

Willoughby Seago

Theoretical Physics

Categories and Quantum Informatics

January 17, 2023

COURSE NOTES

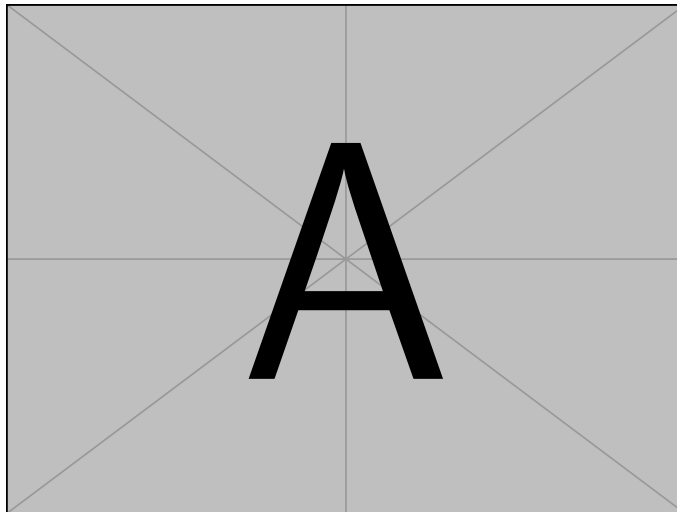
Categories and Quantum Informatics

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 18:18 on February 10, 2023. For notes on other topics see <https://github.com/WilloughbySeago/Uni-Notes>.



Chapters

	Page
Chapters	ii
Contents	iii
List of Figures	vi
1 Introduction	1
I Introduction	2
2 ZX Calculus	3
3 Semantics	12
II Categories	14
4 Categories	15
5 Monoidal Categories	42
6 Scalars	60
7 Braided and Symmetric Monoidal Categories	70
8 Dagger Categories	79
Appendices	87
A Maths Definitions	88
Bibliography	93
Index	94

Contents

	Page
Chapters	ii
Contents	iii
List of Figures	vi
1 Introduction	1
I Introduction	2
2 ZX Calculus	3
2.1 Types of Diagrams	3
2.2 Simplifying Diagrams	4
2.3 Interpretation	7
3 Semantics	12
3.1 Types of Semantics	12
3.2 Motivation	13
3.3 Mathematical Objects	13
II Categories	14
4 Categories	15
4.1 Motivation	15
4.2 Categories: The Idea	16
4.3 Categories: The Definition	17
4.3.1 Technicality	18
4.4 Categories: The Examples	18
4.4.1 Set	18
4.4.2 Rel	20
4.4.3 Vect_k, FVect_k, Hilb, and FHilb	23
4.4.4 More Examples	28
4.5 Diagrams	29
4.6 Terminology	30

4.7	Graphical Notation	31
4.8	Functors	33
4.9	Natural Transformations	37
4.10	Products	39
5	Monoidal Categories	42
5.1	Tensor Products	42
5.2	Monoidal Categories: The Idea	43
5.3	Monoidal Categories: The Need for Specificity	44
5.4	Monoidal Categories: The Definition	44
5.5	Monoidal Categories: The Examples	48
5.5.1	Set	48
5.5.2	Rel	50
5.5.3	Hilb	50
5.5.4	More Examples	51
5.6	Interchange Law	52
5.7	Graphical Notation	53
5.7.1	Correctness Theorem	55
5.8	States	56
5.8.1	States	56
5.8.2	Effects	57
5.8.3	Joint States	57
6	Scalars	60
6.1	Motivation	60
6.2	Scalars in Hilb	60
6.3	Scalars	61
6.4	Scalar Multiplication	64
6.5	Examples	69
7	Braided and Symmetric Monoidal Categories	70
7.1	Braided Monoidal Categories: The Idea	70
7.2	Braided Monoidal Category: The Definition	71
7.3	Braided Monoidal Category: The Examples	72
7.3.1	Set	72
7.3.2	Rel	72
7.3.3	Hilb	73
7.3.4	More Examples	73
7.4	Graphical Notation	73
7.4.1	Correctness Theorem	74
7.5	Symmetric Monoidal Category	76
8	Dagger Categories	79
8.1	Dagger Categories: The Idea	79
8.2	Dagger Categories: The Definition	80
8.3	Dagger Categories: The Examples	80
8.3.1	Set	80
8.3.2	Rel	81
8.3.3	Hilb	82
8.3.4	More Examples	83

8.4	Terminology	83
8.5	Graphical Notation	84
8.6	Monoidal Dagger Categories	84
Appendices		87
A	Maths Definitions	88
A.1	Algebraic Structures	88
A.1.1	One Binary Operation	88
A.1.2	Two Binary Operations	89
A.1.3	Modules	90
A.2	Orderings	91
A.3	Topological Spaces	92
Bibliography		93
Index		94

List of Figures

	Page
2.1 A diagram in ZX calculus taking two qubits to two qubits.	5
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.	59

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

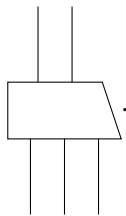
ZX Calculus

In this chapter we will introduce **ZX calculus**. This is a diagrammatic 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 than 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

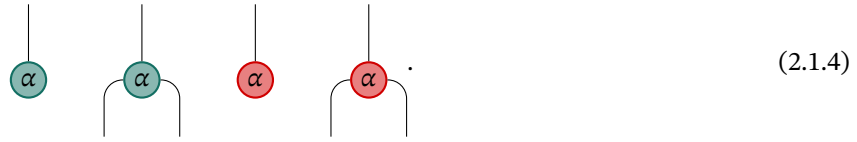
following is equivalent to the previous diagram



We are also free to change the orientation of the box, so long as 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.



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



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:

$$\text{red circle with } 0 = \text{red circle}, \quad \text{and} \quad \text{green circle with } 0 = \text{green circle} \quad (2.1.5)$$

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 they 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

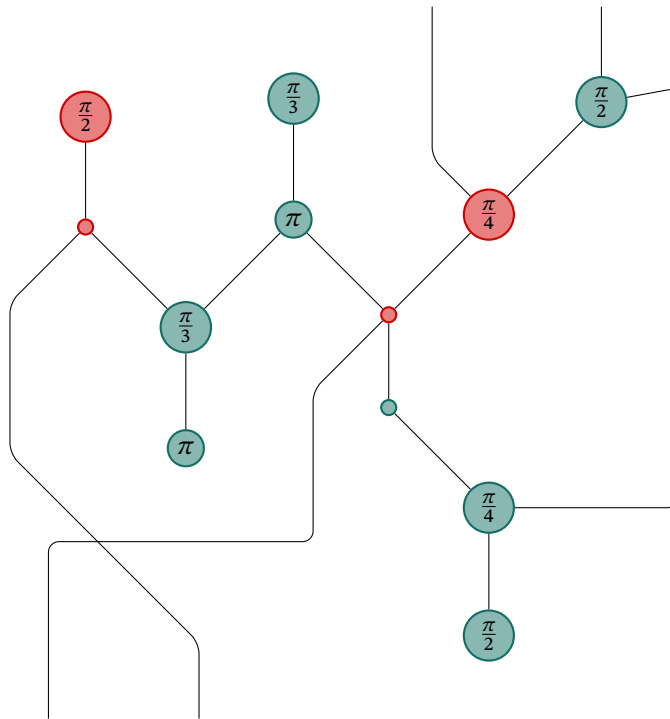
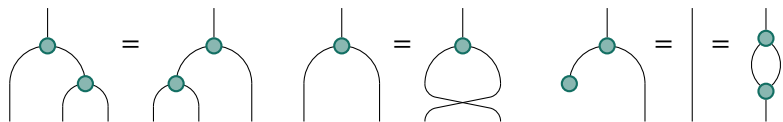


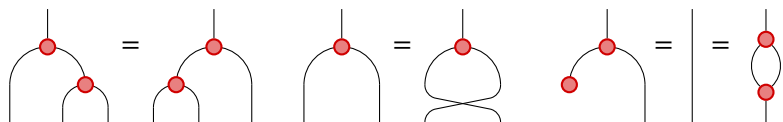
Figure 2.1: A diagram in ZX calculus taking two qubits to two qubits.

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.

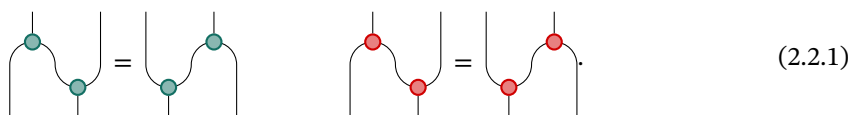
The second type of rule corresponds to specific properties of the basic building blocks. There are quite a few of these, and for now we'll just list them without much explanation. First we have the **monoid rules** which are



and the same for the other colour:



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



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

some circumstances:

$$\begin{array}{c} \text{green dot} \\ \alpha \quad \beta \end{array} = \text{green circle } \alpha + \beta \quad \begin{array}{c} \text{red dot} \\ \alpha \quad \beta \end{array} = \text{red circle } \alpha + \beta, \quad (2.2.2)$$

where addition is taken modulo 2π .

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

$$\begin{array}{c} \text{green circle } \alpha \\ \text{red circle } \alpha \end{array} := \begin{array}{c} \text{green dot} \\ \text{red dot} \end{array} \quad (2.2.3)$$

We also define the **Hadamard**:

$$H := \begin{array}{c} \text{green circle } \frac{\pi}{2} \\ \text{red circle } \frac{\pi}{2} \end{array}. \quad (2.2.4)$$

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:

$$\begin{array}{c} \text{green dot} \\ \text{red dot} \end{array} = \begin{array}{c} \text{green dot} \\ \text{red dot} \end{array} \quad (2.2.5)$$

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**:

$$\begin{array}{c} \text{red dot} \\ \text{green dot} \end{array} = \begin{array}{c} H \\ \text{green dot} \\ H \end{array} \quad (2.2.6)$$

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

$$\begin{array}{c} \text{green dot} \\ \text{red dots} \end{array} = \begin{array}{c} \text{red dots} \end{array}. \quad (2.2.7)$$

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**:



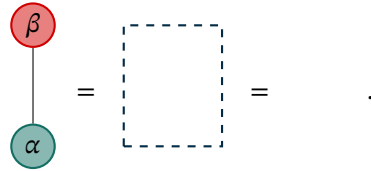
$$(2.2.8)$$

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 $e^{i\alpha}$, and the rules about X spiders tell us how to multiply matrices which are diagonal in the Hadamard transformed basis formed from

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

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

$$\llbracket - \rrbracket : (n\text{-to-}m \text{ qubit ZX diagram}) \rightarrow (2^n \times 2^m \text{ complex matrices}). \quad (2.3.2)$$

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

$$\text{---} \mapsto \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \quad (2.3.3)$$

Unsurprisingly doing nothing to a qubit gives the identity.

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

$$\text{---} \times \text{---} \mapsto \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (2.3.4)$$

In terms of matrices this acts as follows:

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

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:

$$\text{---} \mapsto \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \quad (2.3.6)$$

Similarly we can destroy qubits:

$$\text{---} \mapsto (1 \ 0 \ 0 \ 1). \quad (2.3.7)$$

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

$$\boxed{H} \mapsto \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \quad (2.3.8)$$

We can create a single qubit:

$$\text{---} \circlearrowleft \alpha \mapsto \begin{pmatrix} 1 \\ e^{i\alpha} \end{pmatrix}, \quad \text{and} \quad \text{---} \circlearrowright \alpha \mapsto \begin{pmatrix} 1 + e^{i\alpha} \\ 1 - e^{i\alpha} \end{pmatrix} \quad (2.3.9)$$

The three-arity spiders give the following matrices

$$\text{---} \circlearrowleft \alpha \mapsto \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & e^{i\alpha} \end{pmatrix}, \quad (2.3.10)$$

$$\text{---} \circlearrowright \alpha \mapsto \begin{pmatrix} 1 + e^{i\alpha} & 1 - e^{i\alpha} & 1 - e^{i\alpha} & 1 + e^{i\alpha} \\ 1 - e^{i\alpha} & 1 + e^{i\alpha} & 1 + e^{i\alpha} & 1 - e^{i\alpha} \end{pmatrix} \quad (2.3.11)$$

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



$$\begin{array}{c} \text{---} \\ | \\ \boxed{f} \\ | \\ \text{---} \end{array} \quad \begin{array}{c} \text{---} \\ | \\ \boxed{g} \\ | \\ \text{---} \end{array} \mapsto f \otimes g \quad (2.3.12)$$

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:



$$\begin{array}{c} \text{---} \\ | \\ \boxed{g} \\ | \\ \boxed{f} \\ | \\ \text{---} \end{array} \mapsto g \circ f \quad (2.3.13)$$

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 $\llbracket D_1 \rrbracket = \llbracket D_2 \rrbracket$.

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 $\| \llbracket D \rrbracket - f \| < \varepsilon$ 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 **$\pi/4$ -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 $\llbracket D_1 \rrbracket = \llbracket D_2 \rrbracket$ 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,

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

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:

$$P = \text{if } 1 = 1 \text{ then } F \text{ else } G, \quad (3.1.1)$$

$$Q = \text{if } 1 = 0 \text{ then } F \text{ else } F. \quad (3.1.2)$$

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

$$\llbracket - \rrbracket : \text{Programs} \rightarrow \text{Mathematical Objects}. \quad (3.1.3)$$

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 $\llbracket P \rrbracket = \llbracket Q \rrbracket = \llbracket F \rrbracket$.

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

$$\llbracket F; G \rrbracket = \llbracket F \rrbracket \circ \llbracket G \rrbracket. \quad (3.1.4)$$

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

$$\llbracket \text{paralell } (F, G) \rrbracket = \llbracket F \rrbracket \otimes \llbracket G \rrbracket. \quad (3.1.5)$$

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 combining 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

$$\llbracket - \rrbracket : \text{programs} \rightarrow \text{mathematical objects.} \quad (4.1.1)$$

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 $\llbracket F; G \rrbracket = \llbracket G \rrbracket \circ \llbracket F \rrbracket$ where \circ represents composition in this mathematical formalism;
- a notion of concurrency, if $F \text{ par } G$ corresponds to running F and G at the same time then we should have $\llbracket F \text{ par } G \rrbracket = \llbracket F \rrbracket \otimes \llbracket G \rrbracket$;
- 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 $\llbracket F(X) \rrbracket = \llbracket F \rrbracket(\llbracket X \rrbracket)$.

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

```
1 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, \dots ;
- Morphisms, $f : A \rightarrow B$, 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


```
1 len [] = 0
2 len (x : xs) = 1 + len xs
```
 - Another object might be `Num a => [a]`, and a morphism


```
1 mySum :: (Num a) => [a] -> a
2 mySum [] = 0
3 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**, \mathbf{C} , consists of the following data

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

This data is subject to the following conditions:

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

$$h \circ (g \circ f) = (h \circ g) \circ f, \quad (4.3.2)$$

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

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

$$f \circ \text{id}_A = f = \text{id}_B \circ f. \quad (4.3.3)$$

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 $\text{Ob}(\mathbf{C})$ 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 $\text{Ob}(\mathbf{C})$ 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 $A \rightarrow B$ is simply a function from A to B . Composition of morphisms is composition of functions, defined for $f : A \rightarrow B$ and $g : B \rightarrow C$ by $(g \circ f) : A \rightarrow C$ given by $(g \circ f)(a) = g(f(a))$ for all $a \in A$. The identity morphism is the identity function, $\text{id}_A : A \rightarrow A$, given by $\text{id}_A(a) = a$ for all $a \in A$.

Lemma 4.4.2 **Set** is a category.

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

$$(h \circ (g \circ f))(x) = h((g \circ f)(x)) = h(g(f(x))) = (h \circ g)(f(x)) = ((h \circ g) \circ f)(x) \quad (4.4.3)$$

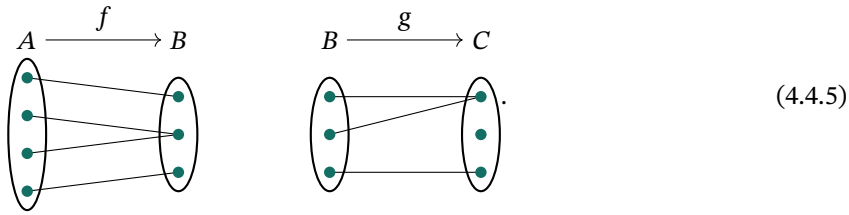
for all $x \in A$, so $h \circ (g \circ f) = (h \circ g) \circ f$. The identity function is then such

that

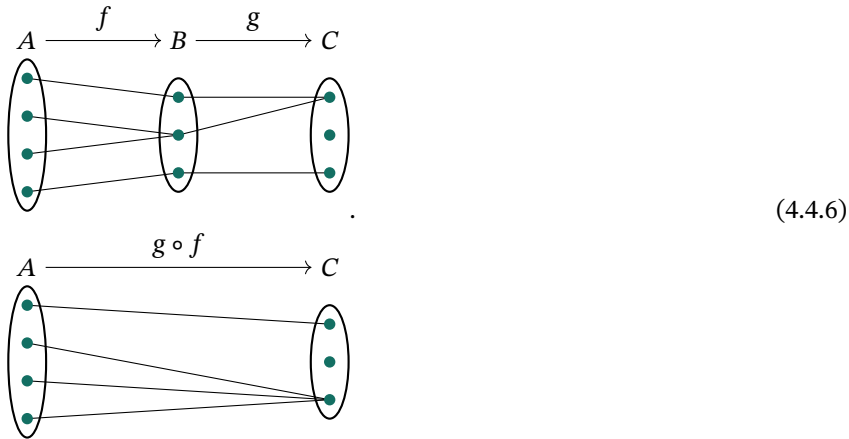
$$(f \circ \text{id}_A)(x) = f(\text{id}_A(x)) = f(x) = \text{id}_B(f(x)) = (\text{id}_B \circ f)(x) \quad (4.4.4)$$

and so $f \circ \text{id}_A = f = \text{id}_B \circ f$. \square

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



Then composition is just following the lines between sets:



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

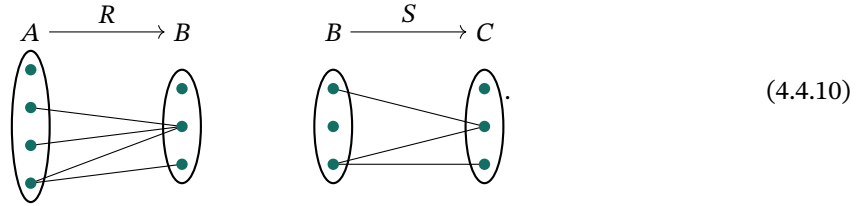
Definition 4.4.7 — Relation Let A and B be sets. A **relation**, R , is a subset of $A \times B$. If $(a, b) \in R$ we write aRb .

Relations are morphisms in a category we will defined shortly (Definition 4.4.15), so for $R \subset A \times B$ we write $R: A \rightarrow B$. 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

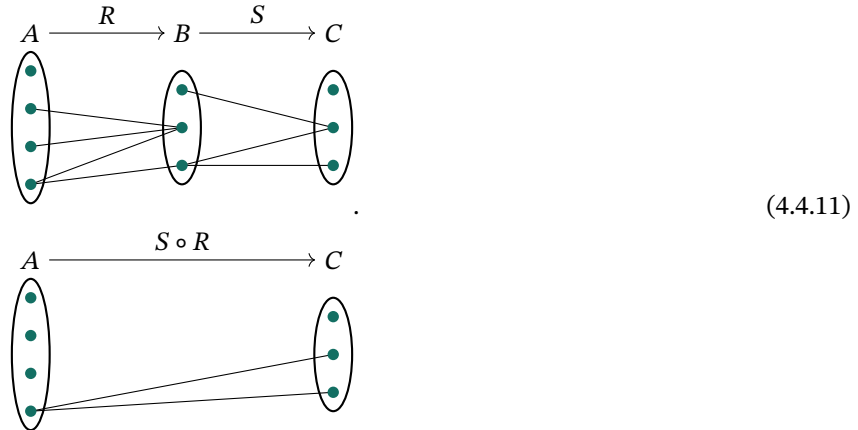
$$R = \{(b, 2), (c, 2), (d, 2), (d, 3)\} \subseteq A \times B, \quad \text{and} \quad (4.4.8)$$

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

can be represented pictorially as



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



That is,

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

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

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

Now we can define a category.

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

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

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

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

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 $R : A \rightarrow B$, $S : B \rightarrow C$, and $T : C \rightarrow D$ 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 $(a, y) \in R$. 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 $R : A \rightarrow B$ be a relation and $\text{id}_A = \{(a, a) \mid a \in A\}$ the identity relation. Then for all $(a, b) \in R$ we have $(a, a) \in \text{id}_A$, meaning that $(a, b) \in R \circ \text{id}_A$. Similarly for all $(a, b) \in R \circ \text{id}_A$ we must have x such that $(x, b) \in R$, and $(x, a) \in \text{id}_A$, which means that $x = a$ and so $(a, b) \in R$. Thus $R \circ \text{id}_A = R$. Similarly $\text{id}_B \circ R = R$. Hence the identity law is satisfied. \square

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 j th element of A and b the i th 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

$$\begin{array}{c} A \xrightarrow{R} B \\ \begin{array}{|c|} \hline \bullet \\ \hline \bullet \\ \hline \bullet \\ \hline \bullet \\ \hline \end{array} \begin{array}{|c|} \hline \bullet \\ \hline \bullet \\ \hline \bullet \\ \hline \bullet \\ \hline \end{array} \end{array} \longleftrightarrow M(R) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (4.4.19)$$

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

$$M: \mathcal{P}(A \times B) \rightarrow \mathcal{M}_{|B| \times |A|}(\mathbb{Z}_2) \quad (4.4.20)$$

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

As an example notice that we have

$$\begin{array}{c} B \xrightarrow{S} C \\ \begin{array}{|c|} \hline \bullet \\ \hline \bullet \\ \hline \bullet \\ \hline \bullet \\ \hline \end{array} \begin{array}{|c|} \hline \bullet \\ \hline \bullet \\ \hline \bullet \\ \hline \bullet \\ \hline \end{array} \end{array} \longleftrightarrow M(S) = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad (4.4.21)$$

and

$$\begin{array}{c} A \xrightarrow{S \circ R} C \\ \begin{array}{|c|} \hline \bullet \\ \hline \bullet \\ \hline \bullet \\ \hline \bullet \\ \hline \end{array} \begin{array}{|c|} \hline \bullet \\ \hline \bullet \\ \hline \bullet \\ \hline \bullet \\ \hline \end{array} \end{array} \longleftrightarrow M(S \circ R) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (4.4.22)$$

The composite can then be calculated as

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

This map, M , actually defines a functor (Definition 4.8.1) $M: \mathbf{Rel} \rightarrow \mathbf{Mat}_{\mathbb{Z}_2}$, where $\mathbf{Mat}_{\mathbb{Z}_2}$ is the category of binary matrices, whose objects are natural numbers and a morphism $n \rightarrow m$ 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.

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

This is actually an example of a more general idea of a representation¹, where we replace objects with matrices preserving the structure of the original space. These ideas can all be understood as functors $\mathbf{C} \rightarrow \mathbf{Mat}_{\mathbb{k}}$ for some appropriate category \mathbf{C} and ring \mathbb{k} . This in turn is more commonly thought of as a map $\mathbf{C} \rightarrow \mathbf{FVect}_{\mathbb{k}}$ identifying matrices with linear maps. This allows for generalisations to infinite dimensional representations, $\mathbf{C} \rightarrow \mathbf{Vect}_{\mathbb{k}}$. We'll define $\mathbf{FVect}_{\mathbb{k}}$ and $\mathbf{Vect}_{\mathbb{k}}$ shortly (Definitions 4.4.28 and 4.4.35).

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 $\mathbf{Vect}_{\mathbb{k}}$, $\mathbf{FVect}_{\mathbb{k}}$, **Hilb**, and **FHilb**

4.4.3.1 $\mathbf{Vect}_{\mathbb{k}}$ and $\mathbf{FVect}_{\mathbb{k}}$

Definition 4.4.24 — Vector Space A **vector space**, $(V, \mathbb{k}, +, \cdot)$, is a set, V , a field^a, \mathbb{k} , and two operations, **vector addition**, $+: V \times V \rightarrow V$, and **scalar multiplication**, $\cdot: \mathbb{k} \times V \rightarrow V$, 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: $\alpha \cdot (u + v) = (\alpha \cdot u) + (\alpha \cdot v)$ for all $\alpha \in \mathbb{k}$ 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 \mathbb{k} ;
- field addition distributes over scalar multiplication: $(\alpha + \beta) \cdot v = (\alpha \cdot v) + (\beta \cdot v)$;
- compatibility of field and scalar multiplication: $\alpha \cdot (\beta \cdot v) = (\alpha \cdot \beta) \cdot v$ for all $\alpha, \beta \in \mathbb{k}$ and $v \in V$ where $\cdot_{\mathbb{k}}$ is multiplication in the field.

^aif 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 $(V, \mathbb{k}, +_V, \cdot_V)$ and $(W, \mathbb{k}, +_W, \cdot_W)$ be vector spaces over the same field, \mathbb{k} . A **linear map** between these vector

spaces is a function $T: V \rightarrow W$ such that

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

for all $\alpha \in \mathbb{k}$ 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

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

We also refer to V and W as vector spaces (over \mathbb{k}) leaving the operations (and potentially the field) implicit.

Definition 4.4.28 — $\mathbf{Vect}_{\mathbb{k}}$ Fix some field \mathbb{k} . The category $\mathbf{Vect}_{\mathbb{k}}$ has vector spaces over \mathbb{k} 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 $\mathbf{Vect}_{\mathbb{k}}$ 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 $T: U \rightarrow V$ and $S: V \rightarrow W$ be linear maps between vector spaces U, V , and W over some field \mathbb{k} . Then for all $u, v \in V$ and $\alpha \in \mathbb{k}$ we have

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

using the linearity of T and then S , and

$$(S \circ T)(\alpha v) = S(T(\alpha v)) = S(\alpha T(v)) = \alpha S(T(v)) = \alpha (S \circ T)(v) \quad (4.4.31)$$

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

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

$$\alpha_1 e_1 + \dots + \alpha_n e_n = 0 \quad (4.4.33)$$

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

$$v = \alpha_1 e_1 + \cdots + \alpha_n e_n \quad (4.4.34)$$

for some finite subset $\{e_1, \dots, e_n\} \subseteq \mathcal{B}$. 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 its dimension is finite.

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

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

Proof. This follows from $\mathbf{Vect}_{\mathbb{k}}$ 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 $T: V \rightarrow W$ defines a matrix with components T_{ij} given by $T(v_i)_j$, where $T(v_i)_j$ is the j th component of $T(v_i)$, which is what we get evaluating the linear map at the i th basis vector, v_i . A matrix, T_{ij} , defines a linear map $T: V \rightarrow W$ 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 — $\mathbf{Mat}_{\mathbb{k}}$ The category $\mathbf{Mat}_{\mathbb{k}}$ has natural numbers as objects with a morphism $n \rightarrow m$ being an $m \times n$ matrix with entries in \mathbb{k} . Composition of morphisms is matrix multiplication and the identity matrix is the identity morphism.

Lemma 4.4.38 $\mathbf{Mat}_{\mathbb{k}}$ 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 $\ell \rightarrow n$ and $n \rightarrow m$ to get a morphism $\ell \rightarrow m$. Associativity follows from associativity of matrix multiplication and the identity matrix is clearly the identity. □

There is an equivalence (Definition 4.8.5) $\mathbf{Mat}_{\mathbb{k}} \rightarrow \mathbf{FVect}_{\mathbb{k}}$ given by sending $n \rightarrow \mathbb{k}^n$ 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 $\mathbb{k} = \mathbb{R}, \mathbb{C}$ equipped with an **inner product** $\langle - | - \rangle : V \times V \rightarrow \mathbb{k}$ 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 $\alpha \in \mathbb{k}$ 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 $\mathbb{k} = \mathbb{R}$ 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 $\| - \| : H \rightarrow \mathbb{R}$ 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

$$\sum_{i=1}^{\infty} \|v_i\| \quad (4.4.41)$$

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

$$\sum_{i=1}^{\infty} v_i \quad (4.4.42)$$

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

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

or equivalently,

$$\lim_n \left\| v - \sum_{i=1}^n v_i \right\| = 0. \quad (4.4.44)$$

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 $\|\cdot\|_H$ and $\|\cdot\|_K$ respectively, both induced by the inner product. A **bounded linear map** is a map $T: H \rightarrow K$ such that

$$\|T(v)\|_K \leq M\|v\|_H \quad (4.4.46)$$

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

Definition 4.4.47 — Hilb and FHilb Fix some field $\mathbb{k} = \mathbb{R}, \mathbb{C}$. 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 $T: H \rightarrow K$ and $S: K \rightarrow J$ 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 $\|S(u)\|_J \leq N\|u\|_K$ 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. \square

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 $T: H \rightarrow K$ is a linear map then we define the **adjoint** to be the linear map $T^\dagger: K \rightarrow H$ 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 $|v\rangle: \mathbb{C} \rightarrow H$ defined by $z \mapsto zv$ and its **bra** is the map $\langle v|: H \rightarrow \mathbb{C}$ 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 $\mathbb{C} \rightarrow H$ is completely defined by where it sends 1, so really linear maps of this form just pick out elements of H , and $|\nu\rangle$ is such that $1 \mapsto 1\nu = \nu$. So we often write $|\nu\rangle$ when we might actually mean ν 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 $w \mapsto \langle \nu | w \rangle$ as simply performing multiplication $\langle \nu | | w \rangle = \langle \nu | w \rangle$.

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

This definition means that V^* is the space of linear maps $V \rightarrow \mathbb{k}$, which in the case of a Hilbert space we can recognise as the space of bras. Note that we are equipping $\mathbf{Vect}_{\mathbb{k}}(V, \mathbb{k})$ 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 $R\text{-Mod}$ 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 continuous maps as morphisms.
- The category **Top_{*}** 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 $f : (X, \bullet) \rightarrow (Y, *)$ 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 $a \rightarrow b$ if $a \leq b$. Since this morphism is unique and $a \leq a$ the identity is simply the unique morphism $a \rightarrow a$. Since $a \leq b$ and $b \leq c$ implies $a \leq c$ there is a unique morphism $a \rightarrow c$ in this case, which must then be the morphism formed by composing the morphisms $a \rightarrow b$ and $b \rightarrow c$.

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 $\bullet \rightarrow \bullet$ 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 is simply reflecting 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

$$A \xrightarrow{f} B \xrightarrow{g} C \quad (4.5.1)$$

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

$$\begin{array}{ccccc} A & \xrightarrow{f} & B & \xrightarrow{g} & C \\ & \searrow & & \nearrow & \\ & & g \circ f & & \end{array} \quad (4.5.2)$$

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

(4.5.3)

(4.5.4)

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

(4.5.5)

(4.5.6)

We now introduce some terminology which will be helpful.

- 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 $f^{-1} : B \rightarrow A$ such that $f^{-1} \circ f = \text{id}_A$ and $f \circ f^{-1} = \text{id}_B$;
- 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 $g, h : B \rightarrow C$;
- we call f a **monomorphism**, or **monic**, if $f \circ g = f \circ h$ implies $g = h$ for all $g, h : C \rightarrow A$.

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 $f : A \rightarrow B$ is an isomorphism then the inverse is unique.

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

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

□

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

$$A \begin{array}{c} \xrightarrow{f} \\ \xleftarrow{f^{-1}} \end{array} B \quad (4.6.4)$$

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 its 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:

$$\text{id}_A = A \left| \begin{array}{c} \hline \hline \end{array} \right. \quad (4.7.1)$$

A general morphism, $f : A \rightarrow B$, 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:

$$f = \begin{array}{c} B \\ | \\ \boxed{f} \\ | \\ A \end{array} \quad (4.7.2)$$

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 $g : B \rightarrow C$ then we have

$$g \circ f = \begin{array}{c} C \\ | \\ \boxed{g} \\ | \\ B \end{array} \circ \begin{array}{c} B \\ | \\ \boxed{f} \\ | \\ A \end{array} = \begin{array}{c} C \\ | \\ \boxed{g} \\ | \\ B \\ | \\ \boxed{f} \\ | \\ A \end{array} \quad (4.7.3)$$

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:

$$\begin{array}{c} B \\ | \\ \boxed{f} \\ | \\ A \\ | \\ \boxed{\text{id}_A} \\ | \\ A \end{array} = \begin{array}{c} B \\ | \\ \boxed{f} \\ | \\ A \end{array} = \begin{array}{c} B \\ | \\ \boxed{\text{id}_B} \\ | \\ B \\ | \\ \boxed{f} \\ | \\ A \end{array}, \quad (4.7.4)$$

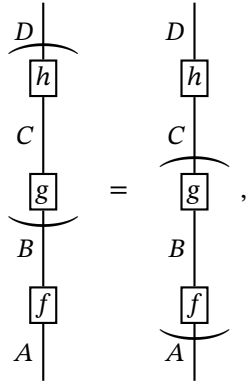
and this is just the identity law

$$f \circ \text{id}_A = f = \text{id}_B \circ f. \quad (4.7.5)$$

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

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

we can write the associativity law as



$$(4.7.7)$$

which is just

$$(h \circ g) \circ f = h \circ (g \circ f). \quad (4.7.8)$$

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 $f : A \rightarrow B$ 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**, $F : \mathbf{C} \rightarrow \mathbf{D}$, is composed of two mappings:

- each object $A \in \text{Ob}(\mathbf{C})$ is mapped to some object $F(A) \in \text{Ob}(\mathbf{D})$;
- 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 $f : A \rightarrow B$ and $g : B \rightarrow C$ morphisms in \mathbf{C} ;
- identities are preserved: $F(\text{id}_A) = \text{id}_{F(A)}$ for $A \in \text{Ob}(\mathbf{C})$.

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 $F : \mathbf{C} \rightarrow \mathbf{D}$ maps each $A \in \text{Ob}(\mathbf{C})$ to some object $FA \in \text{Ob}(\mathbf{D})$, 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 $F\text{id}_A = \text{id}_{FA}$.

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, $C_X : \mathbf{C} \rightarrow \mathbf{D}$, maps every object of \mathbf{C} to $X \in \text{Ob}(\mathbf{D})$ and every morphism $f : A \rightarrow B$ in \mathbf{C} to the identity morphism $\text{id}_X : X \rightarrow X$ in \mathbf{D} .
- The identity functor, $\text{id}_{\mathbf{C}} : \mathbf{C} \rightarrow \mathbf{C}$ maps every object and morphism in \mathbf{C} to itself.
- The power set can be regarded as a functor $\mathcal{P} : \mathbf{Set} \rightarrow \mathbf{Set}$, sending each set to its power set and each function $f : A \rightarrow B$ to the map sending $U \in \mathcal{P}(A)$ to its image

$$f(U) := \{f(u) \mid u \in U\}. \quad (4.8.4)$$

- The map $V \mapsto V^*$ defines a functor $\mathbf{Vect}_{\mathbb{k}} \rightarrow \mathbf{Vect}_{\mathbb{k}}$.
- If G is a group viewed as a one object category then a functor $G \rightarrow \mathbf{Set}$ 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 $s \mapsto g \cdot s$.

- If G is a group viewed as a one object category then a functor $G \rightarrow \mathbf{Vect}_k$ 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 $G \rightarrow H$ 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 $\mathbf{LieGrp} \rightarrow \mathbf{LieAlg}^a$.
- A **forgetful functor**, $\mathbf{C} \rightarrow \mathbf{Set}$ maps a category to **Set** by “forgetting” the structure of the objects in \mathbf{C} . A category, \mathbf{C} , is called **concrete** if there exists a forgetful functor $\mathbf{C} \rightarrow \mathbf{Set}$.
 - The forgetful functor $U: \mathbf{Grp} \rightarrow \mathbf{Set}$ sends each group to its underlying set and each group homomorphism to its underlying function.
 - The forgetful functor $U: \mathbf{Vect}_k \rightarrow \mathbf{Set}$ sends each vector space to its underlying set and each linear transformation to its underlying function.
 - The forgetful functor $U: \mathbf{Top} \rightarrow \mathbf{Set}$ sends each topological space to its underlying set and each continuous function to its underlying function.
- A partially forgetful functor $\mathbf{C} \rightarrow \mathbf{D}$ generalises forgetful functors by forgetting some, but not necessarily all, structure of the objects in \mathbf{C} .
 - The partially forgetful functor $\mathbf{Grp} \rightarrow \mathbf{Mon}$ sends each group to itself, but now viewed as a monoid, essentially forgetting that inverses exist.
 - The partially forgetful functor $\mathbf{Hilb} \rightarrow \mathbf{Vect}_k$ sends each Hilbert space to itself, but forgets the inner product structure.
 - The partially forgetful functor $\mathbf{CRing} \rightarrow \mathbf{Ring}$ sends each commutative ring to itself, forgetting that multiplication is commutative.
 - The partially forgetful functor $\mathbf{Ring} \rightarrow \mathbf{Ab}$ 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 $\mathbf{Set} \rightarrow \mathbf{Grp}$ 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 $\mathbf{Set} \rightarrow \mathbf{Vect}_{\mathbb{k}}$ 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.

^aSee *Symmetries of Quantum Mechanics* or *Symmetries of Particles and Fields*

^bThis 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 $F: \mathbf{C} \rightarrow \mathbf{D}$ is

- **full** when the mapping $f \mapsto Ff$ defines a surjection $\mathbf{C}(A, B) \rightarrow \mathbf{D}(FA, FB)$;
- **faithful** when the mapping $f \mapsto Ff$ defines an injection $\mathbf{C}(A, B) \rightarrow \mathbf{D}(FA, FB)$;
- **essentially surjective on objects** when for each $B \in \text{Ob}(\mathbf{D})$ there is some $A \in \text{Ob}(\mathbf{C})$ 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 $\mathbf{Mat}_{\mathbb{k}} \rightarrow \mathbf{FVect}_{\mathbb{k}}$ 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 $\text{Ob}(\mathbf{C}) \rightarrow \text{Ob}(\mathbf{D})$ and $\mathbf{C}(A, B) \rightarrow \mathbf{D}(FA, FB)$, and the

identity is the identity functor.

To be clear, if $F: \mathbf{C} \rightarrow \mathbf{D}$ and $G: \mathbf{D} \rightarrow \mathbf{E}$ are functors then the functor $(G \circ F): \mathbf{C} \rightarrow \mathbf{E}$ sends $A \in \text{Ob}(\mathbf{C})$ 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 \mathbf{C} and \mathbf{D} be categories and $F, G: \mathbf{C} \rightarrow \mathbf{D}$ functors. For every object $A \in \text{Ob}(\mathbf{C})$ a **natural transformation**, $\zeta: F \Rightarrow G$, assigns a morphism $\zeta_A: FA \rightarrow GA$ in \mathbf{D} such that for every morphism $f: A \rightarrow B$ in \mathbf{C} the following diagram commutes:

$$\begin{array}{ccc} FA & \xrightarrow{\zeta_A} & GA \\ Ff \downarrow & & \downarrow Gf \\ FB & \xrightarrow{\zeta_B} & GB. \end{array} \quad (4.9.2)$$

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 $F: \mathbf{C} \rightarrow \mathbf{D}$ is an equivalence if and only if there is a functor $G: \mathbf{D} \rightarrow \mathbf{C}$ and natural isomorphisms $G \circ F \Rightarrow \text{id}_{\mathbf{C}}$ and $F \circ G \Rightarrow \text{id}_{\mathbf{D}}$. 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

$$\begin{array}{ccc} & F & \\ \mathbf{C} & \begin{array}{c} \curvearrowright \\ \Downarrow \\ \curvearrowleft \end{array} & \mathbf{D}. \\ & G & \end{array} \quad (4.9.3)$$

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, $(\zeta_A)_{A \in \text{Ob}(\mathbf{C})}$.

- 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 $F: \mathbf{C} \rightarrow \mathbf{D}$ is simply a family of objects $(FA)_{A \in \text{Ob}(\mathbf{C})}$. In this case a natural transformation $\zeta: F \Rightarrow G$ is just a family of maps $(\zeta_A: FA \rightarrow GA)_{A \in \text{Ob}(\mathbf{C})}$. 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 $R \rightarrow S$ induces a monoid homomorphism $M_n(R) \rightarrow M_n(S)$ by acting elementwise on the entries of the matrix with the homomorphism. This defines a functor $M_n: \mathbf{CRing} \rightarrow \mathbf{Mon}$.

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

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

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

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

$$(\det_R: M_n(R) \rightarrow U(R))_{R \in \text{Ob}(\mathbf{CRing})}. \quad (4.9.6)$$

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

- Consider the category $\mathbf{FVect}_{\mathbb{k}}$. The map $(-)^*: \mathbf{FVect}_{\mathbb{k}} \rightarrow \mathbf{FVect}_{\mathbb{k}}$ sending each vector space to its dual is a contravariant functor. Thus the map $(-)^{**}: \mathbf{FVect}_{\mathbb{k}} \rightarrow \mathbf{FVect}_{\mathbb{k}}$ sending each vector space to its double dual is also a covariant functor. For each $V \in \text{Ob}(\mathbf{FVect}_{\mathbb{k}})$ we have a canonical isomorphism $\zeta_V: V \rightarrow V^{**}$. Given $v \in V$ the element $\zeta_V(v) \in V^{**}$ is evaluation at v , that is $\zeta_V(v): V^* \rightarrow \mathbb{k}$ maps $\varphi \in V^*$ to $\varphi(v) \in \mathbb{k}$. This defines a natural transformation $\zeta: 1_{\mathbf{FVect}_{\mathbb{k}}} \Rightarrow (-)^{**}$ meaning that each vector space is naturally isomorphic to its double dual. Note that given $F, G: \mathbf{C} \rightarrow \mathbf{D}$ 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 $F, G: \mathbf{C} \rightarrow \mathbf{D}$ 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 to each $A \in \text{Ob}(\mathbf{C})$ the identity morphism id_{FA} in \mathbf{D} .

An example of a functor category is the category of G -sets² for some group G , which is $[G, \mathbf{Set}]$, where G is viewed as a one object category. Similarly $[G, \mathbf{Vect}_{\mathbb{k}}]$ is the category of \mathbb{k} -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) $f : X \rightarrow Y$ such that $f(g \cdot x) = g \cdot f(x)$, that is a map which commutes with the group action, so

²that is sets upon which G acts

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ g \cdot \downarrow & & \downarrow g \cdot \\ X & \xrightarrow{f} & Y \end{array} \quad (4.9.8)$$

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

4.10 Products

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

$$A \times B := \{(a, b) \mid a \in A \text{ and } b \in B\}. \quad (4.10.1)$$

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 $p_A : A \times B \rightarrow A$ and $p_B : A \times B \rightarrow B$ 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

$$A \xleftarrow{p_A} A \times B \xrightarrow{p_B} B. \quad (4.10.2)$$

The key insight comes in requiring that $A \times B$ is, in a sense, the most general³ 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 $f : X \rightarrow A$ and $g : X \rightarrow B$ 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 $\begin{pmatrix} f \\ g \end{pmatrix}$. We can summarise this definition by requiring that the following diagram commutes:

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

$$\begin{array}{ccccc} & & X & & \\ & f \swarrow & \downarrow \begin{smallmatrix} ! \\ f \times g \end{smallmatrix} & \searrow g & \\ A & \xleftarrow{p_A} & A \times B & \xrightarrow{p_B} & B. \end{array} \quad (4.10.3)$$

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.

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 4.10.4 — Product Let \mathbf{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 $p_A : A \times B \rightarrow A$ and $p_B : A \times B \rightarrow B$ such that

$$\begin{array}{ccccc} & & X & & \\ & f \swarrow & \downarrow f \times g & \searrow g & \\ A & \xleftarrow{p_A} & A \times B & \xrightarrow{p_B} & B \end{array} \quad (4.10.5)$$

commutes for all X with morphisms to $f : X \rightarrow A$ and $g : X \rightarrow B$.

Example 4.10.6 — Products

- 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.

Definition 4.10.7 — 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 $A \in \text{Ob}(\mathbf{C})$ and $B \in \text{Ob}(\mathbf{D})$;
- as morphisms $(A_1, B_1) \rightarrow (A_2, B_2)$ pairs of morphisms (f, g) with $f : A_1 \rightarrow A_2$ and $g : B_1 \rightarrow B_2$ 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 $\text{id}_{(A,B)} = (\text{id}_A, \text{id}_B)$.

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 \mathbf{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 $\mathbf{C} \times \mathbf{D} \rightarrow \mathbf{E}$, 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 $\mathbb{R}^2 \rightarrow \mathbb{R}$.

Five

Monoidal Categories

5.1 Tensor Products

Definition 5.1.1 — Bilinear Let U, V and W be vector spaces over \mathbb{k} . A **bilinear map** is a function $T: U \times V \rightarrow W$ which is linear in each variable. That is, the maps $T(-, v): U \rightarrow W$ defined by $u \mapsto T(u, v)$ and $T(u, -): V \rightarrow W$ 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 \mathbb{k} 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 \mathbb{k} . The **tensor product** of U and V is a vector space, which we call $U \otimes V$, equipped with a bilinear map $f: U \times V \rightarrow U \otimes V$ such that for all bilinear maps $g: U \times V \rightarrow W$ there exists a unique linear map $h: U \otimes V \rightarrow W$ such that $g = h \circ f$. That is, such that

$$\begin{array}{ccc} U \times V & \xrightarrow{f \text{ (bilinear)}} & U \otimes V \\ & \searrow g \text{ (bilinear)} & \downarrow h \text{ (linear)} \\ & & W \end{array} \quad (5.1.3)$$

commutes.

^athis definition also holds for R -modules replacing linear with R -linear.

The important idea here is that a linear map $U \otimes V \rightarrow W$ is just as good as a bilinear map $U \times V \rightarrow W$, 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 \mathbb{k} . Let $f: U \rightarrow U'$ and $g: V \rightarrow V'$. Then the **tensor product** of f and g is defined to be the map $f \otimes g: U \otimes V \rightarrow U' \otimes V'$ 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 $(V, \langle - | - \rangle_V)$ be inner product spaces (Hilbert spaces) over \mathbb{k} . Then $(U \otimes V, \langle - | - \rangle_{U \otimes V})$ is an inner product space (Hilbert space) with the inner product

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

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

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

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 $(A \times B) \times C \rightarrow A \times (B \times C)$ 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, \mathbf{C} ;
2. a **tensor product functor**

$$- \otimes - : \mathbf{C} \times \mathbf{C} \rightarrow \mathbf{C}, \quad (5.4.2)$$

also called the **monoidal product**;

3. a **unit object** $I \in \text{Ob}(\mathbf{C})$;
4. a natural isomorphism

$$\alpha : (- \otimes -) \otimes - \Rightarrow - \otimes (- \otimes -) \quad (5.4.3)$$

called the **associator** which has components

$$\alpha_{A,B,C} : (A \otimes B) \otimes C \rightarrow A \otimes (B \otimes C) \quad (5.4.4)$$

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

5. a natural isomorphism

$$\lambda : I \otimes - \Rightarrow - \quad (5.4.5)$$

called the **left unitor** which has components

$$\lambda_A : I \otimes A \rightarrow A \quad (5.4.6)$$

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

6. a natural isomorphism

$$\rho : - \otimes I \Rightarrow - \quad (5.4.7)$$

called the **right unitor** which has components

$$\rho_A : A \otimes I \rightarrow A \quad (5.4.8)$$

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

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

$$\begin{array}{ccc} (A \otimes I) \otimes B & \xrightarrow{\alpha_{A,I,B}} & A \otimes (I \otimes B) \\ \searrow \rho_A \otimes \text{id}_B & & \swarrow \text{id}_A \otimes \lambda_B \\ & A \otimes B & \end{array} \quad (5.4.9)$$

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

$$\begin{array}{ccccc} & & (A \otimes (B \otimes C)) \otimes D & \xrightarrow{\alpha_{A,B \otimes C, D}} & A \otimes ((B \otimes C) \otimes D) \\ & \nearrow \alpha_{A,B,C} \otimes \text{id}_D & & & \searrow \text{id}_A \otimes \alpha_{B,C,D} \\ ((A \otimes B) \otimes C) \otimes D & & & & A \otimes (B \otimes (C \otimes D)) \\ & \searrow \alpha_{A \otimes B, C, D} & & \nearrow \alpha_{A,B,C \otimes D} & \\ & (A \otimes B) \otimes (C \otimes D) & & & \end{array} \quad (5.4.10)$$

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

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

$$\begin{array}{ccc}
 (A \otimes B) \otimes C & \xrightarrow{\alpha_{A,B,C}} & A \otimes (B \otimes C) \\
 (f \otimes g) \otimes h \downarrow & & \downarrow f \otimes (g \otimes h) \\
 (A' \otimes B') \otimes C' & \xrightarrow{\alpha_{A',B',C'}} & A' \otimes (B' \otimes C')
 \end{array} \quad (5.4.11)$$

for all $f: A \rightarrow A'$, $g: B \rightarrow B'$, and $h: C \rightarrow C'$. 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

$$\begin{array}{ccccc}
 I \otimes A & \xrightarrow{\lambda_A} & A & \xleftarrow{\rho_A} & A \otimes I \\
 \text{id}_I \otimes f \downarrow & & f \downarrow & & \downarrow f \otimes \text{id}_I \\
 I \otimes B & \xrightarrow{\lambda_B} & B & \xleftarrow{\rho_B} & B \otimes I
 \end{array} \quad (5.4.12)$$

for all $f: A \rightarrow B$. 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 $\text{id}_I \otimes f$, 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 $f \circ \lambda_A = \lambda_B \circ (\text{id}_I \otimes f)$ 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 let's 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 $\rho_A \otimes \text{id}_B$ 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 $\text{id}_A \otimes \lambda_B$ to $A \otimes (I \otimes B)$. The triangle equality says that both of these are actually the same, that is

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

Finally, let's 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 $\alpha_{A,B,C} \otimes \text{id}_D$. 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 $\text{id}_A \otimes \alpha_{B,C,D}$ 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

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

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 (vertical¹) **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, \mathbf{C} :

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

¹cf. 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.

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.

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 singleton² $\{\bullet\}$ 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.

²Since 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.

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 $\alpha_{A,B,C} : (A \times B) \times C \rightarrow A \times (B \times C)$ defined by $((a, b), c) \mapsto (a, (b, c))$;
4. the left unitor at A to be the function $\lambda_A : \{\bullet\} \times A \rightarrow A$ defined by $(\bullet, a) \mapsto a$;
5. the right unitor at A to be the function $\rho_A : A \times \{\bullet\} \rightarrow A$ 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 $I = \emptyset$. Yet another choice is $A \otimes B = A \sqcup B \sqcup (A \times B)$ with $I = \emptyset$.

This common definition of **Set** as a monoidal category is an example of a more general idea. In some category **C** a **terminal object**, $T \in \text{Ob}(\mathbf{C})$, is an object such that for any object $A \in \text{Ob}(\mathbf{C})$ there exists exactly one morphism $A \rightarrow T$. Similarly a **initial object**, $I \in \text{Ob}(\mathbf{C})$, is an object such that for any object $A \in \text{Ob}(\mathbf{C})$ there exists exactly one morphism $I \rightarrow A$. 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 and category with initial objects and coproducts³ 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.

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

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

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

$$\begin{array}{ccc}
 ((a, \bullet), b) & \xrightarrow{\alpha_{A,I,B}} & (a, (\bullet, b)) \\
 \rho_A \times \text{id}_B \searrow & & \swarrow \text{id}_A \times \lambda_B \\
 (\rho_A(a, \bullet), \text{id}_B(b)) & = (a, b) = & (\text{id}_A(a), \lambda_B(\bullet, b)).
 \end{array} \quad (5.5.3)$$

Similarly, we can demonstrate the commutativity of the pentagon:

$$\begin{array}{ccc}
 (\alpha_{A,B,C}((a, b), c), \text{id}_D(D)) & \xrightarrow{\alpha_{A,B \times C, D}} & \alpha_{A,B \times C, D}((a, (b, c)), d) \\
 = ((a, (b, c)), d) & & = (a, ((b, c), d)) \\
 \uparrow \alpha_{A,B,C} \times \text{id}_D & & \downarrow \text{id}_A \times \alpha_{B,C,D} \\
 (((a, b), c), d) & & \\
 \downarrow \alpha_{A \times B, C, D} & & \downarrow (\text{id}_A(a), \alpha_{B,C,D}((b, c), d)) \\
 \alpha_{A \times B, C, D}(((a, b), c), d) & \xrightarrow{\alpha_{A,B,C \times D}} & \alpha_{A,B,C \times D}((a, b), (c, d)). \\
 = ((a, b), (c, d)) & & = (a, (b, (c, d))) =
 \end{array} \quad (5.5.4)$$

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 $f : A \rightarrow B$:

$$\begin{array}{ccc}
 (\bullet, a) & \xrightarrow{\lambda_A} & \lambda_A(\bullet, a) = a \\
 \text{id}_I \times f \downarrow & & \downarrow f \\
 (\text{id}_I(\bullet), f(a)) = (\bullet, f(a)) & \xrightarrow{\lambda_B} & \lambda_B(\bullet, f(a)) = f(a).
 \end{array} \quad (5.5.5)$$

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

$$\begin{array}{ccc}
 ((a, b), c) & \xrightarrow{\alpha_{A,B,C}} & (a, (b, c)) \\
 (f \times g) \times h \downarrow & & \downarrow f \times (g \times h) \\
 ((f(a), g(b)), h(c)) & \xrightarrow{\alpha_{A',B',C'}} & (f(a), (g(b), h(c)))
 \end{array} \quad (5.5.6)$$

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

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 $S \subseteq C \times D$ 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 $\{\bullet\} \times A \rightarrow A$ 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 4.10.4](#)) in **Rel**, and \emptyset 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 **Vect** _{\mathbb{k}} 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 **Vect** _{\mathbb{k}} as a monoidal category, just replace \mathbb{C} with \mathbb{k} and ignore inner products.

Definition 5.5.8 — Hilb as a Monoidal Category The category **Hilb** can be 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 $\alpha_{H,J,K} : (H \otimes J) \otimes K \rightarrow H \otimes (J \otimes K)$ 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 $\lambda_A : I \otimes A \rightarrow A$ 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 $\rho_A : A \otimes I \rightarrow A$ 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 discuss 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 $T : H \otimes J \rightarrow K$ 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

$$T\left(\sum_i u_i \otimes v_i\right) = \sum_i T(u_i \otimes v_i). \quad (5.5.9)$$

In the specific case of the associator we have that

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

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

$$\lambda_A(z \otimes u) = \lambda_A(z1 \otimes u) = z\lambda_A(1 \otimes u) = zu \quad (5.5.11)$$

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 $R\text{-}\mathbf{Mod}$ and makes $R\text{-}\mathbf{Mod}$ 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, \mathbf{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 $a + \dots + a$ if $n > 0$, e if $n = 0$, and $-a - \dots - a$ 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}_* , is a monoidal category with the smash product as the monoidal product and the pointed 0-sphere (that is the two point space $(\{-1, 1\}, \{\emptyset, \{-1\}, \{1\}, \{-1, 1\})$ with one point chosen as the base point) serving as the unit. The smash product, \wedge , is defined between two pointed spaces, (X, x_*) and (Y, y_*) , by forming $X \times Y$ then identifying $(x, y_*) \sim (x_*, y)$ 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, $A_1 \otimes \dots \otimes A_n$, of objects in \mathbf{C} , then we have a morphism $A_1 \otimes \dots \otimes A_n \rightarrow B_1 \otimes \dots \otimes B_m$ only if $m = n$, in which case the morphism is a finite sequence of morphisms $f_1 \otimes \dots \otimes f_n$, with $f_i : A_i \rightarrow B_i$ a morphism in \mathbf{C} . The monoidal product is then the concatenation of these lists, giving $(A_1 \otimes \dots \otimes A_n) \otimes (B_1 \otimes \dots \otimes B_m) = A_1 \otimes \dots \otimes A_n \otimes B_1 \otimes \dots \otimes B_m$, 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 \mathbf{C} be a monoidal category with monoidal product \otimes . Consider the following objects morphisms in \mathbf{C} :

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

We have

$$(g \circ f) \otimes (j \circ h) = (g \otimes j) \circ (f \otimes h). \quad (5.6.3)$$

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

$$(g \circ f) \otimes (j \circ h) = \otimes(g \circ f, j \circ h). \quad (5.6.4)$$

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 \mathbf{C} and composition on the right is in $\mathbf{C} \times \mathbf{C}$. Thus in $\mathbf{C} \times \mathbf{C}$ we have

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

which we can write as

$$X \xrightarrow{p} Y \xrightarrow{q} Z. \quad (5.6.6)$$

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

$$A \otimes B = A \left| \begin{array}{c} \\ \\ \end{array} \right| B \left| \begin{array}{c} \\ \\ \end{array} \right|. \quad (5.7.1)$$

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

$$f \otimes g = \begin{array}{c} B \\ | \\ \boxed{f} \\ | \\ A \end{array} \quad \begin{array}{c} D \\ | \\ \boxed{g} \\ | \\ C \end{array} . \quad (5.7.2)$$

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

$$I = \begin{array}{c} | \\ | \\ | \end{array} = \begin{array}{c} | \\ | \\ \vdots \end{array} = \begin{array}{c} \cdots \\ \cdots \\ \cdots \end{array} = . \quad (5.7.3)$$

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

$$\begin{array}{c} | \\ | \\ \boxed{\lambda} \\ | \\ \vdots \end{array} = \begin{array}{c} | \\ | \\ \diagdown \\ \vdots \end{array} = \begin{array}{c} | \\ | \\ | \end{array}, \quad \text{and} \quad \begin{array}{c} | \\ | \\ \boxed{\rho} \\ | \\ \vdots \end{array} = \begin{array}{c} | \\ | \\ \diagup \\ \vdots \end{array} = \begin{array}{c} | \\ | \\ | \end{array}. \quad (5.7.4)$$

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,

$$(g \circ f) \otimes (j \circ h) = (g \otimes j) \circ (f \otimes h), \quad (5.7.5)$$

becomes

$$\left\{ \begin{array}{c} C \\ | \\ \boxed{g} \\ | \\ B \\ | \\ \boxed{f} \\ | \\ A \end{array} \right\} \left\{ \begin{array}{c} F \\ | \\ \boxed{j} \\ | \\ E \\ | \\ \boxed{h} \\ | \\ D \end{array} \right\} = \begin{array}{c} \overbrace{\begin{array}{cc} C & F \end{array}} \\ | \\ \overbrace{\begin{array}{cc} \boxed{g} & \boxed{j} \end{array}} \\ | \\ B & E \\ | \\ \overbrace{\begin{array}{cc} \boxed{f} & \boxed{h} \end{array}} \\ | \\ \overbrace{\begin{array}{cc} A & D \end{array}} \end{array} . \quad (5.7.6)$$

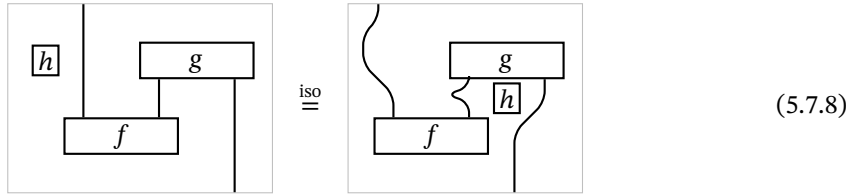
In order to proceed with calculations we need to define what it means for two diagrams to be "the same". Exact equality is too strict, since it's impossible to actually draw two identical diagrams. Besides, we want to be able to move bits of the diagrams around, such as sliding morphisms along the wires,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

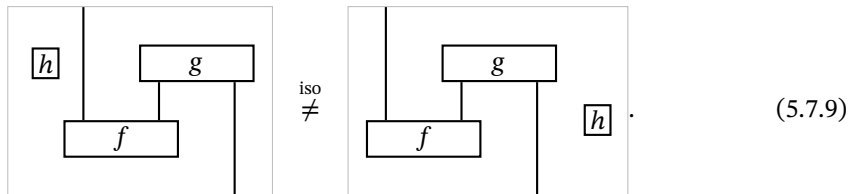
This can be made more rigorous through the idea of an isotopy, which is a homotopy between the two diagrams such that at every point the diagrams remain embedded in the rectangular region of the plane in a non-intersecting way and the inputs and outputs are fixed.

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



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:



5.7.1 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 $f : \{\bullet\} \rightarrow A$ such that $f(\bullet) = a$. Similarly, consider some vector space, V , over \mathbb{k} from which we want to pick a vector, v . We can identify this selection of a single element with a linear map $T : \mathbb{k} \rightarrow V$ such that $T(1) = v$, and then $T(k) = T(k1) = kT(1) = kv$ for all $k \in \mathbb{k}$. 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 $I \rightarrow A$.

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 $\mathbb{C} \rightarrow H$. We can define a state, $T_v : \mathbb{C} \rightarrow H$ 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 $I \rightarrow A$, 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 $a : I \rightarrow A$ is represented as

$$a = \begin{array}{c} A \\ | \\ \triangle \\ a \end{array} . \quad (5.8.2)$$

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 $A \rightarrow I$.

Consider a Hilbert space, H . Then an effect is a linear map $H \rightarrow \mathbb{C}$. We can define one such map for each $v \in H$, $T_v : H \rightarrow \mathbb{C}$ 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 $a : A \rightarrow I$ is drawn as

$$a = \begin{array}{c} \triangle \\ a \\ | \\ A \end{array} . \quad (5.8.4)$$

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 $I \rightarrow A \otimes B$.

This definition generalises to any number of objects.

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

$$c = \begin{array}{c} A \quad B \\ | \quad | \\ \triangle \\ c \end{array} . \quad (5.8.6)$$

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, $c : I \rightarrow A \otimes B$, is a

product state if it is of the form

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

That is, we can write $c = (a \otimes b) \circ \lambda_I^{-1}$ as a product of two states, $a : I \rightarrow A$ and $b : I \rightarrow B$.

A joint state which cannot be written in this form is called an **entangled state**.

R 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

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

Graphically if c is a product state then

$$\begin{array}{c} A \quad B \\ | \quad | \\ \triangle \\ c \end{array} = \begin{array}{c} A \quad B \\ | \quad | \\ \triangle \\ a \end{array} \begin{array}{c} B \\ | \\ \triangle \\ b \end{array}. \quad (5.8.10)$$

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 $|01\rangle + |10\rangle$.

^ahere 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.

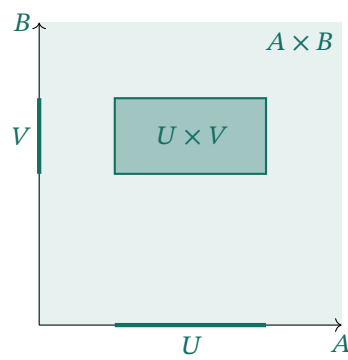


Figure 5.1: An example of a “square” subset of $\mathbb{R} \times \mathbb{R}$, identified with the plane, given $U, V \subseteq \mathbb{R}$ we get a rectangle $U \times V$ in the plane.

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, $\cdot : \mathbb{k} \times V \rightarrow V$, $(a, v) \mapsto av$. We can similarly define a map between (bounded) linear maps, given by¹ $\cdot : \mathbb{k} \times \text{hom}(V, W) \rightarrow \text{hom}(V, W)$, $(a, f) \mapsto a \cdot f$ where if $f : V \rightarrow W$ is a (bounded) linear map then we define $a \cdot f$ to be the map $(a \cdot f) : V \rightarrow W$ 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 discuss 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 $\mathbb{C} \rightarrow \mathbb{C}$ 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 $T : H \rightarrow K$. Given some $a \in \mathbb{C}$ we can consider

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

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

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

¹We use hom here to avoid specifying exactly which category of vector spaces we are working in, it could be **Vect** _{\mathbb{k}} , **FVect** _{\mathbb{k}} , **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

$$\begin{array}{ccc} H & \xrightarrow{a \cdot T} & K \\ \lambda_A^{-1} \downarrow & & \uparrow \lambda_B \\ \mathbb{C} \otimes H & \xrightarrow{f_a \otimes T} & \mathbb{C} \otimes K. \end{array} \quad (6.2.3)$$

We now drop the notation f_a for the map $f_a : \mathbb{C} \rightarrow \mathbb{C}$ defined by $f_a(1) = a$ and instead just write $a : \mathbb{C} \rightarrow \mathbb{C}$ 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

$$\begin{array}{ccc} H & \xrightarrow{a \cdot T} & K \\ \lambda_A^{-1} \downarrow & & \uparrow \lambda_B \\ \mathbb{C} \otimes H & \xrightarrow{a \otimes T} & \mathbb{C} \otimes K. \end{array} \quad (6.2.4)$$

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 \mathbf{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 $I \rightarrow I$.

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 \mathbf{C} be a monoidal category with unit I . Then given any two scalars $a, b : I \rightarrow I$ we have $a \circ b = b \circ a$, that is scalars commute.

Proof. We want to show that the following diagram commutes

$$\begin{array}{ccc} I & \xrightarrow{a} & I \\ b \downarrow & & \downarrow b \\ I & \xrightarrow{a} & I. \end{array} \quad (6.3.3)$$

The naturality square for the left unitor, λ , is

$$\begin{array}{ccc} I \otimes A & \xrightarrow{\lambda_A} & A \\ \text{id}_I \otimes f \downarrow & & \downarrow f \\ I \otimes B & \xrightarrow{\lambda_B} & B. \end{array} \quad (6.3.4)$$

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

$$\begin{array}{ccc} I \otimes I & \xrightarrow{\lambda_I} & I \\ \text{id}_I \otimes f \downarrow & & \downarrow f \\ I \otimes I & \xrightarrow{\lambda_I} & I. \end{array} \quad (6.3.5)$$

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 write this diagram as

$$\begin{array}{ccc} I \otimes I & \xleftarrow{\lambda_I^{-1}} & I \\ \text{id}_I \otimes f \downarrow & & \downarrow f \\ I \otimes I & \xrightarrow{\lambda_I} & I. \end{array} \quad (6.3.6)$$

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

$$\begin{array}{ccc} I & \xrightarrow{f} & I \\ \lambda_I^{-1} \downarrow & & \uparrow \lambda_I \\ I \otimes I & \xrightarrow{\text{id}_I \otimes f} & I \otimes I. \end{array} \quad (6.3.7)$$

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

$$\begin{array}{ccc} I & \xrightarrow{f} & I \\ \rho_I \uparrow & & \uparrow \rho_I \\ I \otimes I & \xrightarrow{f \otimes \text{id}_I} & I \otimes I, \end{array} \quad (6.3.8)$$

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

$$\begin{array}{ccc} I & \xrightarrow{f} & I \\ \rho_I^{-1} \downarrow & & \downarrow \rho_I^{-1} \\ I \otimes I & \xrightarrow{f \otimes \text{id}_I} & I \otimes I. \end{array} \quad (6.3.9)$$

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

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

Now consider the interchange law (Theorem 5.6.1),

$$(g \circ f) \otimes (j \circ h) = (g \otimes j) \circ (f \otimes h). \quad (6.3.11)$$

We have here

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

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

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

$$(6.3.15)$$

This implies commutativity of the following diagram

$$\begin{array}{ccc} A \otimes D & \xrightarrow{f \otimes h} & B \otimes E \\ \text{id}_A \otimes (j \circ h) \downarrow & & \downarrow g \otimes j \\ A \otimes F & \xrightarrow{(g \circ f) \otimes \text{id}_F} & C \otimes F. \end{array} \quad (6.3.16)$$

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

$$\begin{array}{ccc} I \otimes I & \xrightarrow{f \otimes \text{id}_I} & I \otimes I \\ \text{id}_I \otimes j \downarrow & & \downarrow \text{id}_I \otimes j \\ I \otimes I & \xrightarrow{f \otimes \text{id}_I} & I \otimes I. \end{array} \quad (6.3.17)$$

Now $f, j : I \rightarrow I$ 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:

$$\begin{array}{ccccc} I & \xrightarrow{a} & I & & I \\ & \searrow b & & \searrow b & \\ & I & \xrightarrow{a} & I & \\ \lambda_I^{-1} \downarrow \rho_I^{-1} & & \lambda_I^{-1} \downarrow \rho_I^{-1} & & \lambda_I \downarrow \rho_I \\ I \otimes I & \xrightarrow{a \otimes \text{id}_I} & I \otimes I & \xrightarrow{\text{id}_I \otimes b} & I \otimes I \\ & \searrow \text{id}_I \otimes b & & \searrow \text{id}_I \otimes b & \\ & I \otimes I & \xrightarrow{a \otimes \text{id}_I} & I \otimes I & \end{array} \quad (6.3.18)$$

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

$$a \circ b = b \circ a \quad (6.3.19)$$

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

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

$$a = \textcircled{a}. \quad (6.3.20)$$

Commutativity of scalar multiplication becomes

$$\begin{array}{c} \textcircled{b} \\ = \\ \textcircled{a} \end{array} = \begin{array}{c} \textcircled{a} \\ \textcircled{b} \end{array}. \quad (6.3.21)$$

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 $f : A \rightarrow B$, and a scalar $a : I \rightarrow I$ both in \mathbf{C} then we define the **left scalar multiplication** of f by a , denoted $a \cdot f : A \rightarrow B$, to be the morphism

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

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

$$\begin{array}{ccc} A & \xrightarrow{a \cdot f} & B \\ \lambda_A^{-1} \downarrow & & \uparrow \lambda_B \\ I \otimes A & \xrightarrow{f \otimes a} & I \otimes B. \end{array} \quad (6.4.3)$$

Graphically, we have

$$a \cdot f = \textcircled{a} \begin{array}{c} | \\ B \\ \boxed{f} \\ | \\ A \end{array}. \quad (6.4.4)$$

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

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

so that the following diagram commutes

$$\begin{array}{ccc} A & \xrightarrow{f \cdot a} & B \\ \rho_A^{-1} \downarrow & & \uparrow \rho_B \\ A \otimes I & \xrightarrow{a \otimes f} & B \otimes I. \end{array} \quad (6.4.6)$$

This is then drawn as

$$\begin{array}{c} B \\ | \\ \boxed{f} \\ | \\ A \end{array} \quad \textcircled{a}. \quad (6.4.7)$$

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 $\text{id}_{\mathbb{C}}$. 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 \mathbf{C} be a monoidal category with unit I and $f : A \rightarrow B$ a morphism of \mathbf{C} . Then $\text{id}_I \cdot f = f$.

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

$$\begin{array}{ccc} A & \xrightarrow{\text{id}_I \cdot f} & B \\ \lambda_A^{-1} \downarrow & & \uparrow \lambda_B \\ I \otimes A & \xrightarrow{f \otimes a} & I \otimes B. \end{array} \quad (6.4.9)$$

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

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ \lambda_A^{-1} \downarrow & & \uparrow \lambda_B \\ I \otimes A & \xrightarrow{f \otimes a} & I \otimes B. \end{array} \quad (6.4.10)$$

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

$$\begin{array}{ccc} A & \xrightarrow{a \cdot f} & B \\ \lambda_A \uparrow & & \uparrow \lambda_B \\ I \otimes A & \xrightarrow{f \otimes a} & I \otimes B. \end{array} \quad (6.4.11)$$

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

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

$$\begin{array}{c} \textcircled{\text{id}_I} \\ | \\ \boxed{f} \\ | \end{array} = \begin{array}{c} | \\ | \\ | \end{array} = \begin{array}{c} | \\ | \\ \boxed{f} \\ | \end{array}, \quad (6.4.12)$$

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

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

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

$$(a \otimes \text{id}_I) \circ (\text{id}_I \otimes b) = (a \circ \text{id}_I) \otimes (\text{id}_I \circ b) = a \otimes b. \quad (6.4.15)$$

Hence, we have

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

\square

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

$$\begin{array}{c} \textcircled{a} \\ | \\ \textcircled{b} \textcircled{a} \\ | \\ \textcircled{b} \end{array} = \begin{array}{c} \textcircled{a} \\ | \\ \textcircled{b} \end{array}. \quad (6.4.17)$$

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

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

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

Hence,

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

Similarly, we have

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

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

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

$$\begin{array}{ccccccc}
 A & \xrightarrow{\text{id}_A} & A & \xrightarrow{a \cdot (b \cdot f)} & B & \xrightarrow{\text{id}_B} & B \\
 \downarrow \lambda_A^{-1} & & \downarrow \lambda_A^{-1} & & \uparrow \lambda_B & & \uparrow \lambda_B \\
 I \otimes A & \xrightarrow{\text{id}_{I \otimes A}} & I \otimes A & \xrightarrow{a \otimes (b \cdot f)} & I \otimes B & \xrightarrow{\text{id}_{I \otimes B}} & I \otimes B \\
 & & \downarrow \text{id}_I \otimes \lambda_A^{-1} & & \uparrow \text{id}_I \otimes \lambda_B & & \\
 & & I \otimes (I \otimes A) & \xrightarrow{a \otimes (b \otimes f)} & I \otimes (I \otimes B) & & \\
 \lambda_I^{-1} \otimes \text{id}_A \swarrow & & \downarrow \alpha_{I,I,A}^{-1} & & \uparrow \alpha_{I,I,B} & \searrow \lambda_I \otimes \text{id}_B & \\
 & & (I \otimes I) \otimes A & \xrightarrow{(a \otimes b) \otimes f} & (I \otimes I) \otimes B & &
 \end{array} \quad (6.4.24)$$

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 by 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 $\lambda_I^{-1} \otimes \text{id}_A$ 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 $\lambda_I \otimes \text{id}_B$. 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. \square

The following is a proof in the graphical calculus:

$$\textcircled{a} \left\{ \textcircled{b} \quad \boxed{f} \right\} = \left\{ \textcircled{a} \quad \textcircled{b} \right\} \boxed{f}. \quad (6.4.25)$$

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 $\text{id}_I \in \mathbf{C}(I, I)$, since $\text{id}_I \cdot a = a$ 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. \square

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 $f : A \rightarrow B$ and $g : B \rightarrow C$. Then $(b \cdot g) \circ (a \cdot f) = (b \circ a) \cdot (g \circ f)$.

Proof. The left hand side is

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

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

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

The right hand side is

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

Applying the interchange law to this we get

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

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

The following is a proof in the graphical calculus:

$$\begin{array}{c} \text{---} \textcircled{b} \text{---} \boxed{g} \text{---} \\ \text{---} \textcircled{a} \text{---} \boxed{f} \text{---} \end{array} = \begin{array}{c} \left(\textcircled{b} \right) \\ \left(\textcircled{a} \right) \end{array} \cdot \begin{array}{c} \left(\boxed{g} \right) \\ \left(\boxed{f} \right) \end{array}. \quad (6.4.32)$$

6.5 Examples

In **Set** scalars are functions $\{\bullet\} \rightarrow \{\bullet\}$. There is only one such function, $\bullet \mapsto \bullet$, which is simply $\text{id}_{\{\bullet\}}$. Thus there is a single scalar in **Set**, and it is the trivial $\text{id}_{\{\bullet\}}$ which is such that $\text{id}_{\{\bullet\}} \cdot f = f$ 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, $\text{true} = \{(\bullet, \bullet)\} = \text{id}_{\{\bullet\}}$ and $\text{false} = \emptyset$. 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

\cdot	true	false	(6.5.1)
true	true	false	
false	false	false	

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 $\text{true} \cdot R = R$ and $\text{false} \cdot R = \emptyset = \text{false}$, so true and false behave a bit like 1 and 0 for scalar multiplication.

In **Hilb** scalars are linear maps $\mathbb{C} \rightarrow \mathbb{C}$. 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:

- 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, \mathbf{C} equipped with a natural isomorphism, called the **braiding**,

$$\sigma : - \otimes - \Rightarrow - \otimes - \quad (7.2.2)$$

which for objects A and B has components

$$\sigma_{A,B} : A \otimes B \rightarrow B \otimes A. \quad (7.2.3)$$

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

$$\begin{array}{ccc} A \otimes (B \otimes C) & \xrightarrow{\sigma_{A,B \otimes C}} & (B \otimes C) \otimes A \\ \alpha_{A,B,C}^{-1} \swarrow & & \nwarrow \alpha_{B,C,A}^{-1} \\ (A \otimes B) \otimes C & & B \otimes (C \otimes A) \\ \sigma_{A,B} \otimes \text{id}_C \searrow & & \searrow \text{id}_B \otimes \sigma_{A,C} \\ (B \otimes A) \otimes C & \xrightarrow{\alpha_{B,A,C}} & B \otimes (A \otimes C), \end{array} \quad (7.2.4)$$

and

$$\begin{array}{ccc} (A \otimes B) \otimes C & \xrightarrow{\sigma_{A \otimes B, C}} & B \otimes (C \otimes A) \\ \alpha_{A,B,C} \swarrow & & \nwarrow \alpha_{C,A,B} \\ A \otimes (B \otimes C) & & (C \otimes A) \otimes B \\ \text{id}_A \otimes \sigma_{B,C} \searrow & & \searrow \sigma_{A,C} \otimes \text{id}_B \\ A \otimes (C \otimes B) & \xrightarrow{\alpha_{A,C,B}^{-1}} & (A \otimes C) \otimes B. \end{array} \quad (7.2.5)$$

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

$$\begin{array}{ccc} A \otimes B & \xrightarrow{\sigma_{A,B}} & B \otimes A \\ f \otimes g \downarrow & & \downarrow g \otimes f \\ A' \otimes B' & \xrightarrow{\sigma_{A',B'}} & B' \otimes A' \end{array} \quad (7.2.6)$$

for all $f : A \rightarrow A'$ and $g : B \rightarrow B'$. 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 $f : A \rightarrow A'$ and $g : B \rightarrow B'$:

$$\begin{array}{ccc} (a, b) & \xrightarrow{\sigma_{A,B}} & (b, a) \\ \downarrow f \times g & & \downarrow g \times f \\ (f(a), g(b)) & \xrightarrow{\sigma_{A',B'}} & (g(b), f(a)). \end{array} \quad (7.3.3)$$

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

$$\begin{array}{ccccc} & (a, (b, c)) & \xrightarrow{\sigma_{A,B \otimes C}} & ((b, c), a) & \\ \alpha_{A,B,C}^{-1} \swarrow & & & & \swarrow \alpha_{B,C,A}^{-1} \\ ((a, b), c) & & & & (b, (c, a)) = \\ & \searrow \sigma_{A,B} \times \text{id}_C & & \searrow \text{id}_B \times \sigma_{A,C} & \\ & (\sigma_{A,B}(a, b), \text{id}_C(c)) & \xrightarrow{\alpha_{B,A,C}} & (b, (a, c)) & \\ & = ((b, a), c) & & & \end{array}$$

□

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 $\sigma_{A,B} : A \otimes B \rightarrow B \otimes A$ as follows

$$\sigma_{A,B} = \begin{array}{c} B \quad A \\ \diagdown \quad \diagup \\ A \quad B \end{array} . \quad (7.4.1)$$

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 $\sigma_{A,B}^{-1} : B \otimes A \rightarrow A \otimes B$ exists, and we denote it by

$$\sigma_{A,B}^{-1} = \begin{array}{c} A \quad B \\ \diagup \quad \diagdown \\ B \quad A \end{array} . \quad (7.4.2)$$

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.

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

That $\sigma_{A,B}$ and $\sigma_{A,B}^{-1}$ are inverses, tells us that $\sigma_{A,B}^{-1} \circ \sigma_{A,B} = \text{id}_{A \otimes B}$, graphically¹,



$$(7.4.4)$$

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



$$(7.4.5)$$

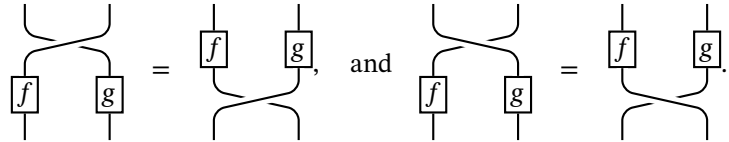
In both of these it looks like if the wires were strings we could just pull them taught, this is the type of thing that is allowed in a spatial isotopy.

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 $(\sigma^{-1})_{A,B} = (\sigma_{A,B})^{-1} = \sigma_{A,B}^{-1}$, which is necessarily also natural. The naturality conditions

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

$$\sigma_{A',B'}^{-1} \circ (f \otimes g) = (g \otimes f) \circ \sigma_{A,B}^{-1}, \quad (7.4.7)$$

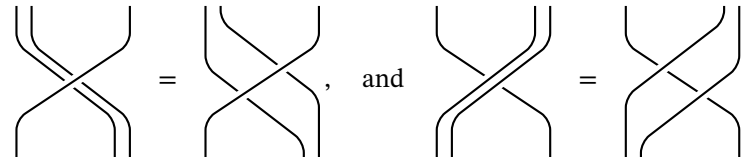
for all $f : A \rightarrow A'$ and $g : B \rightarrow B'$, can be expressed in the graphical notation as



$$(7.4.8)$$

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.

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 $B \rightarrow A \otimes B \otimes C$:

$$(\sigma_{A,B} \otimes \text{id}_B) \circ (\text{id}_A \otimes f) = (\text{id}_A \otimes \sigma_{B,C}^{-1}) \circ (f \otimes \text{id}_B) \quad (7.4.12)$$

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



$$(7.4.13)$$

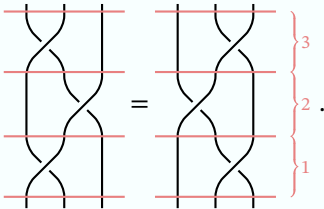
and we can see that these are spatially isotopic, we can just slide the B wire over the f morphism to the other side.

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 $\sigma_{A,B} \otimes \text{id}_C$, region 2 then leaves B alone (which is now the left most wire) and swaps A and C , so it is $\text{id}_B \otimes \sigma_{A,C}$, and region 3 swaps B and C leaving A alone, so it is $\sigma_{B,C} \otimes \text{id}_A$. Doing the same for the diagram on the right we get the equation

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

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

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

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 $\sigma_{A,B}^{-1} = \sigma_{B,A}$.

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



$$(7.5.3)$$

Note the subtle difference to the identity $\sigma_{A,B}^{-1} \circ \sigma_{A,B}$ 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

$$\begin{array}{c} \text{Diagram 1} \end{array} = \begin{array}{c} \text{Diagram 2} \end{array} = \begin{array}{c} \text{Diagram 3} \end{array}. \quad (7.5.5)$$

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 $g \cdot (v \otimes u) = (g \cdot v) \otimes (g \cdot u)$, 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 $f: V \rightarrow W$ 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 $u \otimes v \mapsto v \otimes u$.

^aSee 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 space and consider a bounded linear map $f : H \rightarrow H$. We define the **adjoint**^a of this map, $f^\dagger : H \rightarrow H$, to be the unique map such that $\langle u | f(v) \rangle = \langle f^\dagger(u) | v \rangle$ for all $u, v \in H$.

^aNot to be confused with adjoint functors.

The adjoint has the following properties:

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

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

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

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 $(-)^{\dagger} : \mathbf{Hilb} \rightarrow \mathbf{Hilb}$ 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 $\text{id}_H^{\dagger} = \text{id}_H = \text{id}_{H^{\dagger}}$, satisfying the requirement that $F\text{id}_A = \text{id}_{FA}$, 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 $v, w : \mathbb{C} \rightarrow H$, that is kets $|v\rangle$ and $|w\rangle$. We can turn v into an effect $v^{\dagger} : H \rightarrow \mathbb{C}$. 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

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

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

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

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 $\langle v|$. 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 \mathbf{C} be a category. A **dagger** on \mathbf{C} is an involutive contravariant endofunctor $(-)^{\dagger} : \mathbf{C} \rightarrow \mathbf{C}$ 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 $(-)^{\dagger} : \mathbf{Set} \rightarrow \mathbf{Set}$. 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 $(-)^{\dagger} : \mathbf{Set}(A, B) \rightarrow \mathbf{Set}(B, A)$ 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. \square

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 $R : A \rightarrow B$ then the converse $R^{\dagger} : B \rightarrow A$ is

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

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, $\text{id}_A = \{(a, a) \mid a \in A\}$. We have

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

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

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

The converse of this is

$$(S \circ R)^{\dagger} = \{(c, a) \mid (a, c) \in (S \circ R)\} \quad (8.3.7)$$

$$= \{(c, a) \mid \exists b \in B \text{ such that } aRb \text{ and } bSc\} \quad (8.3.8)$$

$$= \{(c, a) \mid \exists b \in B \text{ such that } bR^{\dagger}a \text{ and } cS^{\dagger}b\} \quad (8.3.9)$$

$$= \{(c, a) \mid \exists b \in B \text{ such that } cS^{\dagger}b \text{ and } bR^{\dagger}a\} \quad (8.3.10)$$

$$= R^{\dagger} \circ S^{\dagger}. \quad (8.3.11)$$

So the converse is indeed a contravariant functor. \square

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 $f : H \rightarrow K$ 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

$$\langle f^\dagger(k) | h \rangle_H = \langle k | f(h) \rangle_K \quad (8.3.13)$$

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 **Vect**_ℓ. To see this we first note that for **Vect**_ℓ and two given vector spaces V and W the set **Vect**_ℓ(V, W) can be made into a vector space by defining pointwise addition, that is given $\varphi, \psi \in \mathbf{Vect}_\ell(V, W)$ define $\varphi + \psi \in \mathbf{Vect}_\ell(V, W)$ to be the linear map

$$(\varphi + \psi)(v) = \varphi(v) + \psi(v). \quad (8.3.14)$$

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

$$(z\varphi)(v) = z\varphi(v). \quad (8.3.15)$$

It is easy to check that these definitions make **Vect**_ℓ(V, W) a vector space.

Now, suppose there was a dagger functor $(-)^\dagger : \mathbf{Vect}_\ell \rightarrow \mathbf{Vect}_\ell$. This gives a mapping of morphisms $\mathbf{Vect}_\ell(V, W) \rightarrow \mathbf{Vect}_\ell(W, V)$. Specifically, choosing $W = \ell$ we have a mapping $\mathbf{Vect}_\ell(V, \ell) \rightarrow \mathbf{Vect}_\ell(\ell, V)$. However, if V is infinite dimensional then $\mathbf{Vect}_\ell(\ell, V)$ has strictly smaller dimension than $\mathbf{Vect}_\ell(V, \ell)$, intuitively there are more ways to squash V down into ℓ than there are to expand ℓ to fill the infinite dimensions of V . However, this means that we cannot have a bijection $\mathbf{Vect}_\ell(V, W) \rightarrow \mathbf{Vect}_\ell(W, V)$ for all vector spaces V and W , as since $\mathbf{Vect}_\ell(V, W)$ and $\mathbf{Vect}_\ell(W, V)$ 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 dimension¹.

¹note 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).

Finite dimensional vector spaces, **FVect**_ℓ, don't have this problem, so we *can* define a dagger $(-)^\dagger : \mathbf{FVect}_\ell \rightarrow \mathbf{FVect}_\ell$, 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.

8.3.4 More Examples

Example 8.3.16

- Any monoid, M , with an involutive map $f : M \rightarrow M$ 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 $F, G : \mathbf{C} \rightarrow \mathbf{D}$ 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 $A \in \text{Ob}(\mathbf{D})$ is ζ_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 $f : A \rightarrow B$ in a dagger category is

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

^asometimes 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:

$$\begin{array}{c} B \\ | \\ \boxed{f} \\ | \\ A \end{array} \xrightarrow{(-)^\dagger} \begin{array}{c} A \\ | \\ \boxed{f^\dagger} \\ | \\ B \end{array} \quad (8.5.1)$$

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:

$$\begin{array}{c} B \\ | \\ \boxed{f} \\ | \\ A \end{array} \xrightarrow{(-)^\dagger} \begin{array}{c} A \\ | \\ \boxed{f}^\dagger \\ | \\ B \end{array} \quad (8.5.2)$$

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

$$\begin{array}{c} A \\ | \\ \triangle a \end{array} \xrightleftharpoons[(-)^\dagger]{(-)^\dagger} \begin{array}{c} \triangle a \\ | \\ A \end{array} \quad (8.5.3)$$

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

$$\langle a|b \rangle := \begin{array}{c} \triangle a \\ | \\ \triangle b \end{array} = \begin{array}{c} \triangle a \\ | \\ \triangle b \end{array} \quad (8.5.4)$$

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

$$\langle a|b \rangle := a \circ b. \quad (8.5.5)$$

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 $f : H \rightarrow K$ and $g : J \rightarrow L$. Then we have a morphism $f \otimes g : H \otimes J \rightarrow K \otimes L$. 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 $(f \otimes g)^\dagger : K \otimes L \rightarrow H \otimes J$ satisfying

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

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

$$\langle v' \otimes w' | (f \otimes g)(v \otimes w) \rangle_{K \otimes L} = \langle v' \otimes w' | f(v) \otimes g(w) \rangle_{K \otimes L} \quad (8.6.4)$$

$$= \langle v' | f(v) \rangle_K \langle w' | g(w) \rangle_L \quad (8.6.5)$$

$$= \langle f^\dagger(v') | v \rangle_H \langle g^\dagger(w') | w \rangle_J \quad (8.6.6)$$

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

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 $\lambda_H : I \otimes H \rightarrow H$. The dagger of this is defined as the unique morphisms such that

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

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

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

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 $\alpha_{H,J,K} : (H \otimes J) \otimes K \rightarrow H \otimes (J \otimes K)$. Then $\alpha_{H,J,K}^\dagger$ is defined as the unique morphism such that

$$\langle \alpha_{H,J,K}^\dagger(u \otimes (v \otimes w)) | (u' \otimes v') \otimes w' \rangle_{(H \otimes J) \otimes K} \quad (8.6.9)$$

$$= \langle u \otimes (v \otimes w) | \alpha_{H,J,K}((u' \otimes v') \otimes w') \rangle_{H \otimes (J \otimes K)} \quad (8.6.10)$$

$$= \langle u \otimes (v \otimes w) | (u' \otimes (v' \otimes w')) \rangle_{H \otimes (J \otimes K)}. \quad (8.6.11)$$

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

$$\langle \alpha_{H,J,K}^{-1}(u \otimes (v \otimes w)) | (u' \otimes v') \otimes w' \rangle_{(H \otimes J) \otimes K} \quad (8.6.12)$$

$$= \langle (u \otimes v) \otimes w | (u' \otimes v') \otimes w' \rangle_{(H \otimes J) \otimes K} \quad (8.6.13)$$

$$= \langle (u \otimes v) | u' \otimes v' \rangle_{H \otimes J} \langle w | w' \rangle_K \quad (8.6.14)$$

$$= \langle u | u' \rangle_H \langle v | v' \rangle_J \langle w | w' \rangle_K \quad (8.6.15)$$

$$= \langle u | u' \rangle_H \langle v \otimes w | v' \otimes w' \rangle_{J \otimes K} \quad (8.6.16)$$

$$= \langle u \otimes (v \otimes w) | u' \otimes (v' \otimes w') \rangle_{H \otimes (J \otimes K)}. \quad (8.6.17)$$

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

Rel is also a symmetric monoidal dagger category.

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 $\cdot : M \times M \rightarrow M$.

A **magma homomorphism**, $(M, \cdot_M) \rightarrow (N, \cdot_N)$, is a function $f : M \rightarrow N$ 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

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c \quad (\text{A.1.3})$$

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**, $(M, \cdot_M, e_M) \rightarrow (N, \cdot_N, e_N)$, is a function $f : M \rightarrow N$ 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**, $(G, \cdot_G, e_G, -^1) \rightarrow (H, \cdot_H, e_H, -^1)$, is a function $f : G \rightarrow H$ 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 $\varphi : G \times S \rightarrow S$ 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 $g \cdot s$ 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 $\psi : S \times G \rightarrow S$ 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 $\rho : G \times V \rightarrow V$. Alternatively, $\rho : V \rightarrow \text{GL}(V)$ is a homomorphism and G acts on V through $g \cdot v = \rho(g)v$.

Note that

$$\begin{aligned} \text{GL}(V) &:= \{\text{invertible linear maps on } V\} \\ &\cong \{n \times n \text{ matrices with entries in } \mathbb{k}\} =: \text{GL}(n, \mathbb{k}) \quad (\text{A.1.8}) \end{aligned}$$

where $n = \dim V$ and V is a vector space over \mathbb{k} .

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, $+$: $R \times R \rightarrow R$ and \cdot : $R \times R \rightarrow R$ such that

- $(R, +, 0, -)$ is an Abelian group.
- $(R, \cdot, 1)$ is a monoid.
- Multiplication distributes over addition:

$$a \cdot (b + c) = a \cdot b + a \cdot c, \quad \text{and} \quad (a + b) \cdot c = a \cdot c + b \cdot c \quad (\text{A.1.10})$$

where we take multiplication to have higher operator precedence than addition.

A **ring homomorphism**, $(R, +_R, -_R, \cdot_R, 1_R, 0_R) \rightarrow (S, +_S, -_S, \cdot_S, 1_S, 0_S)$ is a function $f : R \rightarrow S$ 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**dentify.

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 **negatives**, $-g$ 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 $\cdot : R \times M \rightarrow M$ 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, $\cdot : M \times R \rightarrow M$.

Note that if R is commutative then left and right modules are equivalent.

Definition A.1.14 — Vector Space Let \mathbb{k} be a field. A **vector space** is a \mathbb{k} -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 $m \in M$

$$(r \cdot m) \cdot s = r \cdot (m \cdot s). \quad (\text{A.1.16})$$

Definition A.1.17 If M and N are left R -modules then an **R -linear map**, or **R -module homomorphism**, is a map $f : M \rightarrow N$ such that

$$f(r \cdot m + r' \cdot m') = r \cdot f(m) + r' \cdot f(m') \quad (\text{A.1.18})$$

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 \mathbb{k} be a field and A a \mathbb{k} -vector space. Then A is an **algebra** if it is equipped with a binary operation $\cdot : A \times A \rightarrow A$ satisfying the following for all $u, v, w \in A$ and $a, b \in \mathbb{k}$:

- 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

$$u \cdot (v \cdot w) = (u \cdot v) \cdot w \quad (\text{A.1.20})$$

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** $(P, \leq_P) \rightarrow (Q, \leq_Q)$, is a function $f : P \rightarrow Q$ 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

- $\emptyset \in \mathcal{T}$;
- $X \in \mathcal{T}$;
- 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**, $(X, \mathcal{T}_X) \rightarrow (Y, \mathcal{T}_Y)$, is a function $f: X \rightarrow Y$ such that for all $U \in \mathcal{T}_Y$ we have $f^{-1}(U) \in \mathcal{T}_X$ where

$$f^{-1}(U) := \{x \in X \mid \exists y \in U \text{ such that } y = f(x)\} \subseteq X. \quad (\text{A.3.3})$$

Definition A.3.4 — Homotopy Let X and Y be topological spaces. A **homotopy** between continuous functions $f: X \rightarrow Y$ and $g: X \rightarrow Y$ is a continuous function $H: X \times [0, 1] \rightarrow Y$ 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 $t \in [0, 1]$.

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](https://doi.org/10.48550/ARXIV.1612.09375).
- [3] B. Milewski. *Category Theory for Programmers*. 2019. URL: <https://github.com/hmemcpy/milewski-ctfp-pdf>.

Index

Symbols

$\pi/4$ -ZX calculus, 10
 π -copy rule, 7

A

Abelian group, 89
adjoint, 27, 79, 83
algebra, 91
antisymmetric, 91
arrow, *see* morphism
associative algebra, 91
associativity, 18
associator, 45
automorphism, 30

B

basis, 25
bialgebra rule, 7
bifunctor, 41
bilinear map, 42
bimodule, 90
 homomorphism, 91
bounded linear map, 27
braided monoidal category, 71
braided monoidal dagger category, 85
braiding, 71

C

canonical, 82
Cat, 36
categorification, 47
category, 17
codomain, 30
colour change rule, 6
commutative diagram, 30
commute, 30
completeness, 56
component, 25

composition, 17
concrete category, 19, 35
continuous function, 92
contravariant functor, 34
converse relation, 81
coproduct, 48
copy rule, 6
covariant functor, 34
CRing, 28

D

dagger, 80
 dagger
 Hilb, 80
dagger category, 80
denotational semantics, 12
diagram, 29
dimension, 25
discrete category, 37
domain, 30, 90
dual space, 28

E

effect, 57
endofunctor, 36
endomorphism, 30
entangled state, 58
epic, *see* epimorphism
epimorphism, 31
equivalence, 36
equivariant map, 39
essentially surjective on objects, 36

F

faithful functor, 36
Field, 28
field, 90
finite-dimensional, 25

forgetful functor, 35
 free functor, 35
 Frobenius rule, 5
 full functor, 36
 functor, 34
 functor category, 38
 fusion rule, 5

G

graphical rule, 4
 group, 88
 Abelian, 89
 commutative, *see* Abelian group
 homomorphism, 88
 group action, 89
 groupoid, 29
Grp, 28

H

Hadamard, 6
 matrix, 8
 hexagon equations, 71
Hilb
 as a dagger category, 82
 as a monoidal category, 50
 as a symmetric monoidal category, 73
 Hilbert space, 26
 hom set, 17
 homotopy, 92

I

identity, 18
 identity morphism, 17
 identity rule, 6
 initial object, 48
 inner product, 26
 inner product space, 26
 integral domain, 90
 involution, 80
 isometry, 83
 isomorphic, 31
 isomorphism, 30
 isotopy, 92

J

joint state, 57

K

ket, 27

L

left scalar multiplication, 64
 left unitor, 45
 limit, 39
 linear map, 23
 linearly independent, 24
 locally small category, 18

M

magma, 88
 commutative, 89
 homomorphism, 88
 map, *see* morphism
Mat_k, 25
 module, 90
 homomorphism, 91
Mon, 28
 mono, *see* monomorphism
 monoid, 88
 commutative, 89
 homomorphism, 88
 monoid rule, 5
 monoidal category, 44
 monoidal dagger category, 85
 monoidal product, 44
 monomorphism, 31
 monotone function, 91
 morphism, 17

N

natural isomorphism, 37
 natural transformation, 37
 naturally isomorphic, 37

O

object, 17
 oidification, 29
 operational semantics, 12
 order-preserving function, 91
 orthogonal, 27
 orthonormal, 27

P

partial isometry, 83
 partial order, 91
 partially ordered set, 91
 pentagon equation, 45
 planar isotopic, 55
 poset, 91
 positive, 83

product category, 40
 product state, 58
 projection, 83

R

$R\text{-Mod}$, 28
 reflexive, 91
Rel, 21
 as a dagger category, 81
 as a monoidal category, 50
 as a symmetric monoidal category, 72
 relation, 20
 relation composition, 21
 representation, 89
 rg, 90
 rig, 90
 right R -module, 90
 right unitor, 45
Ring, 28
 ring, 89
 commutative, 89
 homomorphism, 89
 with identity, 89
 with unity, 89
 rng, 89

S

scalar, 61
 scalar multiplication, 23
 scalar rule, 7
 self-adjoint, 83
 semigroup, 88
 commutative, 89
 homomorphism, 88
 semiring, *see* rig
Set, 18
 as a monoidal category, 48
 as a symmetric monoidal category, 72
 small category, 18
 soundness, 56
 source, *see* domain
 spatial isotopic, 73
 spider, 4
 standard model, 7
 state, 56
 symmetric monoidal category, 76
 symmetric monoidal dagger category, 85

T

target, *see* codomain
 tensor product, 42, 44
 of Hilbert spaces, 43
 of linear maps, 43
 tensor product functor, 44
 terminal object, 48
 topological space, 92
 topology, 92
 totally ordered set, 91
 transitive, 91
 triangle equation, 45

U

unit, 90
 unit object, 44
 unitary, 83
 unitor, 45

V

Vect _{\mathbb{k}} , 24
 as a monoidal category, 50
 vector addition, 23
 vector space, 23, 90

Z

zero-divisor, 90
 ZX calculus, 3