

5

Hilbert Space from Diagrams

I would like to make a confession which may seem immoral: I do not believe absolutely in Hilbert space any more.

– *John von Neumann, letter to Garrett Birkhoff, 1935*

We have now seen how processes described by string diagrams already exhibit some quantum-like features. It is natural to ask how much extra work is needed to go from string diagrams to Hilbert spaces and linear maps, the mathematical tools von Neumann used to formulate quantum theory in the late 1920s. The answer is: not that much.

We start by considering what it takes for two processes to be equal. In many process theories, it suffices for them to agree on a relatively small number of states. This feature leads very naturally to the notion of *basis*, and we can use adjoints to identify a particularly handy type of basis, called an *orthonormal basis* (ONB). When all types admit a basis, any process can be completely described by a collection of numbers called its *matrix*.

Now, such a matrix identifies a particular process uniquely, but for any matrix to represent a process we need to add a bit more structure. Therefore, we allow processes with the same input/output wires to be combined into one, or *summed* together. If a process theory admits string diagrams, has ONBs for every type, and has sums of processes, we can describe sums, sequential composition, parallel composition, transpose, conjugate, and adjoint all in terms of operations on matrices. We call this the *matrix calculus* of a process theory.

Thus, by adding ONBs and sums, we have very nearly recovered the full power of *linear algebra*, but with the added generality that the numbers λ are still very unrestricted (in particular, they need not be the elements of some field like the real or complex numbers). In fact, a matrix calculus for relations makes perfect sense, where the numbers are booleans.

The final step towards Hilbert spaces and linear maps consists of requiring the numbers of the process theory to be the *complex numbers*. Thus, we define **linear maps** as the process theory admitting string diagrams where:

1. every type has a finite ONB;
2. for any $n \in \mathbb{N}$ there is a type with an ONB of size n ;
3. processes of the same type can be summed; and
4. the numbers are the complex numbers.

The system-types in this process theory are then called *Hilbert spaces*. For those familiar with the Hilbert space formalism of quantum theory, the most notable thing is the absence of any reference to the *tensor product* of Hilbert spaces, nor to *(multi-)linearity* of maps. The reason for this is that the language of string diagrams gives us these for free!

Our presentation of Hilbert spaces and linear maps is quite different from the ‘bottom-up’ presentation one usually encounters. There, one typically starts with small things, namely vectors, then defines special sets of vectors called vector spaces, and specialises these to Hilbert spaces. Then, to turn this into a process theory, one puts in an awful lot of work defining linear maps, bilinear maps, composition and tensor product using a whole bag of tools from set theory and algebra. These set-theoretic definitions are intrinsically *reductionist* in spirit: they are all about understanding bigger things in terms of smaller things. This is much like the particle physicist’s dream to have a ‘theory of everything’, which explains the world entirely in terms of its smallest parts. By contrast, in our presentation a thing is understood in terms of how it interacts with other things. Thus, it makes sense to adopt a ‘top-down’ approach, where the whole process theory of **linear maps** is defined by first stating how things compose, then filling in the remaining blanks.

Now, this book is about using diagrams for reasoning about quantum processes, so why do we even bother to introduce Hilbert spaces? In fact, in Chapter 8 things like the existence of ONBs and sums will be accounted for in terms of a new diagrammatic primitive, and in Chapter 9 something similar will be done to account for the fact that there are multiple ONBs for a single system. Nonetheless, there are at least three good reasons to introduce Hilbert spaces:

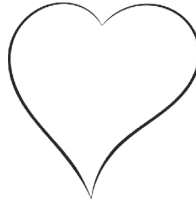
1. Readers who are familiar with Hilbert space quantum theory may find it useful to see the diagrammatic concepts represented in the language of Hilbert spaces, because of their familiarity with that language.
2. Readers who are unfamiliar with Hilbert space quantum theory can translate what they learn in this book into the language used in most other texts on quantum theory.
3. Sometimes the hybrid approach we introduce in this section, combining diagrammatic reasoning with sums, is convenient for calculation. Such a blend has already proved to be very useful in some areas of pure mathematics such as knot theory, where one encounters equations like this one:

$$\begin{array}{c} \diagup \quad \diagdown \\ \diagdown \quad \diagup \end{array} = \lambda \begin{array}{c} | \\ | \end{array} + \lambda^{-1} \begin{array}{c} \cup \\ \cup \end{array}$$

And there is more. Even if one doesn’t care at all about the Hilbert space formulation of quantum theory, one may still ask the question: ‘What can one actually prove using string diagrams?’ When it comes to process equations, the answer to this question is, surprisingly, ‘Exactly what one can prove in Hilbert spaces’!

Using logical terminology, this means there exists a *completeness theorem* for Hilbert spaces with respect to string diagrams. We carefully explain what this means in Section 5.4.1.

On the other hand, string diagrams come with substantially less baggage than Hilbert spaces, so they leave enough room to seek out an alternative to Hilbert space for the formalisation of quantum theory. So, two seemingly unrelated historical developments: abandoning the Hilbert-space formalism for quantum theory (as von Neumann desired), versus embracing non-separability as the crucial feature of quantum theory (as Schrödinger insists), go hand in hand in our tale:



5.1 Bases and Matrices

We now show how some standard notions from linear algebra, most notably bases and matrices, emerge for certain process theories, and how the special processes and operations that we identified in the previous chapters for diagrams then transform into standard linear-algebraic concepts.

Throughout this section we will assume that process theories admit string diagrams and have zero processes for every system-type.

5.1.1 Basis for a Type

Many of the processes that we have studied so far have this property:

$$\left(\text{for all } \begin{array}{c} \downarrow \\ \psi \end{array} : \begin{array}{c} \diagup f \diagdown \\ \downarrow \psi \end{array} = \begin{array}{c} \diagup g \diagdown \\ \downarrow \psi \end{array} \right) \Rightarrow \begin{array}{c} \diagup f \diagdown \\ \downarrow \end{array} = \begin{array}{c} \diagup g \diagdown \\ \downarrow \end{array} \quad (5.1)$$

that is, if two processes do the same thing to all states, then they are equal. In other words, a process is uniquely fixed by what it does to states.

Examples 5.1 Both **functions** and **relations** satisfy (5.1).

For **functions**, in Example 3.35 we saw that the states of A :

$$\begin{array}{c} \downarrow \\ a \end{array} :: * \mapsto a$$

are in a bijective correspondence with the elements $a \in A$. By this correspondence we have:

$$\begin{array}{c} \downarrow \\ \boxed{f} \\ \downarrow \\ \triangle a \end{array} = \begin{array}{c} \downarrow \\ \boxed{g} \\ \downarrow \\ \triangle a \end{array} \iff f(a) = g(a)$$

Thus, we can translate (5.1) as:

$$(\text{for all } a \in A : f(a) = g(a)) \implies f = g$$

which is of course true for any functions f and g .

For **relations**, we saw in Example 3.36 that the states of type A :

$$\begin{array}{c} \downarrow \\ \triangle A' \end{array} :: * \mapsto A'$$

are in a bijective correspondence with the subsets $A' \subseteq A$. However, in the light of (5.1), considering all states $A' \subseteq A$ is overkill. If we just restrict to singletons, we already have:

$$(\text{for all } a \in A : R(a) = S(a)) \implies R = S$$

That is, a relation is fixed by what it does to singletons.

As we just saw for **relations**, it is not always necessary to know what a process does to every state to fix it uniquely. Sometimes it suffices to know how it acts only on a special subset of the states.

Definition 5.2 A *basis* for a type A is a minimal set of states:

$$\mathcal{B} := \left\{ \begin{array}{c} \downarrow \\ \triangle 1 \end{array}, \dots, \begin{array}{c} \downarrow \\ \triangle n \end{array} \right\}$$

such that for all processes f and g :

$$\left(\text{for all } \begin{array}{c} \downarrow \\ \triangle i \end{array} \in \mathcal{B} : \begin{array}{c} \downarrow \\ \boxed{f} \\ \downarrow \\ \triangle i \end{array} = \begin{array}{c} \downarrow \\ \boxed{g} \\ \downarrow \\ \triangle i \end{array} \right) \implies \begin{array}{c} \downarrow \\ \boxed{f} \end{array} = \begin{array}{c} \downarrow \\ \boxed{g} \end{array} \quad (5.2)$$

where ‘minimal’ means that no state can be removed from \mathcal{B} without sacrificing (5.2). The *dimension* $\dim(A)$ of the type A is the minimum size of a basis for A .

Remark* 5.3 For a well-behaved process theory like **relations**, all bases for a particular system will be the same size. In that case, we can just as well define $\dim(A)$ to be the size of any basis. For the theory of **linear maps**, this result is commonly known as the *dimension theorem*.

Exercise 5.4 Show that for a set A in **relations** the singletons:

$$\mathcal{B}_A := \left\{ \left(\begin{array}{c} | \\ \triangle \\ a \end{array} \right) \mid a \in A \right\}$$

form a basis, that is, that no element can be removed from \mathcal{B}_A without losing the property of being a basis. Also show that all bases are of this form and, consequently, that the dimension of a set A in **relations** is its number of elements.

Henceforth, we will use:

$$\left\{ \left(\begin{array}{c} | \\ \triangle \\ i \end{array} \right) \right\}_i$$

as a shorthand for:

$$\left\{ \left(\begin{array}{c} | \\ \triangle \\ 1 \end{array} \right), \dots, \left(\begin{array}{c} | \\ \triangle \\ n \end{array} \right) \right\}$$

The best kinds of bases are those whose states are perfectly distinguishable by testing. That is, if we test the i -th state for being the j -th state, we should get a ‘yes’ outcome with certainty if and only if $i = j$. We give these sets of states (and in particular, bases) a special name.

Definition 5.5 A set of states:

$$\mathcal{A} := \left\{ \left(\begin{array}{c} | \\ \triangle \\ i \end{array} \right) \right\}_i$$

is *orthonormal* if for all i, j we have:

$$\left(\begin{array}{c} \triangle \\ j \\ | \\ \triangle \\ i \end{array} \right) = \delta_i^j \quad (5.3)$$

where δ_i^j is the *Kronecker delta*:

$$\delta_i^j = \begin{cases} \boxed{} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

If \mathcal{A} forms a basis, it is called an *orthonormal basis* (ONB).

Recall from Section 4.3.3 that we can think of the inner product in (5.3) as measuring the ‘overlap’ between states. In that case, we can think of an ONB as a basis whose elements don’t overlap, as in the following example.

Example 5.6 We saw in Example 4.50 that the inner product of two states in **relations** is 1 if and only if they intersect. Since the intersection of any two (different) singletons is empty, the unique bases in **relations** from Exercise 5.4 are ONBs:

$$\begin{array}{c} \triangleup b \\ \hline \triangleleft a \end{array} = \begin{cases} \square & \text{if } a = b \\ \emptyset & \text{if } a \neq b \end{cases}$$

Some ONBs aren't unique.

Example 5.7 In Section 5.3, we will see that **linear maps** admit many different ONBs for a single system-type, and there is no unique choice of 'preferred' ONB. This fact is very important for many quantum phenomena.

Some ONBs are even invisible.

Example 5.8 Since for all states ψ and ϕ of a system A , we have:

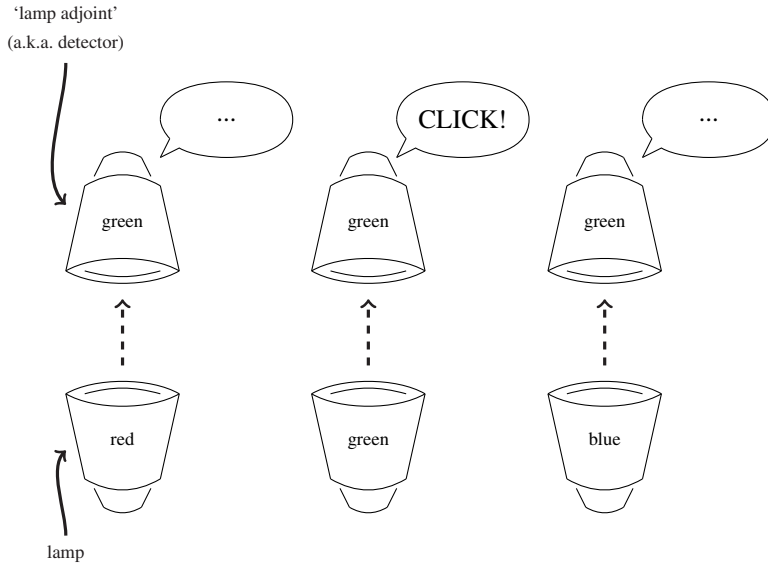
$$\begin{array}{c} \triangleleft \psi \\ \square \end{array} = \begin{array}{c} \triangleleft \phi \\ \square \end{array} \Rightarrow \begin{array}{c} \triangleleft \psi \end{array} = \begin{array}{c} \triangleleft \phi \end{array}$$

the empty diagram forms an ONB for the 'no wire'-type, since in this case orthonormality just means:

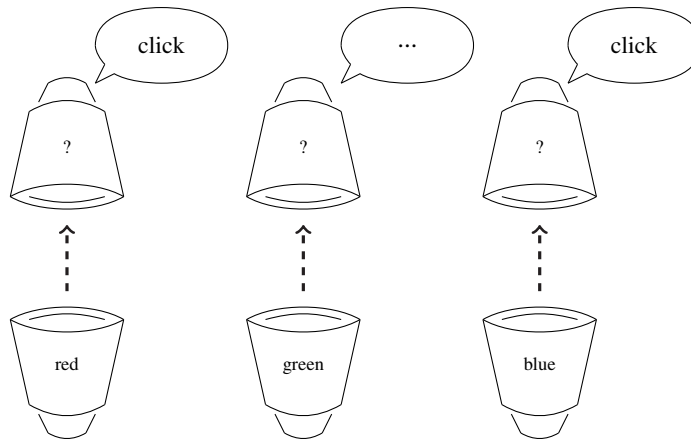
$$\begin{array}{c} \square \\ \square \end{array} = \square$$

Other ONBs are full of colour.

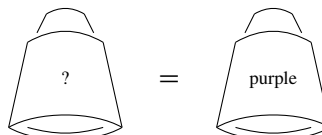
Example 5.9 Consider a process theory of **lamps&detectors**, where states are lamps producing light of a certain colour, and effects are detectors that click when they detect light of a certain colour. Numbers arise when composing a lamp with a detector. The number 0 means 'no click', i.e. nothing detected, and 1 means 'loudest click', i.e. maximal intensity detected. Our interpretation of adjoints dictates that the adjoint of a lamp is the detector for light of the same colour. Then red, green, and blue lamps are 'orthonormal', because they will never detect each other's light:



and they also form a basis. For example, suppose we have some unknown detector, and this happens:



then we can conclude that:



The minimality condition in Definition 5.2 can be tricky to verify for a generic basis. Fortunately, in the case of an ONB, this condition is automatic.

Proposition 5.10 If an orthonormal set of states of type *A* satisfies (5.2) for all pairs of processes, then it must be minimal, and hence an ONB.

Proof Let:

$$\mathcal{A} = \left\{ \begin{array}{c} \downarrow \\ \boxed{i} \\ \nabla \end{array} \right\}_i$$

be an orthonormal set of states satisfying (5.2). Suppose it is not minimal; that is, there is some state i such that:

$$\mathcal{A}' := \mathcal{A} - \left\{ \begin{array}{c} \downarrow \\ \boxed{i} \\ \nabla \end{array} \right\}$$

still satisfies (5.2). First, note that effect i cannot be equal to the zero effect (depicted here as $\boxed{0}$ to avoid confusion) because:

$$\begin{array}{c} \boxed{i} \\ \downarrow \\ \boxed{i} \\ \nabla \end{array} = 1 \neq 0 = \begin{array}{c} \boxed{0} \\ \downarrow \\ \boxed{i} \\ \nabla \end{array}$$

However, since the states in \mathcal{A} are orthonormal, for all $j \neq i$ we have:

$$\begin{array}{c} \boxed{i} \\ \downarrow \\ \boxed{j} \\ \nabla \end{array} = 0 = \begin{array}{c} \boxed{0} \\ \downarrow \\ \boxed{j} \\ \nabla \end{array}$$

But then by (5.2) it follows that

$$\begin{array}{c} \boxed{i} \\ \downarrow \\ \nabla \end{array} = \begin{array}{c} \boxed{0} \\ \downarrow \\ \nabla \end{array}$$

which is a contradiction. □

Now is probably a good time for a quick word of warning about ONBs.

⚠ Warning 5.11 In Section 3.4.3 we introduced the notion of processes being equal ‘up to a number’ and showed that the \approx -relation plays well with diagrams. On the other hand, it does not play well with ONBs. Just because we prove that for all i :

$$\begin{array}{c} \downarrow \\ \boxed{f} \\ \nabla \end{array} \approx \begin{array}{c} \downarrow \\ \boxed{g} \\ \nabla \end{array} \quad (5.4)$$

it does not follow that:

$$\begin{array}{c} \downarrow \\ \boxed{f} \\ \nabla \end{array} \approx \begin{array}{c} \downarrow \\ \boxed{g} \\ \nabla \end{array}$$

In order for this to be true, each of the instances of equation (5.4) should hold up to the same number, otherwise we may not be able to find a single pair of numbers λ, μ such that

$\lambda f = \mu g$. For example, if f and g are effects, this is just saying f and g are non-zero on the same set of ONB states.

For most uses of bases in this book we will use ONBs, with one notable exception: *tomography* (cf. Section 7.4). To account for this, we will prove a number of results in the following sections for the general case of non-orthonormal bases as well. It will also be convenient (or sometimes even necessary) to choose a basis consisting of self-conjugate states. If a basis is self-conjugate, then the corresponding effects are also self-conjugate. We denote these self-conjugate states and effects as follows:

$$\begin{array}{c} \downarrow \\ \triangleleft i \end{array} := \begin{array}{c} \downarrow \\ \square i \end{array} = \begin{array}{c} \downarrow \\ \triangleright i \end{array} \quad \text{and} \quad \begin{array}{c} \uparrow \\ \triangle i \end{array} := \begin{array}{c} \uparrow \\ \square i \end{array} = \begin{array}{c} \uparrow \\ \triangleright i \end{array}$$

Example* 5.12 In linear algebra, self-conjugate ONBs are those whose matrices consist only of real numbers. The canonical example is:

$$\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \dots, \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

We will see in Section 5.2.3 that we can turn any ONB into a self-conjugate one just by choosing the correct cups/caps (which are non-unique in general). An example where it becomes necessary to deal with a non-self-conjugate basis is when studying multiple distinct bases for the same system. It may not, in general, be possible to make a single choice of cap/cup to make all bases simultaneously self-conjugate.

Example* 5.13 In quantum computing, the X -basis, Y -basis, and Z -basis for qubits have the property that only two out of three can be made simultaneously self-conjugate.

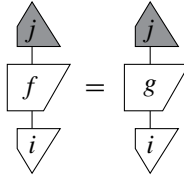
5.1.2 Matrix of a Process

When we have basis states around, we can often prove things about processes by proving things about states. However, because we have adjoints, we also have the associated basis effects, so we can actually do better. To prove equality of processes, it suffices to just look at numbers.

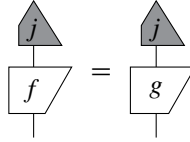
Theorem 5.14 Suppose that \mathcal{B} is a basis for A and that \mathcal{B}' is a basis for B . Then for all f and g with input type A and output type B :

$$\left(\text{for all } \begin{array}{c} \downarrow \\ \triangleleft i \end{array} \in \mathcal{B}, \begin{array}{c} \downarrow \\ \triangleright j \end{array} \in \mathcal{B}' : \begin{array}{c} \begin{array}{c} \triangleleft j \\ f \\ \triangleleft i \end{array} = \begin{array}{c} \triangleleft j \\ g \\ \triangleleft i \end{array} \end{array} \right) \Rightarrow \begin{array}{c} \downarrow \\ \triangleleft f \end{array} = \begin{array}{c} \downarrow \\ \triangleleft g \end{array} \quad (5.5)$$

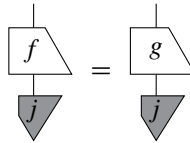
Proof We can prove this using one basis at a time. First, take any state j in \mathcal{B}' . Then, for all states i in \mathcal{B} , if we have:



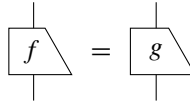
then, since \mathcal{B} is a basis, it follows that:



and applying the adjoint to both sides, we have:



The above equation holds for all states $j \in \mathcal{B}'$, so since \mathcal{B}' is a basis:



Applying the adjoint to both sides again, we conclude that $f = g$. \square

Exercise 5.15 Show that the converse to Theorem 5.14 is also true if in addition we require \mathcal{B} and \mathcal{B}' to be minimal, that is, whenever condition (5.5) holds, then (5.2) holds both for \mathcal{B} and \mathcal{B}' .

When the bases in Theorem 5.14 are orthonormal, we give a familiar name to the numbers that uniquely identify a process.

Definition 5.16 The numbers:

$$\mathbf{f} := \left(f_i^j \mid \begin{array}{c} \downarrow \\ i \end{array} \in \mathcal{B}, \begin{array}{c} j \\ \downarrow \end{array} \in \mathcal{B}' \right)$$

where \mathcal{B} and \mathcal{B}' are ONBs and:

$$f_i^j := \begin{array}{c} j \\ \downarrow \\ f \\ \downarrow \\ i \end{array} \quad (5.6)$$

is called the *matrix* of f . The numbers f_i^j are called the *matrix entries*.

Usually we will use the same notation for a process f and its matrix, but when we wish to distinguish the two, as above, we will use a boldface notation \mathbf{f} for the matrix.

In school, you may have seen a matrix written this way:

$$\begin{pmatrix} f_1^1 & f_2^1 & \cdots & f_m^1 \\ f_1^2 & f_2^2 & \cdots & f_m^2 \\ \vdots & \vdots & \ddots & \vdots \\ f_1^n & f_2^n & \cdots & f_m^n \end{pmatrix}$$

Note how for each matrix entry we give the row index (which arises from an output basis element) as a superscript and the column index (which arises from an input basis element) as a subscript. This ‘tensor-style’ notation (cf. Section* 3.6.1) will come in handy when we have multiple input/output wires.

The matrix of a process with input A and output B will have $\dim(A)$ columns and $\dim(B)$ rows. In Example 5.8 we saw that the ‘no wire’ type has a one-element basis. So, states give $n \times 1$ matrices, called *column vectors*, and effects give $1 \times m$ matrices, called *row vectors*:

$$\begin{array}{c} \downarrow \\ \psi \end{array} \leftrightarrow \begin{pmatrix} \psi^1 \\ \psi^2 \\ \vdots \\ \psi^n \end{pmatrix} \quad \begin{array}{c} \uparrow \\ \phi \end{array} \leftrightarrow (\phi_1 \quad \phi_2 \quad \cdots \quad \phi_m)$$

Numbers, of course, give 1×1 matrices:

$$(\lambda)$$

but we typically don’t bother to write them that way.

Though they are typically associated with linear maps, matrices are actually more general. For example, they provide a convenient alternative representation for relations.

Example 5.17 We saw earlier that each type in **relations** has a unique ONB given by the singletons, and in Example 3.36, we showed that there are only two numbers:

$$0 := \emptyset \text{ (a.k.a. ‘impossible’)} \quad 1 := \boxed{} \text{ (a.k.a. ‘possible’)}$$

Then we have:

$$\begin{array}{c} \triangleup b \\ \square R \\ \triangle a \end{array} = \begin{cases} 1 & \text{if } R :: a \mapsto b \\ 0 & \text{otherwise} \end{cases}$$

Clearly these numbers fully characterise R since they identify precisely the pairs (a, b) such that $R :: a \mapsto b$ by assigning the number 1 to them. We can label the columns of R ’s matrix

by the elements of A , and the rows by the elements of B . Then, we see a 1 whenever the elements of the given row and column are related, and 0 everywhere else:

$$R :: \begin{cases} a_1 \mapsto b_4 \\ a_2 \mapsto b_2 \\ a_2 \mapsto b_3 \\ a_3 \mapsto b_4 \end{cases} \quad \leftrightarrow \quad \begin{matrix} & a_1 & a_2 & a_3 \\ \begin{matrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \end{matrix}$$

This matrix is sometimes called an *adjacency matrix* of R .

Not only can we represent processes by matrices, but the diagrammatic operations of transpose, conjugate, and adjoint also correspond to familiar operations on matrices.

Theorem 5.18 Let f be a process with associated matrix \mathbf{f} . The matrix of f^\dagger is the *adjoint matrix* \mathbf{f}^\dagger , which is defined as:

$$(\mathbf{f}^\dagger)_i^j := \overline{(\mathbf{f}_j^i)}$$

Proof The matrix entries of f^\dagger are computed as:

$$(f^\dagger)_i^j = \begin{array}{c} \triangle_j \\ | \\ \square_f \\ | \\ \triangle_i \end{array} \stackrel{(4.18)}{=} \begin{array}{c} \triangle_i \\ | \\ \square_f \\ | \\ \triangle_j \end{array} = \overline{\begin{pmatrix} \triangle_i \\ | \\ \square_f \\ | \\ \triangle_j \end{pmatrix}} = \overline{(f_j^i)}$$

where (4.18) is the fact that numbers are self-transposed. □

Note that the above theorem works for any ONB, not just the self-conjugate ones. In contrast, for transposition and conjugation to work correctly on matrices, we need to assume that the ONBs are self-conjugate.

Theorem 5.19 Let f be a process with associated matrix \mathbf{f} for self-conjugate ONBs \mathcal{B} and \mathcal{B}' . The matrix of f^T is the *transposed matrix* \mathbf{f}^T , which is defined as:

$$(\mathbf{f}^T)_i^j := \mathbf{f}_j^i$$

The matrix of \bar{f} is the *conjugate matrix* $\bar{\mathbf{f}}$, which is defined as:

$$\bar{\mathbf{f}}_i^j := \overline{(\mathbf{f}_i^j)}$$

Proof The matrix entries of the transpose of f are computed as:

$$\begin{array}{c} \triangle_j \\ | \\ \square_f \\ | \\ \triangle_i \end{array} = \begin{array}{c} \triangle_j \\ | \\ \square_f \\ | \\ \triangle_i \end{array} \stackrel{(*)}{=} \begin{array}{c} \triangle_i \\ | \\ \square_f \\ | \\ \triangle_j \end{array} \quad (5.7)$$

and for the matrix entries of the conjugate we have:

$$\begin{array}{c} \triangleup_j \\ | \\ \square_f \\ | \\ \triangle_i \end{array} \stackrel{(*)}{=} \overline{\left(\begin{array}{c} \triangleup_j \\ | \\ \square_f \\ | \\ \triangle_i \end{array} \right)}$$

where the equations marked $(*)$ rely on \mathcal{B} and \mathcal{B}' being self-conjugate. If this were not the case, the resulting matrices would not be in terms of \mathcal{B} and \mathcal{B}' , but rather in terms of their conjugate bases. \square

So the transpose \mathbf{f}^T of a matrix \mathbf{f} is obtained by interchanging the roles of the row and column indices; the conjugate $\overline{\mathbf{f}}$ is obtained by conjugating each of the entries; and the adjoint \mathbf{f}^\dagger consists of applying both of these operations. For that reason, the adjoint matrix is also sometimes called the conjugate-transpose. As in the case of diagrams, the order doesn't matter:

$$\mathbf{f}^\dagger = \overline{(\mathbf{f}^T)} = (\overline{\mathbf{f}})^T$$

Exercise 5.20 Characterise the matrix of a self-adjoint process.

So, we can already treat several operations on processes as operations on matrices. However, we haven't yet reached the full power of *matrix calculus*. Theorem 5.14 says that $f = g$ if and only if f and g have the same matrix. In other words, when there exists an f with a particular matrix, it is unique. But then, given some arbitrary matrix, nothing guarantees (yet) that there will always exist an f that has that matrix.

Suppose we fix a bunch of numbers, and write them all in an $n \times m$ matrix:

$$\begin{pmatrix} g_1^1 & g_2^1 & \cdots & g_m^1 \\ g_1^2 & g_2^2 & \cdots & g_m^2 \\ \vdots & \vdots & \ddots & \vdots \\ g_1^n & g_2^n & \cdots & g_m^n \end{pmatrix} \tag{5.8}$$

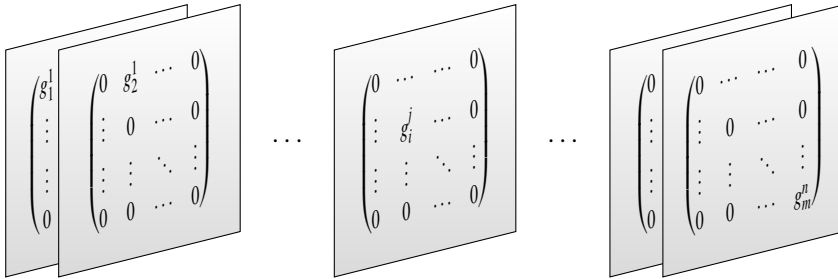
How might we go about obtaining some process g that has this matrix? First, fix ONBs \mathcal{B} and \mathcal{B}' . Then, for any i, j , it is possible to build a process \tilde{g}_{ij} that agrees with g on the i -th input element of \mathcal{B} and the j -th output element of \mathcal{B}' and is zero everywhere else:

$$\begin{array}{c} | \\ \square_{\tilde{g}_i^j} \\ | \end{array} = \begin{array}{c} \diamond_{g_i^j} \\ | \end{array} \begin{array}{c} \triangleup_j \\ | \\ \triangle_i \\ | \end{array}$$

If we compute the matrix of \tilde{g}_i^j , it indeed has precisely one non-zero entry, g_i^j , at the (i, j) -th position:

$$\begin{array}{c} | \\ \hline \tilde{g}_i^j \\ \hline | \end{array} \leftrightarrow \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ \vdots & g_i^j & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

We can define a whole stack of these \tilde{g}_i^j -processes, for all i, j :



Then if we could only ‘overlay’ them somehow, we would get g .

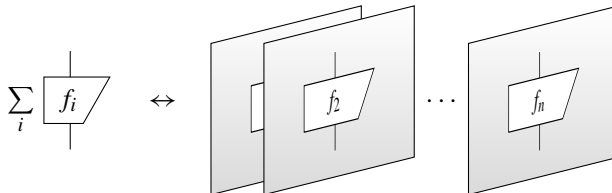
It turns out, for certain process theories, ‘overlaying’ processes makes perfect sense. If we denote this ‘overlaying’ procedure as a *sum* of processes, g can be expressed as:

$$\begin{array}{c} | \\ \hline g \\ \hline | \end{array} := \sum_{ij} \begin{array}{c} \triangle^j \\ \hline g_i^j \\ \hline \triangle^i \end{array} \quad (5.9)$$

Thus, it only remains to make precise what we mean by ‘sums of processes’.

5.1.3 Sums of Processes

Sums are not something you can do for any old processes; for example, what is the sum of two babies? However, for many process theories, it has a perfectly well-defined mathematical meaning. This meaning directly emerges from the intuition behind overlaying diagrams:



Definition 5.21 We say a process theory *has sums* if the following three conditions are satisfied:

- **Condition 1:** For any two processes f, g with the same input and output types, $f + g$ is a process. We always assume that ‘+’ is associative, is commutative, and has a unit given by the zero process:

$$(f + g) + h = f + (g + h) \quad f + g = g + f \quad f + 0 = f = 0 + f$$

and for a set $\{f_i\}_i$ of processes, we write:

$$\sum_i \begin{array}{c} | \\ \hline \diagup f_i \diagdown \\ | \end{array} := \begin{array}{c} | \\ \hline \diagup f_1 \diagdown \\ | \end{array} + \begin{array}{c} | \\ \hline \diagup f_2 \diagdown \\ | \end{array} + \cdots + \begin{array}{c} | \\ \hline \diagup f_N \diagdown \\ | \end{array}$$

- **Condition 2:** Sums *distribute over diagrams*, that is, any time a summation occurs in a diagram, it can be pulled outside:

$$\left(\sum_i \begin{array}{c} | \\ \hline \diagup h_i \diagdown \\ | \end{array} \right) \begin{array}{c} | \\ \hline \diagup g \diagdown \\ | \end{array} = \sum_i \left(\begin{array}{c} | \\ \hline \diagup h_i \diagdown \\ | \end{array} \begin{array}{c} | \\ \hline \diagup g \diagdown \\ | \end{array} \right)$$

- **Condition 3:** Sums preserve adjoints:

$$\left(\sum_i \begin{array}{c} | \\ \hline \diagup f_i \diagdown \\ | \end{array} \right)^\dagger = \sum_i \begin{array}{c} | \\ \hline \diagdown f_i \diagup \\ | \end{array}$$

Note that **Condition 2** subsumes distributivity of sums with respect to parallel and sequential composition, e.g.:

$$\left(\sum_i \begin{array}{c} | \\ \hline \diagup f_i \diagdown \\ | \end{array} \right) \begin{array}{c} | \\ \hline \diagup g \diagdown \\ | \end{array} = \sum_i \left(\begin{array}{c} | \\ \hline \diagup f_i \diagdown \\ | \end{array} \begin{array}{c} | \\ \hline \diagup g \diagdown \\ | \end{array} \right)$$

and:

$$\begin{array}{c} | \\ \hline \diagup g \diagdown \\ | \end{array} \left(\sum_i \begin{array}{c} | \\ \hline \diagup f_i \diagdown \\ | \end{array} \right) = \sum_i \left(\begin{array}{c} | \\ \hline \diagup g \diagdown \\ | \end{array} \begin{array}{c} | \\ \hline \diagup f_i \diagdown \\ | \end{array} \right)$$

An important example of this is *linearity* of maps with respect to states:

$$\left(\sum_i \left(\begin{array}{c} \lambda_i \\ \psi_i \end{array} \right) \right) \begin{array}{c} f \\ \psi_i \end{array} = \sum_i \left(\begin{array}{c} \lambda_i \\ \psi_i \end{array} \right) \begin{array}{c} f \\ \psi_i \end{array}$$

Another one is the inner product. We have full-fledged linearity for states:

$$\left(\sum_i \left(\begin{array}{c} \lambda_i \\ \psi_i \end{array} \right) \right) \begin{array}{c} \phi \\ \psi_i \end{array} = \sum_i \left(\begin{array}{c} \lambda_i \\ \psi_i \end{array} \right) \begin{array}{c} \phi \\ \psi_i \end{array} \quad (5.10)$$

and conjugate-linearity for effects:

$$\left(\sum_i \left(\begin{array}{c} \bar{\lambda}_i \\ \psi_i \end{array} \right) \right) \begin{array}{c} \phi_i \\ \psi_i \end{array} = \sum_i \left(\begin{array}{c} \bar{\lambda}_i \\ \psi_i \end{array} \right) \begin{array}{c} \phi_i \\ \psi_i \end{array} \quad (5.11)$$

Distributivity also helps us derive the matricial counterpart to sums, which unsurprisingly results in something that looks like the linear-algebraic sum of matrices.

Theorem 5.22 Let $\{f_k\}_k$ be processes with associated matrices $\{\mathbf{f}_k\}_k$. The matrix of the process $\sum_k f_k$ is the sum of the matrices $\sum_k \mathbf{f}_k$, where:

$$\left(\sum_k \mathbf{f}_k \right)_i^j := \sum_k (\mathbf{f}_k)_i^j$$

Proof We can use **Condition 2** to compute the matrix of $\sum_k f_k$ as follows:

$$\left(\sum_k \begin{array}{c} \begin{array}{c} j \\ f_k \end{array} \\ i \end{array} \right) = \sum_k \left(\begin{array}{c} \begin{array}{c} j \\ f_k \end{array} \\ i \end{array} \right) = \sum_k (\mathbf{f}_k)_i^j$$

□

If there are multiple summations in a diagram, we can pull them all outside, though we may need to do some re-indexing (as in (*) below):

$$\left(\sum_i \begin{array}{c} g_i \\ f_i \end{array} \right) \stackrel{(*)}{=} \left(\sum_j \begin{array}{c} g_j \\ f_i \end{array} \right) = \sum_i \left(\begin{array}{c} \left(\sum_j \begin{array}{c} g_j \\ f_i \end{array} \right) \end{array} \right) = \sum_{ij} \left(\begin{array}{c} g_j \\ f_i \end{array} \right)$$

where we used:

$$\sum_{ij} \boxed{f_{ij}} \quad \text{as shorthand for} \quad \sum_i \sum_j \boxed{f_{ij}}$$

The order we pull out the sums doesn't matter, so if we always use distinct letters for indexes, we can forget about brackets and write the summation symbols anywhere in the picture:

$$\sum_j \boxed{g_j} \sum_i \boxed{f_i} = \sum_i \sum_j \boxed{g_j} \boxed{f_i} = \sum_j \sum_i \boxed{g_j} \boxed{f_i} = \sum_i \sum_j \boxed{g_j} \boxed{f_i}$$

However, we tend to be boring and stick to writing sums on the left.

Remark 5.23 One can remember all the above rules concerning sums simply by thinking of the sum-symbol as a 'number':



Of course it is not a number in the usual sense, but it can freely move around the diagram, just like a number.

As with ONBs, a quick word of warning about sums is warranted.

Warning 5.24 What we said in Warning 5.11 about the \approx -relation not playing well with ONBs also applies to sums. Just because we prove that for all i :

$$\boxed{f_i} \approx \boxed{g_i} \tag{5.12}$$

it does not follow that:

$$\sum_i \boxed{f_i} \approx \sum_i \boxed{g_i}$$

In order for this to be true, again each of the instances of equation (5.12) should hold up to the same number.

While the existence of sums rules out babies, relations are still hopping along.

Example 5.25 In **relations** sums are unions. Setting:

$$\sum_i R_i := \bigcup_i R_i$$

it can be shown straightforwardly that **Conditions 1–3** are satisfied. Applying it to the two numbers 0 and 1 we obtain:

$$0 + 0 = 0 \quad 0 + 1 = 1 \quad 1 + 0 = 1 \quad 1 + 1 = 1$$

So ‘+’ for numbers in **relations** is the boolean ‘or’ operation. Writing ‘ \cdot ’ for (parallel/sequential) composition of numbers, we also have:

$$0 \cdot 0 = 0 \quad 0 \cdot 1 = 0 \quad 1 \cdot 0 = 0 \quad 1 \cdot 1 = 1$$

That is, we obtain the boolean ‘and’ operation, and **Condition 2** now yields the usual *distributive law* for ‘or’ and ‘and’:

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$

So, in a process theory with sums, numbers always have a ‘plus’ operation as well as a ‘times’ (i.e. composition), with a distributive law between them. Hence they are starting to look a lot more like actual numbers.

Example 5.26 We can regard any natural number n as a number in our process theory. We just take an n -fold sum of 1s (a.k.a. empty diagrams):

$$\langle n \rangle := \underbrace{1 + \cdots + 1}_{n \text{ times}} \quad (5.13)$$

If the numbers of the process theory are the real or complex numbers, then these correspond precisely to the natural numbers (seen as a subset of \mathbb{R} or \mathbb{C}). However, this need not be the case. For instance, if the numbers are booleans as in Example 5.25, all ns are the same:

$$\underbrace{1 + \cdots + 1}_{n \text{ times}} = 1$$

One may even want to consider process theories with *subtraction*, where for every process f , there exists another process $-f$ such that $f + (-f) = 0$. As usual, we can abbreviate $f + (-g)$ as $f - g$. In terms of overlays this can be thought of as a layer that ‘neutralises’ another layer. Note that, by distributivity, this is equivalent to assuming a special number ‘ -1 ’ such that $1 - 1 = 0$. If we include subtraction, the numbers of the process theory form a *ring*. Without subtraction, they form a weaker kind of structure, sometimes called a *unital semiring* or *rig* (because it is a ring without *negative* numbers).

Remark 5.27 The assumption of the existence of additive inverses is actually quite strong. In particular, it rules out the theory of **relations**, because the booleans contain no number that behaves as the additive inverse of 1. Our only two options are 0 and 1, and neither works:

$$1 + 0 = 1 \neq 0 \quad 1 + 1 = 1 \neq 0$$

In the following sections, we will see how every matrix now corresponds to a process, and how composition of processes corresponds to composing the corresponding matrices. All together, this is what we’re aiming for.

Definition 5.28 For a process theory

- that admits string diagrams,
- has a (fixed, self-conjugate) ONB for every system-type, and
- has sums satisfying the conditions in Definition 5.21,

the *matrix calculus* of that process theory refers to the matrices associated with its processes, along with operations for summation, sequential composition, parallel composition, transposition, conjugation, and adjoints of those matrices.

As explained in the proof of Theorem 5.19, we only rely on self-conjugate ONBs to have matricial counterparts to transposition and conjugation.

5.1.4 Processes from Matrices

With sums in hand, we are now able to build processes from matrices, as we discussed at the end of Section 5.1.1.

Theorem 5.29 Fix an ONB \mathcal{B} with m elements and an ONB \mathcal{B}' with n elements. Then, for a collection of numbers g_i^j where i ranges from 1 to m and j ranges from 1 to n , the process:

$$\begin{array}{c} \diagup \\ | \\ \boxed{g} \\ | \end{array} := \sum_{ij} \begin{array}{c} \diagup \\ | \\ \boxed{g_i^j} \\ | \end{array} \begin{array}{c} \diagup \\ | \\ \boxed{j} \\ | \end{array} \begin{array}{c} \diagdown \\ | \\ \boxed{i} \\ | \end{array} \quad (5.14)$$

has the following matrix:

$$\begin{pmatrix} g_1^1 & g_2^1 & \cdots & g_m^1 \\ g_1^2 & g_2^2 & \cdots & g_m^2 \\ \vdots & \vdots & \ddots & \vdots \\ g_1^n & g_2^n & \cdots & g_m^n \end{pmatrix}$$

Proof It suffices to show that the following numbers are equal, for all i, j :

$$\begin{array}{c} \diagup \\ | \\ \boxed{j} \\ | \end{array} \begin{array}{c} \diagup \\ | \\ \boxed{g} \\ | \end{array} \begin{array}{c} \diagdown \\ | \\ \boxed{i} \\ | \end{array} = \begin{array}{c} \diagup \\ | \\ \boxed{g_i^j} \\ | \end{array} \quad (5.15)$$

$$\begin{array}{c} \textcolor{gray}{\triangleup}^j \\ \square^g \\ \triangleleft_i \end{array} = \sum_{kl} \textcolor{gray}{\triangleleft}^l_k = \sum_{kl} \delta_i^k \delta_l^j \textcolor{gray}{\triangleleft}^l_k = \textcolor{gray}{\triangleleft}^j_i$$

- (1) a number,
- (2) an ONB state, and
- (3) an ONB effect.

$$\begin{array}{c} \text{---} \\ | \\ \nabla \\ \psi \end{array} = \sum_i \begin{array}{c} \diamond \\ \psi^i \end{array} \begin{array}{c} \text{---} \\ | \\ \nabla \\ i \end{array}$$

Theorem 5.30 Suppose there exists a basis \mathcal{B} for a type A . Then, another orthonormal set of states:

$$\mathcal{A} := \left\{ \begin{array}{c} | \\ \triangle \\ i \end{array} \right\}_i$$

$$\begin{array}{c} \downarrow \\ \nabla \psi \end{array} = \sum_i \begin{array}{c} \diamond \lambda_i \end{array} \begin{array}{c} \downarrow \\ \nabla i \end{array} \quad (5.16)$$

for all $i \in \mathcal{A}$:

$$f \circ i = g \circ i$$

Since \mathcal{A} spans A , we can express any state ψ as:

$$\begin{array}{c} \downarrow \\ \psi \end{array} = \sum_i \lambda_i \begin{array}{c} \diamond \\ i \end{array}$$

So:

$$\begin{array}{c} \downarrow \\ f \\ \psi \end{array} = \sum_i \lambda_i \begin{array}{c} \downarrow \\ f \\ i \end{array} = \sum_i \lambda_i \begin{array}{c} \downarrow \\ g \\ i \end{array} = \begin{array}{c} \downarrow \\ g \\ \psi \end{array}$$

In particular, f and g agree on all the states in the basis \mathcal{B} , so $f = g$. \square

Another particularly useful special case is the matrix form of an identity process. Given an ONB, the matrix entries of the identity are:

$$\begin{array}{c} \downarrow \\ j \\ \text{---} \\ \text{---} \\ i \end{array} = \delta_i^j$$

If we write these in a matrix, we get 1s down the diagonal (where $i = j$), and 0s everywhere else:

$$\left| \right\rangle \Leftrightarrow \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

i.e. we get the *identity matrix*. Translating this to matrix form yields:

$$\left| \right\rangle = \sum_i \delta_i^j \begin{array}{c} \downarrow \\ j \\ i \end{array} = \sum_i \begin{array}{c} \downarrow \\ i \\ i \end{array}$$

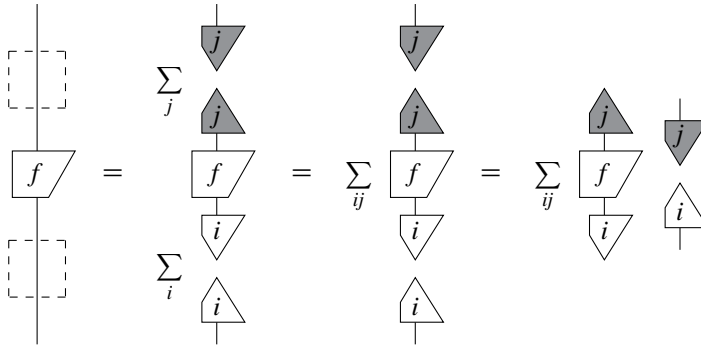
Hence we have the following theorem.

Theorem 5.31 For any ONB we have:

$$\left| \right\rangle = \sum_i \begin{array}{c} \downarrow \\ i \\ i \end{array} \quad (5.17)$$

We refer to such a decomposition as a *resolution of the identity*.

The matrix form for the identity process gives us a handy way to compute the matrix form for an arbitrary process:



The converse of Theorem 5.31 is also true and provides a second, very succinct alternative characterisation of ONBs.

Theorem 5.32 A set of states:

$$\mathcal{A} := \left\{ \begin{array}{c} | \\ \triangle \\ i \end{array} \right\}_i$$

is an ONB if and only if:

$$\begin{array}{c} \begin{array}{c} j \\ \triangle \\ i \end{array} = \delta_i^j \quad \left| \quad \begin{array}{c} | \\ \triangle \\ i \end{array} = \sum_i \begin{array}{c} | \\ \triangle \\ i \end{array} \right. \quad (5.18)$$

Proof From Theorem 5.31, any ONB satisfies (5.18). Conversely, let \mathcal{A} satisfy (5.18), and suppose that for processes f and g we have:

$$\text{for all } \begin{array}{c} | \\ \triangle \\ i \end{array} \in \mathcal{A} : \quad \begin{array}{c} f \\ \triangle \\ i \end{array} = \begin{array}{c} g \\ \triangle \\ i \end{array}$$

Then:

$$\begin{array}{c} f \\ \triangle \\ | \end{array} \stackrel{(5.17)}{=} \sum_i \begin{array}{c} f \\ \triangle \\ i \end{array} = \sum_i \begin{array}{c} g \\ \triangle \\ i \end{array} \stackrel{(5.17)}{=} \begin{array}{c} g \\ \triangle \\ | \end{array}$$

so \mathcal{A} indeed forms a basis, and hence an ONB. □

$$\begin{pmatrix} p^1 \\ \vdots \\ p^n \end{pmatrix} \quad (5.20)$$

with positive real matrix entries p^i summing to 1:

$$\sum_i p^i = \boxed{}$$

Equivalently, probability distributions can also be represented as states of the following form:

$$\begin{array}{c} | \\ \hline \triangleleft p \end{array} := \sum_i p^i \begin{array}{c} | \\ \hline \triangleleft i \end{array}$$

with p^i as above. The probability distributions corresponding to basis states:

$$\begin{array}{c} | \\ \hline \triangleleft 1 \end{array} \Leftrightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \quad \dots \quad \begin{array}{c} | \\ \hline \triangleleft i \end{array} \Leftrightarrow \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \dots \quad \begin{array}{c} | \\ \hline \triangleleft n \end{array} \Leftrightarrow \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

are called *point distributions*.

Since a probability distribution is a matrix (or a state in matrix form), it always comes with a choice of ONB. We take this ONB to be self-conjugate for the simple reason that conjugation has no part in probability theory.

5.1.5 Matrices of Isometries and Unitaries

In this and the following section we characterise isometries, unitaries, positive processes, projectors, and \otimes -positive states in terms of their matrices.

In the case of the first two, which are the ones that we consider in this section, it will be useful to look at the rows and columns of these matrices.

Definition 5.36 Given ONBs for the input and output types of a process f , the *columns* of f are the matrices of the following set of states:

$$\left\{ \begin{array}{c} | \\ \hline \text{f} \\ \hline \triangleleft 1 \end{array}, \dots, \begin{array}{c} | \\ \hline \text{f} \\ \hline \triangleleft m \end{array} \right\}$$

whereas the *rows* are the matrices of the following set of effects:

$$\left\{ \begin{array}{c} \triangleup \\ 1 \\ \hline f \end{array} , \dots , \begin{array}{c} \triangleup \\ n \\ \hline f \end{array} \right\}$$

As the name suggests, the columns of f embed as columns in the overall matrix of f :

and similarly for the rows:

We say a set of column vectors forms an ONB if the associated states do. Similarly, we say row vectors form an ONB if the (adjoints of) the associated effects do. This now gives us a way to recognise matrices of isometries.

Proposition 5.37 For a process f , the following are equivalent:

- (1) f is an isometry;
- (2) f sends orthonormal sets of states to orthonormal sets of states;
- (3) the columns of f are orthonormal; and
- (4) the rows of f^\dagger are orthonormal.

Proof For $(1 \Rightarrow 2)$, given any orthonormal set of states

$$\mathcal{A} := \left\{ \begin{array}{c} \downarrow \\ i \\ \hline \end{array} \right\}_i$$

$$\left\{ \begin{array}{c} \text{---} \\ | \\ \square f \\ | \\ \blacktriangledown i \\ \text{---} \end{array} \right\}_i$$

Diagrammatic equation (4.42) showing the equivalence between a composition of two f nodes and a single δ node. The left side consists of two f nodes (trapezoids) connected in series, with an incoming triangle labeled j and an outgoing triangle labeled i . The right side is a single vertical line with an incoming triangle labeled j and an outgoing triangle labeled i , representing δ_i^j .

9

Proposition 5.38 The following are equivalent:

- Downloaded from <https://www.cambridge.org/core>. Bodleian Libraries of the University of Oxford, on 18 May 2021 at 16:47:36, subject to the Cambridge Core terms of use, available at <https://www.cambridge.org/core/terms>. <https://doi.org/10.1017/9781316219317.006>

$$\begin{array}{c} \boxed{f} \\ \boxed{f} \end{array} = \sum_i \begin{array}{c} \boxed{f} \\ \triangle i \\ \triangle i \\ \boxed{f} \end{array}$$

Since the columns of f form an ONB, by Theorem 5.31, the RHS above is also a resolution of the identity:

$$\sum_i \begin{array}{c} \boxed{f} \\ \triangle i \\ \triangle i \\ \boxed{f} \end{array} = \text{---} \quad \text{so} \quad \begin{array}{c} \boxed{f} \\ \boxed{f} \end{array} = \text{---}$$

Finally, $(3 \Leftrightarrow 4)$ follows from noting that f is unitary if and only if f^\dagger is unitary. \square

The theorem above gives us a characterisation of unitaries as precisely those maps that send ONBs to ONBs. The next corollary immediately follows.

Corollary 5.39 Unitaries can equivalently be represented as:

$$\sum_i \begin{array}{c} \triangle i \\ \triangle i \end{array}$$

for some pair of ONBs (with the same number of elements).

Remark 5.40 Proposition 5.37 tells us that isometries send orthonormal sets of states to orthonormal sets of sets. In particular, it sends an ONB to some *subset* of another ONB. Thus, we can still use the representation (5.39) for isometries, provided we relax the requirement on:

$$\left\{ \triangle i \right\}_i$$

from being an ONB to an orthonormal set. In other words, we can think of isometries as unitaries that have been ‘extended’ to a larger output system.

5.1.6 Matrices of Self-Adjoint and Positive Processes

In Theorem 5.18 we characterised the matrix for the adjoint of a process:

$$(f^\dagger)_i^j = \overline{(f_j^i)}$$

From this, it directly follows that the matrix of a self-adjoint process is:

$$\begin{pmatrix} f_1^1 & f_2^1 & \cdots & f_n^1 \\ \overline{f_2^1} & f_2^2 & \cdots & f_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \overline{f_n^1} & \overline{f_n^2} & \cdots & f_n^n \end{pmatrix}$$

where, in particular, the elements on the diagonal are self-conjugate:

$$f_i^i \stackrel{(5.25)}{=} \begin{array}{c} \triangle i \\ | \\ \square f \\ | \\ \triangle i \end{array} \stackrel{(4.46)}{=} \begin{array}{c} \triangle i \\ | \\ \square f \\ | \\ \triangle i \end{array} \stackrel{(5.25)}{=} (f_i^i)^\dagger = \overline{f_i^i}$$

Unfortunately, matrices of positive processes cannot be recognised quite so easily, but we do at least know what the numbers on the diagonal look like:

$$f_i^i \stackrel{(5.25)}{=} \begin{array}{c} \triangle i \\ | \\ \square f \\ | \\ \triangle i \end{array} \stackrel{(4.45)}{=} \begin{array}{c} \triangle i \\ | \\ \square g \\ | \\ \square g \\ | \\ \triangle i \end{array} = \begin{array}{c} \triangle i \\ | \\ \square g \\ | \\ \square g \\ | \\ \triangle i \end{array} \quad (5.22)$$

i.e. they are positive numbers. Then, by process–state duality we have the following.

Corollary 5.41 For a \otimes -positive state ψ the numbers:

$$\psi^{ii} := \begin{array}{c} \triangle i \quad \triangle i \\ | \quad | \\ \square \psi \end{array}$$

are positive.

Fortunately, for certain processes called *diagonalisable* processes, characterising the numbers on the diagonal is good enough.

Definition 5.42 An *eigenstate* of a process f is a non-zero state ψ such that for some number λ we have:

$$\begin{array}{c} \text{---} \\ | \\ \boxed{f} \\ | \\ \nabla \psi \end{array} = \lambda \begin{array}{c} \text{---} \\ | \\ \nabla \psi \end{array}$$

and f is called *diagonalisable* if there exists an ONB \mathcal{B} such that all of the basis states are eigenstates of f , that is:

$$\text{for all } \begin{array}{c} \text{---} \\ | \\ \nabla i \end{array} \in \mathcal{B}, \text{ there exists } \lambda_i : \begin{array}{c} \text{---} \\ | \\ \boxed{f} \\ | \\ \nabla i \end{array} = \lambda_i \begin{array}{c} \text{---} \\ | \\ \nabla i \end{array} \quad (5.23)$$

The following is the reason for the terminology.

Proposition 5.43 If a process f is diagonalisable, then its matrix in the basis of eigenstates is a *diagonal matrix* :

$$\begin{array}{c} \text{---} \\ | \\ \boxed{f} \\ | \\ \text{---} \end{array} = \sum_i \lambda_i \begin{array}{c} \text{---} \\ | \\ \nabla i \\ | \\ \nabla i \\ | \\ \text{---} \end{array} \Leftrightarrow \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix} \quad (5.24)$$

Proof For the matrix entries of a diagonalisable process f we have:

$$\begin{array}{c} \nabla j \\ | \\ \boxed{f} \\ | \\ \nabla i \end{array} \stackrel{(5.23)}{=} \lambda_i \begin{array}{c} \nabla j \\ | \\ \nabla i \end{array} = \lambda_i \delta_i^j \quad (5.25)$$

from which (5.24) follows. □

If a process is diagonalisable, we can characterise self-adjointness, positivity, and projectors in terms of the numbers on the diagonal.

Proposition 5.44 For a diagonalisable process f :

- (i) f is self-adjoint if and only if all λ_i are self-conjugate;
- (ii) f is positive if and only if all λ_i are positive;
- (iii) f is a projector if and only if all λ_i are positive and satisfy $(\lambda_i)^2 = \lambda_i$.

Proof Part (i) follows from the characterisation of self-adjoint matrices at the beginning of this section. For (ii), first assume f is positive. Then, by (5.22) the λ_i are all positive. Conversely, assume each of the λ_i is positive. Then, for some μ_i , $\lambda_i = \mu_i^\dagger \circ \mu_i$. For all the

process theories in this book, it suffices to assume each μ_i is also a number (as opposed to a more general state), so we first consider that case. Letting:

$$\begin{array}{c} | \\ \hline \text{g} \\ \hline | \end{array} := \sum_i \mu_i \begin{array}{c} | \\ \hline i \\ \hline | \end{array} \begin{array}{c} | \\ \hline i \\ \hline | \end{array}$$

one can easily verify that $f = g^\dagger \circ g$. For a more general process theory, μ_i need not be a number, in which case g needs to be constructed more carefully (cf. Exercise* 5.45). For (iii), let f be a projector. By (i) we know the λ_i must all be positive. Moreover we have:

$$\lambda_i \stackrel{(5.25)}{=} \begin{array}{c} i \\ \hline f \\ \hline i \end{array} \stackrel{(4.50)}{=} \begin{array}{c} i \\ \hline f \\ \hline f \\ \hline i \end{array} \stackrel{(5.23)}{=} \begin{array}{c} i \\ \hline f \\ \hline \lambda_i \\ \hline i \end{array} \stackrel{(5.25)}{=} (\lambda_i)^2$$

Conversely, if the λ_i are positive, so is f , and f is clearly a projector precisely when $(\lambda_i)^2 = \lambda_i$. \square

Put equivalently, a diagonalisable process f is self-adjoint, positive, or a projector if and only if the numbers λ_i are respectively self-adjoint, positive, or projectors themselves.

Exercise* 5.45 Prove (ii) from Proposition 5.44 in the case where the processes μ_i are each states of type A_i . That is, where:

$$\lambda_i = \begin{array}{c} \mu_i \\ \hline A_i \\ \hline \mu_i \end{array}$$

In all of the process theories we will consider in this book, the only numbers satisfying $\lambda^2 = \lambda$ are 0 and 1, so diagonalisable projectors have only zeroes and ones on the diagonal.

Why all this fuss about diagonalisable processes? One might think this is a very restrictive condition. For instance, diagonalisability in some process theories is totally uninteresting.

Example 5.46 For **relations**, the only ONBs available are the singleton bases. As a consequence, the only diagonalisable processes are those whose matrices are already diagonal.

However, as we will see in Section 5.3.3.1, all self-adjoint processes in the theory of **linear maps** are diagonalisable. Since positive processes and projectors are also self-adjoint, Proposition 5.44 gives a complete characterisation in that context.

Exercise 5.47 Show that in any process theory in which non-zero numbers are cancellable (see Section 4.4.2), for any non-zero states ψ, ϕ where:

$$\begin{array}{c} \triangle \\ \phi \\ \hline \psi \\ \nabla \end{array} = 0$$

5.1.7 Traces of Matrices

$$\mathrm{tr}(f) := \text{tr}(f)$$
[illegible]
$$\text{tr} \begin{pmatrix} f_1^1 & \cdots & \cdots & \cdot \\ \vdots & f_2^2 & \cdots & \cdot \\ \vdots & \vdots & \ddots & \vdots \\ \cdot & \cdot & \cdots & f_D^D \end{pmatrix} = \sum_i f_i^i$$

Note that the diagonal entries of a matrix depend on the choice of basis in which the matrix is written. One might be tempted to assume the trace is therefore basis-dependent. However, this is evidently not the case, since we initially defined the trace without any reference to an ONB. The crucial point is in (5.26): we can decompose the identity using any ONB, and the result will be the same.

One can characterise the partial trace similarly. First note that we can decompose the trace as a sum, similar to before:

$$\mathrm{tr}_A(f) = \sum_i \begin{array}{c} \triangleup_i \\ | \\ \text{---} f \text{---} \\ | \\ \triangledown_i \end{array} \begin{array}{c} C \\ B \end{array}$$

However, now rather than *numbers* being summed over, we are summing over processes from B to C . If we compare the matrices of each of these smaller processes with the big matrix of f , we will see them occurring as block matrices along the main diagonal:

$$\begin{pmatrix} \boxed{\mathbf{f}_1} & \cdots & \cdots & \cdot \\ \vdots & \boxed{\mathbf{f}_2} & \cdots & \cdot \\ \vdots & \vdots & \ddots & \vdots \\ \cdot & \cdot & \cdots & \boxed{\mathbf{f}_D} \end{pmatrix}$$

where:

$$\left(\boxed{\mathbf{f}_i} \right) \quad \text{is the matrix of the process} \quad \begin{array}{c} \triangleup_i \\ | \\ \text{---} f \text{---} \\ | \\ \triangledown_i \end{array} \begin{array}{c} C \\ B \end{array}$$

So, the matrix of the partial trace of f is computed by summing up these block matrices:

$$\mathrm{tr}_A \begin{pmatrix} \boxed{\mathbf{f}_1} & \cdots & \cdots & \cdot \\ \vdots & \boxed{\mathbf{f}_2} & \cdots & \cdot \\ \vdots & \vdots & \ddots & \vdots \\ \cdot & \cdot & \cdots & \boxed{\mathbf{f}_D} \end{pmatrix} = \sum_i \left(\boxed{\mathbf{f}_i} \right)$$

where the size and the shape of the blocks depends on the dimensions of the system-types B and C .

5.2 Matrix Calculus

We can now represent all processes by matrices, and conversely, each matrix represents a process. We can identify special processes easily in terms of their matrices, and also have matricial counterparts for transposition, conjugation, and adjoints. However, rather than processes in isolation, what we truly care about is forming diagrams with processes. So we need to figure out how diagrams translate into compositions of matrices.

We have essentially two ways forward here. One way, suggested by Theorem 4.19, is to treat string diagrams as circuit diagrams to which we adjoin cups and caps for each type and provide matricial counterparts to sequential composition, parallel composition, and cups/caps. We will do this over the next three sections. In Section 5.2.4 we will then show a more direct way to compute the matrix of a string diagram, going via the string diagram formulas introduced in Definition 4.21.

Once all of this is in place, we show that matrices provide not only a manner for representing process theories, but also an easy way to construct ‘abstract’ process theories where all that needs to be specified are the numbers that serve as matrix entries.

5.2.1 Sequential Composition of Matrices

First we investigate how sequential composition of processes can be seen as an operation on the matrices of those processes.

Theorem 5.48 Let f and g be processes with associated matrices \mathbf{f} and \mathbf{g} . The matrix of $g \circ f$ is the *matrix product* $\mathbf{g}\mathbf{f}$, which is defined by:

$$(\mathbf{g}\mathbf{f})_i^k := \sum_j \mathbf{g}_j^k \mathbf{f}_i^j \quad (5.27)$$

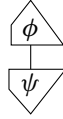
Proof The matrix entries for $g \circ f$ are:

$$(g \circ f)_i^k = \sum_j \mathbf{g}_j^k \mathbf{f}_i^j = (\mathbf{g}\mathbf{f})_i^k$$

The diagrammatic proof shows the composition of two processes g and f . On the left, the composition $g \circ f$ is represented by a string diagram where f (bottom) has input i and output j , and g (top) has input j and output k . This is equated to a sum over j of the product of the matrices of g and f . The matrix of g is \mathbf{g}_j^k and the matrix of f is \mathbf{f}_i^j . The product of these matrices is $\mathbf{g}_j^k \mathbf{f}_i^j$, and the sum over j is $\sum_j \mathbf{g}_j^k \mathbf{f}_i^j$, which is equal to $(\mathbf{g}\mathbf{f})_i^k$.

□

There is an easy way to compute the matrix product (5.27) using rows and columns. First note that the matrix of a state consists of a single column, while the matrix of an effect consists of a single row. Applying the composition formula (5.27) to the case of a state ψ and an effect ϕ :



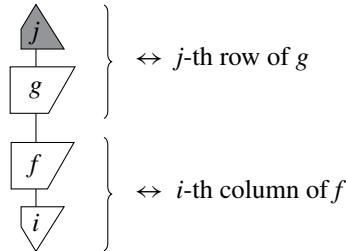
we obtain:

$$(\phi_1 \quad \phi_2 \quad \cdots \quad \phi_n) \begin{pmatrix} \psi^1 \\ \psi^2 \\ \vdots \\ \psi^n \end{pmatrix} = \phi_1 \psi^1 + \cdots + \phi_n \psi^n$$

This is sometimes called the *dot product* of ψ and ϕ , though really it is just sequential composition. Examining the resulting matrix entries in the composition formula (5.27) for general processes f and g :

$$(g \circ f)_i^j = g_1^j f_i^1 + \cdots + g_n^j f_i^n$$

we see that we can compute the entry in the i -th column and the j -th row of the matrix of $g \circ f$ by taking the dot product of the i -th column of f with the j -th row of g :



resulting in:

$$\begin{pmatrix} \vdots \\ g_1^j & \cdots & g_n^j \\ \vdots \end{pmatrix} \begin{pmatrix} f_i^1 \\ \vdots \\ f_i^m \end{pmatrix} = \begin{pmatrix} \vdots \\ g_1^j f_i^1 + \cdots + g_n^j f_i^m \\ \vdots \end{pmatrix}$$

which is of course how most readers will have learned to do matrix composition in school.

5.2.2 Parallel Composition of Matrices

Next up is parallel composition. We will compute the matrix corresponding to $f \otimes g$, given that we have the matrices of f and g . To do this, we must first establish how we obtain an ONB for a joint system-type:



given that we have ONBs for A and B .

Theorem 5.49 Let:

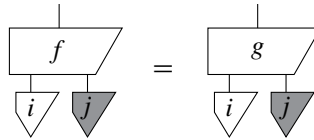
$$\mathcal{B} := \left\{ \begin{array}{c} \downarrow \\ i \end{array} \right\}_i \quad \text{and} \quad \mathcal{B}' := \left\{ \begin{array}{c} \downarrow \\ j \end{array} \right\}_j$$

be ONBs for A and B , respectively. Then the set of states:

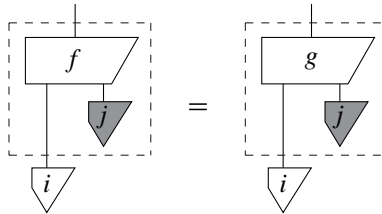
$$\left\{ \begin{array}{c} \downarrow \\ i \end{array} \begin{array}{c} \downarrow \\ j \end{array} \right\}_{ij} \quad (5.28)$$

forms an ONB for $A \otimes B$ called the *product basis*.

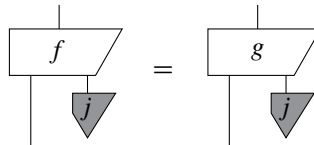
Proof We can show that the basis condition (5.2) holds using one basis at a time, much as we did in the proof of Theorem 5.14. Assume any pair of processes f, g with input type $A \otimes B$ agrees on all of the states in (5.28):



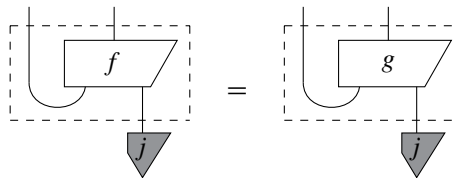
which we can rewrite as:



Since \mathcal{B} is a basis, it follows that:



so for all states j in \mathcal{B}' we also have:



we can rely on the fact that each integer k where $0 \leq k < DD'$ can be decomposed as $iD' + j$ for unique i and j . We can therefore number elements in the product basis as:

$$\begin{array}{|c|} \hline iD' + j \\ \hline \end{array} := \begin{array}{|c|} \hline i \\ \hline \end{array} \begin{array}{|c|} \hline j \\ \hline \end{array} \quad (5.29)$$

This kind of convention extends naturally to any number of systems, and if all systems have the same dimension D , we can think of (5.29) as a representation of some number in base- D . For example, when all system-types have dimension $D = 2$ we have:

$$\begin{array}{|c|} \hline 117 \\ \hline \end{array} := \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{|c|} \hline 0 \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{|c|} \hline 0 \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline \end{array} \quad (5.30)$$

since the number 117 can be written in base-2 as 1110101.

Remark 5.51 The fact that we can encode bit strings as basis states of N systems is very important for quantum computation. We already encountered a similar encoding in Example 3.27 when representing bit strings as N -fold Cartesian products of the set \mathbb{B} .

When working with product bases, it will almost always be more convenient to index a matrix entry by individual digits, thus avoiding the extra number-juggling from equation (5.29). Given a string of individual basis states as in the RHS of (5.30), we call the state on the far left the *most significant* basis state, because changing it will change the ‘encoded’ number the most, and the state on the far right the *least significant* basis state. This follows computer science terminology, where one refers to the ‘most significant bit’ or ‘least significant bit’ in a bit string. This is of course purely a matter of convention. The important thing is to pick a convention and stick to it.

For Q a two-dimensional system the dimension of $Q \otimes Q$ is four, so the matrix of a state ψ of type $Q \otimes Q$ is a column vector with four elements:

$$\begin{array}{|c|} \hline \psi \\ \hline \end{array} \leftrightarrow \begin{pmatrix} \psi^{00} \\ \psi^{01} \\ \psi^{10} \\ \psi^{11} \end{pmatrix} \quad \text{where} \quad \psi^{ij} := \begin{array}{|c|} \hline \begin{array}{|c|} \hline i \\ \hline \end{array} \begin{array}{|c|} \hline j \\ \hline \end{array} \\ \hline \psi \\ \hline \end{array}$$

The use of subscripts and superscripts keeps things tidy, even with multiple indices around:

$$\begin{array}{|c|} \hline g \\ \hline \end{array} \leftrightarrow \begin{pmatrix} g_{00}^0 & g_{01}^0 & g_{10}^0 & g_{11}^0 \\ g_{00}^1 & g_{01}^1 & g_{10}^1 & g_{11}^1 \end{pmatrix} \quad \text{where} \quad g_{ij}^k := \begin{array}{|c|} \hline \begin{array}{|c|} \hline k \\ \hline \end{array} \\ \hline \begin{array}{|c|} \hline g \\ \hline \end{array} \\ \hline \begin{array}{|c|} \hline i \\ \hline \end{array} \begin{array}{|c|} \hline j \\ \hline \end{array} \\ \hline \end{array}$$

So in general, we will encounter matrices with many upper and lower indices:

$$\left(g_{i_1 i_2 \dots i_M}^{j_1 j_2 \dots j_N} \mid 0 \leq i_k < D_k, 0 \leq j_k < D'_k \right) \quad (5.31)$$

Remark* 5.52 This is sometimes called *tensor notation*, where a tensor is just a matrix with lots of indices. It is a precursor to *abstract tensor notation* (cf. Section* 3.6.1), and hence also to diagram formulas, neither of which depend on fixing bases.

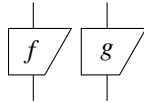
Now, suppose we consider a separable state $\psi \otimes \phi$ of type $Q \otimes Q$. Since this state is formed from states ψ and ϕ , we would expect there to be a relationship between its matrix and the matrices of ψ and ϕ . It can be easily verified that this is indeed the case:

$$\begin{array}{c} \downarrow \\ \psi \\ \downarrow \end{array} \leftrightarrow \begin{pmatrix} \psi^0 \\ \psi^1 \end{pmatrix} \quad \begin{array}{c} \downarrow \\ \phi \\ \downarrow \end{array} \leftrightarrow \begin{pmatrix} \phi^0 \\ \phi^1 \end{pmatrix} \quad \begin{array}{c} \downarrow \quad \downarrow \\ \psi \quad \phi \\ \downarrow \quad \downarrow \end{array} \leftrightarrow \begin{pmatrix} \psi^0 \phi^0 \\ \psi^0 \phi^1 \\ \psi^1 \phi^0 \\ \psi^1 \phi^1 \end{pmatrix}$$

The final matrix consists of all the ways to form a product of one number from the first matrix and one number from the second matrix. This is called the *Kronecker product* of matrices. Let ϕ be the matrix for ϕ . Then we can write this more succinctly using a *block matrix*:

$$\begin{array}{c} \downarrow \quad \downarrow \\ \psi \quad \phi \\ \downarrow \quad \downarrow \end{array} \leftrightarrow \begin{pmatrix} \psi^0 \phi \\ \psi^1 \phi \end{pmatrix}$$

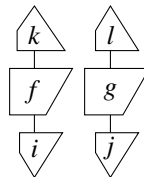
where $\psi^i \phi$ means ‘multiply all of the elements in ϕ by ψ^i ’. The Kronecker product works not only for matrices of states, but also for matrices of any processes of the form:



Theorem 5.53 Let f and g be processes with matrices \mathbf{f} and \mathbf{g} . The matrix of $f \otimes g$ is the *Kronecker product* $\mathbf{f} \otimes \mathbf{g}$, which is defined by:

$$(\mathbf{f} \otimes \mathbf{g})_{ij}^{kl} := \mathbf{f}_i^k \mathbf{g}_j^l \quad (5.32)$$

Proof The matrix entries for $f \otimes g$ are given by:



which exactly matches Definition (5.32). □

In terms of block matrices, for:

$$\mathbf{f} := \begin{pmatrix} f_1^1 & \cdots & f_m^1 \\ \vdots & \ddots & \vdots \\ f_1^n & \cdots & f_m^n \end{pmatrix} \quad \text{and} \quad \mathbf{g} := \begin{pmatrix} g_1^1 & \cdots & g_{m'}^1 \\ \vdots & \ddots & \vdots \\ g_1^{n'} & \cdots & g_{m'}^{n'} \end{pmatrix}$$

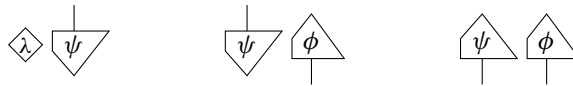
the Kronecker product is the $(nn') \times (mm')$ matrix:

$$\begin{array}{c} \diagup f \diagdown \\ \diagdown g \diagup \end{array} \leftrightarrow \begin{pmatrix} f_1^1 \mathbf{g} & \cdots & f_m^1 \mathbf{g} \\ \vdots & \ddots & \vdots \\ f_1^n \mathbf{g} & \cdots & f_m^n \mathbf{g} \end{pmatrix}$$

This applies to any dimension of matrix, so we can compute Kronecker products of matrices of states, effects, and more general processes. For example, the Kronecker product of the 2×1 matrix of a state ψ and the 2×2 matrix of a process f is computed as follows:

$$\begin{pmatrix} \psi^0 \\ \psi^1 \end{pmatrix} \otimes \begin{pmatrix} f_0^0 & f_1^0 \\ f_0^1 & f_1^1 \end{pmatrix} = \begin{pmatrix} \psi^0 \mathbf{f} \\ \psi^1 \mathbf{f} \end{pmatrix} = \begin{pmatrix} \psi^0 f_0^0 & \psi^0 f_1^0 \\ \psi^0 f_0^1 & \psi^0 f_1^1 \\ \psi^1 f_0^0 & \psi^1 f_1^0 \\ \psi^1 f_0^1 & \psi^1 f_1^1 \end{pmatrix}$$

Exercise 5.54 Give the matrices of these processes:



assuming all of the systems are two-dimensional.

We conclude this section by looking at transposition of matrices over compound systems. Just like when we first computed transposes of matrices in Theorem 5.19, we will assume for the remainder of this section that all ONBs are self-conjugate.

In Section 4.2.2 we pointed out that there are two choices of transposition for processes on joint systems: the usual transpose, which is just a rotation, and the algebraic transpose, which uses the ‘criss-crossed’ cups and caps. For the transpose we have:

$$(5.33)$$

and for the algebraic transpose we have:

$$(5.34)$$

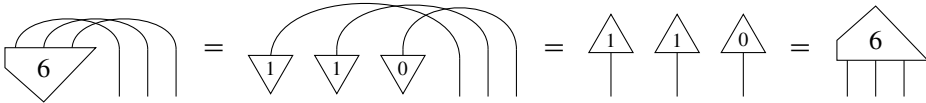
So, we see that the transpose interchanges superscripts and subscripts and reverses the order, whereas the algebraic transpose just interchanges superscripts and subscripts:

$$f_{ij}^{kl} \stackrel{(5.33)}{\rightsquigarrow} f_{lk}^{ji} \quad \text{vs.} \quad f_{ij}^{kl} \stackrel{(5.34)}{\rightsquigarrow} f_{kl}^{ij}$$

Therefore, it is the algebraic transpose that performs the usual transpose of a matrix from linear algebra:

$$\begin{pmatrix} f_{00}^{00} & f_{01}^{00} & f_{10}^{00} & f_{11}^{00} \\ f_{00}^{01} & f_{01}^{01} & f_{10}^{01} & f_{11}^{01} \\ f_{00}^{10} & f_{01}^{10} & f_{10}^{10} & f_{11}^{10} \\ f_{00}^{11} & f_{01}^{11} & f_{10}^{11} & f_{11}^{11} \end{pmatrix} \stackrel{(5.34)}{\rightsquigarrow} \begin{pmatrix} f_{00}^{00} & f_{01}^{00} & f_{10}^{00} & f_{11}^{00} \\ f_{01}^{00} & f_{01}^{01} & f_{10}^{01} & f_{11}^{01} \\ f_{10}^{00} & f_{01}^{10} & f_{10}^{10} & f_{11}^{10} \\ f_{11}^{00} & f_{01}^{11} & f_{10}^{11} & f_{11}^{11} \end{pmatrix}$$

The algebraic transpose also keeps the most significant basis state on the left, and the least significant on the right:



Contrast this with the usual, ‘rotational’ transpose. While handy from a diagrammatic point of view, it’s not very good at counting:



Since conjugation is defined in terms of transposition, this distinction appears there as well.

Exercise 5.55 Show that conjugation reverses the order of input/output indices, whereas algebraic conjugation does not:

$$f_{ij}^{kl} \rightsquigarrow \overline{f_{ji}^{lk}} \quad \text{vs.} \quad f_{ij}^{kl} \rightsquigarrow \overline{f_{ij}^{kl}}$$

5.2.3 Matrix Form of Cups and Caps

We now have matricial counterparts to sequential composition and parallel composition, i.e. a matricial representation for circuits. In order to obtain a matricial representation for string diagrams, we additionally need matricial counterparts for cups and caps. We begin by computing their matrix form.

Proposition 5.56 For any ONB we have:

$$\cup = \sum_i \downarrow_i \downarrow_i \quad \quad \quad \cap = \sum_i \uparrow_i \uparrow_i \quad (5.35)$$

Proof The ONB decomposition of the cup follows immediately from the matrix form of the identity process, as in equation (5.17):

$$\cup = \sum_i \begin{array}{c} \downarrow i \\ \uparrow i \end{array} = \sum_i \begin{array}{c} \downarrow i \\ \downarrow i \end{array}$$

The ONB decomposition of the cap is obtained by taking the adjoint. \square

So (5.35) gives us the matrix form for cups and caps in a product basis. But rather than two copies of the same ONB, this product basis consists of an ONB and its conjugated counterpart. At first it may even seem that this violates the yanking laws, e.g.:

$$\infty = \cup$$

But it doesn't. The key point is that Proposition 5.56 holds for any ONB. In particular, we can apply the proposition to the conjugate of our original ONB and obtain the following equivalent characterisation of cups and caps:

$$\cup = \sum_i \begin{array}{c} \downarrow i \\ \downarrow i \end{array} \quad \cap = \sum_i \begin{array}{c} \uparrow i \\ \uparrow i \end{array} \quad (5.36)$$

Using the two equivalent forms, the yanking laws follow straightforwardly. However, we can avoid this complication by using self-conjugate ONBs, in which case (5.35) becomes:

$$\cup = \sum_i \begin{array}{c} \downarrow i \\ \downarrow i \end{array} \quad \cap = \sum_i \begin{array}{c} \uparrow i \\ \uparrow i \end{array} \quad (5.37)$$

Remark 5.57 Translating equations (5.37) to Dirac notation, we recover the usual definition of the Bell state and the Bell effect that one encounters in the quantum computing literature:

$$\sum_i \begin{array}{c} \downarrow i \\ \downarrow i \end{array} = \sum_i |ii\rangle \quad \sum_i \begin{array}{c} \uparrow i \\ \uparrow i \end{array} = \sum_i \langle ii|$$

The matrices for cups and caps in two dimensions are:

$$\cup \leftrightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad \cap \leftrightarrow \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}$$

Exercise 5.58 Give the matrix of cups and caps in three and four dimensions. What is the relationship between the matrix of a cup/cap and the identity matrix?

In fact, we could just as well take (5.37) to be the definition of the caps and cups for a system of type A . We can then verify the yanking equations directly, e.g.:

$$\begin{array}{c} \text{cup} \end{array} = \begin{array}{c} \sum_i \downarrow_i \quad \sum_j \uparrow_j \end{array} = \sum_{ij} \delta_i^j \begin{array}{c} \downarrow_i \\ \uparrow_j \end{array} = \sum_i \begin{array}{c} \downarrow_i \\ \uparrow_i \end{array} = \text{identity}$$

Exercise 5.59 Prove the yanking equation above without using a self-conjugate basis.

Near to the end of Section 5.1.1 we stated that we can always choose (unique) cups/caps to make a given ONB self-conjugate. We are now ready to show which cups/caps to choose for this purpose.

Proposition 5.60 An ONB:

$$\mathcal{B} := \left\{ \begin{array}{c} \downarrow_i \end{array} \right\}_i$$

is self-conjugate if and only if we have:

$$\text{cup} = \sum_i \begin{array}{c} \downarrow_i \\ \downarrow_i \end{array}$$

Proof Assuming the cup and cap are as given above, we compute conjugates of basis states in \mathcal{B} as the transpose of the adjoint:

$$\begin{array}{c} \text{cap}_j \end{array} = \begin{array}{c} \sum_i \downarrow_i \quad \downarrow_j \end{array} = \sum_i \delta_i^j \begin{array}{c} \downarrow_i \\ \downarrow_j \end{array} = \begin{array}{c} \downarrow_j \end{array}$$

So each of the basis states is indeed self-conjugate. Conversely, suppose caps and cups are chosen such that \mathcal{B} is self-conjugate, that is, for all i :

$$\begin{array}{c} \downarrow_i \end{array} = \begin{array}{c} \downarrow_i \end{array} \quad (5.38)$$

Then, using Proposition 5.56 we obtain:

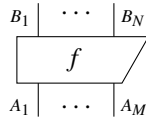
$$\text{cup} = \sum_i \begin{array}{c} \downarrow_i \\ \downarrow_i \end{array} \stackrel{(5.38)}{=} \sum_i \begin{array}{c} \downarrow_i \\ \downarrow_i \end{array}$$

□

5.2.4 String Diagrams of Matrices

In principle, the matricial counterparts of parallel/sequential composition, as well as the matrices of cups and caps, give us everything we need to compute the overall matrix of a string diagram. There is, however, a much more direct method, which we already encountered in Section 3.3.3 for the particular case of **relations**. True, we didn't know then what a string diagram was, but that turns out to not matter that much.

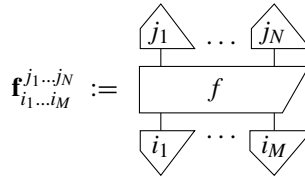
Recall that for a process:



we can write the matrix as:

$$\mathbf{f} = \left(\mathbf{f}_{i_1 \dots i_M}^{j_1 \dots j_N} \mid 0 \leq i_k < D_k, 0 \leq j_k < D'_k \right)$$

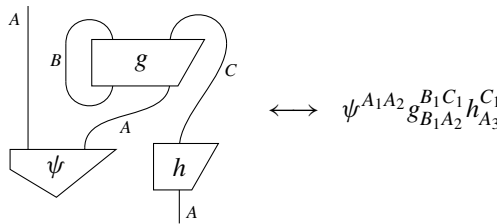
where D_k is the dimension of A_k , D'_k is the dimension of B_k , and:



Theorem 5.61 The matrix of a diagram is the corresponding diagram formula subject to the substitutions:

- box names f become corresponding matrices \mathbf{f} ;
- wire names A_k become indices $0 \leq i_k < D_k$; and
- all repeated indices are summed over.

For example, the matrix \mathbf{m} of:



has entries:

$$\mathbf{m}_{i_3}^{i_1} = \sum_{i_2 j_1 k_1} \psi^{i_1 i_2} \mathbf{g}_{j_1 i_2}^{j_1 k_1} \mathbf{h}_{i_3}^{k_1}$$

If a diagram includes transposes, conjugates, or adjoints, then update the matrix entries of those processes accordingly (cf. Theorems 5.18 and 5.19).

Definition 5.63 Let X be a set of numbers with composition and sums satisfying the conditions spelled out above. We construct the process theory **matrices**(X) as follows:

- (1) The systems are the natural numbers \mathbb{N} .
- (2) The processes with input type $m \in \mathbb{N}$ and output type $n \in \mathbb{N}$ are all $n \times m$ matrices with entries in X .
- (3) Diagrams are computed as in Theorem 5.61.

Of course, instead of the single rule (3) one could take:

- (3a) The cups and caps are the matrices obtained in Exercise 5.58.
- (3b) Adjoints are given by the conjugate-transpose of matrices.
- (3c) Parallel composition is given by the Kronecker product.
- (3d) Sequential composition is given by the matrix product.

Moreover, we have seen in this chapter that matrices can faithfully represent processes in a process theory where every system-type has a finite ONB. The following result should then be unsurprising.

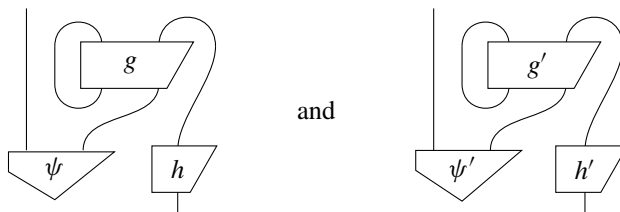
Theorem 5.64 If for a process theory we have:

- (1) each system-type has a finite ONB;
- (2) there is at least one system-type of every dimension $D \in \mathbb{N}$; and
- (3) processes of the same type admit sums

and if the numbers of this process theory are X , then this process theory is *equivalent* to the process theory **matrices**(X).

So what does it mean for two process theories to be equivalent? This is actually a bit more subtle than one may think at first. The naïve approach is to simply require that:

- there is a bijection between system-types A, B, \dots of theory 1 and system-types A', B', \dots of theory 2;
- for all system-types A, B (and corresponding system-types A', B') there is a bijection between processes from A to B in theory 1 and processes from A' to B' in theory 2; and
- these bijections preserve string diagrams, for example, if ψ, g, h are in correspondence with ψ', g', h' , then so too are:



However, this notion of equivalence is often too strict. For example, if a process theory has many types with n -element ONBs, then in **matrices**(X) these all get smooshed onto the same type ' n '. Still, for all practical purposes **matrices**(X) is 'equivalent' to the process theory we started with. Intuitively, equivalent process theories are defined as above, but with the additional ability to 'smoosh together' types that are essentially the same (in this case, types with the same dimension). This description of equivalence will suffice for our purposes, but readers interested in the full details are referred to Section* 5.6.4.

Remark* 5.65 As detailed in Section* 5.6.4, this more subtle notion of equivalence is known in category theory as an *equivalence of categories*, whereas the more naïve notion is called an *isomorphism of categories*.

In the case of the theory of **relations**, we already saw in Example 5.17 that the numbers to consider are the booleans \mathbb{B} . Let **finrelations** be the sub-theory of **relations** that is obtained by restricting the types to finite sets. In order to establish equivalence between the process theory **finrelations** and **matrices**(\mathbb{B}) we need to identify all sets with the same number of elements. Doing so we can conclude the following.

Corollary 5.66 The theory **finrelations** is equivalent to **matrices**(\mathbb{B}).

But we can pick many other sets of numbers X in order to form process theories admitting string diagrams, for example, natural numbers \mathbb{N} , integers \mathbb{Z} , rational numbers \mathbb{Q} , real numbers \mathbb{R} , or some totally wacky 'numbers' like the open sets of a topological space.

Exercise* 5.67 Consider the process theory **matrices**(\mathbb{Z}_2), i.e. matrices over the two-element field, which differs from \mathbb{B} in that $1 + 1 = 0$ rather than $1 + 1 = 1$. What are its properties? How does it relate to **relations**? Note that the 'complete' answer to this question is still a topic of active research.

5.3 Hilbert Spaces

We are now ready to define the process theory of Hilbert spaces and linear maps, which is an important stepping stone towards the theory of quantum processes. In fact, almost all of the pieces of the puzzle are in place. The only remaining piece is to tell you what the numbers are.

5.3.1 Linear Maps and Hilbert Spaces from Diagrams

We saw in Example 3.35 that in **functions**, numbers are trivial. That is, there is only one number. In Example 3.36 we saw that in **relations** there are two numbers 0 and 1, corresponding to 'impossible' and 'possible'. We just saw that given any kind of numbers, we can construct a process theory for these, for example all the real numbers. For Hilbert spaces and linear maps even real numbers are not enough. The new numbers we need will (for the first time in this book) have non-trivial conjugation, and hence will yield adjoints

that do not coincide with the transpose. In other words, finally we will be able to capture the full richness of string diagrams.

Definition 5.68 The process theory of **linear maps** is defined to be the set of all processes described by string diagrams where:

- (1) Each type has a finite ONB.
- (2) There is at least one system-type of every dimension $D \in \mathbb{N}$.
- (3) Processes of the same type admit sums.
- (4) The numbers are the complex numbers \mathbb{C} .

A *Hilbert space* is a system-type in **linear maps**, and we'll denote the D -dimensional systems assumed in (2) as \mathbb{C}^D .

First note that Definition 5.68 doesn't really say what a Hilbert space is. As far as we're concerned, it's just some type that tells us which linear maps can be plugged together. In fact, we know from Theorem 5.64 that we could define **linear maps** equivalently as **matrices**(\mathbb{C}), so for our purposes, the system-types might as well just be natural numbers (i.e. dimensions). This is in line with our usual attitude that processes are more important than systems. For those really worried about this omission, we give the usual set-theoretic definition of Hilbert spaces in Section 5.4.2 and show how it connects to Definition 5.68.

We also haven't recalled yet what a *complex number* is. Before doing so, we want to stress that much of this book can be understood without developing a profound familiarity with complex numbers since, as we mentioned in the introduction, our ultimate goal is to replace matrices of complex numbers with diagrams. A complex number consists of a pair of real numbers $a, b \in \mathbb{R}$, which we write as:

$$a + ib \quad (5.39)$$

One can pretty much compute with complex numbers as one does with real numbers, provided one adopts the following additional equation:

$$i^2 = -1$$

From condition (4) of Definition 5.68, it should be understood that composition of numbers is the usual *multiplication of complex numbers*:

$$\begin{array}{|c|} \hline \lambda_1 \\ \hline \end{array} \begin{array}{|c|} \hline \lambda_2 \\ \hline \end{array} \rightsquigarrow (a_1 + ib_1)(a_2 + ib_2) = (a_1a_2 - b_1b_2) + i(a_1b_2 + b_1a_2)$$

sums are given by *addition of complex numbers*:

$$\begin{array}{|c|} \hline \lambda_1 \\ \hline \end{array} + \begin{array}{|c|} \hline \lambda_2 \\ \hline \end{array} \rightsquigarrow (a_1 + ib_1) + (a_2 + ib_2) = (a_1 + a_2) + i(b_1 + b_2)$$

and, hence, 0 (the 'absorb everything' diagram) and 1 (the empty diagram) are the actual numbers 0 and 1 in \mathbb{C} . Finally, conjugation of numbers corresponds to *complex conjugation*:

$$\begin{array}{|c|} \hline \lambda \\ \hline \end{array} \mapsto \begin{array}{|c|} \hline \bar{\lambda} \\ \hline \end{array} \rightsquigarrow a + ib \mapsto a - ib$$

Crucially, this conjugation is non-trivial!

Remark* 5.69 In fact, if we want an involution that preserves products and sums, we have no choice but to choose the conjugation to be trivial for the real numbers. Furthermore, complex conjugation is the unique (non-trivial) product- and sum-preserving involution for complex numbers that fixes the subset \mathbb{R} .

In light of Example 5.26, complex numbers form a *field*, which means that, along with subtraction, we can have *division by non-zero numbers*. That is, for all $\lambda \neq 0$, there exists a number $\frac{1}{\lambda}$ such that:

$$\diamond \lambda \quad \diamond \frac{1}{\lambda} = \square$$

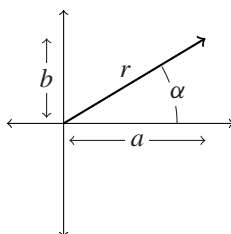
Complex numbers can also be written in *polar form*:

$$re^{i\alpha} \tag{5.40}$$

for r a positive real number and α an angle, called the *complex phase*. The two forms (5.39) and (5.40) are related as follows:

$$\begin{cases} a = r \cos(\alpha) \\ b = r \sin(\alpha) \end{cases} \quad \begin{cases} r = \sqrt{a^2 + b^2} \\ \alpha = \arctan\left(\frac{b}{a}\right) \end{cases}$$

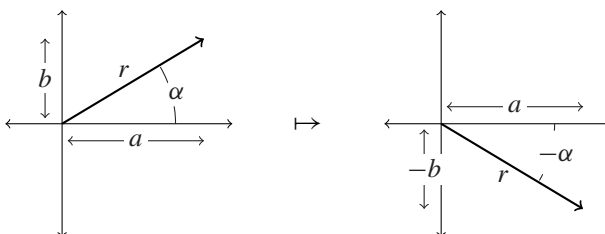
which can be visualised in the *complex plane* as follows:



From this, it's fairly easy to see that conjugation flips the complex phase:

$$re^{i\alpha} \mapsto re^{-i\alpha}$$

Indeed, conjugation simply reflects the complex plane:



Our choice of ‘reflection’ to represent conjugation in diagrams is inspired by this fact:

$$\diamond \lambda = \begin{array}{c} \triangle \phi \\ | \\ \nabla \psi \end{array} \mapsto \begin{array}{c} \triangle \phi \\ | \\ \nabla \psi \end{array} = \diamond \bar{\lambda}$$

5.3.2 Positivity from Conjugation

Using standard terminology, a complex number is ‘positive’ if it is both real and ≥ 0 . Since numbers are, in particular, processes, this definition of ‘positive’ should agree with our prior notion of positivity for processes. This is indeed the case.

Proposition 5.70 For a complex number λ the following are equivalent:

1. It is real and ≥ 0 .
2. There exists a complex number μ such that:

$$\diamond \lambda = \begin{array}{c} \triangle \bar{\mu} \\ | \\ \nabla \mu \end{array} \quad (5.41)$$

3. It is positive in the sense of Definition 4.60; i.e. there exists ψ such that:

$$\diamond \lambda = \begin{array}{c} \triangle \psi \\ | \\ \nabla \psi \end{array} \quad (5.42)$$

Proof For $(1 \Rightarrow 2)$, suppose λ is real and ≥ 0 . Then letting $\mu = \bar{\mu} = \sqrt{\lambda}$, we have equation (5.41). $(2 \Rightarrow 3)$ is immediate, since (5.41) is a special case of (5.42). For $(3 \Rightarrow 1)$, first note that any complex number multiplied by its conjugate is real and ≥ 0 :

$$\bar{\mu}\mu = (a + ib)(a - ib) = a^2 + b^2$$

Computing λ in (5.42) using the matrix of ψ , we have:

$$\lambda = \begin{pmatrix} \bar{\psi}^1 & \bar{\psi}^2 & \dots & \bar{\psi}^n \end{pmatrix} \begin{pmatrix} \psi^1 \\ \psi^2 \\ \vdots \\ \psi^n \end{pmatrix} = \sum_i \bar{\psi}_i \psi_i$$

Since this is a sum of numbers of the form $\bar{\mu}\mu$, it must be real and ≥ 0 . □

In Example* 4.41 we claimed that the transpose in **linear maps** does not provide ‘good’ adjoints. In particular, positive-definiteness:

$$\begin{array}{c} \triangle \psi \\ | \\ \nabla \psi \end{array} = 0 \iff \begin{array}{c} | \\ \nabla \psi \end{array} = 0 \quad (5.43)$$

fails to hold for the transpose. We are now ready to substantiate that claim, then show how complex conjugation fixes the problem.

First, consider this (incorrect) expression of positive definiteness, involving the transpose:

$$(\psi^1 \quad \psi^2 \quad \dots \quad \psi^n) \begin{pmatrix} \psi^1 \\ \psi^2 \\ \vdots \\ \psi^n \end{pmatrix} = 0 \quad \Longleftrightarrow \quad \begin{pmatrix} \psi^1 \\ \psi^2 \\ \vdots \\ \psi^n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

While this works for real numbers, it fails for complex numbers. For example:

$$(1 \quad i) \begin{pmatrix} 1 \\ i \end{pmatrix} = 1 + (-1) = 0 \quad \text{while} \quad \begin{pmatrix} 1 \\ i \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

The problem is, for complex numbers, unlike real numbers, $(\psi^i)^2$ might be less than zero (in this case -1). This is why we need the conjugate-transpose, i.e. the adjoint, for (5.43). Translating this into matrix form, we get:

$$(\overline{\psi^1} \quad \overline{\psi^2} \quad \dots \quad \overline{\psi^n}) \begin{pmatrix} \psi^1 \\ \psi^2 \\ \vdots \\ \psi^n \end{pmatrix} = 0 \quad \Longleftrightarrow \quad \begin{pmatrix} \psi^1 \\ \psi^2 \\ \vdots \\ \psi^n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

or, equivalently:

$$\sum_i \overline{\psi^i} \psi^i = 0 \quad \Longleftrightarrow \quad \text{for all } i : \psi^i = 0 \quad (5.44)$$

By Proposition 5.70 all of the numbers $\overline{\psi^i} \psi^i$ are real and greater than 0. So, if any of the ψ^i are non-zero, the sum in (5.44) will be greater than zero.

5.3.3 Why Mathematicians Love Complex Numbers

In this section, we will look at a number of ways in which passing to complex numbers makes mathematicians' lives a lot easier. While many of the results in this section rely on the explicit matrix forms of linear maps, we will later be able to give their diagrammatic counterparts.

5.3.3.1 The Spectral Theorem

As promised in Section 5.1.6, we present the following theorem.

Theorem 5.71 All self-adjoint linear maps f are *diagonalisable*; that is, there exists some ONB such that:

$$\begin{array}{c} \diagup \\ f \\ \diagdown \end{array} = \sum_i r_i \begin{array}{c} \diagdown \\ i \\ \diagup \\ i \\ \diagdown \end{array} \quad (5.45)$$

where all r_i are real numbers. Moreover, if f is positive, then $r_i \geq 0$ for all i , and if f is a projector, then $r_i \in \{0, 1\}$ for all i . Consequently, every projector P can be written in the following form:

$$\begin{array}{c} \diagup \\ P \\ \diagdown \end{array} = \sum_i \begin{array}{c} \diagdown \\ i \\ \diagup \\ i \\ \diagdown \end{array}$$

for some orthonormal set (i.e. not necessarily a full ONB).

Proof We merely sketch this proof, as the full proof can be found in any book on linear algebra. There are two key ingredients. The first is a standard result that every linear map from a system to itself has at least one eigenstate. This is a consequence of the fact that the numbers in **linear maps** are complex numbers and of the ‘fundamental theorem of algebra’. The second is that self-adjoint linear maps preserve orthogonality with eigenstates. That is, for any eigenstate ψ , if ϕ is orthogonal to ψ , then so too is $f \circ \phi$:

$$\begin{array}{c} \diagup \\ \psi \\ \diagdown \\ f \\ \diagup \\ \phi \end{array} = \begin{array}{c} \diagup \\ \psi \\ \diagdown \\ f \\ \diagup \\ \phi \end{array} = \bar{\lambda} \begin{array}{c} \diagup \\ \psi \\ \diagdown \\ \phi \end{array} = 0$$

This enables us to keep choosing new eigenstates for f orthogonal to all the previous ones until we build an ONB. Once we have established that f is indeed diagonalisable, the fact that each of the r_i in Theorem 5.71 are real, positive, or $\in \{0, 1\}$ follows from Proposition 5.44. \square

Theorem 5.71 is known as the *spectral theorem*. The following equivalent presentation is particularly relevant for us. First we feed (5.45) into process–state duality:

$$\begin{array}{c} \diagup \\ f \\ \diagdown \end{array} = \sum_i r_i \begin{array}{c} \diagdown \\ i \\ \diagup \\ i \\ \diagdown \end{array}$$

and then we exploit the correspondences established in Section 4.3.6:

Corollary 5.72 For all self-conjugate bipartite states ψ in **linear maps** there exists some ONB such that:

$$\begin{array}{c} \downarrow \\ \psi \\ \downarrow \end{array} = \sum_i r_i \begin{array}{c} \downarrow \\ i \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ i \\ \downarrow \end{array} \quad (5.46)$$

where all r_i are real numbers. If ψ is \otimes -positive, then $r_i \geq 0$ for all i .

The origin of the word ‘spectral’ in ‘spectral theorem’ is the following.

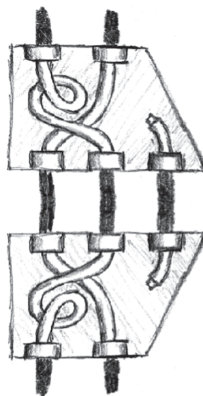
Definition 5.73 For a self-adjoint linear map f we call the list $(r_i)_i$ from Theorem 5.71 the *spectrum* of f , and we denote it by $\text{spec}(f)$. Similarly, for a self-conjugate bipartite state ψ , we also call the numbers from Corollary 5.72 the *spectrum* of ψ , denoted $\text{spec}(\psi)$.

In Section 8.2.5 we will provide a diagrammatic counterpart to Theorem 5.71 and hence also to Corollary 5.72.

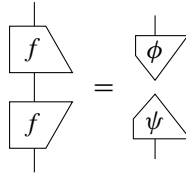
Now we will use the spectral theorem to prove a fact about adjoints of **linear maps** that comes from taking the idea that an adjoint really is a reflection seriously. Suppose we have a \circ -non-separable process. One could then imagine that it has some internal structure, say a collection of tubes or machines connecting some inputs to outputs:



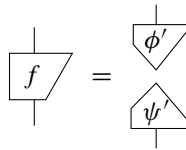
If we now compose this process with its adjoint, i.e. its vertical reflection, then these internal connections match up:



so one expects the resulting process also to be \circ -non-separable. Consequently, if the composite of a process with its adjoint is \circ -separable:



then also that process should be \circ -separable:

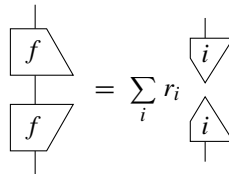


The spectral theorem indeed guarantees that $f^\dagger \circ f$ detects separability of f .

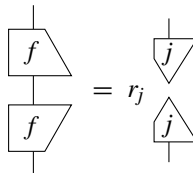
Proposition 5.74 For any linear map f we have:

$$\left(\exists \psi, \phi : \begin{array}{c} \text{box } f \\ \text{box } f \end{array} = \begin{array}{c} \text{triangle } \phi \\ \text{triangle } \psi \end{array} \right) \iff \left(\exists \psi', \phi' : \text{box } f = \begin{array}{c} \text{triangle } \phi' \\ \text{triangle } \psi' \end{array} \right) \quad (5.47)$$

Proof The direction (\Leftarrow) is trivial. For (\Rightarrow) , assume $f^\dagger \circ f$ is \circ -separable. Then, since it is positive, we can diagonalise $f^\dagger \circ f$:



but then, since $f^\dagger \circ f$ is \circ -separable, it must be the case that:



for some j . Otherwise, $f^\dagger \circ f$ could produce (non-zero) orthogonal states as output, which can never happen if it is \circ -separable. Thus, for all $i \neq j$:

$$\begin{array}{c} \triangleup_i \\ | \\ \text{f} \\ | \\ \text{f} \\ | \\ \triangleup_i \end{array} = 0 \quad \text{and hence} \quad \begin{array}{c} | \\ \text{f} \\ | \\ \triangleup_i \end{array} = 0$$

by positive-definiteness. A resolution of the identity completes the proof:

$$\begin{array}{c} | \\ \text{f} \end{array} = \sum_i \begin{array}{c} \text{f} \\ | \\ \triangleup_i \\ | \\ \triangleup_i \end{array} = \begin{array}{c} \text{f} \\ | \\ \triangleup_j \\ | \\ \triangleup_j \end{array}$$

□

Remark 5.75 Note that there is an analogy between condition (5.47) and positive-definiteness. While positive-definiteness tells us that the number $\psi^\dagger \circ \psi$ allows one to detect whether the state ψ is zero, condition (5.47) tells us that $f^\dagger \circ f$ allows one to detect whether f is \circ -separable.

And now comes the really interesting bit. While in Example 4.40 we established that **relations** obey the positive-definiteness condition, **relations** do not satisfy condition (5.47).

Exercise 5.76 Verify that the relation with matrix:

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

does not satisfy (5.47).

So in **relations** the intuition explained above that a gap in $f^\dagger \circ f$ causes a gap in f fails to hold, and this can be traced back to the failure of the spectral theorem for **relations**. The main reason for this is that in **relations** there simply aren't enough numbers to guarantee a diagonal representation for every relation.

Exercise 5.77 Along the same vein as Proposition 5.74, show using the spectral theorem that $f^\dagger \circ f$ detects whether f is invariant under a projector; that is, show that for any projector P :

$$\left(\begin{array}{c} \boxed{P} \\ \diagdown \\ f \\ \diagup \\ f \\ \diagdown \\ \boxed{P} \end{array} \right) = \left(\begin{array}{c} \diagdown \\ f \\ \diagup \\ f \end{array} \right) \iff \left(\begin{array}{c} \diagdown \\ f \\ \diagup \\ \boxed{P} \end{array} \right) = \left(\begin{array}{c} \diagdown \\ f \end{array} \right)$$

5.3.3.2 The Dimension Theorem

We (somewhat perversely) defined $\dim(A)$ in Section 5.1.1 as the size of the smallest basis of a type A . This is quite annoying, since if we want to figure out the dimension of A we need to look at all bases of A , just to make sure that we have the smallest one. Fortunately, in **linear maps** we don't have to do that, because the complex numbers form a field. We can therefore rely on the following theorem, called the *dimension theorem*.

Theorem 5.78 If a process theory admits a matrix calculus and its numbers form a field, then all bases for a given type A are the same size, namely $\dim(A)$.

Thanks to the dimension theorem, in **linear maps** $\dim(A)$ is the size of any basis for A . As a consequence, any orthonormal set of size $\dim(A)$ is an ONB. To see this, we only need the following proposition.

Proposition 5.79 Any orthonormal set:

$$\mathcal{A} = \left\{ \begin{array}{c} \downarrow \\ \boxed{i} \end{array} \right\}$$

extends to an ONB containing \mathcal{A} .

Proof It is straightforward to show that:

$$\boxed{P} := \sum_i \begin{array}{c} \downarrow \\ \boxed{i} \end{array} \quad \text{and} \quad \boxed{Q} := \left| - \right. \boxed{P}$$

are projectors. Using the spectral theorem we can diagonalise Q in terms of a second orthonormal set of states \mathcal{A}' :

$$\boxed{Q} = \sum_i \begin{array}{c} \downarrow \\ \boxed{i} \end{array}$$

Since P is a projector, it easily follows that $P \circ Q = 0$, and from this it then also easily follows that all of states in \mathcal{A}' must be orthogonal to those in \mathcal{A} . Moreover, since:

$$\sum_i \begin{array}{c} \text{---} \\ \diagdown \text{ } i \text{ } \diagup \\ \text{---} \end{array} + \sum_i \begin{array}{c} \text{---} \\ \diagdown \text{ } i \text{ } \diagup \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \diagdown \text{ } P \text{ } \diagup \\ \text{---} \end{array} + \begin{array}{c} \text{---} \\ \diagdown \text{ } Q \text{ } \diagup \\ \text{---} \end{array} = \left| \right.$$

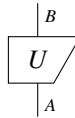
$\mathcal{A} \cup \mathcal{A}'$ gives a resolution of the identity, and thus, by Theorem 5.32, $\mathcal{A} \cup \mathcal{A}'$ is an ONB, which, evidently, contains \mathcal{A} . \square

Thus, if we have an orthonormal set \mathcal{A} of size $\dim(A)$, there is an ONB \mathcal{B} containing \mathcal{A} . But, by Theorem 5.78, \mathcal{B} must be of size $\dim(A)$, so $\mathcal{A} = \mathcal{B}$.

Corollary 5.80 Any orthonormal set of size $\dim(A)$ is an ONB.

Let's now have a look at how the dimension theorem genuinely can make life easier (rather than just 'less annoying'). Recall that in Propositions 5.37 and 5.38, we gave characterisations of isometries and unitaries in terms of their behaviour on orthonormal sets. Combining this with the dimension theorem yields the following.

Proposition 5.81 For any isometry:



$\dim(A) \leq \dim(B)$, and U is furthermore unitary if $\dim(A) = \dim(B)$. So in particular, any isometry from a Hilbert space to itself must be unitary.

Proof Proposition 5.79 implies that any orthonormal set on B must be of size $\leq \dim(B)$. Then, by Proposition 5.37, U sends any ONB \mathcal{B} on A to an orthonormal set \mathcal{B}' of size $\dim(A)$. Thus, $\dim(A) \leq \dim(B)$. If $\dim(A) = \dim(B)$, then by Corollary 5.80, \mathcal{B}' is an ONB for B . Thus by Proposition 5.38, U is a unitary. \square

5.3.4 Classical Logic Gates as Linear Maps

We assumed the existence of Hilbert spaces \mathbb{C}^n for every n . Of particular interest is \mathbb{C}^2 , because of its role in quantum computation, namely in the description of quantum bits, or *qubits*. The Hilbert space \mathbb{C}^2 has a two-element basis:

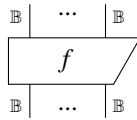
$$\left\{ \begin{array}{c} \text{---} \\ \diagdown \text{ } 0 \text{ } \diagup \\ \text{---} \end{array}, \begin{array}{c} \text{---} \\ \diagdown \text{ } 1 \text{ } \diagup \\ \text{---} \end{array} \right\}$$

which is referred to as the *computational basis* or *Z-basis*. By analogy with (classical) bits, we choose to count the basis elements of \mathbb{C}^2 from zero.

Remark 5.82 In Dirac notation, these basis states are denoted as:

$$\{|0\rangle, |1\rangle\}$$

Classical *logic gates* are functions:



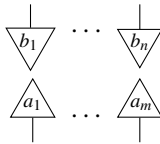
which form the basic building blocks in digital circuits. Thanks to the computational basis, we can now write any logic gate as a linear map, by means of the following translation, for $a, b \in \{0, 1\}$:

$$a \mapsto b \quad \rightsquigarrow \quad \begin{array}{|c} \downarrow \\ \triangle \\ a \end{array} \mapsto \begin{array}{|c} \downarrow \\ \triangle \\ b \end{array}$$

or more generally, for $a_1, \dots, a_m, b_1, \dots, b_n \in \{0, 1\}$:

$$a_1 \cdots a_m \mapsto b_1 \cdots b_n \quad \rightsquigarrow \quad \begin{array}{|c} \downarrow \\ \triangle \\ a_1 \end{array} \cdots \begin{array}{|c} \downarrow \\ \triangle \\ a_m \end{array} \mapsto \begin{array}{|c} \downarrow \\ \triangle \\ b_1 \end{array} \cdots \begin{array}{|c} \downarrow \\ \triangle \\ b_n \end{array}$$

Each such mapping of a particular input contributes a term:



and the linear map is obtained as the sum of all the terms of this kind. For instance, the AND gate:

$$\text{AND} :: \begin{cases} (0, 0) \mapsto 0 \\ (0, 1) \mapsto 0 \\ (1, 0) \mapsto 0 \\ (1, 1) \mapsto 1 \end{cases}$$

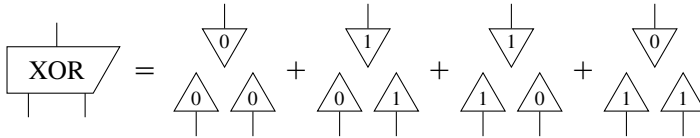
induces the linear map:

$$\begin{array}{|c} \downarrow \\ \text{AND} \end{array} = \begin{array}{|c} \downarrow \\ 0 \end{array} \begin{array}{|c} \uparrow \\ 0 \end{array} \begin{array}{|c} \uparrow \\ 0 \end{array} + \begin{array}{|c} \downarrow \\ 0 \end{array} \begin{array}{|c} \uparrow \\ 0 \end{array} \begin{array}{|c} \uparrow \\ 1 \end{array} + \begin{array}{|c} \downarrow \\ 0 \end{array} \begin{array}{|c} \uparrow \\ 1 \end{array} \begin{array}{|c} \uparrow \\ 0 \end{array} + \begin{array}{|c} \downarrow \\ 1 \end{array} \begin{array}{|c} \uparrow \\ 1 \end{array} \begin{array}{|c} \uparrow \\ 1 \end{array}$$

Similarly, for an *exclusive-OR*, or XOR gate :

$$\text{XOR} :: \begin{cases} (0, 0) \mapsto 0 \\ (0, 1) \mapsto 1 \\ (1, 0) \mapsto 1 \\ (1, 1) \mapsto 0 \end{cases}$$

the induced linear map is:



We will see in Chapter 12 that this process of turning logic gates into linear maps plays a key role in the *circuit model of quantum computing*. However, in that context one must restrict to logic gates that yield unitary linear maps, for reasons that will become clear in the following chapter. The AND linear map defined above is not a unitary map. In fact, it doesn't even have an inverse.

Exercise 5.83 Show that a logic gate will define a unitary linear map if and only if it has an inverse (see Definition 4.24).

A simple example of a logic gate with an inverse is the NOT gate:

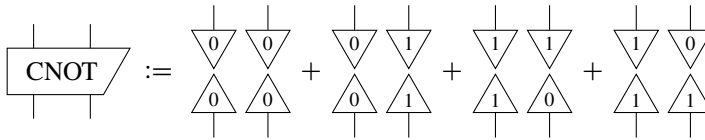
$$\text{NOT} :: \begin{cases} 0 \mapsto 1 \\ 1 \mapsto 0 \end{cases}$$

which induces the linear map:

Another important example is the *controlled-NOT*, or CNOT, gate:

$$\text{CNOT} :: \begin{cases} (0, 0) \mapsto (0, 0) \\ (0, 1) \mapsto (0, 1) \\ (1, 0) \mapsto (1, 1) \\ (1, 1) \mapsto (1, 0) \end{cases}$$

The reason it's called a controlled-NOT gate is that the first bit 'controls' whether we apply a NOT to the second bit. Its induced linear map is:

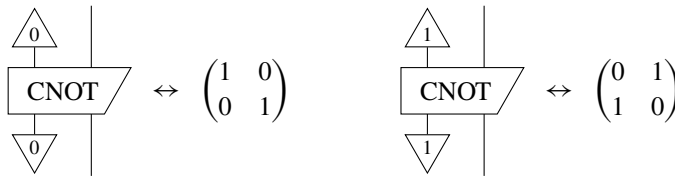


the matrix of which is:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

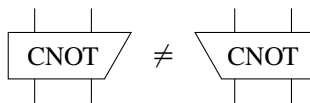
Exercise 5.84 Use Proposition 5.38 to prove that the NOT and CNOT linear maps are both unitary.

Note how the matrix of the CNOT linear map has an identity matrix in the top left corner and a NOT matrix in the bottom right corner. This is a common shape for matrices of 'controlled-(...)' gates, as the first basis element is 'selecting' which of these two smaller matrices to apply to the second:



Note also that the CNOT linear map is self-adjoint, which justifies a diagrammatic representation that is invariant under vertical reflection, which we will encounter in the next section. On the other hand, it is not invariant under horizontal reflection.

Exercise 5.85 Prove that the CNOT linear map is not invariant under horizontal reflection:



Hence, its diagrammatic representation shouldn't be, either.

5.3.5 The X-Basis and the Hadamard Linear Map

An alternative perspective on the CNOT gate involves the COPY gate:

$$\text{COPY} :: \begin{cases} 0 \mapsto (0, 0) \\ 1 \mapsto (1, 1) \end{cases}$$

whose induced linear map is:

$$\text{COPY} := \begin{array}{c} \downarrow \quad \downarrow \\ \text{COPY} \end{array} := \begin{array}{c} \downarrow \quad \downarrow \\ 0 \quad 0 \\ \downarrow \quad \downarrow \\ 0 \end{array} + \begin{array}{c} \downarrow \quad \downarrow \\ 1 \quad 1 \\ \downarrow \quad \downarrow \\ 1 \end{array}$$

One way to perform a CNOT is to COPY the first bit, then XOR a copy of the first bit with the second bit. That is:

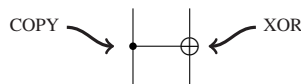
$$\text{CNOT} = \begin{array}{c} \downarrow \quad \downarrow \\ \text{CNOT} \end{array} = \begin{array}{c} \downarrow \quad \downarrow \\ \text{XOR} \\ \downarrow \quad \downarrow \\ \text{COPY} \end{array} \quad (5.48)$$

Exercise 5.86 Prove equation (5.48).

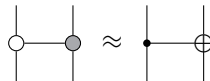
In most textbooks, one encounters the CNOT gate written this way:



This evokes the same ‘COPY+XOR’ interpretation of CNOT, given that copying in classical circuits is usually expressed as a small black dot, and the XOR operation as the symbol \oplus :



However, we will adopt a more symmetric-looking notation:



This is because these XOR and COPY linear maps are very closely related. They actually both arise as COPY maps, but for different bases. To see this, let's consider another basis for \mathbb{C}^2 . First, note that since $-1 \in \mathbb{C}$, we can subtract states:

$$\begin{array}{c} \downarrow \\ \psi \end{array} - \begin{array}{c} \downarrow \\ \phi \end{array} := \begin{array}{c} \downarrow \\ \psi \end{array} + \begin{array}{c} \downarrow \\ -1 \end{array} \begin{array}{c} \downarrow \\ \phi \end{array}$$

Let the X -basis be:

$$\begin{array}{c} \downarrow \\ 0 \end{array} := \frac{1}{\sqrt{2}} \left(\begin{array}{c} \downarrow \\ 0 \end{array} + \begin{array}{c} \downarrow \\ 1 \end{array} \right) \quad \begin{array}{c} \downarrow \\ 1 \end{array} := \frac{1}{\sqrt{2}} \left(\begin{array}{c} \downarrow \\ 0 \end{array} - \begin{array}{c} \downarrow \\ 1 \end{array} \right)$$

It is easy to check that:

$$\begin{array}{c} \triangleup^j \\ \triangleleft_i \end{array} = \delta_i^j$$

and since both of these states take real-valued coefficients in the Z-basis, they are self-conjugate. Thus we now have two self-conjugate ONBs for \mathbb{C}^2 :

$$\left\{ \begin{array}{c} \triangleleft_0 \\ \triangleleft_1 \end{array} \right\} \quad \text{and} \quad \left\{ \begin{array}{c} \triangleleft_0 \\ \triangleleft_1 \end{array} \right\}$$

Remark 5.87 In Dirac notation, the X-basis is usually denoted as:

$$\{|+\rangle, |-\rangle\}$$

referring to the fact that we take the sum and the difference of the states of the Z-basis. We will use these bases in Section 6.1.2 to label the Z- and X-axes on a sphere of states for a qubit, which justifies the seemingly ad hoc names ‘Z-basis’ and ‘X-basis’.

Now we can define two versions of all of the maps before, one for the Z-basis and one for the X-basis. For instance, the COPY linear map now has two versions:

$$\begin{array}{c} \text{Z-COPY} \end{array} := \sum_i \begin{array}{c} \triangleleft_i \quad \triangleleft_i \\ \triangleleft_i \end{array} \quad \begin{array}{c} \text{X-COPY} \end{array} := \sum_i \begin{array}{c} \triangleleft_i \quad \triangleleft_i \\ \triangleleft_i \end{array}$$

Evidently, such COPY maps make sense for Hilbert spaces of all dimensions n . Later on, we use these maps so much that it’s convenient to write them just as dots of the appropriate colour:

$$\begin{array}{c} \text{white dot} \end{array} := \sum_i \begin{array}{c} \triangleleft_i \quad \triangleleft_i \\ \triangleleft_i \end{array} \quad \begin{array}{c} \text{grey dot} \end{array} := \sum_i \begin{array}{c} \triangleleft_i \quad \triangleleft_i \\ \triangleleft_i \end{array}$$

By writing the adjoints of these maps by vertical reflection as usual:

$$\begin{array}{c} \text{white dot} \end{array} := \sum_i \begin{array}{c} \triangleleft_i \\ \triangleleft_i \quad \triangleleft_i \end{array} \quad \begin{array}{c} \text{grey dot} \end{array} := \sum_i \begin{array}{c} \triangleleft_i \\ \triangleleft_i \quad \triangleleft_i \end{array}$$

we can write the XOR-part of a CNOT gate as a grey dot, up to a number.

Proposition 5.88

$$\begin{array}{c} \text{XOR} \end{array} \approx \begin{array}{c} \text{grey dot} \end{array}$$

Proof We show this by computing matrix entries for the Z-basis. Using:

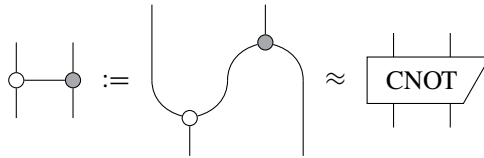
$$\begin{array}{c} \triangleup 0 \\ \downarrow i \end{array} = \begin{array}{c} \triangleup i \\ \downarrow 0 \end{array} = \frac{1}{\sqrt{2}} \quad \begin{array}{c} \triangleup 1 \\ \downarrow i \end{array} = \begin{array}{c} \triangleup i \\ \downarrow 1 \end{array} = (-1)^i \frac{1}{\sqrt{2}}$$

for the RHS we have:

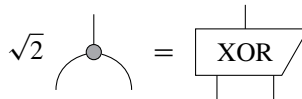
$$\begin{array}{c} \triangleup k \\ \swarrow i \searrow j \end{array} = \left(\begin{array}{c} \triangleup k \\ \downarrow 0 \end{array} \begin{array}{c} \triangleup k \\ \downarrow 1 \end{array} \right) = \frac{1}{\sqrt{2}} \cdot \frac{1}{2} (1 + (-1)^{i+j+k})$$

This will be equal to $\frac{1}{\sqrt{2}}$ if $i+j+k$ is even, and 0 otherwise. But then, $i+j+k$ is even if and only if $i \oplus j = k$. If we ignore the $\frac{1}{\sqrt{2}}$ -factor, this gives precisely the matrix of XOR. \square

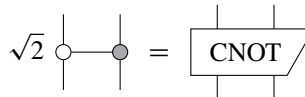
Hence, we let:



If we don't ignore numbers we have:



and hence:



Example 5.89 Since it depends on subtraction, the X-basis doesn't seem to make sense for the theory of **relations** (see Remark 5.27). However, since the COPY process of the X-basis is the same as XOR, this process does live in **relations**. Thus the 'shadow' of this basis, and crucially its interaction with the Z-basis, occurs there.

Exercise 5.90 Prove that CNOT obeys:



We will see in Chapter 9 that these are not just random diagrammatic equations, but rather capture the fundamental concept of *complementarity* in quantum theory.

We can also define a unitary linear map that translates the X -basis into the Z -basis by following the recipe of Corollary 5.39.

Definition 5.91 The *Hadamard map* is:

$$\boxed{H} := \sum_i \begin{array}{c} \text{---} \downarrow i \\ \uparrow i \text{---} \end{array}$$

The reason why we don't draw the Hadamard map asymmetrically, like most boxes, is the following.

Proposition 5.92 The Hadamard linear map H is self-conjugate and self-adjoint and, hence, also self-transposed:

$$\boxed{H} = \begin{array}{c} \text{---} \downarrow \\ \uparrow \text{---} \end{array} \boxed{H} = \boxed{H} \begin{array}{c} \text{---} \downarrow \\ \uparrow \text{---} \end{array}$$

Proof We can see that H is self-adjoint and self-conjugate if we expand the X -basis in terms of the Z -basis:

$$\boxed{H} = \begin{array}{c} \text{---} \downarrow 0 \\ \uparrow 0 \text{---} \end{array} + \begin{array}{c} \text{---} \downarrow 1 \\ \uparrow 1 \text{---} \end{array} = \frac{1}{\sqrt{2}} \left(\begin{array}{c} \text{---} \downarrow 0 \\ \uparrow 0 \text{---} \end{array} + \begin{array}{c} \text{---} \downarrow 1 \\ \uparrow 0 \text{---} \end{array} + \begin{array}{c} \text{---} \downarrow 0 \\ \uparrow 1 \text{---} \end{array} - \begin{array}{c} \text{---} \downarrow 1 \\ \uparrow 1 \text{---} \end{array} \right) \quad (5.49)$$

□

From (5.49), we see that the matrix of H in the Z -basis is:

$$\boxed{H} \leftrightarrow \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Exercise 5.93 What is the matrix of H in the X -basis?

Since the adjoint of a unitary map is its inverse, we also have the following.

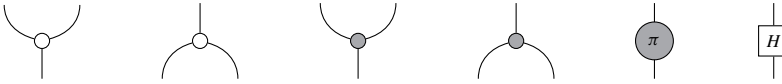
Corollary 5.94 The Hadamard linear map is self-inverse:

$$\boxed{H} \begin{array}{c} \text{---} \downarrow \\ \uparrow \text{---} \end{array} \boxed{H} = \text{---}$$

Remark 5.95 Just as we did for the CNOT gate, later we will also introduce an alternative notation for the NOT gate, namely:



The reason for it being grey is that it, like XOR, can be expressed naturally in terms of the X-basis, which will be explained later. So, here are all of the operations we met in this section:



We will encounter the copy maps again in Section 8.2.2 when expressing *classical data* in diagrammatic terms. Decorations such as the π in the case of the NOT gate will be explained in Section 9.1, and the interaction of the two colours will be explained in Section 9.2. All of these pieces together will play an important role in a very powerful diagrammatic language called the *ZX-calculus*, described in Section 9.4.

5.3.6 Bell Basis and Bell Maps

We will now give a basis for $\mathbb{C}^2 \otimes \mathbb{C}^2$ that is not simply a product of two bases for \mathbb{C}^2 . For the following four states:

$$\left\{ \begin{array}{l} B_0 := \frac{1}{\sqrt{2}} \left(\begin{array}{|c|} \hline \downarrow \\ \hline 0 \end{array} \begin{array}{|c|} \hline \downarrow \\ \hline 0 \end{array} + \begin{array}{|c|} \hline \downarrow \\ \hline 1 \end{array} \begin{array}{|c|} \hline \downarrow \\ \hline 1 \end{array} \right) \\ B_1 := \frac{1}{\sqrt{2}} \left(\begin{array}{|c|} \hline \downarrow \\ \hline 0 \end{array} \begin{array}{|c|} \hline \downarrow \\ \hline 1 \end{array} + \begin{array}{|c|} \hline \downarrow \\ \hline 1 \end{array} \begin{array}{|c|} \hline \downarrow \\ \hline 0 \end{array} \right) \\ B_2 := \frac{1}{\sqrt{2}} \left(\begin{array}{|c|} \hline \downarrow \\ \hline 0 \end{array} \begin{array}{|c|} \hline \downarrow \\ \hline 0 \end{array} - \begin{array}{|c|} \hline \downarrow \\ \hline 1 \end{array} \begin{array}{|c|} \hline \downarrow \\ \hline 1 \end{array} \right) \\ B_3 := \frac{1}{\sqrt{2}} \left(\begin{array}{|c|} \hline \downarrow \\ \hline 0 \end{array} \begin{array}{|c|} \hline \downarrow \\ \hline 1 \end{array} - \begin{array}{|c|} \hline \downarrow \\ \hline 1 \end{array} \begin{array}{|c|} \hline \downarrow \\ \hline 0 \end{array} \right) \end{array} \right\} \quad (5.50)$$

it is straightforward to show that they form an orthonormal set, e.g.:

$$\begin{aligned} \begin{array}{|c|} \hline B_0 \\ \hline \text{---} \\ \hline B_0 \\ \hline \end{array} &= \frac{1}{2} \left(\begin{array}{|c|} \hline \triangle \\ \hline 0 \end{array} \begin{array}{|c|} \hline \triangle \\ \hline 0 \end{array} + \begin{array}{|c|} \hline \triangle \\ \hline 1 \end{array} \begin{array}{|c|} \hline \triangle \\ \hline 0 \end{array} + \begin{array}{|c|} \hline \triangle \\ \hline 0 \end{array} \begin{array}{|c|} \hline \triangle \\ \hline 1 \end{array} + \begin{array}{|c|} \hline \triangle \\ \hline 1 \end{array} \begin{array}{|c|} \hline \triangle \\ \hline 1 \end{array} \right) \\ &= \frac{1}{2} (1 + 0 + 0 + 1) = 1 \end{aligned}$$

$$\begin{aligned} \begin{array}{|c|} \hline B_1 \\ \hline \text{---} \\ \hline B_0 \\ \hline \end{array} &= \frac{1}{2} \left(\begin{array}{|c|} \hline \triangle \\ \hline 0 \end{array} \begin{array}{|c|} \hline \triangle \\ \hline 1 \end{array} + \begin{array}{|c|} \hline \triangle \\ \hline 1 \end{array} \begin{array}{|c|} \hline \triangle \\ \hline 0 \end{array} + \begin{array}{|c|} \hline \triangle \\ \hline 0 \end{array} \begin{array}{|c|} \hline \triangle \\ \hline 1 \end{array} + \begin{array}{|c|} \hline \triangle \\ \hline 1 \end{array} \begin{array}{|c|} \hline \triangle \\ \hline 0 \end{array} \right) \\ &= \frac{1}{2} (0 + 0 + 0 + 0) = 0 \end{aligned}$$

$$\begin{aligned}
 \begin{array}{|c|} \hline B_2 \\ \hline B_0 \\ \hline \end{array} &= \frac{1}{2} \left(\begin{array}{|c|} \hline 0 \\ \hline 0 \end{array} \begin{array}{|c|} \hline 0 \\ \hline 0 \end{array} - \begin{array}{|c|} \hline 1 \\ \hline 0 \end{array} \begin{array}{|c|} \hline 1 \\ \hline 0 \end{array} + \begin{array}{|c|} \hline 0 \\ \hline 1 \end{array} \begin{array}{|c|} \hline 0 \\ \hline 1 \end{array} - \begin{array}{|c|} \hline 1 \\ \hline 1 \end{array} \begin{array}{|c|} \hline 1 \\ \hline 1 \end{array} \right) \\
 &= \frac{1}{2} (1 - 0 + 0 - 1) = 0
 \end{aligned}$$

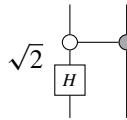
and so on. So the B_i are an orthonormal set of size $\dim(\mathbb{C}^2 \otimes \mathbb{C}^2) = 4$, hence by Corollary 5.80, they form an ONB. Indeed, we can write each of the elements of the product basis for $\mathbb{C}^2 \otimes \mathbb{C}^2$ in terms of the states (5.50):

$$\begin{aligned}
 \begin{array}{|c|} \hline 0 \\ \hline 0 \end{array} \begin{array}{|c|} \hline 0 \\ \hline 0 \end{array} &= \frac{1}{\sqrt{2}} \left(\begin{array}{|c|} \hline B_0 \\ \hline \end{array} + \begin{array}{|c|} \hline B_2 \\ \hline \end{array} \right) \\
 \begin{array}{|c|} \hline 0 \\ \hline 1 \end{array} \begin{array}{|c|} \hline 1 \\ \hline 0 \end{array} &= \frac{1}{\sqrt{2}} \left(\begin{array}{|c|} \hline B_1 \\ \hline \end{array} + \begin{array}{|c|} \hline B_3 \\ \hline \end{array} \right) \\
 \begin{array}{|c|} \hline 1 \\ \hline 0 \end{array} \begin{array}{|c|} \hline 0 \\ \hline 1 \end{array} &= \frac{1}{\sqrt{2}} \left(\begin{array}{|c|} \hline B_1 \\ \hline \end{array} - \begin{array}{|c|} \hline B_3 \\ \hline \end{array} \right) \\
 \begin{array}{|c|} \hline 1 \\ \hline 1 \end{array} \begin{array}{|c|} \hline 1 \\ \hline 1 \end{array} &= \frac{1}{\sqrt{2}} \left(\begin{array}{|c|} \hline B_0 \\ \hline \end{array} - \begin{array}{|c|} \hline B_2 \\ \hline \end{array} \right)
 \end{aligned}$$

Definition 5.96 The ONB (5.50) is called the *Bell basis*.

By Proposition 5.38 we know that unitaries always send ONBs to ONBs. So, another way to show that the states (5.50) form an ONB is to show that they arise from applying some unitary to another ONB.

Exercise 5.97 Show that the Bell basis arises from applying the following unitary linear map:



to the product basis:

$$\left\{ \begin{array}{|c|} \hline i \\ \hline j \end{array} \right\}_{ij}$$

In other words, show that the states (5.50) can be expressed as:

$$\left\{ \sqrt{2} \begin{array}{|c|} \hline \begin{array}{|c|} \hline \begin{array}{|c|} \hline i \\ \hline \end{array} \end{array} \end{array} \right\}_{ij}$$

and hence that they form an ONB.

The first of the Bell states is just a cup multiplied by $\frac{1}{\sqrt{2}}$, while the other three are variations obtained by flipping bits or signs. From this, it is easy to see that the Bell states are all non-separable. Moreover, they are all maximally non-separable in the sense of Definition 4.80, a fact that plays a crucial role in quantum teleportation. We can show maximal non-separability by relating certain unitary linear maps, called the *Bell maps*, to the Bell states as follows:

$$\text{Diagram of } B_i \text{ (inverted triangle)} = \text{Diagram of } B_i \text{ (triangle) with a cup and a dashed box containing } \frac{1}{\sqrt{2}} \quad (5.51)$$

where:

$$\text{Diagram of } B_i \text{ (triangle)} := \sqrt{2} \times \text{Diagram of } B_i \text{ (inverted triangle) with a cup}$$

Exercise 5.98 Show that the matrix forms of the Bell maps are:

$$\begin{aligned} B_0 &:= \begin{array}{c} \text{Diagram of } B_0 \\ \text{triangle} \end{array} = \begin{array}{c} \text{Diagram of } B_0 \\ \text{triangle} \end{array} + \begin{array}{c} \text{Diagram of } B_0 \\ \text{triangle} \end{array} \\ B_1 &:= \begin{array}{c} \text{Diagram of } B_1 \\ \text{triangle} \end{array} = \begin{array}{c} \text{Diagram of } B_1 \\ \text{triangle} \end{array} + \begin{array}{c} \text{Diagram of } B_1 \\ \text{triangle} \end{array} \\ B_2 &:= \begin{array}{c} \text{Diagram of } B_2 \\ \text{triangle} \end{array} = \begin{array}{c} \text{Diagram of } B_2 \\ \text{triangle} \end{array} - \begin{array}{c} \text{Diagram of } B_2 \\ \text{triangle} \end{array} \\ B_3 &:= \begin{array}{c} \text{Diagram of } B_3 \\ \text{triangle} \end{array} = \begin{array}{c} \text{Diagram of } B_3 \\ \text{triangle} \end{array} - \begin{array}{c} \text{Diagram of } B_3 \\ \text{triangle} \end{array} \end{aligned} \quad (5.52)$$

and hence their corresponding matrices are:

$$\begin{aligned} B_0 &\Leftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & B_1 &\Leftrightarrow \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ B_2 &\Leftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} & B_3 &\Leftrightarrow \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \end{aligned} \quad (5.53)$$

Also show that these linear maps are unitary.

Unsurprisingly, we call the matrices associated with the Bell maps the *Bell matrices*. However, in most textbooks, one doesn't tend to encounter the Bell matrices, but rather their close cousins.

Remark 5.99 The following matrices are known as the *Pauli matrices*:

$$\begin{aligned}\sigma_0 = 1 &:= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & \sigma_1 = \sigma_X &:= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ \sigma_2 = \sigma_Y &:= \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} & \sigma_3 = \sigma_Z &:= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\end{aligned}$$

They are almost exactly the same as the Bell matrices:

$$\sigma_0 = B_0 \quad \sigma_1 = B_1 \quad \sigma_3 = B_2$$

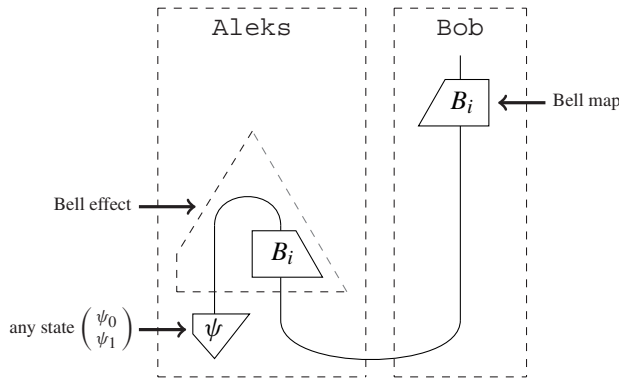
where the only difference is that:

$$\sigma_2 = iB_3 \quad (5.54)$$

We shall see in the next chapter that multiplying by i has no effect on the physical meaning of this process. So why should one care about these Pauli matrices? We don't! But others do. The reason for (5.54) is to make all of the Pauli matrices self-adjoint. In the usual presentation of quantum theory, self-adjoint matrices are used to define measurements (see Remark 7.25). However, our definition of measurement doesn't depend on self-adjoint linear maps, so we'll stick to Bell maps.

Note from equation (5.51) that one obtains all Bell states by applying certain processes to a fixed state. This fact is directly related to how the Bell states and Bell maps play a role in quantum teleportation. We will explain this in detail in the next chapter, but the following is a preview.

Exercise 5.100 Compute the state:



using the matrix representations of all of its ingredients and conclude that, up to a number, we indeed obtain ψ , for every i .

The relationship that the Bell basis and the Bell maps enjoy is a particular case of a general relationship between ONBs and certain families of linear maps induced by processes-state duality.

Definition 5.101 A set of linear maps:

$$\left\{ \begin{array}{c} |B \\ \boxed{i} \\ |A \end{array} \right\}_i \quad (5.55)$$

is called a *Hilbert–Schmidt ONB* if the associated bipartite states:

$$\begin{array}{c} | \\ \diagdown \\ \boxed{i} \\ \diagup \\ | \end{array} = \frac{1}{\sqrt{D}} \begin{array}{c} | \\ \boxed{i} \\ | \end{array} \quad (5.56)$$

form an ONB, for $D = \dim(A)$.

Taking the inner product of states (5.56) yields:

$$\begin{array}{c} \diagup \\ \boxed{j} \\ \diagdown \\ \boxed{i} \\ \diagup \end{array} = \frac{1}{D} \begin{array}{c} \boxed{j} \\ \boxed{i} \end{array} = \frac{1}{D} \text{tr} \left(\begin{array}{c} \boxed{j} \\ \boxed{i} \end{array} \right)$$

The rightmost expression (without the $\frac{1}{D}$) is sometimes known as the *Hilbert–Schmidt inner product* of linear maps. Evidently, we have an alternative expression for orthonormality of (5.55) in terms of this inner product:

$$\text{tr} \left(\begin{array}{c} \boxed{j} \\ \boxed{i} \end{array} \right) = D \delta_i^j$$

Recalling Definition 4.80 of maximally non-separable states, the following are moreover equivalent:

- the basis states $\begin{array}{c} | \\ \diagdown \\ \boxed{i} \\ \diagup \\ | \end{array}$ are maximally non-separable, and
- the corresponding basis maps $\begin{array}{c} | \\ \boxed{i} \\ | \end{array}$ are unitary.

5.4 Hilbert Spaces versus Diagrams

So what do string diagrams really have to do with Hilbert spaces and linear maps? We defined Hilbert spaces and linear maps as a special kind of string diagrams. However, in order to get there, we needed to assume bases, sums, and the fact that the numbers are

complex numbers, which all has a bit of a brute-force feel to it. Moreover, many other things are also described by string diagrams, for example, sets and relations, so the relationship with Hilbert spaces and linear maps may not be that special.

However, we will present a surprising result, tightly relating diagrammatic reasoning to Hilbert spaces and linear maps. Given that the additional structure of Hilbert spaces allows us to prove lots and lots of stuff about processes, one might be tempted to think that we can prove more equations between diagrams of linear maps than we could for just plain old diagrams, i.e. without using bases, sums, etc. This turns out not to be the case. The equations that are true between diagrams of linear maps are precisely those that can be proven just by deforming diagrams. So, the first thing we will see in this section is a precise statement of that fact.

The second thing we'll do is derive the usual, set-theoretic definition of Hilbert space from the one based on string diagrams. We'll see in Section 5.4.2 that this definition, along with linear maps and the tensor product, falls out automatically by regarding the system-types for **linear maps** not simply as labels, but instead as the sets of all their states.

Note that material in the next two sections is optional and won't be used explicitly later in this book. However, many readers may find them useful for understanding the power of diagrammatic proofs and connecting our notion of Hilbert space to the usual one encountered in the literature.

5.4.1 String Diagrams Are Complete for Linear Maps

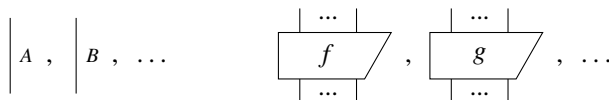
In this section we aim to characterise which new equations between diagrams appear once we assume that the diagrams actually represent linear maps, with the extra structure of sums, bases, and complex numbers around. And as already indicated above, the quite surprising answer is: none! That is, the language of string diagrams is already powerful enough to capture all of the equations that hold for generic diagrams of linear maps. Using terminology borrowed from logic:

String diagrams are *complete* for **linear maps**.

Another technical way to state this, which emphasises the fact that adding ONBs, sums, and complex numbers is 'innocent', is as follows:

linear maps are a *conservative extension* of string diagrams.

Recall from Section 3.1.2 that the data making up a string diagram are the boxes in it, including specification of the input and output wires, and how these boxes are wired together. That is, by a 'string diagram' we mean the 'drawing' of boxes and wires without interpretation within a process theory. In the same vein, string diagrams are equal if they can be deformed into each other. In order to distinguish wires and boxes, we used labels:

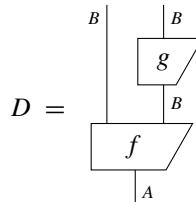


Diagrams also come with a notion of vertical reflection, which is crucial to the completeness result we will soon present. This result is about comparing these ‘free’, or ‘interpretation-free’, diagrams, with diagrams interpreted in the process theory of **linear maps**. Let us make the idea of interpretation a bit more formal.

Definition 5.102 An *interpretation in linear maps* $\llbracket \cdot \rrbracket$ of a string diagram D consists of a choice of Hilbert space $\llbracket A \rrbracket$ for every type A occurring in D and a choice of linear map $\llbracket f \rrbracket$ (respecting input/output types) for every box f occurring in D .

In particular, an interpretation defines a linear map $\llbracket D \rrbracket$ obtained by plugging all the linear maps $\llbracket f \rrbracket$, $\llbracket g \rrbracket$, ... into the diagram D and evaluating them as linear maps (using e.g. Theorem 5.61).

Example 5.103 Consider this diagram:



The diagram D has two types and two boxes, so here is an interpretation:

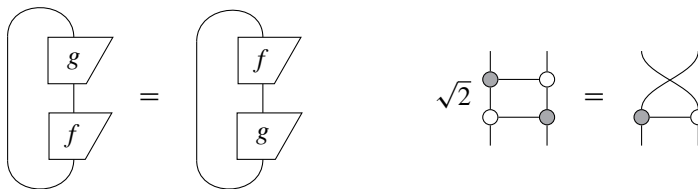
$$\left\{ \begin{array}{l} \llbracket A \rrbracket := \mathbb{C}^3, \quad \llbracket B \rrbracket := \mathbb{C}^2, \quad \llbracket f \rrbracket := \begin{pmatrix} 1 & 2 & 0 \\ 3 & 1 & 4 \\ 5 & 5 & 5 \\ 0 & 1 & 0 \end{pmatrix}, \quad \llbracket g \rrbracket := \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \end{array} \right\}$$

Then, under this interpretation:

$$\llbracket D \rrbracket = \begin{pmatrix} 1 & 2 & 0 \\ 4 & 3 & 4 \\ 5 & 5 & 5 \\ 5 & 6 & 5 \end{pmatrix}$$

Similarly, we can give an interpretation for a collection of diagrams D, E, F, \dots by interpreting all of the boxes and wires occurring in any of the diagrams, which in turn yields linear maps $\llbracket D \rrbracket, \llbracket E \rrbracket, \llbracket F \rrbracket, \dots$

Now consider the following two equations, which we encountered earlier:



Both equations are valid in the process theory of **linear maps**; however, in the first one f and g can be any linear maps (with appropriately matching types), while the second one only holds in the specific case of CNOT. In other words, the second equation involves only a single interpretation of the diagrams in **linear maps**, while the first one ranges over all possible interpretations in **linear maps**. The equations we are concerned with here are those of first kind, involving all possible interpretations in **linear maps**.

We are now ready to state the completeness theorem.

Theorem 5.104 String diagrams are complete for **linear maps**. That is, for any two string diagrams D, E , the following are equivalent:

- $D = E$
- For all interpretations of D, E into **linear maps**, $\llbracket D \rrbracket = \llbracket E \rrbracket$.

Proof (sketch) The proof of this theorem goes beyond the scope of this book. Roughly speaking, it involves taking some diagram D and defining an interpretation that is ‘characteristic’ of D . That is, the interpretation yields some fixed linear map L for D , and for any other $D' \neq D$, it is possible to show that $\llbracket D' \rrbracket$ cannot possibly be equal to L . \square

So the bottom line is:

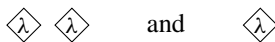
An equation between string diagrams holds for all Hilbert spaces and linear maps if and only if the string diagrams are the same.

This provides compelling evidence for the fact that string diagrams capture the compositional content of linear maps.

The condition that $\llbracket D \rrbracket = \llbracket E \rrbracket$ for all interpretations is fairly strong, so one still might ask if Theorem 5.104 holds for any process theory that admits string diagrams. This is not the case.

Theorem 5.105 String diagrams are not complete for **relations**. That is, there exist non-equal string diagrams $D \neq E$, where for all interpretations as **relations**, $\llbracket D \rrbracket = \llbracket E \rrbracket$.

Proof We just need to find two distinct string diagrams that are equal for all interpretations as **relations**. Recall that the only numbers in **relations** are 0 and 1. In both cases $\lambda^2 = \lambda$, so the following two diagrams:



are equal for all interpretations as **relations**. However, they are clearly not equal as string diagrams. \square

In other words, there are equations between string diagrams that hold for (all) sets and relations that are not string diagram equalities. In light of the fact that the only real difference between **linear maps** and **relations** is the choice of numbers, it might come as a surprise that something very simple like the booleans breaks completeness, whereas \mathbb{C} , with all of its extra structure, retains it.

5.4.2 The Set-Theoretic Definition of Hilbert Spaces

This section can safely be skipped by the reader who is perfectly happy with our definition of the process theory of **linear maps** (and hence Hilbert spaces). The main goal here is to connect our definition to the traditional one. In other words, we will define a Hilbert space H as a set with some extra structure (sums, scalar multiplication, inner product) and linear maps as functions that preserve (some of) this structure. Compound systems are formed by constructing the *tensor product* of Hilbert spaces, which we will also define. We provide this alternative definition for two reasons:

- for comprehensiveness, so the reader has some exposure to the traditional definition of Hilbert space, and
- for contrast, to emphasise the different spirit of the two definitions.

Recall from Section 3.4.1 that for **functions**, the states of type A are in one-to-one correspondence with the elements of the set A . We will define a set-theoretic counterpart \tilde{H} to ‘our Hilbert spaces’ H (i.e. system-types in **linear maps**) by mimicking this one-to-one correspondence:

$$\tilde{H} := \left\{ \begin{array}{c} | \\ \text{ } \\ \psi \\ \text{ } \end{array} \middle| \psi \text{ is a state of type } H \right\}$$

That is, given a type H in **linear maps** we define a *Hilbert space* \tilde{H} to be its set of states. Instead of states, we call the elements of \tilde{H} *vectors*. In what follows we show that we also get all of the set-theoretic Hilbert space operations (along with the tensor product!) ‘for free’ by expressing them in terms of states. We begin with sums and scalar multiplication.

Since **linear maps** has sums, we have the following:

- $\begin{array}{c} | \\ 0 \\ \text{ } \end{array} \in \tilde{H}$
- if $\begin{array}{c} | \\ \psi \\ \text{ } \end{array} \in \tilde{H}$, $\begin{array}{c} | \\ \phi \\ \text{ } \end{array} \in \tilde{H}$ then $\begin{array}{c} | \\ \psi \\ \text{ } \end{array} + \begin{array}{c} | \\ \phi \\ \text{ } \end{array} \in \tilde{H}$
- if $\begin{array}{c} \lambda \\ \text{ } \end{array} \in \mathbb{C}$, $\begin{array}{c} | \\ \psi \\ \text{ } \end{array} \in \tilde{H}$ then $\begin{array}{c} \lambda \\ \text{ } \end{array} \begin{array}{c} | \\ \psi \\ \text{ } \end{array} \in \tilde{H}$

In other words, the set of states in H forms a *complex vector space*.

Definition 5.106 A *complex vector space* is a set \tilde{V} with a distinguished element $0 \in \tilde{V}$ and two operations:

- $\psi + \phi$ for all $\psi, \phi \in \tilde{V}$
- $\lambda \cdot \psi$ for all $\lambda \in \mathbb{C}$ and $\psi \in \tilde{V}$

called *sum* and *scalar multiplication*, respectively, satisfying:

1. $0 + \psi = \psi$
2. $(\psi + \phi) + \xi = \psi + (\phi + \xi)$

3. $\psi + \phi = \phi + \psi$
4. $0 \cdot \psi = 0$
5. $\lambda \cdot (\lambda' \cdot \psi) = (\lambda\lambda') \cdot \psi$
6. $\lambda \cdot (\psi + \phi) = \lambda \cdot \psi + \lambda \cdot \phi$
7. $(\lambda + \lambda') \cdot \psi = \lambda \cdot \psi + \lambda' \cdot \psi$

for all $\psi, \phi, \xi \in \tilde{V}$ and $\lambda, \lambda' \in \mathbb{C}$.

We typically write the scalar product $\lambda \cdot \psi$ simply as $\lambda\psi$. As in Section 5.1.3, once it is established that $(- + -)$ is associative and commutative, it is more convenient to use summation notation, and replace equations 6 and 7 above with their equivalents using the summation symbol:

$$6'. \quad \lambda \left(\sum_i \psi_i \right) = \sum_i (\lambda \psi_i)$$

$$7'. \quad \left(\sum_i \lambda_i \right) \psi = \sum_i (\lambda_i \psi)$$

Exercise 5.107 Use **Conditions 1–3** in Definition 5.21 to show that \tilde{H} satisfies the seven equations of a complex vector space.

In **linear maps** every system-type has a finite basis:

$$\mathcal{B} = \left\{ \begin{array}{c} \downarrow \\ \boxed{i} \end{array} \right\}_i$$

From Theorem 5.30 it follows that any $\psi \in \tilde{H}$ can be written in this basis:

$$\begin{array}{c} \downarrow \\ \boxed{\psi} \end{array} = \sum_i \begin{array}{c} \downarrow \\ \boxed{\psi_i} \end{array} \begin{array}{c} \downarrow \\ \boxed{i} \end{array}$$

This means that our complex vector space is *finite dimensional*.

Definition 5.108 A *basis* for a vector space \tilde{V} is a minimal set of vectors

$$\{\phi_i\}_i \subseteq \tilde{V}$$

that *spans* \tilde{V} ; that is, for all $\psi \in \tilde{V}$ there exist $\{\phi_i\}_i \subseteq \mathbb{C}$ such that:

$$\psi = \sum_i \lambda_i \phi_i \tag{5.57}$$

If $\{\phi_i\}_i$ is finite, then \tilde{V} is *finite-dimensional* and the number of elements in $\{\phi_i\}_i$ is the *dimension* of \tilde{V} .

The theory of **linear maps** has adjoints, so we can form:

$$\begin{array}{c} \boxed{\psi} \\ \downarrow \\ \boxed{\phi} \end{array}$$

for all $\psi, \phi \in \tilde{H}$. This gives us all the structure we need to make the following definition.

Definition 5.109 A *finite-dimensional Hilbert space* is a finite-dimensional vector space \tilde{H} , with an additional operation:

$$\langle - | - \rangle : \tilde{H} \times \tilde{H} \rightarrow \mathbb{C}$$

called the *inner product*, satisfying:

1. $\langle \psi | \lambda \phi + \xi \rangle = \lambda \langle \psi | \phi \rangle + \langle \psi | \xi \rangle$
2. $\langle \lambda \psi + \phi | \xi \rangle = \bar{\lambda} \langle \psi | \xi \rangle + \langle \phi | \xi \rangle$
3. $\langle \psi | \phi \rangle = \overline{\langle \phi | \psi \rangle}$
4. $\langle \psi | \psi \rangle > 0$ for all $\psi \neq 0$

where $\overline{(-)}$ is complex conjugation.

Note that 1 and 3 imply 2. We already encountered most of these properties in Proposition 4.51, with the exception of the sum-preservation part of condition 1, which as we saw in Section 5.1.3, amounts to:

$$\sum_i \begin{array}{c} \triangle \\ \phi \\ \hline \psi_i \\ \triangle \end{array} = \sum_i \begin{array}{c} \triangle \\ \lambda_i \\ \hline \psi_i \\ \triangle \end{array}$$

and similarly for condition 2.

Remark* 5.110 We only define finite-dimensional Hilbert spaces here, because that is all we need for this book. The definition of an arbitrary Hilbert space is more complicated, as the set of vectors is required to be topologically closed under taking limits of certain sequences of vectors. This condition is automatically satisfied in finite dimensions.

The usual notion of ONB for a Hilbert space matches ours.

Definition 5.111 A basis $\{\phi_i\}_i$ for a finite-dimensional Hilbert space \tilde{H} is *orthonormal* if we have:

$$\langle \phi_i | \phi_j \rangle = \delta_i^j$$

The next thing to do is to introduce linear maps as certain functions between sets. General *multilinear* maps (i.e. maps with many inputs/outputs) require a bit of work to define using this definition of Hilbert space. However, maps with precisely one input and one output are straightforward. Given Hilbert spaces \tilde{H} and \tilde{K} , by considering functions of the form:

$$\tilde{f} : \tilde{H} \rightarrow \tilde{K} :: \begin{array}{c} \triangle \\ \psi \\ \triangle \end{array} \mapsto \begin{array}{c} \triangle \\ f \\ \hline \psi \\ \triangle \end{array}$$

we indeed recover the usual properties of linear maps.

Definition 5.112 For vector spaces (or Hilbert spaces) \tilde{H} and \tilde{K} , a *linear map* $f : \tilde{H} \rightarrow \tilde{K}$ is a function from the set \tilde{H} to the set \tilde{K} satisfying:

$$f(\lambda\psi) = \lambda f(\psi) \quad \text{and} \quad f(\psi + \phi) = f(\psi) + f(\phi)$$

These two conditions are referred to as *linearity*.

We can express both conditions more compactly using summation:

$$f\left(\sum_i \lambda_i \psi_i\right) = \sum_i \lambda_i f(\psi_i) \quad (5.58)$$

As we saw in Section 5.1.3, this amounts to:

The diagram shows an equality between two expressions. On the left, there is a summation over index i of a diamond-shaped node labeled λ_i connected to a trapezoidal node labeled f , which is then connected to an inverted trapezoidal node labeled ψ_i . On the right, there is a summation over index i of a diamond-shaped node labeled λ_i connected to a trapezoidal node labeled f , which is then connected to an inverted trapezoidal node labeled ψ_i . The two expressions are separated by an equals sign.

Remark 5.113 We initially defined the set-theoretic counterpart to Hilbert spaces by mimicking the one-to-one correspondence between sets A and functions from the set representing the ‘no wire’-type to the set A . In the case of Hilbert spaces, it’s linearity that makes this one-to-one correspondence work. First, note that \mathbb{C} forms a particularly simple vector space, for which the number 1 by itself forms a one-element ONB, since all the other numbers $\lambda \in \mathbb{C}$ can be decomposed as follows (cf. equation (5.57)):

$$\lambda = \lambda \cdot 1$$

and the orthonormality condition is trivially satisfied. Addition of ‘vectors’ in \mathbb{C} is just addition of complex numbers, by rule 7 from Definition 5.106:

$$\lambda + \lambda' = \lambda \cdot 1 + \lambda' \cdot 1 = (\lambda + \lambda') \cdot 1 = \lambda + \lambda'.$$

Now, a vector $\psi \in \tilde{H}$ defines a unique linear map:

$$\tilde{\psi} : \mathbb{C} \rightarrow \tilde{H} :: 1 \mapsto \psi$$

since by linearity we now have, for all $\lambda \in \mathbb{C}$:

$$\tilde{\psi} :: \lambda = \lambda \cdot 1 \mapsto \lambda \cdot \psi$$

Taking \tilde{H} to be \mathbb{C} itself, each $\lambda \in \mathbb{C}$ defines a unique linear map:

$$\tilde{\lambda} : \mathbb{C} \rightarrow \mathbb{C} :: 1 \mapsto \lambda$$

Hence there also is a one-to-one correspondence between the elements of \mathbb{C} and the linear maps from \mathbb{C} to itself. Thus linear maps, vectors, and numbers in \mathbb{C} are all instances of the same thing, which you knew already, because they are all certain kinds of processes.

For string diagrams, and hence **linear maps**, the adjoint is:

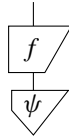
For traditional linear maps we have the following standard result.

Proposition 5.114 For Hilbert spaces \tilde{H}, \tilde{K} and a linear map $f : \tilde{H} \rightarrow \tilde{K}$ there exists a unique linear map f^\dagger such that for all $\psi \in \tilde{H}, \phi \in \tilde{K}$:

$$\langle \phi | f(\psi) \rangle = \langle f^\dagger(\phi) | \psi \rangle \quad (5.59)$$

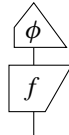
The unique linear map f^\dagger in the proposition is also called the *adjoint*. With string diagrams, equation (5.59) is a tautology. Indeed, since:

• $|f(\psi)\rangle$ translates as

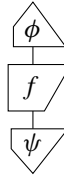


it follows that:

• $\langle f^\dagger(\phi) |$ translates as



and so both the LHS $\langle \phi | f(\psi) \rangle$ and RHS $\langle f^\dagger(\phi) | \psi \rangle$ of (5.59) are:

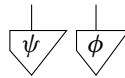


The notions of unitary and positive maps are identical for both the traditional presentation of Hilbert spaces and ours. That is, they are maps that take the form of Definition 4.56 and Definition 4.60, respectively.

The final, and trickiest, piece of the set-theoretic counterpart to **linear maps** is defining the parallel composition of Hilbert spaces. We need to define an operation $\tilde{\otimes}$ that mimics at the level of Hilbert spaces \tilde{H} and \tilde{K} the result of composing the types H and K in parallel. That is:

$$\tilde{H} \tilde{\otimes} \tilde{K} \text{ should match } \widetilde{H \otimes K}$$

For two Hilbert spaces \tilde{H} and \tilde{K} , let's look at some of the properties of $\widetilde{H \otimes K}$. Any states $\psi \in \tilde{H}$ and $\phi \in \tilde{K}$ yield a state:



in $\widetilde{H \otimes K}$. So, at first glance, one might be tempted to define $\tilde{H} \tilde{\otimes} \tilde{K}$ as the Cartesian product $H \times K$, which is what we did for **functions**. However, this quickly goes wrong, since $\widetilde{H \otimes K}$ contains lots of states that are non-separable. Recall from Section 5.2.2 that we can build a basis for $H \otimes K$ in terms of the individual bases of H and K . In particular, given ONBs:

$$\left\{ \begin{array}{c} | \\ i \end{array} \right\}_i \quad \text{and} \quad \left\{ \begin{array}{c} | \\ j \end{array} \right\}_j$$

for H and K respectively, we can write all states $\Phi \in \widetilde{H \otimes K}$ as follows:

$$\text{Diagram of } \Phi = \sum_i \text{Diagram of } \Phi_{ij} \text{ (diamond)} \text{Diagram of } i \text{ (triangle)} \text{Diagram of } j \text{ (triangle)}$$

So given Hilbert spaces \tilde{H} and \tilde{K} , we want to form a Hilbert space for which:

$$\left\{ \text{Diagram of } i \text{ (triangle)} \text{Diagram of } j \text{ (triangle)} \right\}_{ij}$$

plays the role of an ONB. First we need to form the set of vectors, and one way to do this is to simply consider the set of ‘formal sums’:

$$\tilde{H} \tilde{\otimes} \tilde{K} := \left\{ \sum_{ij} \lambda_{ij} \phi_i \tilde{\otimes} \phi'_j \mid \lambda_{ij} \in \mathbb{C} \right\}$$

where $\{\phi_i\}_i$ and $\{\phi'_j\}_j$ are ONBs of \tilde{H} and \tilde{K} , respectively. We say ‘formal sums’ since at the moment the sum-symbol really doesn’t have any meaning. Nor does the $\tilde{\otimes}$ -symbol between ϕ_i and ϕ'_j . We could just as well have represented these elements as lists of numbers indexed by (i, j) , i.e. matrices of the λ_{ij} coefficients. Next, we need to make this set into a vector space, by letting these formal sums act like ‘actual’ sums in our definition of vector addition and scalar multiplication:

$$\left(\sum_{ij} \lambda_{ij} \phi_i \tilde{\otimes} \phi'_j \right) + \left(\sum_{ij} \lambda'_{ij} \phi_i \tilde{\otimes} \phi'_j \right) := \sum_{ij} (\lambda_{ij} + \lambda'_{ij}) \phi_i \tilde{\otimes} \phi'_j$$

$$\lambda \cdot \left(\sum_{ij} \lambda_{ij} \phi_i \tilde{\otimes} \phi'_j \right) := \sum_{ij} \lambda \lambda_{ij} \phi_i \tilde{\otimes} \phi'_j$$

Also, in this form, we can clearly relate the elements of \tilde{H} and \tilde{K} to the corresponding ones in $\tilde{H} \tilde{\otimes} \tilde{K}$. More specifically, to

$$\sum_i \lambda_i \phi_i \in \tilde{H} \quad \text{and} \quad \sum_i \lambda'_i \phi'_i \in \tilde{K}$$

we can associate the element:

$$\sum_{ij} \lambda_i \lambda'_j \phi_i \tilde{\otimes} \phi'_j \in \tilde{H} \tilde{\otimes} \tilde{K}$$

We denote this mapping also by the symbol $\tilde{\otimes}$, and hence obtain:

$$\left(\sum_i \lambda_i \phi_i \right) \tilde{\otimes} \left(\sum_j \lambda'_j \phi'_j \right) = \sum_{ij} \lambda_i \lambda'_j \phi_i \tilde{\otimes} \phi'_j$$

This equation is called $\tilde{\otimes}$ -bilinearity. Also as the case of linearity, this follows diagrammatically just from shuffling numbers and summations around:

$$\sum_i \diamond \lambda_i \triangleright \phi_i \sum_j \diamond \lambda'_j \triangleright \phi'_j = \sum_{ij} \diamond \lambda_i \diamond \lambda'_j \triangleright \phi_i \triangleright \phi'_j$$

Horizontal composition of linear maps on separable vectors is defined as:

$$(f \tilde{\otimes} g)(\psi \tilde{\otimes} \phi) := f(\psi) \tilde{\otimes} g(\phi)$$

The definition for non-separable vectors then follows from linearity:

$$(f \tilde{\otimes} g)(\Psi) = (f \tilde{\otimes} g) \left(\sum_i \psi_i \tilde{\otimes} \phi_i \right) = \sum_i f(\psi_i) \tilde{\otimes} g(\phi_i)$$

Diagrammatically this is nothing but:

$$\begin{array}{c} \begin{array}{|c|c|} \hline f & g \\ \hline \end{array} \\ \hline \Psi \end{array} = \sum_i \begin{array}{c} \begin{array}{|c|c|} \hline f & g \\ \hline \end{array} \\ \hline \begin{array}{|c|c|} \hline \phi_i & \psi_i \\ \hline \end{array} \end{array} = \sum_i \begin{array}{c} \begin{array}{|c|c|} \hline f & g \\ \hline \end{array} \\ \hline \begin{array}{|c|c|} \hline \phi_i & \psi_i \\ \hline \end{array} \end{array}$$

Finally, it remains to show that $\tilde{H} \tilde{\otimes} \tilde{K}$ is a Hilbert space, i.e. that it has an inner product. For separable vectors in $\tilde{H} \tilde{\otimes} \tilde{K}$, we can define it as:

$$\langle \phi \tilde{\otimes} \phi' | \psi \tilde{\otimes} \psi' \rangle_{\tilde{H} \tilde{\otimes} \tilde{K}} := \langle \phi | \psi \rangle_{\tilde{H}} \langle \phi' | \psi' \rangle_{\tilde{K}}$$

The definition for non-separable vectors again follows from linearity:

$$\langle \Phi | \Psi \rangle_{\tilde{H} \tilde{\otimes} \tilde{K}} = \left\langle \sum_i \phi_i \tilde{\otimes} \phi'_i \left| \sum_j \psi_j \tilde{\otimes} \psi'_j \right. \right\rangle_{\tilde{H} \tilde{\otimes} \tilde{K}} = \sum_{ij} \langle \phi_i | \psi_j \rangle_{\tilde{H}} \langle \phi'_i | \psi'_j \rangle_{\tilde{K}}$$

As with the rest, this equation falls right out of the picture:

$$\begin{array}{c} \begin{array}{|c|} \hline \Phi \\ \hline \end{array} \\ \hline \begin{array}{|c|} \hline \Psi \\ \hline \end{array} \end{array} = \sum_i \sum_j \begin{array}{c} \begin{array}{|c|c|} \hline \phi_i & \phi'_i \\ \hline \end{array} \\ \hline \begin{array}{|c|c|} \hline \psi_j & \psi'_j \\ \hline \end{array} \end{array} = \sum_{ij} \begin{array}{c} \begin{array}{|c|c|} \hline \phi_i & \phi'_i \\ \hline \end{array} \\ \hline \begin{array}{|c|c|} \hline \psi_j & \psi'_j \\ \hline \end{array} \end{array}$$

Definition 5.115 The operation $\tilde{\otimes}$ on Hilbert spaces and linear maps that we constructed above is called the *tensor product*.

Remark 5.116 Above we explicitly relied on ONBs of Hilbert spaces in order to form their tensor product. This can be avoided, which then also shows that the tensor doesn't depend on the choices of ONBs. However, this ONB-independent construction is more involved.

Now we have all the ingredients to define a new process theory:

- Systems types are set-theoretic Hilbert spaces \tilde{H} .
- Processes are set-theoretic linear maps \tilde{f} .
- Horizontal composition is set-theoretic composition.

- Vertical composition is the tensor product $\widetilde{\otimes}$.
- The no-wire type is the one-dimensional Hilbert space \mathbb{C} .

Let's call it **linear maps**.

Theorem 5.117 **linear maps** and **linear maps** are equivalent process theories in the sense explained in Section 5.2.5.

5.5 Summary: What to Remember

1. **Linear maps** is a process theory admitting string diagrams such that:

- (1) Each type has a finite *orthonormal basis* (ONB).
- (2) There is at least one system-type of every dimension $D \in \mathbb{N}$.
- (3) Processes of the same type admit sums.
- (4) The numbers are the complex numbers \mathbb{C} .

Here, an ONB is a set of states satisfying:

$$\begin{array}{c} \triangleup_j \\ \downarrow \\ \triangleleft_i \end{array} = \delta_i^j$$

and:

$$\left(\text{for all } \triangleleft_i : \begin{array}{c} f \\ \triangleleft_i \end{array} = \begin{array}{c} g \\ \triangleleft_i \end{array} \right) \Rightarrow f = g$$

Admitting sums means that for any set $\{f_i\}_i$ of processes of the same type there exists a process of that type:

$$\sum_i \begin{array}{c} f_i \end{array}$$

such that these sums can always be pulled out of diagrams:

$$\left(\sum_i \begin{array}{c} h_i \\ \downarrow \\ g \end{array} \right) \begin{array}{c} g \\ \downarrow \\ f \end{array} = \sum_i \left(\begin{array}{c} h_i \\ \downarrow \\ g \end{array} \right) \begin{array}{c} g \\ \downarrow \\ f \end{array}$$

and preserve adjoints:

$$\left(\sum_i \begin{array}{|c|} \hline \diagup f_i \diagdown \\ \hline \end{array} \right)^\dagger = \sum_i \begin{array}{|c|} \hline \diagdown f_i \diagup \\ \hline \end{array}$$

A *Hilbert space* is a type in **linear maps**, and the traditional set-theoretic notion of Hilbert space arises as the set of all the states for a fixed type:

$$\tilde{H} := \left\{ \begin{array}{|c|} \hline \begin{array}{c} H \\ \diagdown \psi \diagup \end{array} \\ \hline \end{array} \mid \psi \text{ is a state of type } H \right\}$$

2. In **linear maps** each process admits a *matrix*:

$$\begin{array}{|c|} \hline \diagup f \diagdown \\ \hline \end{array} \leftrightarrow \begin{pmatrix} f_1^1 & f_1^2 & \cdots & f_1^m \\ f_2^1 & f_2^2 & \cdots & f_2^m \\ \vdots & \vdots & \ddots & \vdots \\ f_n^1 & f_n^2 & \cdots & f_n^m \end{pmatrix} \quad \text{where} \quad \begin{array}{|c|} \hline \begin{array}{c} j \\ \diagdown f_i^j \diagup \\ i \end{array} \\ \hline \end{array} := \begin{array}{|c|} \hline \begin{array}{c} j \\ \diagup f \diagdown \\ i \end{array} \\ \hline \end{array}$$

which allows them to be written in *matrix form*:

$$\begin{array}{|c|} \hline \diagup f \diagdown \\ \hline \end{array} = \sum_{ij} \begin{array}{|c|} \hline \begin{array}{c} j \\ \diagdown f_i^j \diagup \\ i \end{array} \\ \hline \end{array}$$

The matricial counterparts to sums, sequential composition, parallel composition, transposes, and adjoints of processes are usual matrix operations from linear algebra. The matrix **m** of a diagram:

$$\begin{array}{c} \begin{array}{c} A \\ \diagdown \psi \diagup \\ A \end{array} \begin{array}{c} B \\ \diagup g \diagdown \\ C \end{array} \begin{array}{c} C \\ \diagdown h \diagup \\ A \end{array} \end{array} \leftrightarrow \psi^{A_1 A_2} g_{B_1 A_2}^{B_1 C_1} h_{A_3}^{C_1}$$

is computed via its diagram formula:

$$\mathbf{m}_{i_3}^{i_1} = \sum_{i_2 j_1 k_1} \psi^{i_1 i_2} g_{j_1 i_2}^{j_1 k_1} h_{i_3}^{k_1}$$

3. The process theory of **linear maps** is equivalent to **matrices**(\mathbb{C}):

- (1) The systems are the natural numbers \mathbb{N} .
- (2) The processes with input type $m \in \mathbb{N}$ and output type $n \in \mathbb{N}$ are all $n \times m$ matrices with entries in \mathbb{C} .
- (3) Diagrams are computed as explained above.

Moreover, for any set of numbers X admitting sums, the matrix calculus **matrices**(X) is a process theory admitting string diagrams. The process theory **relations** (restricted to finite sets) also arises in this manner, as **matrices**(\mathbb{B}).

4. We introduced a handful of linear maps that will come in handy later:

- The *Bell basis* is:

$$B_0 := \frac{1}{\sqrt{2}} \left(\begin{array}{|c|} \hline 0 \\ \hline \end{array} \begin{array}{|c|} \hline 0 \\ \hline \end{array} + \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline \end{array} \right)$$

$$B_1 := \frac{1}{\sqrt{2}} \left(\begin{array}{|c|} \hline 0 \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline \end{array} + \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{|c|} \hline 0 \\ \hline \end{array} \right)$$

$$B_2 := \frac{1}{\sqrt{2}} \left(\begin{array}{|c|} \hline 0 \\ \hline \end{array} \begin{array}{|c|} \hline 0 \\ \hline \end{array} - \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline \end{array} \right)$$

$$B_3 := \frac{1}{\sqrt{2}} \left(\begin{array}{|c|} \hline 0 \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline \end{array} - \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{|c|} \hline 0 \\ \hline \end{array} \right)$$

- The corresponding *Bell maps* are:

$$B_0 \leftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$B_2 \leftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$B_1 \leftrightarrow \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$B_3 \leftrightarrow \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

- The *copy* maps for the Z-basis (white) and the X-basis (grey) are:

$$\text{Copy}_Z := \begin{array}{|c|} \hline 0 \\ \hline \end{array} \begin{array}{|c|} \hline 0 \\ \hline \end{array} + \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline \end{array}$$

$$\text{Copy}_X := \begin{array}{|c|} \hline 0 \\ \hline \end{array} \begin{array}{|c|} \hline 0 \\ \hline \end{array} + \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline \end{array}$$

- The *Hadamard* and *CNOT* maps are:

$$H := \begin{array}{|c|} \hline 0 \\ \hline \end{array} \begin{array}{|c|} \hline 0 \\ \hline \end{array} + \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline \end{array}$$

$$\sqrt{2} \text{CNOT} \leftrightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

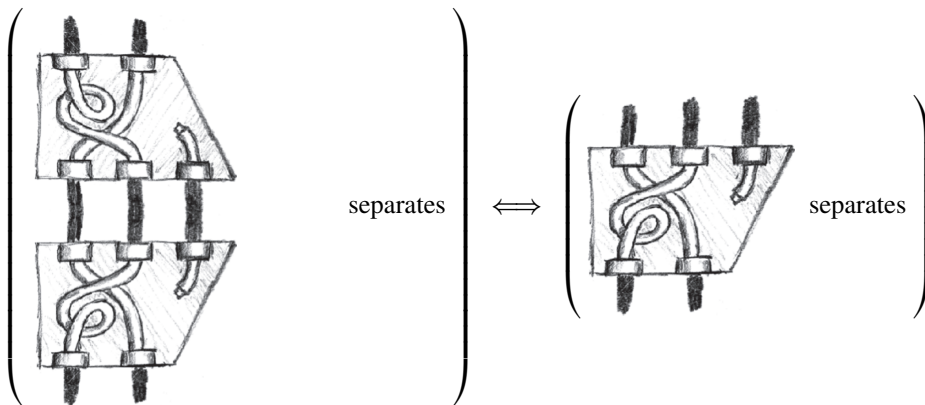
5. The *spectral theorem* states that self-adjoint linear maps *diagonalise*:

$$\begin{array}{c} | \\ \hline \diagup \quad \diagdown \\ f \\ \diagdown \quad \diagup \\ \hline | \end{array} = \sum_i r_i \begin{array}{c} | \\ \hline \diagdown \quad \diagup \\ i \\ \diagup \quad \diagdown \\ \hline | \end{array}$$

for some ONB and real numbers r_i . This has many consequences, an important one being that $f^\dagger \circ f$ detects separability of f :

$$\left(\exists \psi, \phi : \begin{array}{c} | \\ \hline \diagup \quad \diagdown \\ f \\ \diagdown \quad \diagup \\ \hline | \end{array} = \begin{array}{c} | \\ \hline \diagdown \quad \diagup \\ \phi \\ \diagup \quad \diagdown \\ \hline | \end{array} \right) \iff \left(\exists \psi', \phi' : \begin{array}{c} | \\ \hline \diagdown \quad \diagup \\ f \\ \diagup \quad \diagdown \\ \hline | \end{array} = \begin{array}{c} | \\ \hline \diagdown \quad \diagup \\ \phi' \\ \diagup \quad \diagdown \\ \hline | \end{array} \right)$$

Interpretationally, this fact naturally follows from taking seriously the idea that an adjoint really is a reflection:



⚠ The statement of the spectral theorem involving sums is only a temporary one. In Chapter 8, we will pictorialise this theorem.

6. String diagrams are *complete* for **linear maps**: the equations that hold between diagrams of linear maps are exactly all string diagram equations.
7. Many diagrammatic concepts have a corresponding representation using a (self-conjugate) ONB. Here is a summary:

Concept	Diagram	In terms of basis
process		$\sum_{ij} \diamond f_i^j$ with $\diamond f_i^j = \begin{array}{c} \triangle j \\ \\ f \\ \\ \triangle i \end{array}$
state		$\sum_i \diamond \psi^i$ with $\diamond \psi^i = \begin{array}{c} \triangle i \\ \\ \psi \\ \\ \triangle \end{array}$
identity		$\sum_i \begin{array}{c} \triangle i \\ \\ \triangle i \end{array}$
cups		$\sum_i \begin{array}{c} \triangle i \\ \\ \triangle i \end{array}$
caps		$\sum_i \begin{array}{c} \triangle i \\ \\ \triangle i \end{array}$
dimension		$\sum_i \square = \dim(H)$
trace		$\sum_i \begin{array}{c} \triangle i \\ \\ \square \\ \\ \triangle i \end{array}$
transpose		$\sum_{ij} \diamond f_j^i$ with $\diamond f_i^j$ as above
conjugate		$\sum_{ij} \diamond f_i^j$ with $\diamond f_i^j$ as above
adjoint		$\sum_{ij} \diamond \widehat{f_j^i}$ with $\diamond f_i^j$ as above

5.6 Advanced Material*

In this section, we look at the definition of Hilbert space beyond finite dimensions and highlight some interesting examples, as well as some of the difficulties they cause. After that, we continue our venture into category theory a bit more, by looking at how category theoreticians represent sums and ONBs. We also explain how knot theoreticians use a similar blend of diagrammatic reasoning and sums in their work. Last but not least, we explain how the pioneers of category theory effectively formulated what it means for two process theories to be equivalent.

5.6.1 Beyond Finite Dimensions*

Recall that the inner product gives us a norm: $\|v\| := \sqrt{\langle v|v \rangle}$. This plays a much bigger role in the infinite-dimensional case, because it allows one to talk about the convergence of sequences, which is the missing piece of the definition of a Hilbert space, which we can safely ignore in the finite-dimensional case. The full definition of a Hilbert space goes like this.

Definition 5.118 A (possibly infinite-dimensional) *Hilbert space* is a complex inner product space that is additionally *Cauchy complete*. That is, for any (Cauchy) convergent sequence $(v_i)_i$ of vectors in H , there exists $v \in H$ such that $\|v_i - v\| \rightarrow 0$.

This means we can take limits of sequences, so we can hit Hilbert spaces (and hence quantum mechanics) with a whole big bag of tools from *functional analysis*. For one thing, having limits around allows one to define infinite sums:

$$\sum_{i=0}^{\infty} \psi_i := \lim_{n \rightarrow \infty} \sum_{i=0}^n \psi_i$$

There are two examples of Hilbert spaces that have historically played a major role in quantum mechanics as (equivalent) presentations of the quantum mechanical state space.

Example 5.119 L^2 is the set of all functions $\psi : \mathbb{R}^n \rightarrow \mathbb{C}$ whose ‘squared-integrals’ are finite:

$$\int \overline{\psi(x)} \psi(x) dx < \infty$$

This forms a Hilbert space by letting:

$$(\lambda_1 \psi + \lambda_2 \phi)(x) := \lambda_1 \psi(x) + \lambda_2 \phi(x) \quad \text{and} \quad \langle \psi | \phi \rangle := \int \overline{\psi(x)} \phi(x) dx$$

Example 5.120 ℓ^2 is the set of all (countably infinite) sequences of complex numbers, whose ‘squared sums’ are finite:

$$\sum_{i=0}^{\infty} \overline{a_i} a_i < \infty$$

$$\lambda_1(a_i)_i + \lambda_2(b_i)_i := (\lambda_1 a_i + \lambda_2 b_i)_i \quad \text{and} \quad \langle (a_i)_i | (b_i)_i \rangle := \sum_{i=0}^{\infty} \overline{a_i} b_i$$

So, this brings us back to **Q3** from Section 2.3:

Or in the majority of quantum computing, for that matter? The short answer is that many things just don't seem to work as well. Suppose, for example, that we take the caps and cups that we know (and love) by now, and try to run them in infinite dimensions. Naïvely, things look good at first:

$$\begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} = \sum_{i=0}^{\infty} \left(\begin{array}{c} \text{---} \\ | \\ \triangle_i \end{array} \right) \sum_{j=0}^{\infty} \left(\begin{array}{c} \triangle_j \\ | \\ \text{---} \end{array} \right) = \sum_{ij=0}^{\infty} \left(\begin{array}{c} \triangle_j \\ | \\ \triangle_i \end{array} \right) = \sum_{i=0}^{\infty} \left(\begin{array}{c} \triangle_i \\ | \\ \triangle_i \end{array} \right) =$$

$$\boxed{\bigcirc} = \sum_{j=0}^{\infty} \sum_{i=0}^{\infty} \boxed{\begin{array}{cc} \triangle_j & \triangle_j \\ | & | \\ \triangle_i & \triangle_i \end{array}} = \sum_{i,j=0}^{\infty} \boxed{\begin{array}{cc} \triangle_j & \triangle_j \\ | & | \\ \triangle_i & \triangle_i \end{array}} = \sum_{i=0}^{\infty} \boxed{\phantom{\begin{array}{cc} \triangle_j & \triangle_j \\ | & | \\ \triangle_i & \triangle_i \end{array}}} = \infty \quad (5.60)$$

One might argue that this is a problem with string diagrams. On the other hand, situation (5.60) occurs not just for caps and cups, but for any perfect correlations. Thus:

It's far from clear whether this is a 'property of nature' or simply an artifact of the mathematics of Hilbert spaces. This suggests two interesting questions:

1. How much of the process-theoretic development of quantum theory can be extended to infinite dimensions (i.e. without recourse to string diagrams)?
2. Can the theory of Hilbert spaces and bounded linear maps be modified to accommodate string diagrams and/or perfect correlations in infinite dimensions?

Some progress has been made towards both of these questions (see Section 5.7), but there is still much to do.

5.6.2 Categories with Sums and Bases*

A large subdiscipline of category theory (sometimes called Abelian category theory) studies categories whose morphisms can be added. A category that has some extra structure on its sets of morphisms is called an *enriched category*. The extra structure we are interested in is that of a *commutative monoid*, i.e. a set M with an associative, commutative, and unital sum. For example, by Definition 5.21, **Condition 1**, processes of the same type in a process theory with sums form such a commutative monoid. **Condition 2** moreover guarantees that the categories corresponding to process theories with sums also have the following structure.

Definition 5.121 A category \mathcal{C} is said to be *enriched in commutative monoids* if for every pair of objects A and B , the set $\mathcal{C}(A, B)$ forms a commutative monoid and the monoid structure is compatible with \circ -composition:

$$\begin{cases} 0 \circ f = 0 = g \circ 0 \\ h \circ (f + g) = (h \circ f) + (h \circ g) \\ (g + h) \circ f = (g \circ f) + (h \circ f) \end{cases}$$

A related notion to the *sums* of morphisms is the *direct sum* of objects (e.g. vector spaces). In categorical terms, this is called a *biproduct*.

Definition 5.122 Let \mathcal{C} be a category that is enriched in commutative monoids. Then, \mathcal{C} has *biproducts* if for every pair of objects A_1, A_2 , there exists a third object $A_1 \oplus A_2$, along with a pair of maps:

$$\iota_j : A_j \rightarrow A_1 \oplus A_2$$

called *injections*, and a pair of maps:

$$\pi_j : A_1 \oplus A_2 \rightarrow A_j$$

called *projections*, satisfying:

$$\iota_1 \circ \pi_1 + \iota_2 \circ \pi_2 = 1_{A_1 \oplus A_2} \quad \pi_k \circ \iota_j = \begin{cases} 1_{A_j} & \text{if } j = k \\ 0 & \text{otherwise} \end{cases}$$

Look familiar yet? If not, let's get the dagger involved.

Definition 5.123 Let \mathcal{C} be a dagger compact closed category that is enriched in commutative monoids. Then, \mathcal{C} has *dagger biproducts* if for every pair of objects A_1, A_2 , there exists a third object:

$$A_1 \oplus A_2$$

along with a pair of maps:

$$\iota_j : A_j \rightarrow A_1 \oplus A_2$$

satisfying:

$$\iota_1 \circ \iota_1^\dagger + \iota_2 \circ \iota_2^\dagger = 1_{A_1 \oplus A_2} \quad \iota_k^\dagger \circ \iota_j = \begin{cases} 1_{A_j} & \text{if } j = k \\ 0 & \text{otherwise} \end{cases}$$

If $A_1 = A_2 := I$, the trivial object, these conditions look like this:

That's right:

$$\left\{ \begin{array}{c} \downarrow \\ \iota_k \\ \uparrow \end{array} \right\}_k$$

forms a two-dimensional ONB! Using the equations in Definition 5.123, it is possible to show that \oplus is associative (up to isomorphism), so we can define biproducts of three objects:

$$A \oplus B \oplus C := (A \oplus B) \oplus C \cong A \oplus (B \oplus C)$$

or, similarly, N objects. Objects that are N -fold biproducts of I :

$$A \cong I \oplus I \oplus \cdots \oplus I \quad (5.61)$$

then have N -dimensional ONBs. To see if an object admits something like an ONB, we can first decompose it into its *irreducible* components.

Definition 5.124 An object is called *irreducible* if it cannot be written as a biproduct of two non-zero objects.

The reason **relations** and **linear maps** have ONBs for every type is that the only irreducible object is the trivial system. In other words, we can decompose any object as in (5.61). However, there are many interesting algebraic categories (rings, modules, (representations of) algebras, etc.) that have a much richer collection of irreducibles. Interestingly, we still have a useful matrix calculus, but now instead of matrices of numbers we get matrices of processes. Suppose we look at some complicated map like this:

$$g : A_1 \oplus \cdots \oplus A_m \rightarrow B_1 \oplus \cdots \oplus B_n$$

Then g can be decomposed into a ‘matrix form’ that generalises the matrix form given in equation (5.14):

$$\begin{array}{c} \diagup \\ g \\ \diagdown \end{array} = \sum_{jk} \begin{array}{c} \diagup \\ \iota_k \\ g_j^k \\ \iota_j \\ \diagdown \end{array} \leftrightarrow \begin{pmatrix} g_1^1 & g_2^1 & \cdots & g_m^1 \\ g_1^2 & g_2^2 & \cdots & g_m^2 \\ \vdots & \vdots & \ddots & \vdots \\ g_1^n & g_2^n & \cdots & g_m^n \end{pmatrix}$$

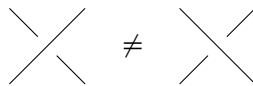
Rather than numbers, the components in the matrices are all morphisms

$$g_j^k : A_j \rightarrow B_k$$

If we start to compose matrices, everything looks just like normal matrix composition, where sums are sums and ‘multiplication’ is composition of morphisms. So, we can pretend we are doing normal matrix calculus, but actually, we’re doing something much more general (and powerful!).

5.6.3 Sums in Knot Theory*

In this book wires are like electrical wires, in the sense that only connections matter. In contrast, if they were ropes, then one may care if they are knotted up or braided together. In particular, one may wish to distinguish these two diagrams:



There is an area of mathematics called knot theory that is all about ropes either being knotted/braided or not. In fact, figuring out if a rope is knotted/braided at all turns out to be extremely difficult.

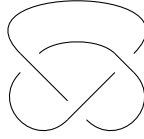
The study of knots and braids not only is of interest for its own sake but is also the foundation of a particular model of quantum computation called *topological quantum computation*, where the structure of braids is used to encode quantum gates. In the historical material at the end of this section we provide some pointers to the existing literature.

In the introduction we already gave an example of an equation that one encounters in any modern knot theory textbook:

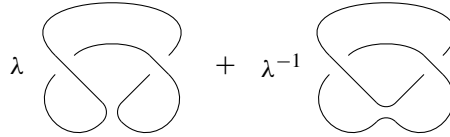
$$\begin{array}{c} \diagup \\ \diagdown \end{array} = \lambda \begin{array}{c} | \\ | \end{array} + \lambda^{-1} \begin{array}{c} \frown \\ \smile \end{array} \quad (5.62)$$

We will now explain its use, as an example of a hybrid calculus that uses both diagrams and traditional mathematical notions. The *Kauffman bracket* (5.62) allows one to associate a polynomial to any given knot.

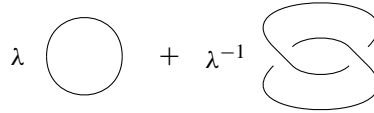
Consider the *trefoil knot*:



Applying (5.62) to the lowermost crossing we obtain:



Deforming the diagrams this becomes:



Then, a second application of (5.62) yields:

$$\lambda \bigcirc + \bigcirc + \lambda^{-2} \bigcirc = (\lambda + 1 + \lambda^{-2}) \bigcirc$$

Besides (5.62) a second equation is:

$$\bigcirc = 1$$

so we are left with a polynomial in λ :

$$\lambda + 1 + \lambda^{-2}$$

This polynomial is called the *bracket polynomial*, and a suitably normalised version of it is called the *Jones polynomial*, which is a *knot invariant*. That is, it is the same for all knots that can be deformed into each other. This makes it an extremely valuable tool for the classification of knots.

5.6.4 Equivalence of Symmetric Monoidal Categories*

In Section 5.2.5, we gave an informal definition of equivalence for process theories. In this section, we will make this definition formal, using the language of category theory.

In order to talk about equivalence of categories, we first must say what it means to interpret the morphisms (i.e. processes) in one symmetric monoidal category as morphisms in another. We do this using *functors*, which are the standard kind of ‘map’ one considers between categories.

Definition 5.125 A (strict) *symmetric monoidal functor* F from SMC \mathcal{C} to SMC \mathcal{D} consists of a function mapping objects of \mathcal{C} to objects of \mathcal{D} :

$$F : \text{ob}(\mathcal{C}) \rightarrow \text{ob}(\mathcal{D})$$

and for every pair of objects A, B , a function mapping morphisms of \mathcal{C} to morphisms of \mathcal{D} (also written as ‘ F ’):

$$F : \mathcal{C}(A, B) \rightarrow \mathcal{D}(F(A), F(B))$$

such that ‘ F preserves diagrams’, i.e.:

1. F preserves parallel composition and unit for objects:

$$F(A \otimes B) = F(A) \otimes F(B) \qquad F(I) = I$$

2. F preserves parallel and sequential composition for morphisms:

$$F(f \otimes g) = F(f) \otimes F(g) \qquad F(g \circ f) = F(g) \circ F(f)$$

3. F preserves identity morphisms and swaps:

$$F(1_A) = 1_{F(A)} \qquad F(\sigma_{A,B}) = \sigma_{F(A), F(B)}$$

As in the definition of monoidal category, the adjective ‘strict’ means equations involving parallel composition of objects are ‘real equations’; i.e. they hold on the nose, rather than only up to isomorphism. If we dropped it, condition 1 above would become:

$$F(A \otimes B) \cong F(A) \otimes F(B) \qquad F(I) \cong I$$

and we would need to require that these isomorphisms obey certain coherence equations like those for a non-strict SMC.

The simplest functor is the identity functor:

$$1_{\mathcal{C}} : \mathcal{C} \rightarrow \mathcal{C}$$

which just sends every object and morphism to itself. We can define sequential composition of functors in the obvious way:

$$(G \circ F)(X) = G(F(X)) \qquad (G \circ F)(f) = G(F(f))$$

in which case it clearly follows that:

$$F \circ 1_{\mathcal{C}} = F = 1_{\mathcal{D}} \circ F$$

The stricter sense in which two categories can be ‘the same’ is isomorphism, which is defined just like isomorphism between other kinds of objects.

Definition 5.126 Two symmetric monoidal categories are *isomorphic* if there exist symmetric monoidal functors:

$$F : \mathcal{C} \rightarrow \mathcal{D} \qquad G : \mathcal{D} \rightarrow \mathcal{C}$$

such that:

$$G \circ F = 1_{\mathcal{C}} \qquad F \circ G = 1_{\mathcal{D}}$$

In other words, if we take a round trip through F and G , we get back to exactly where we started. However, in many cases, such as the equivalence between **linear maps** and **matrices**(\mathbb{C}), we don't get back where we started. However, we end up somewhere that 'looks exactly the same' as where we started. In other words, for every object X , the object $G(F(X))$ is isomorphic to X . Let's give this isomorphism a name:

$$\eta_X : X \rightarrow G(F(X))$$

Of course, we don't get just a single such isomorphism but a whole family of them η_X, η_Y, \dots for every object in \mathcal{C} .

That tells us what it means for objects to look exactly the same after a round trip, but what does it mean for morphisms to look the same? In other words, what does it mean for:

$$f : X \rightarrow Y \quad \text{and} \quad G(F(f)) : G(F(X)) \rightarrow G(F(Y))$$

to do 'the same thing'? It means if we 'encode' X as $G(F(X))$ via the isomorphism η_X , then do $G(F(f))$, then 'decode' the result, this should be the same thing as just doing f :

$$\eta_Y^{-1} \circ G(F(f)) \circ \eta_X = f \tag{5.63}$$

This means that η is not just any old family of isomorphisms, but is what's called a *natural isomorphism*. By moving η_Y to the right, we have:

$$G(F(f)) \circ \eta_X = \eta_Y \circ f \tag{5.64}$$

a condition that is usually expressed using a *commutative diagram*:

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ \eta_X \downarrow & & \downarrow \eta_Y \\ G(F(X)) & \xrightarrow{G(F(f))} & G(F(Y)) \end{array} \tag{5.65}$$

A commutative diagram is a visual way to express equations between morphisms. It asserts that any two paths with the same start and end point yield the same morphism by sequential composition. So, the diagram above says exactly the same thing as equation (5.64).

Remark 5.127 The definition of natural isomorphism actually makes sense for any pair of functors:

$$D, E : \mathcal{C} \rightarrow \mathcal{D}$$

in which case it becomes a family of isomorphisms:

$$\{\iota_X : D(X) \rightarrow E(X)\}_{X \in \mathcal{C}}$$

satisfying:

$$\begin{array}{ccc} D(X) & \xrightarrow{D(f)} & D(Y) \\ \downarrow \iota_X & & \downarrow \iota_Y \\ E(X) & \xrightarrow{E(f)} & E(Y) \end{array}$$

Then, (5.65) is the special case where $D := 1_{\mathcal{C}}$ and $E := G \circ F$. We can also drop the requirement that each ι_X be an isomorphism, in which case such a family is called a *natural transformation*.

Example 5.128 Let $\mathcal{C} \times \mathcal{C}$ be the category whose objects are pairs (A, B) of objects in \mathcal{C} and whose morphisms are pairs (f, g) of morphisms in \mathcal{C} . Parallel composition in an SMC then induces two functors from the *product category* $\mathcal{C} \times \mathcal{C}$ to the category \mathcal{C} :

$$P_1 :: \begin{cases} (A, B) \mapsto A \otimes B \\ (f, g) \mapsto f \otimes g \end{cases} \quad P_2 :: \begin{cases} (A, B) \mapsto B \otimes A \\ (g, f) \mapsto g \otimes f \end{cases}$$

The swap morphisms in \mathcal{C} then give a natural isomorphism from P_1 to P_2 . Naturality in this case gives the following commutative diagram:

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

which, as an equation between diagrams, should look very familiar:

Similarly, the associativity and unit isomorphisms for non-strict SMCs mentioned in Section* 3.6.2, which respectively are:

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

$$\lambda_A : I \otimes A \rightarrow A \qquad \rho_A : A \otimes I \rightarrow A$$

also give natural isomorphisms for suitably defined functors.

The final ingredient we need for capturing ‘sameness’ of SMCs is to define a natural isomorphism that respects parallel composition of objects:

$$\eta_{X \otimes Y} = \eta_X \otimes \eta_Y$$

which is called a *monoidal natural isomorphism*. Then, an equivalence of SMCs is a pair of functors where each round trip ($G \circ F$ and $F \circ G$) yields something (monoidally, naturally) isomorphic to what we started with.

Definition 5.129 Two symmetric monoidal categories are *equivalent* if there exist symmetric monoidal functors $F : \mathcal{C} \rightarrow \mathcal{D}$ and $G : \mathcal{D} \rightarrow \mathcal{C}$ such that there exist monoidal natural isomorphisms:

$$\{\eta_X : X \rightarrow G(F(X))\}_{X \in \text{ob } \mathcal{C}} \quad \text{and} \quad \{\epsilon_Y : Y \rightarrow F(G(Y))\}_{Y \in \text{ob } \mathcal{D}}$$

So, we should be able to see the relationship between **linear maps** and **matrices**(\mathbb{C}) in these terms. As SMCs, these are usually called FHilb (for finite-dimensional Hilbert spaces) and $\text{Mat}(\mathbb{C})$, respectively. For Hilbert spaces A, B, \dots , fix bases:

$$\left\{ \begin{array}{c} \downarrow \\ \phi_i \end{array} \right\}_i, \quad \left\{ \begin{array}{c} \downarrow \\ \phi_j \end{array} \right\}_j, \quad \dots$$

Then, let:

$$F : \text{FHilb} \rightarrow \text{Mat}(\mathbb{C})$$


be the functor that sends each Hilbert space to its dimension and each linear map to the matrix with elements:

$$f_i^j = \begin{array}{c} \begin{array}{c} \triangleup \\ \phi_j \end{array} \\ \downarrow \\ \begin{array}{c} \square \\ f \end{array} \\ \downarrow \\ \begin{array}{c} \triangle \\ \phi_i \end{array} \end{array}$$

In the other direction, let:

$$G : \text{Mat}(\mathbb{C}) \rightarrow \text{FHilb}$$

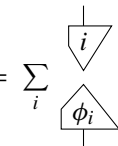
be the functor that sends each natural number $d \in \text{ob}(\text{Mat}(\mathbb{C}))$ to the Hilbert space \mathbb{C}^d and each matrix with entries f_i^j to the linear map:

$$\sum_{ij} f_i^j$$


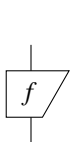
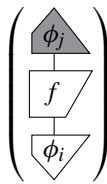
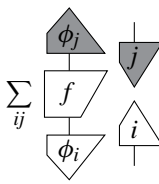
So, we have:

$$A \xrightarrow{F} d = \dim(A) \xrightarrow{G} \mathbb{C}^d$$

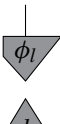
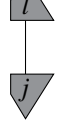




Now, $A \neq \mathbb{C}^d$, but since these two Hilbert spaces have the same dimension, there is a unitary:

$$\eta_A := \sum_i$$


which is, of course, also an isomorphism. If we now look at the round trip of a linear map, we have:

$$f \xrightarrow{F} \left(\begin{array}{c} \phi_j \\ f \\ \phi_i \end{array} \right)_{ij} \xrightarrow{G} \sum_{ij} \left(\begin{array}{c} \phi_j \\ f \\ \phi_i \end{array} \right)_{ij}$$




which is not exactly the same linear map. But it can be ‘decoded’ as we did in equation (5.63) using the natural isomorphisms:

$$\eta_B \circ G(F(f)) \circ \eta_A = \sum_{ij} \left(\begin{array}{c} \phi_l \\ l \\ \phi_j \\ f \\ \phi_i \\ i \\ k \\ \phi_k \end{array} \right) = \sum_{ij} \left(\begin{array}{c} \phi_j \\ f \\ \phi_i \end{array} \right) = f$$







The other round trip is even easier, because it just sends a matrix to itself:

$$\mathbf{f} \xrightarrow{G} \sum_{ij} \mathbf{f}_i^j \begin{array}{c} \downarrow j \\ \triangle \\ \uparrow i \end{array} \xrightarrow{F} \mathbf{f}$$

So the second natural isomorphism is just given by identity morphisms:

$$\begin{array}{c} \downarrow \\ \triangle \\ \uparrow \end{array} \epsilon_d := \left| \right.$$

for all $d \in \text{ob}(\text{Mat}(\mathbb{C}))$.

5.7 Historical Notes and References

The name ‘Hilbert space’ was coined by John von Neumann in his mathematical formulation of quantum theory, ultimately culminating in his famous book on the subject (von Neumann, 1932). Hilbert spaces united two seemingly different mathematical formalisations of quantum mechanics: the wave-function picture due to Schrödinger (1926) and matrix mechanics originating with Heisenberg (1925) and formalised by Born and Jordan (1925). In mathematics, many examples of Hilbert spaces had already been studied extensively, including by David Hilbert when studying integral equations, but it was von Neumann who was the first one to provide an axiomatic treatment, so they may as well have been coined ‘von Neumann spaces’. A detailed account on the history of Hilbert spaces can be found in Bourbaki (1981, 1987).

The name ‘matrix’ was coined by Sylvester in 1848. The origins of the complex numbers trace back way over 1000 years, to Arabic algebra. Euler introduced the notation i and Hamilton was the first to treat complex numbers as a pair of real numbers, which constituted the first algebraic definition of the complex numbers. Gauss coined the name ‘complex number’.

As mentioned in Remark 5.99, the Bell maps are typically introduced as the (marginally different) Pauli matrices, which first occurred in the Pauli equation, a variation on Schrödinger’s equation that takes into account the interaction of the spin of a particle with an external electromagnetic field.

Unfortunately, in the literature, the terminology concerning Bell states and the Bell basis is a bit all over the place, mainly for historical reasons. First, the ‘Bell state’ typically refers to the first element of the Bell basis (i.e. the cup), whereas the ‘Bell states’ refer to the whole basis. Sometimes the Bell state is also called the ‘EPR-state’, but more commonly, *EPR-state* is understood to be B_3 from (5.50). The reason for singling out this state is that it is the only *antisymmetric* state of the Bell states, and in fact, the only antisymmetric state for a pair of two-dimensional quantum systems. Here, *antisymmetry* means that if we swap the two systems, we get the same state up to (-1) :

term-wise swap
↓

$$\begin{array}{c} \text{---} \end{array} \begin{array}{c} \diagup \quad \diagdown \\ \diagdown \quad \diagup \end{array} \begin{array}{c} \text{---} \end{array} = \begin{array}{c} \text{---} \end{array} \begin{array}{c} \diagup \quad \diagdown \\ \diagdown \quad \diagup \end{array} \begin{array}{c} \text{---} \end{array} - \begin{array}{c} \text{---} \end{array} \begin{array}{c} \diagup \quad \diagdown \\ \diagdown \quad \diagup \end{array} \begin{array}{c} \text{---} \end{array} = \begin{array}{c} \text{---} \end{array} \left(\begin{array}{c} \diagup \quad \diagdown \\ \diagdown \quad \diagup \end{array} \begin{array}{c} \text{---} \end{array} - \begin{array}{c} \text{---} \end{array} \begin{array}{c} \diagup \quad \diagdown \\ \diagdown \quad \diagup \end{array} \begin{array}{c} \text{---} \end{array} \right)$$

Systems that obey this antisymmetry are called *fermions*. The three other Bell states are *bosons*, which means that if we swap the two systems, we get the same state on the nose.

The use of sums and matrix structure in addition to string diagrams for modelling quantum systems was initiated by Abramsky and Coecke (2004), in the form of categorical biproducts, the category-theoretic counterpart to having a matrix calculus, as explained in Section* 5.6.2. On a personal note, the second author of that paper never really liked biproducts. If one were to stick with biproducts, this chapter would have been where this book ends. Aiming beyond biproducts has produced the remaining 66.6%.

Biproducts arose from the study of Abelian categories, or categories that resemble the category of Abelian groups in several important ways. These first occurred in Mac Lane (1950), following discussions with Eilenberg about the appropriate categories for treating homology and cohomology of topological spaces. Following further developments by Buchsbaum (1955) and Grothendieck (1957), such categories grew to play a role in mainstream algebraic topology and geometry. A standard text is Freyd (1964).

Category theory was invented by Eilenberg and Mac Lane (1945) precisely to say what it means for process theories to be equivalent. However, the concept that a morphism should be thought of as a process didn't become widespread until much later (see Baez and Lauda, 2011). The process theories Eilenberg and Mac Lane originally had in mind only had sequential composition, not parallel composition, so their notion of equivalence only had to preserve \circ . However, with the advent of parallel composition, i.e. monoidal categories (Benabou, 1963; Mac Lane, 1963), came the notion of monoidal equivalence of categories introduced in Section* 5.6.4.

Selinger (2011a) proved completeness of string diagrams for **linear maps**, building further on a similar result for vector spaces (i.e. without adjoints) due to Hasegawa, Hofmann, and Plotkin (2008).

Discussions of quantum protocols in terms of non-self-dual cups and caps, including the upshot of doing so, are in Coecke et al. (2008a), Paquette (2008), and Kissinger (2012a). Much earlier they had been extensively studied in mathematics, for example in the area of quantum groups (Kassel, 1995). The Jones polynomial, which first appeared in Jones (1985), earned Vaughan Jones the Fields Medal, sometimes also called the Nobel Prize for mathematics. A few years later Louis Kauffman introduced his bracket as an easy manner to produce the Jones polynomial, in Kauffman (1987). Kauffman (1991) surveys the interplay between knots and physics. Panangaden and Paquette (2011) survey topological quantum computing, and the interplay between crossings and the kind of diagrams this book is all about is surveyed in Fauser (2013).