation

R For more details see my notes from *Principles of Quantum Mechanics*.

Suppose Alice is in possession of a quantum state, which for simplicity we take to be a single qubit, $|\psi\rangle$. Quantum teleportation is a protocol by which Alice and Bob can prepare a joint set of states and then by communicating only classical information manipulate a system in Bob's presence into the state $|\psi\rangle$. Alice's state is necessarily destroyed in this process, otherwise the no cloning theorem is violated, and so this process makes a state disappear from Alice's possession and appear in Bob's possession, i.e. it teleports.

The process is as follows. Alice and Bob prepare ahead of time an extra two qubits in the Bell state

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}[|+-\rangle - |-+\rangle]. \tag{11.1.1}$$

Alice then takes the first qubit here into her possession, and Bob takes the second qubit into their possession. This creates a "quantum channel" between Alice and Bob.

Alice now has two qubits, the original qubit $|\psi\rangle$, and her qubit from the entangled pair. In general Alice's qubit will be in the state

$$|\psi\rangle = a|+\rangle + b|-\rangle \tag{11.1.2}$$

for some $a, b \in \mathbb{C}$ with $a^2 + b^2 = 1$.

If we then consider the joint system of all three qubits the state of this system is

$$|\varphi\rangle = |\psi\rangle \otimes |\Psi^-\rangle \tag{11.1.3}$$

$$= [a|+\rangle + b|-\rangle] \otimes \frac{1}{\sqrt{2}}[|+-\rangle - |-+\rangle] \tag{11.1.4}$$

$$= \frac{c}{\sqrt{2}}[|++-\rangle - |+-+\rangle] + \frac{d}{\sqrt{2}}[|-+-\rangle - |--+\rangle]. \tag{11.1.5}$$

The four Bell states form a basis for the space $\mathbb{C}^2 \otimes \mathbb{C}^2$, so we can write Alice's pair of states as a superposition of Bell states. For example,

$$|++\rangle = \frac{1}{\sqrt{2}}[|\Phi^+\rangle + |\Phi^-\rangle]. \tag{11.1.6}$$

If we do this with all of Alice's states then we find that

$$|\varphi\rangle = \frac{1}{2}[|\Psi^-\rangle \otimes (-a|+\rangle - b|-\rangle) + |\Psi^+\rangle \otimes (-a|+\rangle + b|-\rangle) \\ + |\Phi^-\rangle \otimes (a|-\rangle + b|+\rangle) + |\Phi^+\rangle \otimes (a|-\rangle - b|+\rangle)]. \quad (11.1.7)$$

Thus when Alice measures her states, changing them in the process, she will measure them to be in one of the Bell states. There are four of these, so by sending two classical bits to Bob, establishing a "classical channel", Alice can inform Bob of the result of her measurement.

This tells Bob what their state is, since the wavefunction collapses[1] to leave just one of the terms above. For example, if Alice measures $|\Phi^+\rangle$ then Bob knows that their qubit is in the state $a|-\rangle - b|+\rangle$. Bob knows that they can then put their qubit into the state $|\psi\rangle$ by applying a unitary transformation, in this example,

$$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}. \quad (11.1.8)$$

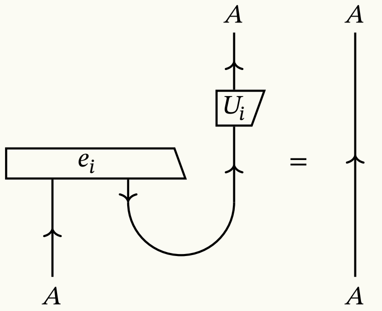
This can be done regardless of which state Alice measures, and so Bob can always put their qubit in the state $|\psi\rangle$.

[1] or insert your favourite interpretation of quantum mechanics here

11.2 Abstract Teleportation

We will now define a general notion of teleportation, which becomes quantum teleportation in **Hilb**.

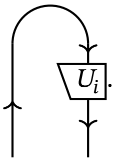
Definition 11.2.1 — Teleportation Consider a monoidal dagger category with right duals. A **teleportation procedure** is a finite family of effects, $e_i : A \otimes A^* \to I$, and unitary morphisms $U_i : A \to A$ such that



(11.2.2)

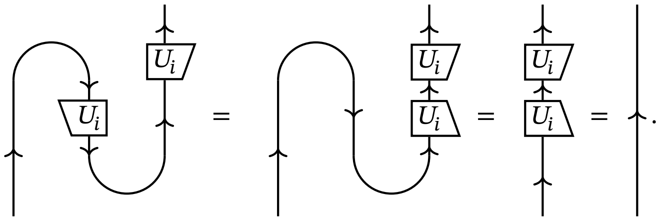
for all i.

This definition actually carries more information than required, since given unitary morphisms U_i we can always take e_i to be



(11.2.3)

This works since



(11.2.4)

This uses the sliding of Lemma 10.5.3, then the snake equation, and then the definition of unitarity, namely $U_i \circ U_i^\dagger = \mathrm{id}_A$.

Consider the first diagram above, we can interpret this as a procedure as follows:

1. Start with a single system, A.
2. Independently prepare a joint system, $A^* \otimes A$, in the state η, so the combined system is $A \otimes (A^* \otimes A)$.
3. Reassociate, so that we consider the initial system and one of the two systems in the joint system, $(A \otimes A^*) \otimes A$.
4. Perform a joint measurement on the first two systems, with the result given by the effect $\varepsilon \circ (\mathrm{id}_L \otimes U_i^*)$.
5. Perform a unitary operation, $U_i^\dagger$, on the remaining system.

11.3 Quantum Teleportation Again

In **Hilb** a teleportation procedure is exactly the process of quantum teleportation. We can see this by considering the steps listed above and comparing to the qubit teleportation procedure at the start of the chapter.

1. Start with Alice's system that she wants to send to Bob. This is some state in in a Hilbert space $A = \mathbb{C}^2$.
2. Prepare a state in the Bell state, this is an element of the joint system $\mathbb{C}^2 \otimes \mathbb{C}^2$ (note that $\mathbb{C}^2 \cong (\mathbb{C}^2)^*$).
3. Reassociate, which means considering the pair of qubits in Alice's possession.
4. Alice performs a measurement.
5. Bob can then perform a unitary operation to get their qubit into the same state as Alice's at the start of the process.

We can be more explicit for the case of teleporting a qubit. We have $A = \mathbb{C}^2$ and $A \cong \mathbb{C}^2$. We can represent morphisms with matrices after choosing some orthonormal basis. Then we have

$$\eta^\dagger = \varepsilon = \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}. \tag{11.3.1}$$

If we choose the unitary operators

$$\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}, \quad \begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}, \quad \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}, \quad \text{and} \quad \begin{pmatrix}0 & 1\\-1 & 0\end{pmatrix}, \tag{11.3.2}$$

which we might recognise as the identity, I, and the Pauli matrices, σ^3, σ^1, and $i\sigma^2$ respectively. The family of effects are then given by $\varepsilon \circ (I \otimes U_i^*)$, which gives

$$\begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix}, \quad \begin{pmatrix}1 & 0 & 0 & -1\end{pmatrix}, \quad \begin{pmatrix}0 & 1 & 1 & 0\end{pmatrix}, \quad \text{and} \quad \begin{pmatrix}0 & 1 & -1 & 0\end{pmatrix}.$$

This is a complete set of effects, since it is a basis for **Hilb**$(\mathbb{C}^2 \otimes \mathbb{C}^2, \mathbb{C}) = (\mathbb{C}^2 \otimes \mathbb{C}^2)^* \cong \mathbb{C}^2 \otimes \mathbb{C}^2 \cong \mathbb{C}^4$, and so this teleportation procedure is guaranteed to work no matter which result we produce at the measurement step.

11.4 One Time Pads

Consider an implementation of the teleportation procedure in **Rel**. We'll take the simple case of $L = R = \{0, 1\}$ and

$$\eta^\dagger = \varepsilon = \begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix}. \tag{11.4.1}$$

There are only two unitary relations $\{0, 1\} \to \{0, 1\}$, and they are represented by the matrices

$$U_1 = \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}, \quad \text{and} \quad U_3 = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}. \tag{11.4.2}$$

Taking these as our family of unitary morphisms, U_i, we get the effects

$$\begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix}, \quad \text{and} \quad \begin{pmatrix}0 & 1 & 1 & 0\end{pmatrix}. \tag{11.4.3}$$

These form a complete set of effects.

The teleportation procedure is then as follows.

1. Alice starts with a single bit she wishes to communicate to Bob.
2. Alice and Bob each take one bit from a preprepared joint state of $\{0, 1\} \times \{0, 1\}$. This state, η, is $\{(0, 0), (1, 1)\}$, so they will both take the same bit.
3. Reassociate, which means considering the pair of bits in Alice's possession.
4. Alice looks at the shared bit and sees it either matches the bit she wants to send or it doesn't.
5. Alice tells Bob which of these outcomes and if it matches he does nothing, U_1, and if it doesn't match he swaps it, U_2. Bob's bit from the shared system will now be the bit that Alice wanted to send in the first place.

This is how a one time pad works. The joint state is the one time pad, Alice then communicates information about the one time pad, and Bob can turn this into information about the state that Alice is trying to communicate. One can imagine a one time pad containing all messages Alice could possibly have reason to send, generating a shared message and a set of unitary operations swapping that message with a message at a different position on the one time pad. These are unitary since they are transpositions and so square to the identity. In reality at some point it becomes easier to have a fixed number of possible messages, say the 26 letters, and then break down longer messages into these. To do so securely Alice and Bob should use a different one time pad for each letter they send.

Twelve

Compact Categories

12.1 Compact Categories

Definition 12.1.1 — Compact Category A **compact category** is a symmetric monoidal category where every object has a dual.

All of **FVect**$_{\mathbb{k}}$, **FHilb**, **Mat**$_{\mathbb{k}}$, and **Rel** are compact categories.

In the graphical notation wires now come with arrows, this gives us a new notion of the orientation of a wire, and we have to preserve this property. This leads to us updating what it means for two diagrams to be equal.

Theorem 12.1.2 — Correctness of the Graphical Calculus for Compact Categories. Any well-formed equation of the form $f = g$ for morphisms f and g in a compact category follows from the axioms of a compact category if and only if it holds in the graphical notation up to four-dimensional oriented isotopy.

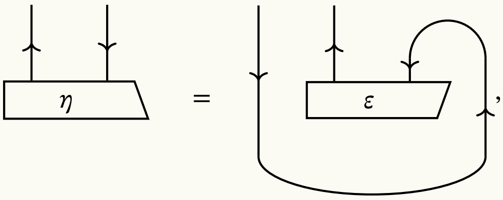
Intuitively this just means that we can't turn arrows around, which shouldn't happen when manipulating a diagram anyway.

12.2 Dagger Compact Category

Lemma 12.2.1 In a monoidal dagger category if $L \dashv R$ then $R \dashv L$.

Proof. Suppose $\eta : I \to R \otimes L$ and $\varepsilon : L \otimes R \to I$ are the unit and counit of the $L \dashv R$ duality. Then $\eta' = \varepsilon^\dagger : L \otimes R \to I$ and $\varepsilon' = \eta^\dagger : R \otimes L \to I$ can be taken as the unit and counit of the $R \dashv L$ duality. It is possible to show that the diagrams defining a duality commute by taking one, and applying the dagger functor, flipping all the arrows, and then using $(f \otimes g)^\dagger = f^\dagger \otimes g^\dagger$, which turns one of these diagrams into the other but with L and R exchanged. □

Definition 12.2.2 — Dagger Dual In a monoidal dagger category a **dagger dual** is a duality $A \dashv A^*$ witnessed by morphisms $\eta\colon I \to A^* \otimes A$ and $\varepsilon\colon A \otimes A^* \to I$ satisfying



(12.2.3)

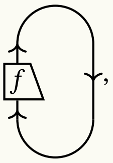
A **compact dagger category** is a symmetric dagger category which can be made into a compact category by taking all duals to be dagger duals.

FHilb, **Mat**$_{\Bbbk}$, and **Rel** are all compact dagger categories.

12.3 Trace

Definition 12.3.1 — Trace In a compact dagger category the **trace** of an endomorphism, $f\colon A \to A$, is the scalar

$\mathrm{tr}_A f :=$



(12.3.2)

that is

$$\mathrm{tr}_A f = \varepsilon \circ (f \otimes \mathrm{id}_A^*) \circ \eta. \tag{12.3.3}$$

Definition 12.3.4 — Dimension In a compact dagger category the **dimension** of an object, A, is the trace of the identity on that object, $\dim A := \mathrm{tr}_A(\mathrm{id}_A)$.

The names trace and dimension are used as these generalise these concepts from **FHilb**.

Lemma 12.3.5 In **FHilb** the trace as defined in Definition 12.3.1 is the ordinary trace and the dimension is the ordinary dimension.

Proof. Recall that in **FHilb** the unit is given by

$$\eta\colon \mathbb{C} \to H^* \otimes H \tag{12.3.6}$$

$$1 \mapsto \sum_i \langle i| \otimes |i\rangle \tag{12.3.7}$$

where $\{i\}$ is a basis for H. The counit is defined by

$$\varepsilon\colon H\otimes H^* \to \mathbb{C} \tag{12.3.8}$$
$$|i\rangle \otimes \langle j| \mapsto \langle i|j\rangle. \tag{12.3.9}$$

Then the trace of f in H is

$$\begin{aligned}
\mathrm{tr}_H f &= (\varepsilon \circ (f\otimes \mathrm{id}_{H^*})\circ \eta)(f) && (12.3.10)\\
&= (\varepsilon \circ (f\otimes \mathrm{id}_{H^*}))\left(\sum_i \langle i| \otimes |i\rangle\right) && (12.3.11)\\
&= \varepsilon\left(\sum_i \langle f(i)| \otimes |i\rangle\right) && (12.3.12)\\
&= \langle i|f(i)\rangle && (12.3.13)\\
&= \langle i|f|i\rangle && (12.3.14)
\end{aligned}$$

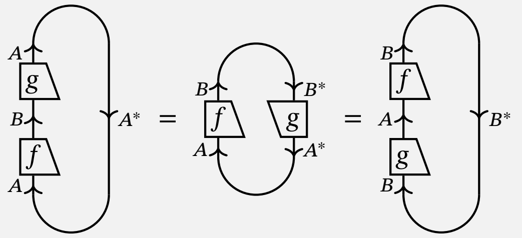
which is just the usual definition of the trace.
The trace of an identity matrix, and hence an identity linear transformation, is just the number of rows/columns in the matrix, and hence is the dimension of the vector space. So since the trace in **FHilb** is just the usual trace the dimension is just the usual dimension. □

The trace has many of the expected properties. For example, it is cyclic.

Lemma 12.3.15 — Trace is Cyclic The trace is cyclic, that is $\mathrm{tr}_A(g\circ f) = \mathrm{tr}_B(f\circ g)$ for all morphisms $f\colon A \to B$ and $g\colon B\to A$ in a compact category.

Proof.

$\mathrm{tr}_A(g\circ f) =$



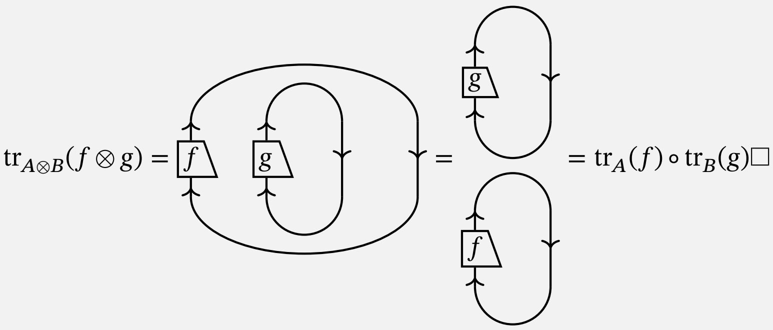
$= \mathrm{tr}_B(f\circ g)$.□

Lemma 12.3.16 The trace of a scalar is that scalar, that is $\mathrm{tr}_I\, s = s$.

Proof. The identity is self dual with the unit and counit given by λ_I^{-1} and λ_I respectively. We don't draw these in the diagram, and the objects are all unit objects, which we also don't draw. Hence, the diagram defining the trace is simply the diagram consisting of the scalar s. □

Lemma 12.3.17 In a compact category with morphisms $f\colon A \to A$ and $g\colon B \to B$ we have $\operatorname{tr}_{A\otimes B}(f \otimes g) = \operatorname{tr}_A(f) \circ \operatorname{tr}_B(g)$.

Proof.


$\operatorname{tr}_{A\otimes B}(f \otimes g) =$
f
g
=
g
f
$= \operatorname{tr}_A(f) \circ \operatorname{tr}_B(g)$□

Lemma 12.3.18 In a compact dagger category with morphism $f\colon A \to A$ we have $(\operatorname{tr}_A f)^\dagger = \operatorname{tr}_A(f^\dagger)$.

Proof. Take the diagram defining the trace, flip it upside-down to take the dagger, this is the diagram defining the trace of $f^\dagger$. □

Lemma 12.3.19 In a compact dagger category with unit object I we have $\dim I = \mathrm{id}_I$.

Proof. Using Lemma 12.3.16 we have

$$\dim I = \operatorname{tr}_I \mathrm{id}_I = \mathrm{id}_I\,. \qquad \square$$

Lemma 12.3.20 In a compact dagger category with objects A and B we have $\dim(A \otimes B) = \dim(A) \circ \dim(B)$.

Proof. Using the functoriality of the tensor product and Lemma 12.3.17

$$\begin{aligned}\dim(A \otimes B) = \operatorname{tr}_{A\otimes B}(\mathrm{id}_{A\otimes B}) &= \operatorname{tr}_{A\otimes B}(\mathrm{id}_A \circ \mathrm{id}_B) \\ &= \operatorname{tr}_A(\mathrm{id}_A) \circ \operatorname{tr}_B(\mathrm{id}_B) = \dim(A) \circ \dim(B). \quad \square\end{aligned}$$

We can now show that any infinite-dimensional space does not have a dual. To do so we need to accept a few facts first. To start with given two vector spaces, V and W, we can form their direct sum, the vector space

$$V \oplus W \coloneqq \{(v, w) \mid v \in V \text{ and } w \in W\} \tag{12.3.21}$$

with pairwise vector addition:

$$(v, w) + (v', w') = (v + v', w + w'), \tag{12.3.22}$$

and scalar multiplication according to

$$z(v, w) = (zv, zw). \tag{12.3.23}$$

Writing $v + w$ for (v, w) and treating this sum as a formal (non-commutative) sum of vectors these operations naturally become

$$(v+w)+(v'+w') = (v+v')+(w+w'), \qquad \text{and} \qquad z(v+w) = zv+zw. \tag{12.3.24}$$

The dimension of $V \oplus W$ is simply

$$\dim(V \oplus W) = \dim V + \dim W, \tag{12.3.25}$$

which is clear for the traditional definition of the dimension and finite dimensional vector spaces, but holds for the generalised definition and infinite dimensional vector spaces.

For an infinite dimensional vector space V we have an isomorphism $V \oplus \Bbbk \cong V$. An explicit construction is to take a basis[1] $\{v_i\}$ for V and then $\{v_i \oplus 1\}$ is a basis for $V \oplus \Bbbk$, which clearly has the same cardinality. This can then be extended linearly to all elements of $V \oplus \Bbbk$.

[1] assuming the axiom of choice

Lemma 12.3.26 Infinite dimensional Hilbert spaces don't have duals.

Proof. Suppose H is an infinite dimensional vector space with dual. Then we can construct a scalar $\dim H$. We have an isomorphism $H \oplus \mathbb{C} \cong H$, and hence $\dim(H \oplus \mathbb{C}) = \dim(H) + 1$. But isomorphic vector spaces have the same dimension, so $\dim(H \oplus \mathbb{C}) = \dim H$. This is not possible since in **Hilb** the scalars can be identified with the complex numbers, and there is no complex number, z making $z + 1 = z$ true, so there is no way to identify $\dim H$ with a complex number, a contradiction since $\dim H$ is by assumption a scalar. □

Part IV

Structures Within Categories

Thirteen

(Co)Monoids

I'm going to diverge from the order in which things were approached in lectures, because I think it makes more sense to define monoids before comonoids.

13.1 Universal Algebra

This section will be a bit technical, and is beyond the scope of the course, the important parts will be covered again at the start of the next section.

In category theory we aim to work with morphisms between objects, rather than with "elements" of the objects, which isn't even a meaningful concept in some categories. Something like a group is defined as a set, G, a binary operation, $G \times G \to G$, a distinguished element, 1, and for every element, g, another element, g^{-1}. This is too much talk about elements for category theory. Let's have a look at an alternative characterisation without reference to elements.

A set, G, and binary operation, $G \times G \to G$ is fine, no mention of elements here. A binary operation is a 2-ary operation, taking two things to one, we might think of a binary operation as a map $m\colon G^2 \to G$. Next we need to define the identity. We can do so by introducing a 0-ary operation $e\colon G^0 \to G$? This takes in no things and produces one thing, which we can take to be 1, the identity. Finally, we need the inverse, we can characterise this as a map $i\colon G \to G$ taking in one element, g, and returning one element, $i(g) = g^{-1}$.

The axioms of a group then become requirements on these operations, namely for all $g, h, k \in G$

- $m(e(), g) = m(g, e()) = g$, writing $e()$ to remind us that e is an operation with no arguments
- $m(g, i(g)) = m(i(g), g) = e()$,
- $m(g, m(h, k)) = m(m(g, h), k)$.

Notice that there are no quantifiers here apart from the "for all" at the start, this is a desirable feature.

Universal algebra studies algebraic structures by characterising them as a set and some collection of operations on that set. The signature of an algebraic structure is just a list of the arity of the operations. We can then define a variety of algebras as a collection of all sets with some fixed signature and all satisfying some set of equational laws, like those above for a group, with only outermost quantifiers.

Examples are

- The variety of groups which has signature $(2, 1, 0)$ and satisfy the equational laws above.
- The variety of abelian groups which has the signature $(2, 1, 0)$ and the additional equational law that for all $g, h \in G$ we have $m(g, h) = m(h, g)$
- The variety of monoids which has signature $(2, 0)$ and satisfies the equational laws above, apart from the second one with inverses.
- The variety of rings, which has the signature $(2, 2, 0, 0, 1)$ and satisfies the mononoid and abelian group laws for appropriately chosen operations.
- The variety of vector spaces, which has the signature $(2, 1, 1, 1, \dots)$ with one unary operation for each scalar, namely multiplication by that scalar. These satisfy equational laws like $m_\lambda(p(v, w)) = p(m_\lambda(v), m_\lambda(w))$ where $m_\lambda : V \to V$ is multiplication by the scalar λ and $p : V^2 \to V$ is vector addition.

One of the nice things about varieties of algebras is once we've defined one we can generalise it by replacing the set with an object in some other category, so long as it has finite products (to form G^2 etc.) and an initial object (to take the place of G^0), and functions with morphisms of this category. This is what we will do here, focusing on varieties of monoids.

An example of this is group objects, which are objects of the variety of groups.

- Interpreted in **Set** we just get normal groups.
- Interpreted in **Grp** we instead get Abelian groups This follows since we require the inverse to be a homomorphism, that is $g \mapsto g^{-1}$ is a homomorphism, and this is only the case in an Abelian group.
- Interpreting in **Top** we get topological groups, which are topological spaces with a group structure in which the product and inversion maps are continuous, this isn't that interesting and is basically spelling out the definition of a group object in the specific context of **Top**.
- Interpreting in **SmthMan**, the category of smooth manifolds, with smooth functions as morphisms, a group object is exactly a Lie group.

13.2 Monoids: The Idea

A monoid can be characterised without reference to explicit elements as follows.

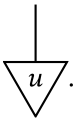
> **Definition 13.2.1 — Monoid** A **monoid**, (M, m, u), is a set, M, equipped with a binary operation, $m : M \times M \to M$, and a 0-ary operation, $u : \{\bullet\} \to M$ satisfying the following axioms for all $x, y, z \in M$:
>
> - $m(u(\bullet), x) = m(x, u(\bullet)) = x$,
> - $m(x, m(y, z)) = m(m(x, y), z)$.

Clearly we can identify this with the usual definition of a monoid by taking $m(x, y) = xy$, multiplication, and $u(\bullet) = 1$, the identity, and then the laws become $1x = x1 = 1$, the identity law, and $x(yz) = (xy)z$, the associativity law.

The nice thing about this definition is that removing explicit reference to elements[1] we can make this more general by interpreting it in any monoidal category making the following replacements:

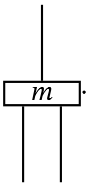
- Replace “a set, M” with “an object, M”.
- Replace $M \times M$ with $M \otimes M$.
- Replace functions with morphisms.
- Replace $\{\bullet\}$ with the unit object.

Before we give the more general definition of a monoid lets look at the monoid laws in the graphical notation. First, the identity is characterised by a morphism $e : I \to M$, so this is exactly a state,



(13.2.2)

Multiplication is then characterised by a morphism $M \otimes M \to M$, so this can be drawn as

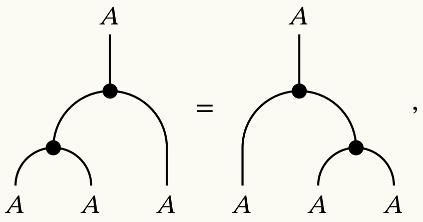


(13.2.3)

Rather than give a name to these morphisms we can just represent them with a dot with the appropriate number of inputs.

13.3 Monoids: The Definition

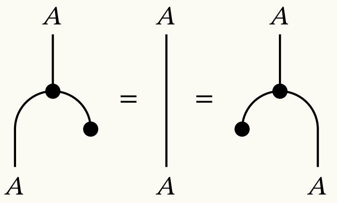
Definition 13.3.1 — Monoid A **monoid** in a monoidal category, **C**, is a triple, $(A, ⋏, ⫰)$, consisting of an object, $A \in \mathrm{Ob}(\mathbf{C})$, a morphism, $⋏ : A \otimes A \to A$, and a state, $⫰ : I \to A$, satisfying the **associativity** and **unitality** equations,



(13.3.2)

[1] apart from $\bullet \in \{\bullet\}$, but this is characterised as being the initial object in **Set** and so is unique up to isomorphism and doesn’t really count as talking about an element

and



(13.3.3)

respectively.

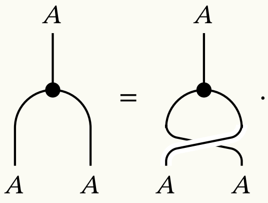
These relations can also be expressed as commutative diagrams for a monoid (A, m, e) we require

$$\begin{tikzcd}
A \otimes (A \otimes A) \arrow[r, "\mathrm{id}_A \otimes m"] \arrow[dd, "\alpha_{A,A,A}"'] & A \otimes A \arrow[dr, "m"] & \\
& & A, \\
(A \otimes A) \otimes A \arrow[r, "m \otimes \mathrm{id}_A"'] & A \otimes A \arrow[ur, "m"'] &
\end{tikzcd} \tag{13.3.4}$$

and

$$\begin{tikzcd}
A \otimes I \arrow[d, "\mathrm{id}_A \otimes u"'] & A \arrow[l, "\rho_A^{-1}"'] \arrow[r, "\lambda_A^{-1}"] \arrow[d, "\mathrm{id}_A"'] & I \otimes A \arrow[d, "\mathrm{id}_A \otimes u"] \\
A \otimes A \arrow[r, "m"'] & A & A \otimes A. \arrow[l, "m"]
\end{tikzcd} \tag{13.3.5}$$

Definition 13.3.6 — Commutative Monoid A **commutative monoid** in a braided monoidal category, **C**, is a monoid, $(A,$ ⅄, ↓$)$, satisfying the **commutativity** equation



(13.3.7)

This can also be expressed as a commutative diagram for the monoid (A, m, e):

$$\begin{tikzcd}
A \otimes A \arrow[r, "m"] \arrow[d, "\sigma_{A,A}"'] & A \\
A \otimes A. \arrow[ur, "m"'] &
\end{tikzcd} \tag{13.3.8}$$

13.4 Monoids: The Examples

In any monoidal category $(I, \lambda_I, \mathrm{id}_I)$ is trivially a monoid, all diagrams we write for the monoid laws are empty.

A monoid in **Set** is exactly the normal definition of a monoid.

A monoid in **Mon** is a commutative monoid. To see this note that if $((M, \cdot), m, u)$ is a monoid object in **Mon** then m must be a monoid homomorphism between $M \otimes M \to M$, and elements of $M \otimes M$ are of the form (a, b). Then for $a, b, a', b' \in M$ we have

$$aba'b' = m(a, b)m(a', b') = m((a, b), (a', b')) = m(aa', bb') = aa'bb'. \quad (13.4.1)$$

Cancelling the a and b', which can be done without requiring inverses, we have $ba' = a'b$.

A monoid $(G, \otimes_{\mathbb{Z}}, \mathbb{Z})$ in the category of Abelian groups, **Ab**, is a ring. That is, if we treat Abelian groups as $\mathbb{Z}$-modules and use the tensor product of $\mathbb{Z}$-modules as the product in our monoid then we already have addition, the Abelian group operation, and now we have multiplication, which follows the monoid laws, so this is both an Abelian group, $(G, +)$ and a monoid, $(G, \otimes_{\mathbb{Z}})$, and it is possible to demonstrate distributivity from distributivity of the tensor product, so this is a ring.

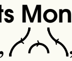
A monoid $(V, \times, 1)$ in the category of vector spaces, $\mathbf{Vect}_{\Bbbk}$, is a unital (associative) algebra. The monoid product in this case gives us a way of multiplying two vectors and is such that $1 \times v = v = v \times 1$ for all vectors $v \in V$. This is exactly a unital (associative) algebra. Examples of algebras are $\mathbb{C}^n$ under pointwise multiplication $((a, \dots, z) \times (a', \dots, z') = (aa', \dots, zz')$, with the unit $(1, \dots, 1))$ and $n \times n$ matrices, $\mathcal{M}_n(\Bbbk)$ with the usual matrix multiplication and the identity matrix as the unit.

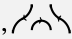
A monoid (a, mappend, mempty) in the category **Hask** of *Haskell* types and functions is exactly an instance of the Monoid typeclass.

A monoid (T, μ, η) in the category of endofunctors, $[\mathbf{C}, \mathbf{C}]$, is a monad. That is, $T \colon \mathbf{C} \to \mathbf{C}$ is a functor for some category $\mathbf{C}$, and $\mu \colon T \otimes T \to T$ and $\eta \colon \mathrm{id}_{\mathbf{C}} \Rightarrow T$ are natural transformations. This leads to the famously confusing statement

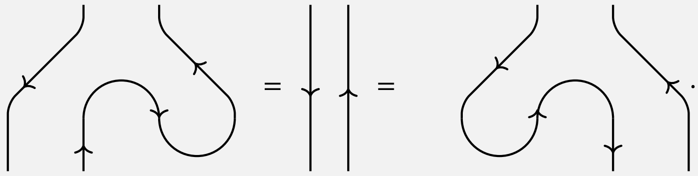
> A **monad** is a monoid in the category of endofunctors.

13.4.1 Pair of Pants Monoid

Definition 13.4.2 — Pair of Pants Monoid In a category in which every object has a chosen dual ($A^* \otimes A$, ,) is a monoid.

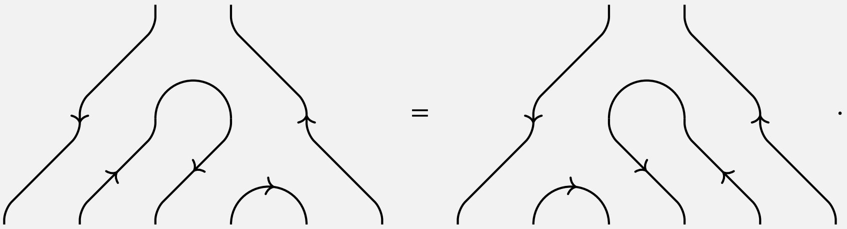
Lemma 13.4.3 — Pair of Pants is Monoid In a category where every object has a chosen dual ($A^* \otimes A$, ,) is a monoid.

Proof. First we demonstrate unitality:



. (13.4.4)

Next we demonstrate associativity:



(13.4.5)

□

Note that this monoid is not, in general, commutative.

It turns out that lots of monoids can be recast as monoids of this form. For example, consider the algebra $\mathcal{M}_n(\mathbb{C})$ viewed as a monoid object in **FHilb**. A matrix can be considered as linear map, so it can be interpreted as an element of **FHilb**$(\mathbb{C}^n, \mathbb{C}^n)$. This is isomorphic to $(\mathbb{C}^n)^* \otimes \mathbb{C}^n$. More generally, **FVect**$_{\Bbbk}(V, V) \cong V^* \otimes V$. Making this identification leads to the following lemma.

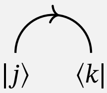
Lemma 13.4.6 — Pair of Pants on FHilb The pair of pants monoid **FHilb** is simply the matrix algebra $\mathcal{M}_n(\mathbb{C})$.

Proof. Let $A = \mathbb{C}^n$, then $A^* = (\mathbb{C}^n)^* \cong \mathbb{C}^n$. Fix some orthonormal basis $\{|i\rangle\}$ for $\mathbb{C}^n$, then $\{\langle i|\}$ is a basis for $(\mathbb{C}^n)^*$. Define a map

$$\varphi\colon (\mathbb{C}^n)^* \otimes \mathbb{C}^n \to \mathcal{M}_n(\mathbb{C}) \tag{13.4.7}$$

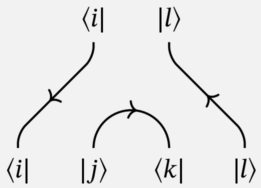
$$\langle i| \otimes |j\rangle \mapsto e_{ij} \tag{13.4.8}$$

where $e_{ij} \in \mathcal{M}_n(\mathbb{C})$ is defined as the matrix with zeros everywhere except a single entry on row i and column j which is 1.
This map is clearly a bijection, and it respects multiplication. Recall that the cap on **FHilb** is defined by



$$= \varepsilon(|j\rangle \otimes \langle k|) = \langle k|j\rangle = \delta_{kj}. \tag{13.4.9}$$

Then we have



$$= \delta_{jk}\langle i| \otimes |l\rangle = \begin{cases} \langle i| \otimes |\ell\rangle & j = k, \\ 0 & j \neq k. \end{cases} \qquad (13.4.10)$$

Applying the map φ this result becomes

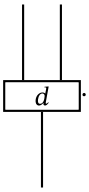
$$\begin{cases} e_{il} & j = k \\ 0 & j \neq k \end{cases} = e_{ij}e_{kl} \qquad (13.4.11)$$

where $e_{ij}e_{kl}$ is the product of these two matrices (note that i, j, k, and l are not indexing particular components of the matrix, they label the matrices). This shows that φ preserves multiplication.

The identity in the pair of pants monoid is $\sum_i \langle i| \otimes |i\rangle$. This is just the usual completeness relation for Hilbert spaces, but with the order of the bras and kets swapped. We then have $\sum_i \langle i| \otimes |i\rangle \mapsto \sum_i e_{ii}$, which is just the identity matrix. So φ also preserves identities, and so is a bijective monoid homomorphism, and hence is a monoid isomorphism. □

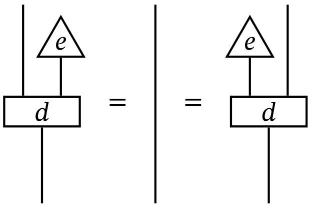
13.5 Comonoids: The Idea

Consider a process which takes an object and copies it. This should give a copying, or duplication, operation $d : A \to A \otimes A$. We can draw this as



(13.5.1)

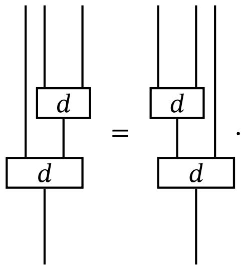
What other properties should this copying have? The first thing we might look for is that this is really copying. One way to test this is to take an object, copy it, then throw away one of the copies, the result should be the original object, and it shouldn't matter which of the two objects we delete. That is, we should have



(13.5.2)

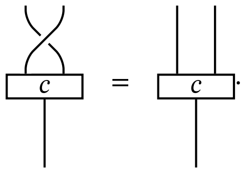
where $e : A \to I$ can be thought of as deletion.

Another property that we should consider is what happens if we make multiple copies? It shouldn't matter which of the two copies from the first copying process we use to make the copies, the result should be the same both ways. That is,



(13.5.3)

Sometimes, but not always, swapping the two objects after we copy won't matter, if this is the case then we have



(13.5.4)

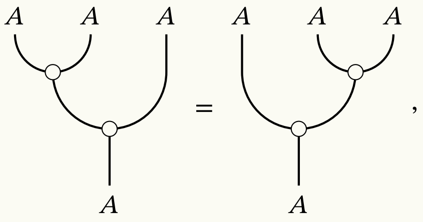
Now consider the monoid laws, and we see that

- the "true copy rule" is just the unitality law upside-down,
- the "multiple copy rule" is just the associativity law upside-down,
- the "copy then swap rule" is just the commutativity rule upside-down.

More formally, by upside-down we mean that the direction of the morphisms has been reversed, so we're in the opposite category. We can therefore consider these rules to be counitality, coassociativity, and cocommutativity respectively, and define a comonoid as a monoid in the opposite category. Expanding this definition leads to the definition of the comonoid.

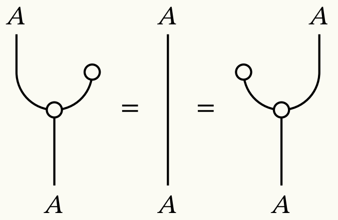
13.6 Comonoids: The Definition

Definition 13.6.1 — Comonoid A **comonoid** in a monoidal category, $\mathbf{C}$, is a triple, $(A, ⋎, ⫯)$, consisting of an object, $A \in \mathrm{Ob}(\mathbf{C})$, a morphism, $⋎ : A \to A \otimes A$, and an effect, $⫯ : A \to I$, satisfying the **coassociativity** and **counitality** equations,



(13.6.2)

and



(13.6.3)

respectively.

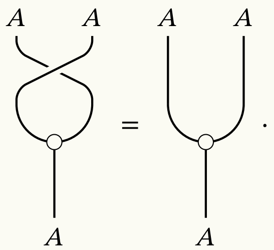
These relations can also be expressed as commutative diagrams for a comonoid (A, d, e) we require

$$\begin{tikzcd} & A \otimes A \arrow[r, "d\otimes \mathrm{id}_A"] & (A \otimes A) \otimes A \arrow[dd, "\alpha_{A,A,A}"] \\ A \arrow[ur, "d"] \arrow[dr, "d"'] & & \\ & A \otimes A \arrow[r, "\mathrm{id}_A \otimes d"'] & A \otimes (A \otimes A), \end{tikzcd} \tag{13.6.4}$$

and

$$\begin{tikzcd} A \otimes I \arrow[r, "\rho_A"] & A \arrow[d, "\mathrm{id}_A"] & I \otimes A \arrow[l, "\lambda_A"'] \\ A \otimes A \arrow[u, "\mathrm{id}_A \otimes e"] & A \arrow[l, "d"] \arrow[r, "d"'] & A \otimes A \arrow[u, "e\otimes \mathrm{id}_A"'] \end{tikzcd} \tag{13.6.5}$$

Definition 13.6.6 — Cocommutative Comonoid A **cocommutative comonoid** in a braided monoidal category, $\mathbf{C}$, is a comonoid, $(A, ⋎, ⫯)$, satisfying the **cocommutativity** equation



(13.6.7)

This can also be expressed as a commutative diagram for the comonoid (A, d, e):

$$\begin{tikzcd} A \arrow[r, "d"] \arrow[d, "d"'] & A \otimes A \\ A \otimes A. \arrow[ur, "\sigma_{A,A}"'] & \end{tikzcd} \tag{13.6.8}$$

It doesn't matter if we use the swap or the inverse swap in a cocommutative monoid, things work out the same as the next lemma shows.

Lemma 13.6.9 In a cocommutative monoid $\sigma_{A,A}^{-1} \circ d = d$,

Proof. In a cocommutative monoid by definition $\sigma_{A,A} \circ d = d$. Applying this to the equation $\sigma_{A,A}^{-1} \circ \sigma_{A,A} \circ d = \mathrm{id}_A \circ d = d$ we are left with $\sigma_{A,A}^{-1} \circ d = d$. □

13.7 Comonoids: The Examples

In **Set** any given nonempty set, A, forms exactly one comonoid, (A, c, d), with $c(a) = (a, a)$ and $d(a) = \bullet$. This is cocommutative. Since there is only one monoid we can define for any set there is no "a comonoid is a set with..." definition of a comonoid.

Take some group, G, and we can form a comonoid in **Rel** by defining a comultiplication with the relation $g \sim (h, h^{-1}g)$ for any $g, h \in G$ and the counit as the relation $1 \sim \bullet$. Here $\{\bullet\}$ is the unit object in **Rel**. We can check that this is a comonoid:

- Counitality: Take some $g \in G$, and apply c, so now we have $g \sim (h, h^{-1}g)$ for any $h \in G$. Now apply $\mathrm{id}_G \otimes d$, the result depends on $h^{-1}g$. If $h^{-1}g = 1$ then we get $(h, h^{-1}g) = (h, 1) \sim (h, \bullet)$, on the other hand if $h^{-1}g \neq 1$ then there is no relation. So, we must have $h^{-1}g = 1$, which we can rearrange to get $g = h$. Finally applying ρ_I we get $(h, \bullet) \sim h = g$. Chaining these relations we have $g \sim g$, which is to say that this chain is the identity relation, as required. Counitality on the other side can be demonstrated similarly.

- Coassociativity: Take some $g \in G$, and apply c, we get $g \sim (h, h^{-1}g)$ for any $h \in G$. Now apply $c \otimes \mathrm{id}_A$ and we get $(h, h^{-1}g) \sim ((k, k^{-1}h), h^{-1}g)$ for any $k \in G$. Reassociating gives $((k, k^{-1}h), h^{-1}g) \sim (k, (k^{-1}h, h^{-1}g))$, so $g \sim (k, (k^{-1}h, h^{-1}g))$. If instead we had applied $\mathrm{id}_A \otimes c$ we would get $(h, h^{-1}g) \sim (h, (k, k^{-1}h^{-1}g))$. Chaining these relations together we have $g \sim (h, (k, k^{-1}h^{-1}g))$. Now set $h \to k$ and $k \to k^{-1}h$, then we get $g \sim (k, (k^{-1}h, (k^{-1}h)^{-1}k^{-1}g)) = (k, (k^{-1}h, h^{-1}kk^{-1}g)) = (k, (k^{-1}h, h^{-1}g))$, which is exactly the result we got before.

- Cocommutativity: Take some $g \in G$ and apply c, giving $g \sim (h, h^{-1}g)$. If we apply $\sigma_{A,A}$ we get $(h, h^{-1}g) \sim (h^{-1}g, h)$. This comonoid will be cocommutative if and only if $(h, h^{-1}g) = (k^{-1}g, k)$. Clearly this holds if and only if $k = h^{-1}g$, that is $hk = g$, but we also require $h = k^{-1}g$, so $kh = g$. Thus, in order for this to be a cocommutative monoid we need to have $hk = g = kh$, so the group must be Abelian.

In **FHilb** choose some basis, $\{e_i\}$ for a Hilbert space H. We can define a cocommutative comonoid by defining comultiplication according to $e_i \mapsto e_i \otimes e_i$ and the counit according to $e_i \mapsto 1$, and extending these by linearity to all of H. We can check that this defines a cocommutative comonoid, and thanks to linearity we need only check the laws hold on each basis vectors:

- Coassociativity: Going one way around the diagram we have

$$e_i \overset{c}{\mapsto} e_i \otimes e_i \xmapsto{c \otimes \mathrm{id}_H} (e_i \otimes e_i) \otimes e_i \xmapsto{\alpha_{H,H,H}} e_i \otimes (e_i \otimes e_i) \tag{13.7.1}$$

  and going the other way we get

$$e_i \overset{c}{\mapsto} e_i \otimes e_i \xmapsto{\mathrm{id}_H \otimes c} e_i \otimes (e_i \otimes e_i), \tag{13.7.2}$$

  which are both the same so coassociativity is satisfied.

- Counitality: Going one way around the diagram we have

$$e_i \overset{c}{\mapsto} e_i \otimes e_i \xmapsto{\mathrm{id}_H \otimes d} e_i \otimes 1 \xmapsto{\rho_H} e_i \tag{13.7.3}$$

  so this chain is simply the identity. The other counitality law can be checked similarly.

- Cocommutativity: We have

$$e_i \overset{c}{\mapsto} e_i \otimes e_i \xmapsto{\sigma_{H,H}} e_i \otimes e_i \tag{13.7.4}$$

  so clearly this is cocommutative.

Fourteen

Uniform Copying and Deleting

14.1 Uniform Deleting

Consider a comonoid, (A, d, e). The counit, $e : A \to I$ gives us a way to "delete" copies of the system, A. Suppose we have a way of systematically deleting every object. Such a process must respect the tenor product, leading to the following definition.

Definition 14.1.1 — Uniform Deleting Consider a monoidal category, $\mathbf{C}$, with unit object I. This category has **uniform deleting** if there exists a natural transformation $e : \mathrm{id}_{\mathbf{C}} \Rightarrow \mathrm{const}_I$, with components $e_A : A \to I$, such that $e_I = \mathrm{id}_I$ and

$$\begin{tikzcd} & A \otimes B \arrow[dl, "e_A \otimes e_B"'] \arrow[dr, "e_{A\otimes B}"] & \\ I \otimes I \arrow[rr, "\lambda_I"'] & & I \end{tikzcd} \tag{14.1.2}$$

commutes.

Writing out the naturality condition for e the diagram

$$\begin{tikzcd} A \arrow[r, "e_A"] \arrow[d, "f"'] & I \arrow[d, "\mathrm{id}_I"] \\ B \arrow[r, "e_B"'] & I \end{tikzcd} \tag{14.1.3}$$

must commute.

This definition states that uniform deleting is a collection of isomorphisms deleting systems, that is sending them to the empty system, I. The commutativity definition is simply that deleting both systems independently, then combining the two empty systems, is the same as deleting the combined system.

Lemma 14.1.4 Let $\mathbf{C}$ be a monoidal category with unit object I. Then $\mathbf{C}$ has uniform deleting if and only if I is terminal.

Proof. Suppose that **C** has uniform deleting Then for each object A we have a morphism $e_A : A \to I$. We need to show that this is unique. The naturality condition for e, specialised to the case where the second object is I, becomes the commuting square

$$\begin{tikzcd} A \arrow[r, "e_A"] \arrow[d, "f"'] & I \arrow[d, "\mathrm{id}_I"] \arrow[dl, equal] \\ I \arrow[r, "e_I=\mathrm{id}_I"'] & I. \end{tikzcd} \tag{14.1.5}$$

which commutes if and only if $f = e_A$, proving that there is a *unique* morphism $A \to I$ for all objects A.
Conversely, suppose that I is terminal. Then simply define e such that $e_A : A \to I$ is the unique morphism $A \to I$. The commutativity property is satisfied by uniqueness of morphisms into I. □

Deleting causes problems in quantum mechanics. To delete a state we must lose information, which is not generally possible. A quantum system can be modelled as a monoidal category with duals. The way that this issue with deleting arises in category theory is that any quantum system with uniform deleting will be boring, in the sense it will be a preorder.

Definition 14.1.6 — Preorder A **preorder** is a category with at most one morphism $A \to B$ for each pair of objects A and B.

Note that a preorder can be defined similarly to a poset, but relaxing the requirement of antisymmetry, and then interpreted as a category in the same way we think of a preorder as a category.

Viewed as processes preorders are particularly boring categories, there is at any one point at most one process which can occur.

Theorem 14.1.7 — No Deleting. If a monoidal category with duals has uniform deleting then it is a preorder.

Proof. Let $f, g : A \to B$ be morphisms in this category. We will show that $f = g$. Recall that we can define the coname, $\llcorner f \lrcorner : A \otimes B^* \to I$. Naturality of e for this morphism gives

$$\begin{tikzcd} A \otimes B^* \arrow[r, "e_{A\otimes B^*}"] \arrow[d, "\llcorner f \lrcorner"'] & I \arrow[d, "\mathrm{id}_I"] \\ I \arrow[r, "e_I=\mathrm{id}_I"'] & I. \end{tikzcd} \tag{14.1.8}$$

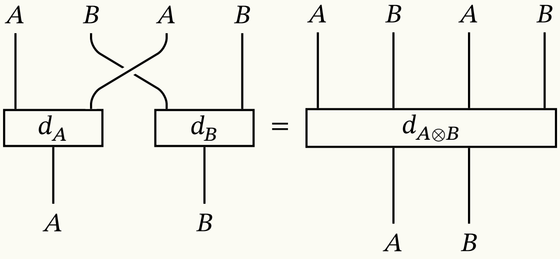
Terminality of the unit (Lemma 14.1.4) implies that $\llcorner f \lrcorner = e_{A\otimes B^*}$.
Exactly the same logic implies that $\llcorner g \lrcorner = e_{A\otimes B^*}$. Since the coname uniquely specifies the morphism we must have that $f = g$. Hence this

category is a preorder. □

14.2 Uniform Copying

Consider a comonoid, (A, d, e). The comultiplication, $d\colon A \to A \otimes A$ gives us a way to create an extra copy of our system. Suppose we have a way of systematically copying every object. Such a process must respect the tensor product, leading to the following definition. A subtlety in this is that if we copy $A \otimes B$ as a single object we get $(A \otimes B) \otimes (A \otimes B)$, but if we copy each part independently we get $(A \otimes A) \otimes (B \otimes B)$, so we must work in a braided monoidal category.

Definition 14.2.1 — Uniform Copying Consider a braided monoidal category, $\mathbf{C}$, with unit object I. This category has **uniform copying** if there exists a natural transformation $d\colon \mathrm{id}_{\mathbf{C}} \Rightarrow \Delta$, with components $d\colon A \to A \otimes A$, satisfying coassociativity and cocommutativity, with $d_I = \rho_I^{-1}$ and



(14.2.2)

that is

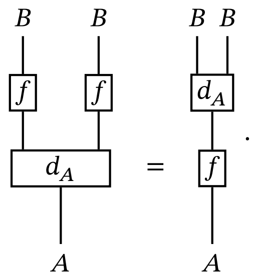
$$\begin{tikzcd}
A \otimes B \arrow[r, "d_A \otimes d_B"] \arrow[dd, "d_{A\otimes B}"'] & (A \otimes A) \otimes (B \otimes B) \arrow[r, "\alpha_{A,A,B\otimes B}"] & A \otimes (A \otimes (B \otimes B)) \arrow[d, "\mathrm{id}_A \otimes \alpha^{-1}_{A,B,B}"] \\
 & & A \otimes ((A \otimes B) \otimes B) \arrow[d, "\mathrm{id}_A \otimes (\sigma_{A,B} \otimes B)"] \\
(A \otimes B) \otimes (A \otimes B) & A \otimes (B \otimes (A \otimes B)) \arrow[l, "\alpha_{A,B,A\otimes B}"] & A \otimes ((B \otimes A) \otimes B) \arrow[l, "\mathrm{id}_A \otimes \alpha_{B,A,B}"]
\end{tikzcd}$$

commutes.

The naturality condition for d, commutativity of

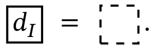
$$\begin{tikzcd}
A \arrow[r, "d_A"] \arrow[d, "f"'] & A \otimes A \arrow[d, "f \otimes f"] \\
B \arrow[r, "d_B"'] & B \otimes B,
\end{tikzcd} \tag{14.2.3}$$

can be expressed as



(14.2.4)

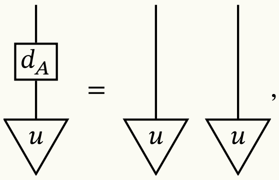
That $d_I = \mathrm{id}_I$ can be expressed as



(14.2.5)

Example 14.2.6 **Set** has uniform copying defined by the maps $a \mapsto (a, a)$. Writing $1 = \{\bullet\}$ for the singleton set, which is the unit of **Set**, and recalling that $\rho_A(a, \bullet) = a$ for all sets A, we have $d_1(\bullet) = (\bullet, \bullet) = \rho_1^{-1}(\bullet)$.

Definition 14.2.7 — Copyable In a braided monoidal category a state, $u : I \to A$, is **copyable** with respect to a map $d_A : A \to A \otimes A$ when



(14.2.8)

that is, $d_A \circ u = (u \otimes u) \circ \rho_I$.

Claim 14.2.9 In a braided monoidal category with uniform copying all states are copyable.

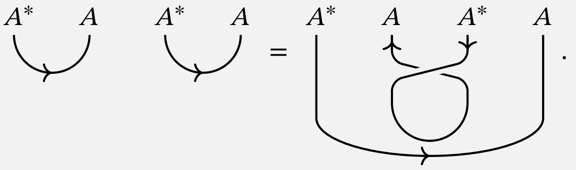
Proof. Naturality of the uniform copying map specialised to morphisms $u : I \to A$ gives the commuting square

$$\begin{tikzcd} I \arrow[r, "d_I = \rho_I^{-1}"] \arrow[d, "u"'] & I \otimes I \arrow[d, "u \otimes u"] \\ A \arrow[r, "d_A"'] & A \otimes A. \end{tikzcd} \tag{14.2.10}$$

That is $d_A \circ u = (u \otimes u) \circ \rho_I$, which is exactly the requirement for u to be copyable. □

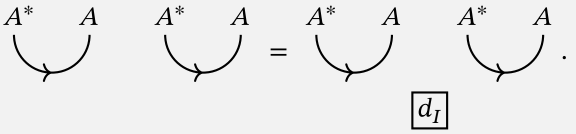
We saw that uniform deleting rendered a quantum theory trivial, in the sense that we lost all but one process per system and were left with a preorder. We will similarly show that uniform copying renders a quantum theory trivial, although this time in the sense that all maps simply become scalar multiples of the identity, so we can't do anything interesting, especially since we usually normalise states and ignore overall phases, so these scalings don't change the physics. Before we show this we'll need a couple of lemmas, which already start to show that if we have duals and uniform copying then things are intuitively not quite right.

Lemma 14.2.11 In a braided monoidal category with duals and uniform copying



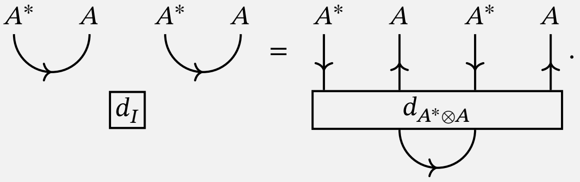
(14.2.12)

Proof. Recalling that $d_I = \rho_I^{-1}$ we can insert the scalar d_I into our diagram without changing anything, since it's really the empty diagram. This gives



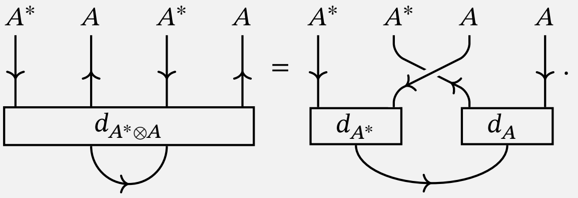
(14.2.13)

Naturality tells us that we can copy and then perform a process on each copy or perform the process and then copy. In this case the process is creating the system $A^* \otimes A$. As written above we are copying the empty system then applying the process to create $A^* \otimes A$. Instead, we can create $A^* \otimes A$ from the empty system and then copy it. In terms of diagrams this gives



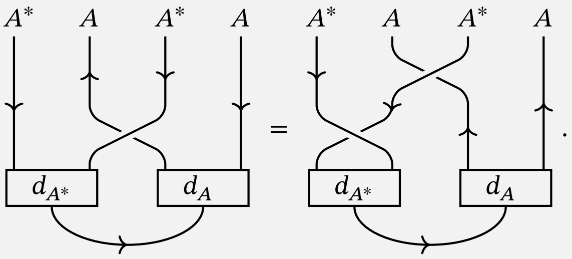
(14.2.14)

We can now apply the definition of uniform copying (Equation (14.2.2)) to write



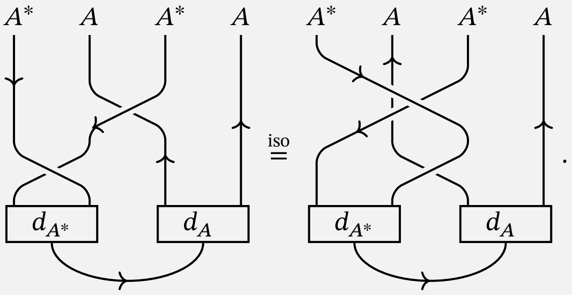
(14.2.15)

Applying cocommutativity of d_{A^*}, using the inverse swap, we have



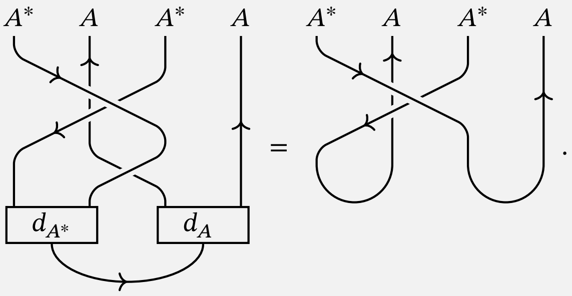
(14.2.16)

Redrawing this diagram we have



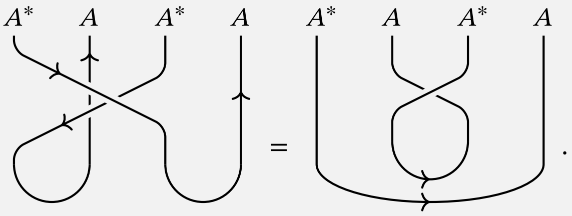
(14.2.17)

Now notice that the lower half of this diagram is exactly the right hand side of Equation (14.2.15), so we can replace it with the double cup making up the left hand side of Equation (14.2.13). This gives



(14.2.18)

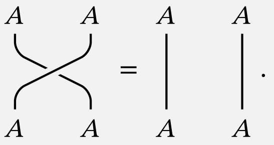
Finally, we can rewrite this with an isotopy of diagrams giving



(14.2.19)

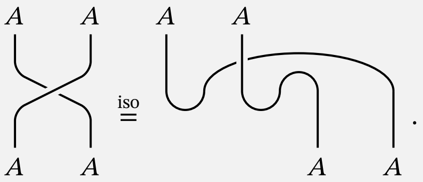
This is intuitively not quite right because on the left hand side we have the second system, A, and the third system, A^*, unconnected, but on the right they become connected. This makes it hard to create independent states.

Lemma 14.2.20 In a braided monoidal category with duals and uniform copying the swap is trivial, that is



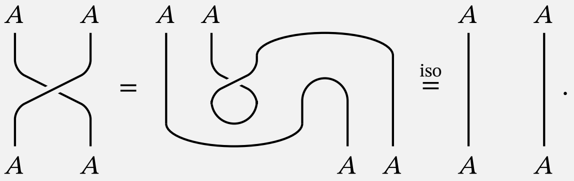
(14.2.21)

Proof. Start with the left hand side above and deform it according to the snake equation:



(14.2.22)

Now apply Lemma 14.2.11 to get

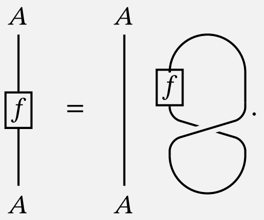


(14.2.23)

So exchanging two identical systems is the same as doing nothing. From a quantum mechanical point of view this means that exchanging two electrons is the same as doing nothing, but we know that exchanging two electrons should result in an overall minus sign, since electrons are fermions and so their wave functions are antisymmetric. Thus we cannot have duals and uniform copying if we want fermions, and we most certainly want fermions as they make up all matter.

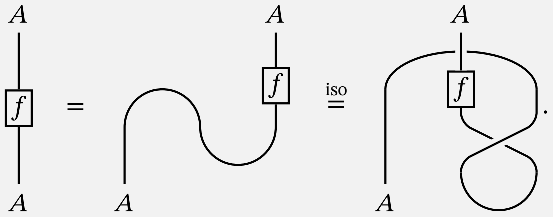
The failure of these structures to model reality results in the **no cloning theorem**, which says that a system modelling quantum reality cannot have uniform copying. Rephrased in the following theorem it says that if we do have uniform copying then the only endomorphisms are multiples of the identity. From a quantum mechanical perspective this means that all observables are simply defined by a single eigenvalue and take the same value on all states, so there are no interesting observables.

Theorem 14.2.24 — No Cloning Theorem. In a braided monoidal category with duals and uniform copying every endomorphism is a multiple of the identity. In particular, if $f : A \to A$ is an endomorphism in this category then $f = \mathrm{tr}(f) \cdot \mathrm{id}_A$, that is



(14.2.25)

Proof. Start by applying the snake equation to f, then sliding the lower wire over f:



(14.2.26)

Now apply Lemma 14.2.20 to the swap and we are left with $\mathrm{tr}(f) \cdot \mathrm{id}_A$. □

Fifteen

Frobenius Structures

15.1 Frobenius Structures: The Idea

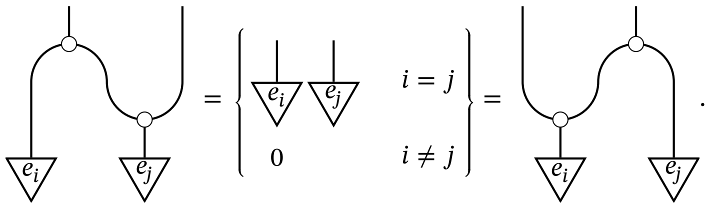
While quantum mechanics doesn't allow for copying of states there is nothing preventing us copying classically. In this way classical mechanics can be phrased as quantum mechanics plus the ability to copy and delete. We take **FHilb** as our motivation. Fix some Hilbert space with orthogonal basis $\{e_i\}$. Then copying is the map defined by

$$d = \curlyvee : e_i \mapsto e_i \otimes e_i \tag{15.1.1}$$

extended by linearity to all inputs. The adjoint to this copying map is the **comparison map**

$$d^\dagger = \curlywedge : e_i \otimes e_j \mapsto \begin{cases} e_i & i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{15.1.2}$$

This map defines a monoid on this Hilbert space. These two structures can be combined to give

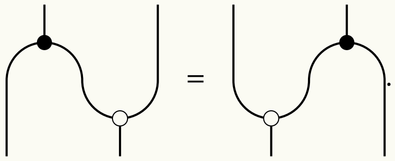


(15.1.3)

The shape of this equation motivates our next definition.

15.2 Frobenius Structures: The Definition

Definition 15.2.1 — Frobenius Structure In a monoidal category a **Frobenius structure** is formed from a monoid, $(A, \curlywedge, \bullet)$, and comonoid, $(A, \curlyvee, \circ)$, on some object A, such that the product and coproduct satisfy the **Frobenius law**



(15.2.2)

If ⋏ = ⋏̊ then we call this a **dagger Frobenius structure**.

15.3 Frobenius Structures: The Examples

15.3.1 Group Algebra

We've seen that in **FHilb** we get a Frobenius structure by taking the comparison monoid and copying comonoid. There is another way to form a Frobenius structures in **FHilb**. Start with a finite group, G, and define the **group algebra**, $\mathbb{C}[G]$, to be the Hilbert space spanned by the elements of the group. A vector in this space is of the form $\sum_i z_i g_i$ where $g_i \in G$ and $z_i \in \mathbb{C}$. The sum of two such vectors is $\sum_i z_i g_i + \sum_i w_i g_i = \sum_i (z_i + w_i) g_i$, scalar multiplication is given by $w \sum_i z_i g_i = \sum_i (w z_i) g_i$, the product of two vectors is $\left(\sum_i z_i g_i\right)\left(\sum_j w_j g_j\right) = \sum_{i,j}(z_i w_j)(g_i g_j)$, and the scalar product is $\langle \sum_i z_i g_i | \sum_j w_j g_j \rangle = \sum_i z_i^* w_i$. These sums are purely formal, we don't try to evaluate them.

The group algebra comes equipped with a natural Frobenius structure. Define a monoid on $\mathbb{C}[G]$ in the obvious way using the fact that every group is a monoid and defining

$$m = ⋏ \colon\ g \otimes h \mapsto gh, \qquad ⫯ \colon\ z \mapsto z 1_G \tag{15.3.1}$$

for $g, h \in G$ and $1_G \in G$ the group identity, and extending this by linearity to all of $\mathbb{C}[G]$. The adjoint of this monoid is the comonoid defined by

$$d = ⋎ \colon\ g \mapsto \sum_{h \in G} gh^{-1} \otimes h, \qquad ⫰ \colon \begin{cases} 1_G \mapsto 1 & g = 1_G, \\ g \mapsto 0 & g \neq 1_G. \end{cases} \tag{15.3.2}$$

again, extending this by linearity to all of $\mathbb{C}[G]$.

The left hand side of the Frobenius law takes in two vectors in $\mathbb{C}[G]$, but we can simplify the analysis by acting just on the basis vectors, which are elements of the group, G, embedded in the group algebra. So consider the input $g \otimes h$. Algebraically the left hand side of the Frobenius law is

$$f = (m \otimes \mathrm{id}_A) \circ \alpha^{-1}_{A,A,A} \circ (\mathrm{id}_A \otimes d) \tag{15.3.3}$$

where m and d are the monoid multiplication and comonoid copy maps. Applying this to $g \otimes h$ we get

$$f(g \otimes h) = [(m \otimes \mathrm{id}_A) \circ \alpha^{-1}_{A,A,A}]\left(g \otimes \left[\sum_{k \in G} hk^{-1} \otimes k\right]\right) \tag{15.3.4}$$

$$= \sum_{k \in G} [(m \otimes \mathrm{id}_A) \circ \alpha^{-1}_{A,A,A}](g \otimes (hk^{-1} \otimes k)) \tag{15.3.5}$$

$$= \sum_{k\in G} [(m \otimes \mathrm{id}_A)]((g \otimes hk^{-1}) \otimes k) \tag{15.3.6}$$

$$= \sum_{k\in G} ghk^{-1} \otimes k \tag{15.3.7}$$

$$= \sum_{k\in G} gk^{-1} \otimes kh. \tag{15.3.8}$$

In the last step here we've used Cayley's theorem to recognise that if k takes on all values in G then so does kh as this is simply the group acting on itself by permutation. Thus, these two sums are equal but the order of terms in the sum may change. The right hand side of the Frobenius law is

$$\tilde{f} = (\mathrm{id}_A \otimes m) \circ \alpha_{A,A,A} \circ (d \otimes \mathrm{id}_A). \tag{15.3.9}$$

Applying this to $g \otimes h$ we get

$$\tilde{f}(g \otimes h) = [(\mathrm{id}_A \otimes m) \circ \alpha_{A,A,A}]\left(\left[\sum_{k\in G} gk^{-1} \otimes k\right] \otimes h\right) \tag{15.3.10}$$

$$= \sum_{k\in G} [(\mathrm{id}_A \otimes m) \circ \alpha_{A,A,A}]((gk^{-1} \otimes k) \otimes h) \tag{15.3.11}$$

$$= \sum_{k\in G} [(\mathrm{id}_A \otimes m)](gk^{-1} \otimes (k \otimes h)) \tag{15.3.12}$$

$$= \sum_{k\in G} [gk^{-1} \otimes kh]. \tag{15.3.13}$$

These are equal, so the Frobenius law holds.

15.3.2 Groupoids

Recall that a **groupoid** is a category in which all morphisms are invertible. In order to work with groupoids it is often easier to work with a slightly different, but completely equivalent, definition of a category.

Definition 15.3.14 — Category A **category**, $\mathbf{C}$, consists of the following data:

- a class, $\mathrm{Ob}(\mathbf{C})$, of objects;
- a class, $\hom(\mathbf{C})$, of morphisms;
- two maps $s, t \colon \hom(\mathbf{C}) \to \mathrm{Ob}(\mathbf{C})$, called the **source** and **target** maps;
- for all pairs of morphisms $f, g \in \hom(\mathbf{C})$ with $t(f) = s(g)$ a morphism $g \circ f \in \hom(\mathbf{C})$;
- for every object $A \in \mathrm{Ob}(\mathbf{C})$ a morphism $\mathrm{id}_A \in \hom(\mathbf{C})$.

This data is constrained by the following:

- source and target respect composition: $s(g \circ f) = s(f)$ and $t(g \circ f) = t(g)$;
- source and target respect identities: $s(\mathrm{id}_A) = t(\mathrm{id}_A) = A$;

- associativity: $(h \circ g) \circ f = h \circ (g \circ f)$ whenever these maps are defined;
- unit laws: $\mathrm{id}_{t(f)} \circ f = f = f \circ \mathrm{id}_{s(f)}$.

Basically, take the definition we usually use of a category, do away with individual hom-sets and put all morphisms into one bag, hom(**C**), and then define functions s and t which give the domain and codomain. To map back to the original definition of a category simply define $\mathrm{hom}(A, B) = \{f \in \mathrm{hom}(\mathbf{C}) \mid s(f) = A \text{ and } t(f) = B\}$. With this new definition of a category a groupoid, **G**, consists of objects, Ob(**G**), and isomorphisms hom(**G**).

In **Rel** we can construct a Frobenius structure on a groupoid, **G**. First, let $G = \mathrm{hom}(\mathbf{G})$ and we can then construct relations on this set (assuming that **G** is a small category). Then define multiplication:

$$⅄ :\ G \times G \to G \tag{15.3.15}$$

$$(g, h) \sim g \circ h \text{ if } s(g) = t(h). \tag{15.3.16}$$

That is, identify the pair of isomorphisms (h, g) with their composite $h \circ g$ if this is defined, and with nothing if this composite doesn't exist. The unit is then defined as

$$⫰ :\ 1 \to G \tag{15.3.17}$$

$$\bullet \sim \mathrm{id}_A \quad \text{for all } A \in \mathrm{Ob}(\mathbf{G}) \tag{15.3.18}$$

where $1 = \{\bullet\}$ is the singleton set. That is, we identify the element of the singleton set with identities and nothing else.

The adjoint of these maps are defined by the converse relations, in particular,

$$Y :\ g \sim (h, h^{-1} \circ g) \text{ for all } h \in G \text{ with } t(g) = s(h^{-1}) = t(h) \tag{15.3.19}$$

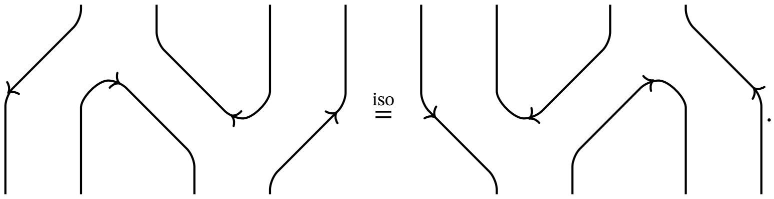
and

$$⫯ :\ \mathrm{id}_A \sim \bullet \text{ for all } A \in \mathrm{Ob}(\mathbf{G}). \tag{15.3.20}$$

This defines a Frobenius structure on G.

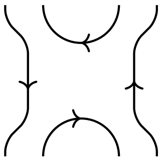
15.3.3 Pair of Pants

In a dagger monoidal category with duality $A \dashv A^*$ the pair of pants monoid on $A^* \otimes A$ forms a dagger Frobenius structure. This can be seen directly as the left and right hand sides of the Frobenius law are isotopic for the pair of pants monoid:



. (15.3.21)

Notice that both sides here are isotopic to

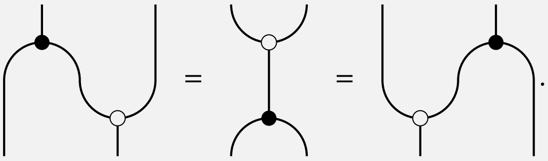


(15.3.22)

which motivates the following lemma.

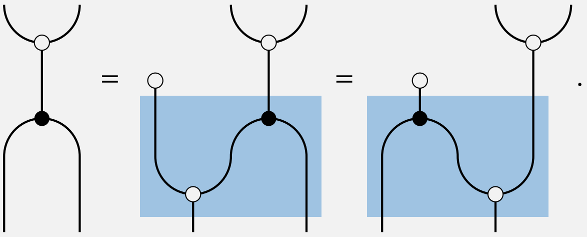
15.4 Extended Frobenius Law

Lemma 15.4.1 — Extended Frobenius Law Any Frobenius structure satisfies



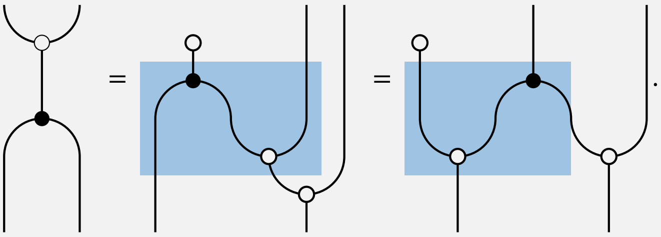
(15.4.2)

Proof. Start with the middle term above and apply the counitality law to the bottom left leg. Then identify a subdiagram we can apply the Frobenius law to. This gives



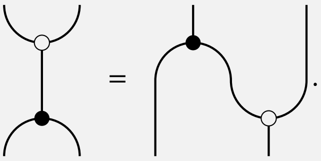
(15.4.3)

We can now apply coassociativity to the right hand side of this diagram and then again identify a subdiagram to which we can apply the Frobenius law:



(15.4.4)

Finally, apply counitality to the left hand side of this diagram and we are left with



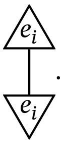
(15.4.5)

Proving the second equality is done in exactly the same way but starting with the other leg. □

15.5 Special Frobenius Structure

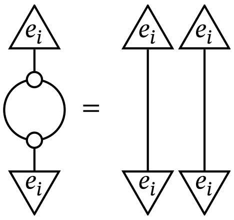
Consider the Frobenius structure given by an orthogonal basis, $\{e_i\}$, on some Hilbert space. The norm squared of a basis vector is given by

$$\|e_i\|^2 =$$



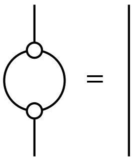
. (15.5.1)

Since the comonoid in this case copies the basis we have



$$= \|e_i\|^4. \quad (15.5.2)$$

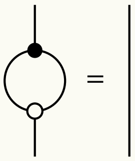
From this we see that



(15.5.3)

if and only if $\|e_i\| = 1$. This motivates the following definition.

Definition 15.5.4 — Special Frobenius Structure A Frobenius structure is special if



, (15.5.5)

that is $m = d^{-1}$ where m is monoid multiplication and d is comonoid comultiplication.

As stated before the orthogonal basis Frobenius structure in **FHilb** is special exactly when the basis is orthonormal.

The group algebra Frobenius structure in **FHilb** is only special for the trivial group, in which case $\mathbb{C}[\{1\}] \cong \mathbb{C}$. This whole example is trivial though, since $m(1_G \otimes 1_G) = 1_G 1_G = 1_G$ and $d(1_G) = 1_G \otimes 1_G$ are the only operations, neither of which allows any interesting computations.

The groupoid Frobenius structure in **Rel** is always special.

15.6 Classical Structures

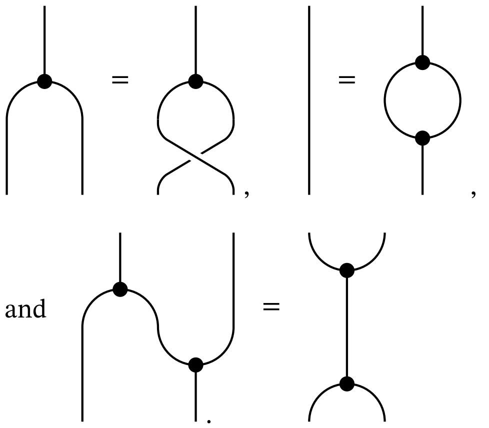
A classical structure, which we shall define shortly, captures all of the structure of a classical computation, which is roughly speaking a quantum computation with the ability to copy and delete.

> **Definition 15.6.1 — Classical Structure** A **classical structure** in a braided monoidal dagger category is a special commutative dagger Frobenius structure.

The (second) "dagger" part of this definition means that the comonoid and monoid forming the Frobenius structure are adjoints, so we typically only specify one of them. The commutative part applies to the monoid, it is a commutative monoid, which then means that the comonoid is cocommutative.

A classical structure in **FHilb** corresponds to an orthonormal basis. A classical structure in **Rel** corresponds to an Abelian group.

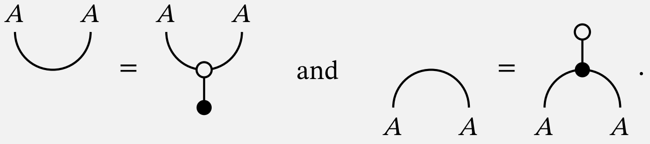
The definition of a classical turns out to have several redundancies. It can be shown that $(A,$ ⋏$,$ ⫰$)$ is a classical structure if all of the following hold:



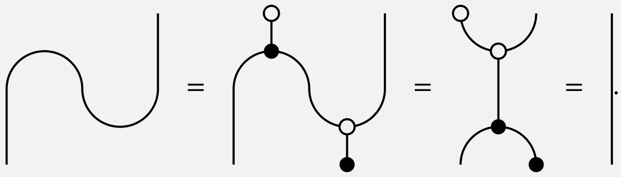
(15.6.2)

That is, a classical structure is a commutative map ⋏$: A \otimes A \to A$ in a dagger category such that the speciality condition holds and half of the extended Frobenius law holds. The other requirements from these. For example, combining speciality and the Frobenius law gives us associativity

15.7 Self Duality

> **Lemma 15.7.1 — Frobenius Structures are Self Dual** If $(A,$ ⋎$,$ ⫯$,$ ⋏$,$ ⫰$)$ is a Frobenius structure in some monoidal category then A is self dual with this duality expressed through
>
> 
>
>
> (15.7.2)

Proof. We need to demonstrate that the snake equations hold with this definition. This follows immediately by the extended Frobenius law and the definition of the (co)unit:



(15.7.3)

The other snake equation is proven similarly. □

To state the next theorem we'll need a few definitions first.

Definition 15.7.4 — Zero Object An object, 0, is a **zero object** if it is both initial and terminal. A **zero morphism** $0_{A,B} : A \to B$ is the unique morphism $A \to 0 \to B$ factoring through the zero object.

Note that uniqueness of $0_{A,B}$ follows since by the definition of an initial object the morphism $A \to 0$ is unique and by the definition of a terminal object the morphism $0 \to B$ is unique.

Definition 15.7.5 — Dagger Kernel In a dagger category with a zero object given a map $f : A \to B$ a **dagger kernel** of this map is a pair (K, k) consisting of an object K and an isometry[a] $k : K \to A$ such that when $f \circ k = 0_{K,B}$ every morphism satisfying $x : X \to A$ factors through k so that the following commutes:

$$\begin{tikzcd}
K \arrow[r, "k"] & A \arrow[r, shift left, "f"] \arrow[r, shift right, "0_{A,B}"'] & B \\
X. \arrow[u, dashed] \arrow[ur, "x"'] & &
\end{tikzcd} \tag{15.7.6}$$

[a] recall that k is an isometry if $k^\dagger \circ k = \mathrm{id}_K$

The morphism $m : X \to K$ posited by this commuting diagram is unique, and must by equal to $k^\dagger \circ x$ since k is an isometry so

$$m = \mathrm{id}_K \circ m = k^\dagger \circ k \circ m = k^\dagger \circ x \tag{15.7.7}$$

The intuition here is to generalise the notion of the kernel of a linear map as the subspace which is mapped to the zero vector, since the zero-dimensional subspace $\{0\}$ is exactly the zero object in the category of vector spaces. The extra map k that we have to introduce here is simply the inclusion map from this zero-dimensional space into the space on which f is defined.

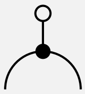
Definition 15.7.8 — Nondegenerate Fix some object A in a dagger monoidal category. A state $a : I \to A$ is **nondegenerate** if $a \neq 0_{I,A}$ implies $a^\dagger \circ a \neq 0_{I,I}$. A nondegenerate form is an effect $\alpha : A \to I$ such that $\alpha^\dagger$ is a nondegenerate state.

In the language of geometry a **form** (aka a covariant vector in physics) is something which takes in a vector (aka a contravariant vector in physics) and returns a scalar, the interpretation here being that a nondegenerate form $\alpha : A \to I$ takes in a state $a : I \to A$ and returns the scalar $\alpha \circ a : I \to I$.

It can be shown that all morphisms in a category with zero objects and arbitrary dagger kernels are nondegenerate or identically zero.

These definitions are quite a lot, so just think in terms of Hilbert spaces where a zero morphism is a map sending everything to the zero vector and a nondegenerate form is a linear map $\alpha : V \to \mathbb{C}$ such that $\alpha(v) = 0 \implies v = 0$. In more familiar notation, $\langle \alpha | v \rangle = 0$ if and only if $v = 0$ (assuming $\alpha \neq 0$). Don't worry too much about these concepts, they're just technicalities required for the following theorem.

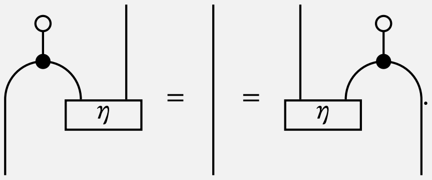
Theorem 15.7.9. Given a monoid $(A,$, $)$ we can define a Frobenius structure with comonoid $(A,$, $)$ (to be specified in the proof) if and only if A admits a nondegenerate form $: A \to I$ such that



(15.7.10)

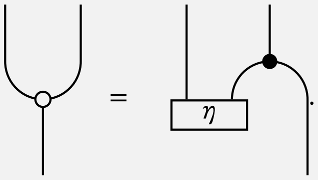
is the cap of the self duality $A \dashv A$.

Proof. Lemma 15.7.1 proves that all Frobenius structures give rise to a self duality with the required cap. So we need only show that the existence of such a self duality with unspecified cup $\eta : I \to A \otimes A$ ensures this monoid-comonoid pair is a Frobenius structure. The snake equations for this self duality take the form



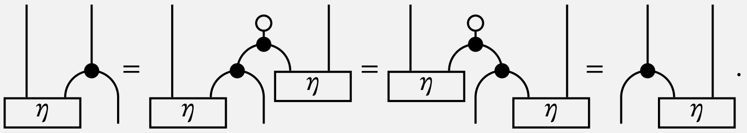
(15.7.11)

We can then define comultiplication through



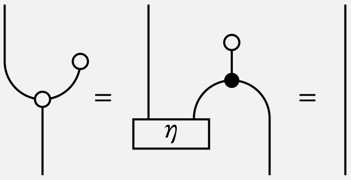
(15.7.12)

Note that we could have defined comultiplication with η on the other side since applying the snake equation to the line leaving the monoid product dot and then applying associativity of the monoid product we can exchange one of these diagrams for the other:



(15.7.13)

Counitality is then simply the snake equation:



(15.7.14)

Coassociativity follows similarly using the symmetry of the definition of comultiplication in terms of the cup. The Frobenius law holds by applying associativity of the multiplication after writing the comultiplication in terms of the cup. □

Nondegneracy was implicitly used at several points in this proof. Without this assumption we would have been doing the equivalent of dividing by zero.

Sixteen

Normal Forms

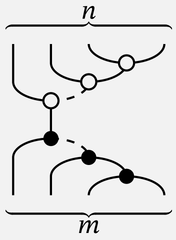
There are two ways to think about diagrams in the graphical notation. The first is as a representation of a morphism. The second is to treat the diagram as an object in its own right, in which case diagrams are just graphs, although not all graphs are valid diagrams. The morphisms are nodes and the objects are edges. This way of thinking about diagrams allows us to do things like consider subdiagrams, and replace them with other equivalent diagrams, which we've been doing already without thinking too much about the details.

If two diagrams are different when viewed as graphs they may still be equal as morphisms. We've seen already that really we want to consider homotopy classes of diagrams, which is just a fancy way of saying we can move things around and the morphism is the same so long as the connectivity stays the same. There are other operations we can perform on diagrams-as-graphs without changing the morphisms they represent. We can define an equivalence relation on graphs where two graphs are equivalent if they represent the same morphism. The particularly nice thing about this, which we shall prove shortly, is it is possible to define a normal form for a graph such that any equivalent graph can be manipulated into this normal form without changing the morphism it represents. This allows us to work with a fixed form for the representatives of the equivalence classes of diagrams representing morphisms.

Similar to the coherence theorem, which says that given the data of a monoidal category there is a unique morphism between any two objects we will show that given the ability to copy, discard, fuse, and create data there is a unique way to do so if we start and finish with a fixed amount of data. This, stated formally, is the spider theorem.

16.1 Spider Theorem

Theorem 16.1.1 — Spider Theorem. Let $(A,$ ⋏$,$ ⫰$,$ ⋎$,$ ⫯$)$ be a special Frobenius structure. Any connected[a] morphism $A^{\otimes m} \to A^{\otimes n}$ built from finitely many pieces ⋏, ⫰, ⋎, ⫯, and id_A, using only $\circ$ and $\otimes$ is equal to the morphism represented by the diagram



(16.1.2)

[a] By connected we mean that any diagram representing it is a connected graph, if this isn't the case we can treat the disconnected parts separately, so this isn't an issue.

Proof. The proof proceeds by (strong) induction on the number of dots, that is the total number of multiplications, units, comultiplications, and counits. The base case of $N = 1$ dots is trivial, since the diagram must simply be one of , , , or , all of which are already in this normal form. Now suppose that all diagrams with at most N dots can be brought into normal form. Consider a diagram with $N + 1$ dots. We can use naturality to write this diagram in a form where one of the dots is physically higher than all other dots. We then proceed casewise.

- First, suppose that the topmost dot is the counit, . Then we can take the subdiagram consisting of the N lower dots and put this into normal form. There are two subcases:
  - If there are any other white dots in the diagram after placing it in normal form then they will be a comultiplications, and the counit will be directly connected to one of these comultiplications, in which case it can be eliminated, leaving an N dot diagram which can be placed into normal form.
  - If there are no other white dots after placing the lower diagram into normal form then this diagram is already in normal form, specifically, the counit corresponds to the white dot on top of the vertical line in the middle of the normal form.
- If the topmost dot is instead comultiplication, , then place the N lower dots into normal form and use coassociativity repeatedly to move the topmost dot all the way to the right. Since the diagram is finite in extent this process terminates. This process will also result in the final diagram being in normal form.
- If the topmost dot is the unit, , then the connectivity requirement tells us that this is the only dot present in the diagram, and hence the diagram is already in normal form.
- If the topmost dot is the multiplication, , then put the lower N dots into normal form. There are then two subcases, imagine removing

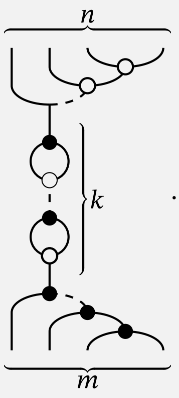
the topmost dot and the wires leading into it. There are two possibilities, first, we could end up with two disconnected parts to the diagram, or we could have just one connected part if there were other connections between the two halves of the diagram

- Suppose we are left one connected diagram after removing the topmost dot. Then we can repeatedly us coassociativity and speciality to move the multiplication dot down until it is below all comultiplications and counits. Then repeatedly use associativity to more it all the way to the right and the diagram will be in normal form.
- If instead we are left with two disconnected diagrams then we can put these individually in normal form. Then applying the extended Frobenius law will allow us to push the multiplication dot downwards. Doing so repeatedly we move the multiplication to be below all comultiplications and counits, and then we can use associativity to pus it all the way to the right, placing the diagram in normal form.

□

There are many variants of the spider theorem which we can obtain by allowing slight modifications on the initial set up.

Theorem 16.1.3 — Non-special Spider Theorem. Let $(A,$ ⅄, ⫰, Y, ⫯$)$ be a not-necessarly special Frobenius structure. Any connected morphism $A^{\otimes m} \to A^{\otimes n}$ built out of finitely many pieces ⅄, ⫰, Y, ⫯, and id_A using only $\circ$ and $\otimes$ is equal to the morphism represented by the diagram



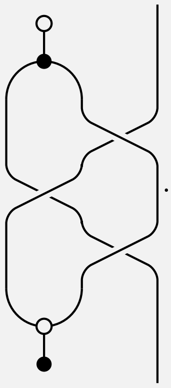
. (16.1.4)

Theorem 16.1.5 — Symmetric Spider Theorem. Let $(A,$ ⅄, ⫰, Y, ⫯$)$ be a symmetric Frobenius structure. Any connected morphism $A^{\otimes m} \to A^{\otimes n}$ built out of finitely many pieces ⅄, ⫰, Y, ⫯, ×, id_A using only $\circ$ and $\otimes$ is equal to the morphism represented by the diagram in Equation (16.1.4).

However, a general Frobenius structure does not allow for a normal form.

Theorem 16.1.6 — No Braided Spider Theorem. Let $(A,$ ⅄, ⫰, Y, ⫯$)$ be a commutative Frobenius structure. There is no normal form for a connected morphism $A^{\otimes m} \to A^{\otimes n}$ built out of finitely many pieces ⅄, ⫰, Y, ⫯, ⤫, id_A using only $\circ$ and $\otimes$.

Proof. Consider the following diagram representing a morphism $A \to A$:



(16.1.7)

This is a knot (once we connect up the free ends). The Frobenius axioms, without the addition of speciality or symmetry, induce homotopy equivalences on diagrams. By definition a knot is *not* homotopy equivalent to the circle, but clearly, for example, the identity $\mathrm{id}_A : A \to A$, after connecting the ends, is homotopy equivalent to the circle. Hence, these two diagrams do not correspond to the same morphism. □

Appendices

A

Maths Definitions

A.1 Algebraic Structures

A.1.1 One Binary Operation

Definition A.1.1 — Magma A **magma**, $(M, \cdot)$, is a set, M, equipped with a binary operation $\cdot : M \times M \to M$.
A **magma homomorphism**, $(M, \cdot_M) \to (N, \cdot_N)$, is a function $f : M \to N$ such that $f(m \cdot_M m') = f(m) \cdot_N f(m')$ for all $m, m' \in M$.

Definition A.1.2 — Semigroup A **semigroup**, $(S, \cdot)$, is an associative magma, meaning that

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c \tag{A.1.3}$$

for all $a, b, c \in S$.
A **semigroup homomorphism** is a magma homomorphism between semigroups.

Definition A.1.4 — Monoid A **monoid**, $(M, \cdot, e)$, is a semigroup with an identity element, $e \in M$, such that $e \cdot m = m \cdot e = m$ for all $m \in M$.
A **monoid homomorphism**, $(M, \cdot_M, e_M) \to (N, \cdot_N, e_N)$, is a function $f : M \to N$ such that $f(m \cdot_M m') = f(m) \cdot_N f(m')$ for all $m, m' \in M$ and $f(e_M) = e_N$.

Definition A.1.5 — Group A **group**, $(G, \cdot, e, -^{-1})$, is a monoid with inverses, meaning for each $g \in G$ there exists $g^{-1} \in G$ such that $g \cdot g^{-1} = g^{-1} \cdot g = e$.
A **group homomorphism**, $(G, \cdot_G, e_G, -^{-1}) \to (H, \cdot_H, e_H, -^{-1})$, is a function $f : G \to H$ such that $f(g \cdot_G g') = f(g) \cdot_H f(g')$ for all $g, g' \in G$.

Note that the definition of a group homomorphism implies that $f(e_G) = e_H$ and $f(g^{-1}) = f(g)^{-1}$ for all $g \in G$, unlike in the monoid case.

For magmas, semigroups, and monoids if $x \cdot y = y \cdot x$ we say it is a **commutative magma/semigroup/monoid**. For groups we call a commutative group an **Abelian group**.

Definition A.1.6 — Group Action Let G be a group and S a set. Then a **left group action** is a map $\varphi\colon G \times S \to S$ such that

- the identity acts as the identity: $\varphi(e, s) = s$;
- group multiplication is preserved: $\varphi(g, \varphi(h, s)) = \varphi(gh, s)$.

Often we write $g \,.\, s$ or gs in place of $\varphi(g, s)$, in which case the axioms are $es = s$ and $g(hs) = (gh)s$, which look very similar to the group identity and associativity laws. Note that a right action is a map $\psi\colon S \times G \to S$ such that $\psi(s, e) = s$ and $\psi(\psi(s, g), h) = \psi(s, gh)$, or $se = s$ and $(sg)h = s(gh)$.

Definition A.1.7 — Group Representation A **representation** of a group G is a pair (ρ, V) consisting of a vector space V and a group action $\rho\colon G \times V \to V$. Alternatively, $\rho\colon V \to \mathrm{GL}(V)$ is a homomorphism and G acts on V through $g \,.\, v = \rho(g)v$.

Note that

$$\begin{aligned} \mathrm{GL}(V) &\coloneqq \{\text{invertible linear maps on } V\} \\ &\cong \{n \times n \text{ matrices with entries in } \Bbbk\} \eqqcolon \mathrm{GL}(n, \Bbbk) \end{aligned} \tag{A.1.8}$$

where $n = \dim V$ and V is a vector space over $\Bbbk$.

A.1.2 Two Binary Operations

Definition A.1.9 — Ring A **ring**, $(R, +, -, \cdot, 1, 0)$ is a set, R, equipped with two binary operations, $+\colon R \times R \to R$ and $\cdot\colon R \times R \to R$ such that

- $(R, +, 0, -)$ is an Abelian group.
- $(R, \cdot, 1)$ is a monoid.
- Multiplication distributes over addition:

  $$a \cdot (b + c) = a \cdot b + a \cdot c, \qquad \text{and} \qquad (a + b) \cdot c = a \cdot c + b \cdot c \tag{A.1.10}$$

  where we take multiplication to have higher operator precedence than addition.

A **ring homomorphism**, $(R, +_R, -_R, \cdot_R, 1_R, 0_R) \to (S, +_S, -_S, \cdot_S, 1_S, 0_S)$ is a function $f\colon R \to S$ such that $f(a +_R b) = f(a) +_S f(b)$, $f(a \cdot_R b) = f(a) \cdot_S f(b)$, and $f(1_R) = 1_S$.
If $x \cdot y = y \cdot x$ for all $x, y \in R$ we call R a **commutative ring**.

Note that some sources only require that $(R, \cdot)$ is a semigroup, in which case what we define here is called a **ring with identity** or **ring with unity**. Sometimes a ring without identity is called a **rng**, with i taken out as there is no **i**dentity.

It is also common to define **rig** or **semiring|seerig** by only requiring that $(R, +)$ is a commutative monoid, meaning we drop inverses, which for an additive group would be **n**egatives, $-g$ being the additive inverse of g. A **rg** is then defined by dropping the requirements for multiplicative identities and additive inverses.

Definition A.1.11 — Integral Domain Let R be a ring. A **zero-divisor** is some element $a \in R$ such that $ab = 0$ and/or $ba = 0$ for some $b \in R$ with $b \neq 0$. A ring with *no* zero-divisors is called a **domain**. A commutative domain is an **integral domain**.

Definition A.1.12 — Field Let R be a ring. A **unit** is some element $a \in R$ with $a \neq 0$ such that there exists $a^{-1} \in R$ with $a \cdot a^{-1} = a^{-1} \cdot a = 1$. A division ring is a ring in which all nonzero elements are units. That is, $(R^{\times}, \cdot, e, -^{-1})$ is a group, where $R^{\times} = R \setminus \{0\}$.
A commutative division ring is a **field**.

A.1.3 Modules

Definition A.1.13 — Module Let R be a ring. A **left R-module**, is an Abelian group, $(M, +)$, equipped with an operation $\cdot : R \times M \to M$ such that for all $r, s \in R$ and $u, v \in M$ we have

- left distributivity over module addition: $r \cdot (u + v) = r \cdot u + r \cdot v$;
- right distributivity over ring addition: $(r + s) \cdot u = r \cdot u + s \cdot u$;
- compatibility of multiplications: $(rs) \cdot u = r \cdot (s \cdot u)$;
- compatibility identities: $1 \cdot u = u$.

A **right R-module** is similarly defined except for the order of the arguments, $\cdot : M \times R \to M$.

Note that if R is commutative then left and right modules are equivalent.

Definition A.1.14 — Vector Space Let $\Bbbk$ be a field. A **vector space** is a $\Bbbk$-module.

Definition A.1.15 — Bimodule Let R and S be rings. An **R-S-bimodule** is an Abelian group, $(M, +)$, such that

- M is a left R-module and a right S-module;
- for all $r \in R$, $s \in S$, and $m \in M$

$$(r \cdot m) \cdot s = r \cdot (m \cdot s). \tag{A.1.16}$$

Definition A.1.17 If M and N are left R-modules then an R-**linear map**, or R-**module homomorphism**, is a map $f: M \to N$ such that

$$f(r \cdot m + r' \cdot m') = r \cdot f(m) + r' \cdot f(m') \tag{A.1.18}$$

for all $r, r' \in R$ and $m, m' \in M$. Right R-module homomorphisms are defined analogously.
An R-S-**bimodule homomorphism** is a map f which is both a left R-module homomorphism and a right S-module homomorphism.

Definition A.1.19 — Algebra Let $\Bbbk$ be a field and A a $\Bbbk$-vector space. Then A is an **algebra** if it is equipped with a binary operation $\cdot : A \times A \to A$ satisfying the following for all $u, v, w \in A$ and $a, b \in \Bbbk$:

- right distributivity over vector addition: $(u + v) \cdot w = u \cdot w + v \cdot w$;
- left distributivity over vector addition: $u \cdot (v + w) = u \cdot v + u \cdot w$;
- compatibility of multiplication: $(au) \cdot (bv) = (ab)(u \cdot v)$.

If in addition

$$u \cdot (v \cdot w) = (u \cdot v) \cdot w \tag{A.1.20}$$

for all $u, v, w \in A$ then we say that A is an **associative algebra**.

A.2 Orderings

Definition A.2.1 — Partial Order A (non-strict) **partially ordered set**, or **poset**, $(P, \leq)$, is a set, P, equipped with a (non-strict) **partial order**, which is a reflexive, antisymmetric, transitive relation. That is,

- **reflexivity**: for all $a \in P$ we have $a \leq a$;
- **antisymmetric**: for $a, b \in P$ if $a \leq b$ and $b \leq a$ then $a = b$;
- **transitivity**: for $a, b, c \in P$ if $a \leq b$ and $b \leq c$ then $a \leq c$.

Definition A.2.2 — Total Order A (non-strict) **totally ordered set**, $(X, \leq)$, is a poset with the added condition of totality, that is for all $a, b \in X$ we have either $a \leq b$ or $b \leq a$ (or both, in which case $a = b$ by antisymmetry).

Definition A.2.3 — Monotone Function Let $(P, \leq_P)$ and $(Q, \leq_Q)$ be partially ordered sets. A **monotone function**, or **order-preserving function** $(P, \leq_P) \to (Q, \leq_Q)$, is a function $f: P \to Q$ such that if $a \leq_P b$ for

$a, b \in P$ then $f(a) \leq_Q f(b)$.

A.3 Topological Spaces

Definition A.3.1 — Topological Space A **topological space**, $(X, \mathcal{T})$ is a set, X, equipped with a **topology**, $\mathcal{T}$, which is a subset of $\mathcal{P}(X)$ such that

- $\varnothing \in \mathcal{T}$;
- $X \in \mathcal{T}$;
- an arbitrary (potentially infinite) union of elements of $\mathcal{T}$ is again an element of $\mathcal{T}$;
- a finite intersection of elements of $\mathcal{T}$ is again an element of $\mathcal{T}$.

Elements of $\mathcal{T}$ are called open sets.

Definition A.3.2 — Continuous Function Let $(X, \mathcal{T}_X)$ and $(Y, \mathcal{T}_Y)$ be topological spaces. A **continuous function**, $(X, \mathcal{T}_X) \to (Y, \mathcal{T}_Y)$, is a function $f : X \to Y$ such that for all $U \in \mathcal{T}_Y$ we have $f^{-1}(U) \in \mathcal{T}_X$ where

$$f^{-1}(U) \coloneqq \{x \in X \mid \exists y \in U \text{ such that } y = f(x)\} \subseteq X. \tag{A.3.3}$$

Definition A.3.4 — Homotopy Let X and Y be topological spaces. A **homotopy** between continuous functions $f : X \to Y$ and $g : X \to Y$ is a continuous function $H : X \times [0,1] \to Y$ such that $H(x, 0) = f(x)$ and $H(x, 1) = g(x)$.
If $x \mapsto H(x, t)$ is an embedding for all $t \in [0,1]$ then we call this an **isotopy**.

We can imagine a homotopy as smoothly morphing f into g, with the parameter in $[0,1]$ determining how far we are along in this process of morphing. An isotopy is then the special case where we can also draw $H(x, t)$ at any point $t \in [0,1]$.

B

Adjoint Functors

Adjoint functors are, strictly, beyond the scope of this course, although we have already seen several examples. Broadly adjoint functors are to categories as dual objects are to objects in a monoidal category.

B.1 Adjoint Functors: The Definition

There are multiple equivalent definitions of functors, we'll look at two here. The first is easier to use but the second makes the relation to dual objects more explicit.

Before we can give the first definition we need to introduce a special collection of functors.

> **Definition B.1.1 — Hom-Functor** Given a category, C, then for objects A and B there are functors $\mathsf{C} \to \mathsf{Set}$, namely the covariant and contravariant **hom-functors**
>
> $$\mathsf{C}(A,-) \colon \mathsf{C} \to \mathsf{Set}, \qquad \text{and} \qquad \mathsf{C}(-,B) \colon \mathsf{C} \to \mathsf{Set}. \tag{B.1.2}$$
>
> The first sends the object $B \in \mathrm{Ob}(\mathsf{C})$ to the set of morphisms $\mathsf{C}(A,B)$, and the second sends the object $A \in \mathrm{Ob}(\mathsf{C})$ to the set of morphisms $\mathsf{C}(A,B)$. The covariant hom-functor acts on maps $f \colon X \to Y$ by sending them to maps
>
> $$\mathsf{C}(A,f) \colon \mathsf{C}(A,X) \to \mathsf{C}(A,Y) \tag{B.1.3}$$
> $$g \mapsto f \circ g. \tag{B.1.4}$$
>
> The contravariant hom-functor acts on maps $h \colon X \to Y$ by sending them to maps
>
> $$\mathsf{C}(h,B) \colon \mathsf{C}(X,B) \to \mathsf{C}(Y,B) \tag{B.1.5}$$
> $$gd \mapsto g \circ h. \tag{B.1.6}$$

It is simple enough to check that these define functors, we'll do it here for the covariant hom-functor. First, $\mathsf{C}(A, \mathrm{id}_X)$, is the map which sends g to $g \circ \mathrm{id}_X = g$, so it is the identity. Second, $\mathsf{C}(A, f \circ f')$ is the map which sends g to $f \circ f' \circ g$, which is the map that $\mathsf{C}(A, f)$ sends $f' \circ g$ to.

Definition B.1.7 — Adjoint Functors Let $\mathbf{C}$ and $\mathbf{D}$ be categories. Fix two functors $F\colon \mathbf{C} \to \mathbf{D}$ and $G\colon \mathbf{D} \to \mathbf{C}$. An **adjunction** between F and G is a natural isomorphism

$$\Phi\colon \mathbf{D}(F-,-) \Rightarrow \mathbf{C}(-,G-). \tag{B.1.8}$$

That is, a collection of bijections

$$\Phi_{A,B}\colon \mathbf{D}(FA,B) \to \mathbf{C}(A,GC) \tag{B.1.9}$$

for $A \in \mathrm{Ob}(\mathbf{C})$ and $B \in \mathrm{Ob}(\mathbf{D})$ such that the following diagram commutes

$$\begin{tikzcd}
\mathbf{D}(FA,B) \arrow[r, "\Phi_{A,B}"] \arrow[d, "\mathbf{C}(Fg,f)"'] & \mathbf{C}(A,GB) \arrow[d, "\mathbf{D}(g,Gf)"] \\
\mathbf{D}(FA',B') \arrow[r, "\Phi_{A',B'}"'] & \mathbf{C}(A',GB').
\end{tikzcd} \tag{B.1.10}$$

for all objects $A, A' \in \mathrm{Ob}(\mathbf{C})$ and $B, B' \in \mathrm{Ob}(\mathbf{D})$ and morphisms $f\colon A \to A'$ and $g\colon B \to B'$.
We say that F is **left adjoint** to G and G is **right adjoint** to F. We write $F \vdash G$

This definition is a bit complex, but essentially it comes down to the existence of a bijection

$$\mathbf{D}(FA,B) \cong \mathbf{C}(A,GB) \tag{B.1.11}$$

for all objects in such a way that this bijection defines a natural isomorphism. We'll explore this with lots of examples.

The idea is that adjoint functors translate between categories without changing the structure of the categories too much. In particular an equivalence of categories can be defined as an adjunction with a few more requirements. Consider categories as countries, with the objects being citizens and morphisms conversations in the countries language. Then functors are translations from one language to another preserving meaning. Then F and G are adjoints if it doesn't matter if Alice travel's to Bob's country and speaks Bob's language or Bob travel's to Alice's country and speaks Alice's language.

B.2 Adjoint Functors: The Examples

B.2.1 Free-Forgetful Adjunction

Recall that a forgetful functor interprets objects of one category in another category by "forgetting" some of the details, such as the forgetful functor **Grp** → **Set**, which interprets each group as its underlying set, or the forgetful functor **CRing** → **Ring** which forgets that a given ring is commutative. Often it is possible to go the other way, to add in some structure to turn a set into a group, or a ring into a commutative ring. There are usually many ways to do this, but often there is only one minimal way to do it, to add in only the strictly necessary structure in order for a set to become a group or a ring to become a commutative ring. Minimal here refers to the added structure, not the size of the free objects we produce, which tend to be much larger than the object we started with. Such a minimal addition of structure is called a free construction.

Example B.2.1 — Free Vector Space Fix some field $\Bbbk$. Write $U\colon$ $\mathbf{Vect}_\Bbbk \to \mathbf{Set}$ for the forgetful functor which sends a vector space to its underlying set of vectors, simply forgetting the existence of the scalars and forgetting anything about adding vectors. This functor sends a linear map to the underlying function, which is all it really can do since we've forgotten about scalars and vector addition.

Write $F\colon$ $\mathbf{Set} \to \mathbf{Vect}_\Bbbk$ for the functor sending a set, A, to the vector space of all formal linear combinations of elements of A, that is

$$F(A) = \{\varphi\colon A \to \mathbb{C} \mid \varphi(a) \neq 0 \text{ for only finitely many } a \in A\}. \quad (B.2.2)$$

Taking A as the basis for $F(A)$ we can interpret $\varphi(a)$ as the a component of some vector φ, in more standard notation we might write φ_a. That is, a vector of $F(A)$ is of the form

$$\sum_{a\in A} \varphi(a)a = \sum_{a\in A} \varphi_a a. \quad (B.2.3)$$

The second part, that $\varphi(a)$ is nonzero for a finitely many $a \in A$, is a technical requirement for the vector space to be well-defined.

Addition in this vector space $F(A)$ is given by $(\varphi + \psi)(a) = \varphi(a) + \psi(a)$, and scalar multiplication by $(z\varphi)(a) = z[\varphi(a)]$. The zero vector is the zero map, $a \mapsto 0$ for all $a \in A$. For a finite set, A, we have that $F(A) \cong \Bbbk^{|A|}$.

We also need to define how F acts on functions. Given a function (of sets) $f\colon A \to B$ we need a linear map $F(f)\colon F(A)\colon F(B)$. We can construct such a map: $\varphi \mapsto \varphi \circ f$.

We will show that F is left adjoint to U. To do this we need to construct a one-to-one correspondence

$$(\text{Linear functions } g\colon F(A) \to V) \leftrightarrow (\text{Functions } f\colon A \to U(V)). \quad (B.2.4)$$

Given some function $f\colon A \to U(V)$ define g_f by $\varphi \mapsto \sum_{a\in A} \varphi(a)f(a)$. Then the map $f \mapsto g_f$ evaluates a formal linear combination in $F(A)$ as a (not-necessarily-formal) linear combination in V. Conversely, given a linear map $g\colon F(A) \to V$ define f_g by sending $a \in A$ to $g(\delta_{a-})$ where $\delta_{a-} \in F(A)$ is the characteristic function $c \mapsto \delta_{ac}$, that is $\delta_{a-}(b) = 1$ if $a = b$ and $\delta_{a-}(b) = 0$ otherwise.

Then we have $f_{g_f} = f$ and $g_{f_g} = g$, so these two constructions are inverses. To see this notice that f_{g_f} takes in an element $a \in A$, sends it to $g_f(\delta_{a-})$, which according to the definition of g_f is the map sending δ_{a-} to $\sum_{a'\in A} \delta_{aa'} f(a') = f(a)$.

Similarly, g_{f_g}, which is linear, takes a basis vector, $a \in A$, and sends it to $f_g(\delta_{a-})$, which is by definition $\sum_{a'\in A} \delta_{aa'} g(a') = g(a)$, which we can then extend by linearity to all of $F(A)$.

It is also possible to show that this defines a natural transformation, but we won't do so here.

Example B.2.5 — Free Monoid Write $U\colon$ **Mon** $\to$ **Set** for the forgetful functor sending a monoid to its underlying set.
Write $F\colon$ **Set** $\to$ **Mon** for the functor which takes a set, A, to the free monoid on A. The free monoid on A being the monoid made form all formal combinations of “letters” in A with string concatenation as the monoid product and the empty string as the identity. For example, the free monoid on one object is simply (isomorphic to) the natural numbers, $F(\{\bullet\}) \cong \mathbb{N}$. The free monoid on two generators has set of elements

$$F(\{0,1\}) = \{\varnothing, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 110, 101, 111, 0000, \dots\}. \tag{B.2.6}$$

Given a function (of sets) $f\colon A \to B$ the free monoid functor sends this to the function $F(f)\colon F(A) \to F(B)$ given by acting on each element of one of these “words” at a time, for example, $F(f)(001) = f(0)f(0)f(1)$, and sending the empty string to the empty string. This is clearly a monoid homomorphism.
To show that $F \vdash U$ we need to establish a one-to-one correspondence

$$(\text{monoid homomorphisms } g\colon F(A) \to M) \leftrightarrow (\text{functions } f\colon A \to U(M)). \tag{B.2.7}$$

Given f define g_f to send the word $a_1 \cdots a_n$ to the product $f(a_1) \cdots f(a_2)$ in M, and the empty string to 1, the identity of M. Conversely, given g define f_g by applying g to the one-letter word $a \in A$. Then we have $f_{g_f} = f$ and $g_{f_g} = g$, so these two constructions are inverses.
To see this simply unwrap the definitions and we see that $f_{g_f}(a) = g_f(a) = f(a)$ and $g_{f_g}(a_1 \cdots a_n) = f_g(a_1) \cdots f_g(a_n) = g(a_1) \cdots g(a_n) = g(a_1 \cdots a_n)$, the last equality following as g is a monoid homomorphism.
This construction is also natural.

Example B.2.8 — Free Category Write **Graph** for the category of directed graphs. For a graph G if $V(G)$ is the set of vertices and $E(G)$ the set of edges then morphisms $f\colon G \to G'$ in **Graph** are pairs of functions $f_V\colon V(G) \to V(G')$ and $f_E\colon E(G) \to E(G')$ such that the source and target of an edge is preserved. That is, given an edge $e = (v_1, v_2)$, recall an edge of a directed graph is just an ordered pair of vertices, we have an edge $f_E(e) = (f_V(v_1), f_V(v_2))$.
Write **Cat** for the category of all (small) categories with functors as morphisms. Then we can write down a directed graph $U(\mathbf{C})$ whose vertex set is the objects of $\mathbf{C}$, $V(U(\mathbf{C})) = \mathrm{Ob}(\mathbf{C})$, with an edge between two vertices if there is a morphism between these vertices. Then U is a forgetful functor $U\colon$ **Cat** $\to$ **Graph**.

> Write $F\colon \mathsf{Graph} \to \mathsf{Cat}$ for the functor taking a graph, G to the category whose objects are elements of $V(G)$ and morphisms $v \to w$ are paths between the two vertices, $v \xrightarrow{e_1} \dots \xrightarrow{e_n} w$. Composition is concatenation of paths, and the identity is the path of length zero from a vertex to itself.
> On a graph morphisms, $f\colon G \to G'$, the functor F acts to give a functor $F(f)\colon F(G) \to F(G')$ sending objects $v \in \mathrm{Ob}(F(G))$ to $f(v)$, and morphisms $v \xrightarrow{e_1} \dots \xrightarrow{e_n} w$ to the morphism $f(v) \xrightarrow{f(e_1)} \dots \xrightarrow{f(e_n)} f(w)$.
> We will show that $F \vdash U$. To do so we need to establish a one-to-one correspondence
>
> $$(\text{functors } G\colon F(H) \to \mathsf{C} \to) \leftrightarrow (\text{graph morphisms } f\colon H \to U(\mathsf{C})) \tag{B.2.9}$$
>
> for a graph H.
> Given $f\colon H \to U(\mathsf{C})$ define G_f as follows: send an object v to $f(v)$ and a morphism $v \xrightarrow{e_1} \dots \xrightarrow{e_n} w$ to the composite $f(e_n) \circ \cdots \circ f(e_1)$. Conversely, given G define f_G as follows: send a vertex v to the object $G(v)$, and send an edge e to the morphism $G(e)$. Then $f_{G_f} = f$ and $G_{f_G} = G$, so these constructions are inverses. This is also natural.

In general, given a category, C, of algebraic structures and structure preserving morphisms we can define a forgetful functor $U\colon \mathsf{C} \to \mathsf{Set}$. If that functor has a left adjoint, $F\colon \mathsf{Set} \to \mathsf{C}$, then this has a natural interpretation of constructing a free object from a set.

This construction is useful because free objects are, as the name suggests, the most free objects of a given type of structure. We can then define more interesting objects as quotients of these free ones, basically imposing some other property by quotienting out by the equivalence relation it defines. For example, we can construct the free algebra on a vector space, V, and the result is the tensor algebra,

$$T(V) = \bigoplus_{k=0}^{\infty} V^{\otimes k}. \tag{B.2.10}$$

Then we can define other algebras of interest as quotients of this:

- $T(V)/(x \otimes y - y \otimes x)$ is the symmetric algebra, the state space for a boson.
- $T(V)/(x \otimes y + y \otimes x)$ is the exterior algebra, the state space for a fermion.
- If $Q\colon V \to \Bbbk$ is a quadratic form, that is a map such that $Q(zv) = z^2Q(v)$ for all $v \in V$ and $z \in \Bbbk$, then we can define the Clifford algebra associated with this quadratic form to be $T(V)/(v \otimes v - Q(v)1)$ where $(v \otimes v - Q(v)1)$ is the ideal generated by all elements of the form $v \otimes v - Q(v)1$ for all elements $v \in V$, where 1 is the unit of the tensor algebra.
- If $\mathfrak{g}$ is a Lie algebra then we can define a bracket $\mathfrak{g} \otimes \mathfrak{g} \to \mathfrak{g}$ as $[a,b] = a \otimes b - b \otimes a$. We can then generalise this recursively to $\mathfrak{g}^{\otimes n}$ through $[a \otimes b, c] = a \otimes [b,c] + [a,c] \otimes b$ and $[a, b \otimes c] = [a,b] \otimes c + b \otimes [a,c]$. Then we can define the universal enveloping algebra to be the algebra $U(\mathfrak{g}) = T(\mathfrak{g})/(a \otimes b - b \otimes a - [a,b])$.

Bibliography

[1] C. Heunen and J. Vicary. *Categories for Quantum Theory*. Oxford: Oxford University Press, 2019.

[2] T. Leinster. *Basic Category Theory*. Cambridge: Cambridge University Press, 2014. DOI: 10.48550/ARXIV.1612.09375.

[3] B. Milewski. *Category Theory for Programmers*. 2019. URL: https://github.com/hmemcpy/milewski-ctfp-pdf.

Index