4

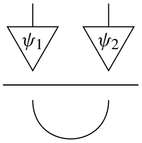
String Diagrams

> When two systems, of which we know the states by their respective representatives, enter into temporary physical interaction due to known forces between them, and when after a time of mutual influence the systems separate again, then they can no longer be described in the same way as before, viz. by endowing each of them with a representative of its own. I would not call that one but rather the characteristic trait of quantum mechanics, the one that enforces its entire departure from classical lines of thought.
>
> – *Erwin Schrödinger, 1935*

By 1935, Schrödinger had already realised that the biggest gulf between quantum theory and our received classical preconceptions is that, when it comes to quantum systems, the whole is more than the sum of its parts. In classical physics, for instance, it is possible to totally describe the state of two systems – say, two objects sitting on a table – by first totally describing the state of the first system then totally describing the state of the second system. This is a fundamental property one expects of a classical, *separable* universe. However, as Schrödinger points out, there exist states predicted by quantum theory (and observed in the lab!) that do not obey this 'obvious' law about the physical world. Schrödinger called this new, totally non-classical phenomenon *Verschränkung*, which later became translated to the dominant scientific language as *entanglement*.

Quantum picturalism is all about studying the way parts compose to form a whole. In the good company of Schrödinger, we believe that the role of multiple interacting systems – especially non-separable systems – should take centre stage in the study of quantum theory.

We shall see in the next section that it is easy to say what it means for a process to be *separable* in terms of diagrams. Literally, it means that it can be broken up into pieces that are not connected to each other. On the other hand, enforcing *non*-separability requires us to refine our diagrammatic language. To this end, we introduce special states and effects called *cups* and *caps*, respectively. Intuitively, cups and caps act like pieces of wire that have been 'bent sideways'. Whereas all states in the classical world separate, these states and effects are obviously non-separable and thus witness a crucial quantum feature:



$$= \frac{\text{separable}}{\text{non-separable}} = \frac{\text{classical}}{\text{quantum}} \tag{4.1}$$

In addition to caps and cups, we'll add *adjoints* to the diagrammatic language, which we picture as the vertical reflection of a diagram. This reflection operation associates each state ψ to a corresponding effect that tests 'how similar to ψ' a given state is. This has a direct analogue in Dirac notation, namely reflecting a ket into a bra. It will play a key role in modelling things such as reversible state evolution (unitary processes), quantum measurements (projectors and positive processes), and the probabilities associated with quantum measurements (via the inner product).

The new diagrams we obtain by adding caps, cups, and reflection are called *string diagrams*, and we will already start to see some strikingly non-classical behaviour arising just from using the language of string diagrams.

For example, we will already encounter quantum teleportation in Section 4.4.4. Related to this feature is the fact that caps and cups confound our ability to give a fixed time-ordering to processes composed in sequence, which we will see in Section 4.4.3. We will also encounter several 'no-go' theorems for string diagrams. That is, we will show that several properties that we expect to hold for classical (deterministic) physical processes are in fact impossible in a process theory that admits string diagrams. One such no-go theorem is the *no-cloning theorem*, which says that in any process theory that admits string diagrams it is impossible to have a *cloning process*, that is, one that takes any state as input and returns two copies.

Remark 4.1 Throughout this chapter we are careful to say 'non-separable' rather than 'entangled'. For a class of quantum states called 'pure states' (introduced in Section 6.1 of Chapter 6), the notions of 'non-separable' and 'entangled' do indeed coincide. However, for more general states, identifying quantum entanglement is a bit more subtle. Thus, we will hold off on giving the most general notion of quantum entanglement until Section 8.3.5.

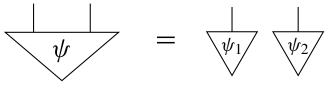
4.1 Cups, Caps, and String Diagrams

In this section, we first give a formal definition for *separability*, and hence *non-separability*. Next we motivate string diagrams via the principle of *process–state duality*, a correspondence between states of compound systems and processes – known in the special case of quantum theory as the *Choi–Jamiołkowski isomorphism* – that is fundamentally connected to non-separability.

We then present the definition of string diagrams in two equivalent ways. The first way is to *extend* the language of circuit diagrams with the 'caps' and 'cups' used to define process–state duality. The second, easier way is to simply allow the wires in a diagram to connect processes in arbitrary ways, including connecting inputs to inputs and outputs to outputs.

4.1.1 Separability

Definition 4.2 A *bipartite state* ψ is a state of two systems, and we call such a state $\otimes$-*separable* if there exist states ψ_1 and ψ_2 such that:

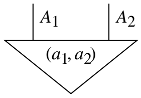


(4.2)

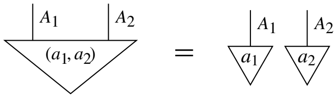
The following is an example of a process theory with all bipartite states $\otimes$-separable.

Proposition 4.3 All bipartite states in **functions** are $\otimes$-separable.

Proof In Example 3.35 we saw that states for **functions** are totally defined by the element that they ‘point to’. If for a bipartite state this element is $(a_1, a_2) \in A_1 \times A_2$, then we denote this state as:



Now we have:

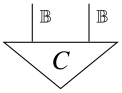


since both these states point to the same element. □

On the other hand, there are many (far more interesting!) process theories that contain bipartite states that are not $\otimes$-separable, i.e. states that are $\otimes$-*non-separable*.

Proposition 4.4 There are $\otimes$-non-separable bipartite states in **relations**.

Proof Recall from Example 3.36 that states in **relations** are represented by subsets. Consider the bipartite state:

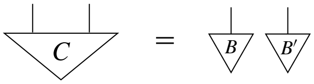


$$:: \begin{cases} * \mapsto (0,0) \\ * \mapsto (1,1) \end{cases}$$

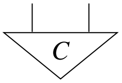
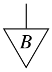
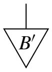
Then:

$$C = \{(0,0), (1,1)\} \subseteq \mathbb{B} \times \mathbb{B}$$

Suppose for the sake of contradiction that:



for some $B, B' \subseteq \mathbb{B}$. Then, by (3.22):

 $:: * \mapsto (b, b') \iff$  $:: * \mapsto b$ and  $:: * \mapsto b'$

that is:

$$(b, b') \in C \iff b \in B \text{ and } b' \in B'$$

Since $(0,0) \in C$, it follows that $0 \in B$. Similarly, $(1,1) \in C$ implies that $1 \in B'$. But these two facts imply $(0,1) \in C$, which is a contradiction. □

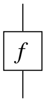
The crux of this proof is the fact that the (non-singleton) set:

$$\{(0,0), (1,1)\}$$

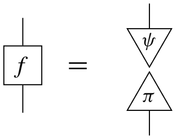
cannot be written as a Cartesian product of subsets of $\mathbb{B}$. If as in **functions**, every such set is a singleton, then this is always possible.

A closely related notion to ⊗-separability is the following.

Definition 4.5 We call a process:



∘-*separable* if there is an effect π and a state ψ such that:

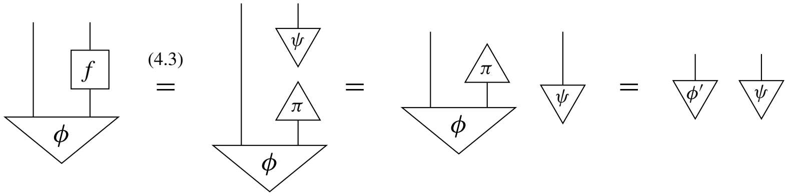


(4.3)

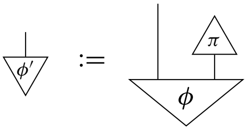
Both ⊗-separability of bipartite states and ∘-separability of processes are examples of 'disconnectedness' of the corresponding diagrams. Whenever no confusion is possible we will simply say 'separable' in both cases.

Moreover, these two notions of separability are related to each other in the following sense.

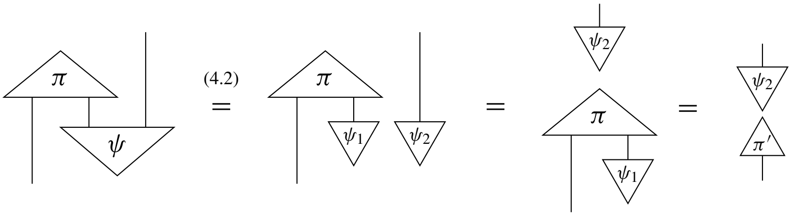
- If the process f is ∘-separable, then for any bipartite state ϕ the following state is ⊗-separable:



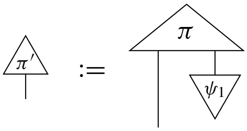
where:



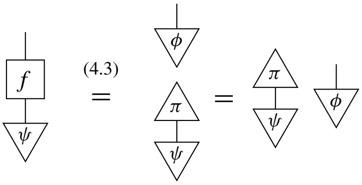
- If the state ψ is $\otimes$-separable, then for any bipartite effect π the following process is $\circ$-separable:



where:



We also saw earlier that in **functions** all bipartite states are $\otimes$-separable. So what kinds of process theories have all $\circ$-separable processes? The answer is, ‘only the really boring ones!’ When we apply a $\circ$-separable process to an arbitrary state:



we cannot obtain anything other than a fixed state ϕ (ignoring the number $\pi \circ \psi$), regardless of what the input ψ was. That is, every process is essentially a *constant* process. In other words:

nothing ever happens!

Thus it shouldn’t be surprising that the theory of **functions** *does* contain $\circ$-non-separable processes.

Exercise 4.6 Show that identities (i.e. plain wires) in **functions** are $\circ$-non-separable, and characterise those functions that are $\circ$-separable.

Remark 4.7 The ultimate boringness of process theories where all processes are $\circ$-separable will play an important role in several ‘no-go’ theorems that we develop later in this chapter. In particular, we will show that some statement P is not true by proving a theorem of the form:

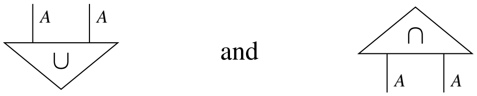
If P is true, then all processes are $\circ$-separable.

As this is an absurd condition for any reasonable process theory, and in particular any theory of physics, we can take this as a proof that P is false.

4.1.2 Process–State Duality

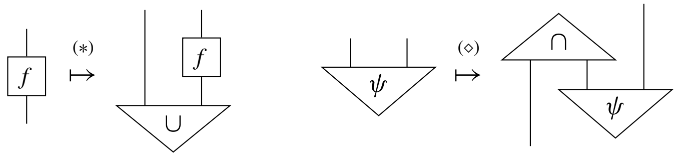
One feature of quantum theory is that one can turn a process into a bipartite state and vice versa. This is in itself not very significant: we already demonstrated such a conversion in the previous section in order to relate $\otimes$-separability of bipartite states to $\circ$-separability of processes. What is significant about quantum theory is that this can be done in a reversible manner. That is, we have a way to turn a process into a bipartite state and then back into a process such that we obtain the original process, and vice versa. In other words, the operations that send processes to bipartite states and bipartite states to processes are inverses of each other. Therefore, in quantum theory, processes and bipartite states are in bijective correspondence.

As in the previous section, we will make use of bipartite states and effects to interconvert processes and states. However, we will not be using just any state/effect pair but instead fix a special state and effect:



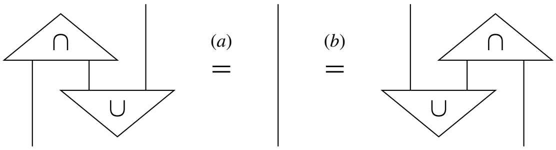
(4.4)

for each system A. We call these states and effects *cups* and *caps*, respectively. They can be used to convert processes to states and back via the following operations:



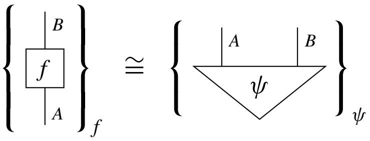
(4.5)

Theorem 4.8 The operations given in (4.5) are inverses of each other, i.e. they give *process–state duality*, if and only if:

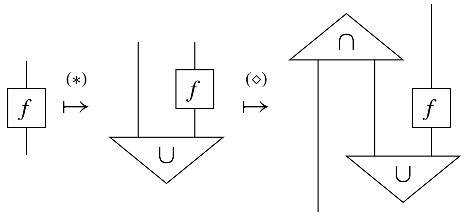


(4.6)

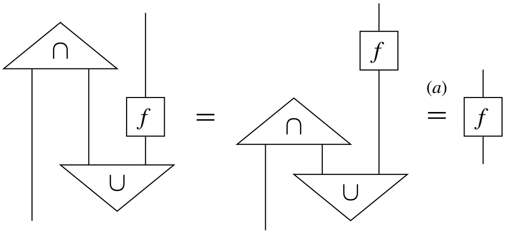
In that case, the set of all processes with input type A and output type B is in one-to-one correspondence with the set of all bipartite states of type $A \otimes B$:



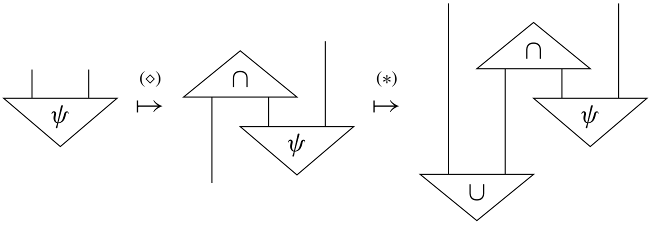
Proof First, assume equations (a) and (b). Then, if we first turn a process into a state and back:



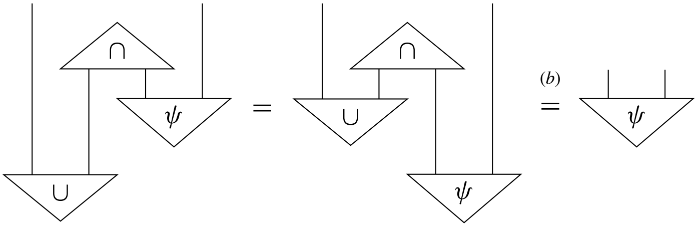
we can simplify back to the original process using (a):



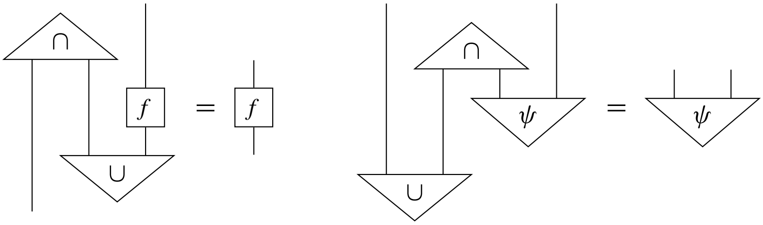
Similarly, starting with a state:



we can use (b):



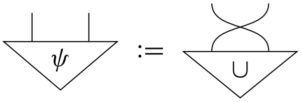
Conversely, if the operations given in (4.5) are mutually inverse, then:



(4.7)

for all f and ψ. Taking f to be the identity yields (a) immediately. Proving (b) using the above equations is left as an exercise. □

Exercise 4.9 Prove equation (4.6b) using both of the process–state duality equations (4.7), by setting:

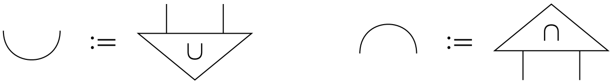


Exercise 4.10 Show that process–state duality does not hold for **functions**, but that it does hold for **relations**.

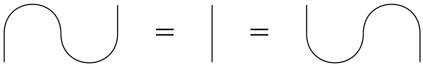
Remark* 4.11 A reader familiar with quantum theory may know about a correspondence between processes and states called the *Choi–Jamiołkowski isomorphism*. Process–state duality is in fact a generalisation of the C-J isomorphism. In Chapter 6, we will introduce the process theory of **quantum maps**, in which case this generalised C-J isomorphism will actually be the familiar correspondence between bipartite (possibly mixed) quantum states and completely positive maps.

4.1.3 The Yanking Equations

Equation (4.6) is not very intuitive when viewed diagrammatically, but by means of a simple change of notation we can fix this. We write the special states and the special effects, respectively, as ∪-shaped and ∩-shaped wires:



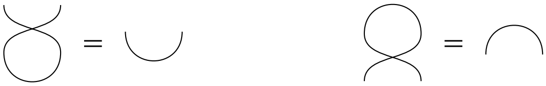
Equation (4.6) now becomes:



(4.8)

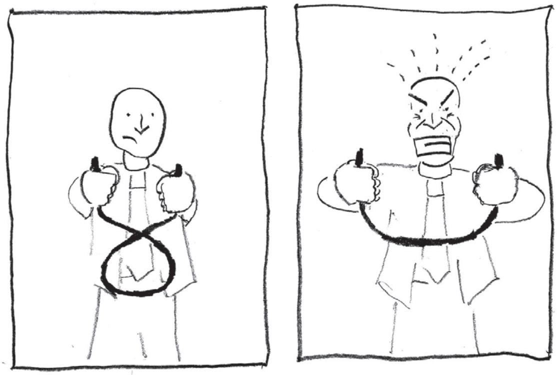
This notation expresses the fact that bent wires involving a ∪-shape and a ∩-shape can be yanked into a straight wire:

If we take this diagrammatic interpretation seriously, then there are other equations that should also hold for cups and caps. For example, we should be able to 'yank out crossings':

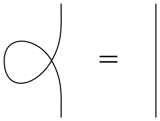


(4.9)

Or, stated with a little bit more effort:

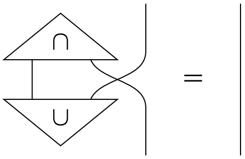


From (4.6) and (4.9), we can also derive the equation to pull out loops:



(4.10)

Exercise 4.12 Prove that equation (4.10), or equivalently:

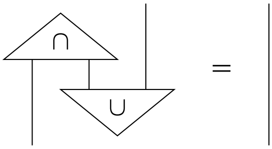


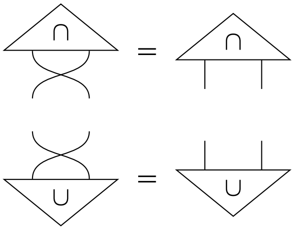
follows from (4.6) and (4.9).

These are actually all of the equations we need to 'yank' any tangled-up piece of wire straight. In fact, we already have some redundancy, as seen in the following proposition.

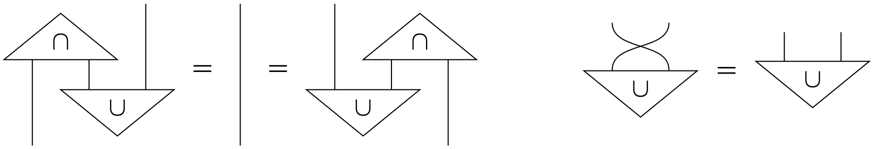
Proposition 4.13 The following are equivalent:

(i) a state and an effect satisfying:



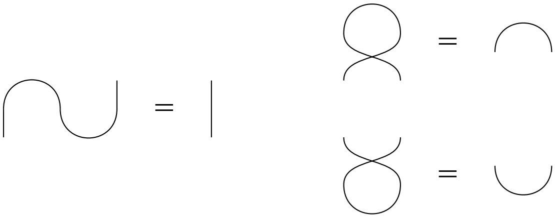


(ii) a state and an effect satisfying:



Exercise 4.14 Prove Proposition 4.13.

From now on, we will refer to the three equations:



(4.11)

as the *yanking equations*.

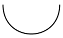
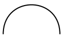
Example* 4.15 The *Bell state* and *Bell effect* from quantum theory:

$$\frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \qquad \frac{1}{\sqrt{2}}\left(\langle 00| + \langle 11|\right)$$

provide the quintessential example of cups and caps.

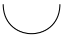
In Propositions 4.3 and 4.4 we already saw that, while in **functions** all bipartite states are ⊗-separable, this is no longer the case in **relations**, as we can easily find cups and caps for every system.

Exercise 4.16 Prove that for any set A the following relations:

 $:: * \mapsto \{(a,a) \mid a \in A\}$  $:: \forall a \in A : (a,a) \mapsto *$

satisfy equations (4.11).

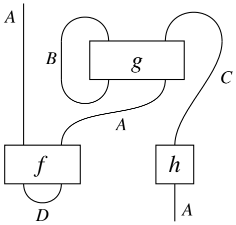
Example 4.17 Taking $A := \mathbb{B}$, we rediscover the state that we used as an example of a non-separability in the proof of Proposition 4.4:

 $:: * \mapsto \{(0,0),(1,1)\}$

4.1.4 String Diagrams

Equations (4.11) do look very appealing now, but we can do even better than that. If every system-type has cups and caps satisfying (4.11), then we can introduce a more liberal notion of a diagram that has them built in.

Definition 4.18 A *string diagram* consists of boxes and wires, where we additionally allow inputs to be connected to inputs and outputs to be connected to outputs, for example:



(4.12)

Above we already indicated that we can replace the special states and the special effects of the previous section by cup-shaped wires and by cap-shaped wires, respectively. This is how we obtain a string diagram from a circuit diagram in which there are states and effects satisfying (4.11). Conversely, starting from a string diagram we can replace a wire connecting two inputs with the special cup state and a wire connecting two outputs with the special cap effect. Hence the following theorem.

Theorem 4.19 The following two notions are equivalent:

(i) string diagrams and
(ii) circuit diagrams to which we adjoin a special state and a special effect for each type, and for which (4.11) holds.

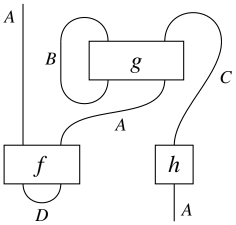
in the sense that (ii) can be unambiguously expressed as (i) and vice versa.

Bingo! We expressed non-separability in purely diagrammatic terms, and not just by adding some new boxes to (the now boring) circuit diagrams, but also by simply considering a different, more liberal kind of diagram. Thus, the most important feature of quantum theory (according to Schrödinger) is built right into the kinds of diagrams we use!

Exercise 4.20 Write diagram (4.12) in $\circ$ and $\otimes$ notation using cups and caps.

String diagrams also admit a formula-like presentation (cf. Section 3.1.3).

Definition 4.21 A *string diagram formula* is the same as a diagram formula except for the fact that matched pairs of wire names, besides consisting of an upper and a lower name, can now also consist either of two upper names or of two lower names, for example:

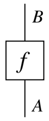


$$\longleftrightarrow \quad f^{A_1A_2}_{D_1D_1}\, g^{B_1C_1}_{B_1A_2}\, h^{C_1}_{A_3}$$

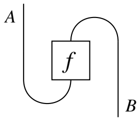
Remark 4.22 In Section 3.1.3, we introduced special box names $1_{A_1}^{A_2}$ to represent plain wires in diagram formulas. If we need explicit cups or caps in string diagram formulas, we can use two 'plain wires' with their inputs or outputs, respectively, connected together:

$$\cup^{A_1A_2} := 1_{A_3}^{A_1} 1_{A_3}^{A_2} \qquad\qquad \cap_{A_1A_2} := 1_{A_1}^{A_3} 1_{A_2}^{A_3}$$

This allows us to distinguish, for example:



$\longleftrightarrow \quad f_{A_1}^{B_1}$ and

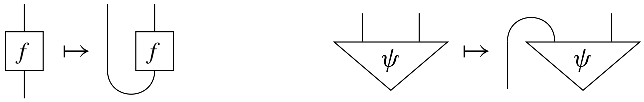


$\longleftrightarrow \quad \cup^{A_2A_1} f_{A_1}^{B_1} \cap_{B_1B_2}$

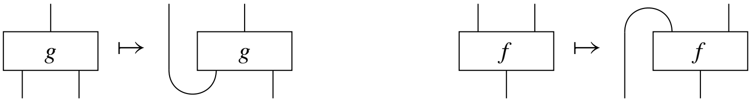
We now explore the richness of these diagrams.

4.2 Transposition and Trace

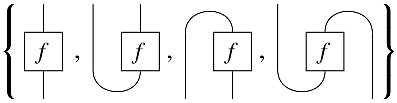
Transposition and trace are notions you may know from linear algebra. Quite remarkably, these already arise at the very general level of string diagrams. We already know from the previous section that cups and caps let us form new processes from old ones by turning inputs into outputs and vice versa. In the case of process–state duality we have:



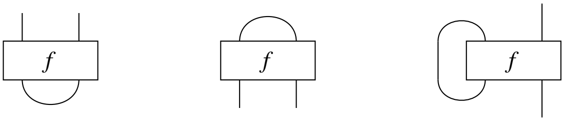
More generally, we can always inter-convert inputs and outputs:



This ability to inter-convert inputs and outputs for process theories that admit string diagrams means that inputs and outputs have a less profound status than in the case of other process theories that only admit circuits. In particular, any process with an input and an output has (at least) the following four inter-convertible representations:



and in the case of several inputs and outputs there are many more. We also can obtain new processes by connecting inputs and outputs together using cups and caps:

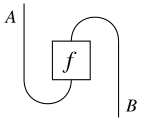


The *transpose* and the *trace* are the most prominent examples where caps and cups are used to convert processes into new processes.

4.2.1 The Transpose

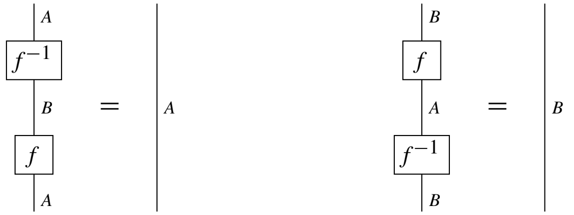
In a process theory that admits string diagrams we can associate to each process another process going in the opposite direction.

Definition 4.23 The *transpose* f^T of a process f is another process:



This of course should not be confused with another well-known way to obtain a process going in the other direction:

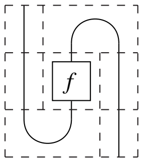
Definition 4.24 A process f from type A to type B has an *inverse* if there exists another process f^{-1} from type B to type A such that:



(4.13)

Exercise 4.25 Show that if a process f has an inverse, it is unique.

While the transpose can be realised by a diagram involving f:

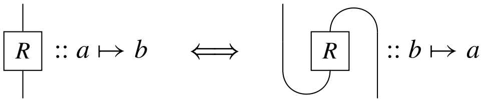


$$= (1_A \otimes \cap_B) \circ (1_A \otimes f \otimes 1_B) \circ (\cup_A \otimes 1_B) \qquad (4.14)$$

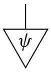
this is not the case for an inverse, otherwise every process would have an inverse, which is usually not true.

Remark* 4.26 The simple fact that the transpose in linear algebra can be written using a cup and a cap as in decomposition (4.14) is surprisingly not very well known, even among specialists.

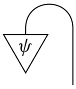
Exercise 4.27 Prove that in **relations** the transpose of a relation R is the converse relation, that is:



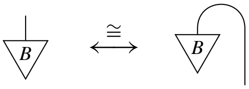
A state:



has no input, so we only have to convert the output into an input:

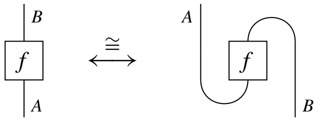


Hence, it is no coincidence that in **relations** any system-type A has the same number of states as effects (see Example 3.36), since we have a bijective correspondence between states and effects:



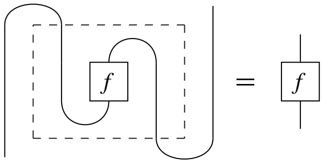
This is a general phenomenon.

Proposition 4.28 For any process theory that admits string diagrams there is a bijective correspondence between states and effects of the same type. More generally, the correspondence:



yields a bijective correspondence between the processes of a fixed input and output type, and the processes with the opposite input and output type.

The transpose of a transpose yields the original process:

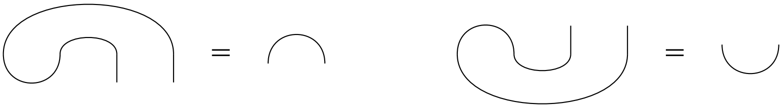


Or, symbolically:

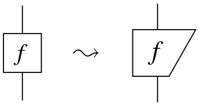
$$(f^T)^T = f$$

An operation that 'undoes itself' like this is called an *involution*.

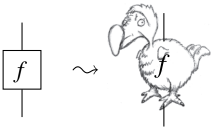
Example 4.29 The transpose of a cup is a cap and vice versa:



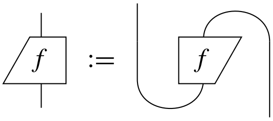
Now comes the really cool bit. We can build the definition of transpose into our diagrammatic notation. First, we deform our boxes a bit:



It doesn't matter too much <u>how</u> we deform boxes, only that we break the symmetry. For example,



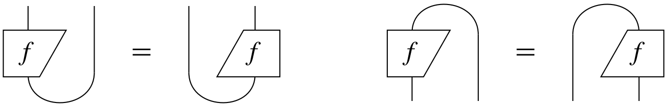
would also work, but Dave might not appreciate being skewered. Now, we express the transpose of f as a box labelled 'f', but rotated 180°:



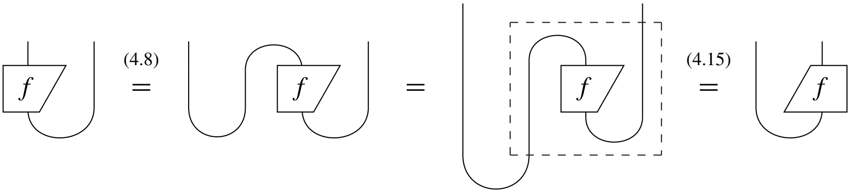
(4.15)

This notation is clearly consistent with the fact that the transpose is an involution, since, if one rotates by 180° twice, one obtains the original box again. However, the real advantage of this choice of notation comes when transposes interact with cups and caps.

Proposition 4.30 For any process f we have:

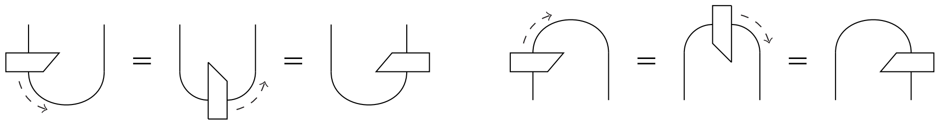


Proof The first equality is established as follows:

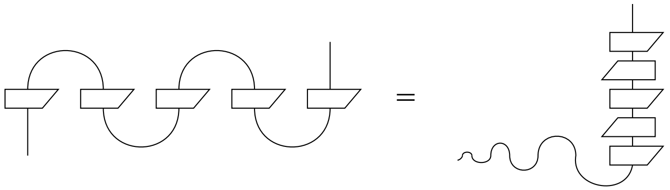


and the proof of the second one proceeds similarly. □

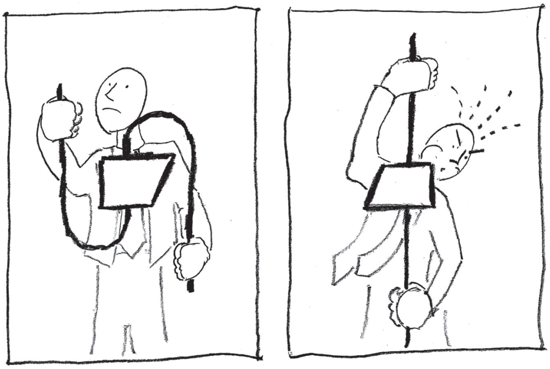
Thanks to the clever shorthand for the transpose, it now looks as if we can slide boxes across the ∪-shaped wires and ∩-shaped wires:



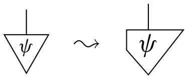
Thus we can slide boxes around on wires like beads on a necklace. For example, this is a valid equation:



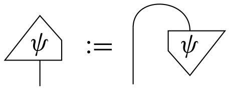
This is also very easy to remember. Just think about what would happen if we took the definition of the transpose (4.15) and yanked on the wires:



The same notational trick also applies to states and effects. To make it clear that they have been rotated, we chop off a corner:



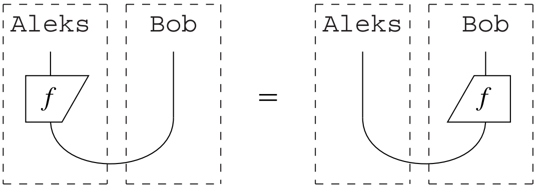
and we set



(4.16)

It is not obvious why this is necessary at this point, since it's pretty easy to tell if a triangle as been turned upside-down. However, it will soon become very important (starting in Section 4.3), so we might as well get used to it.

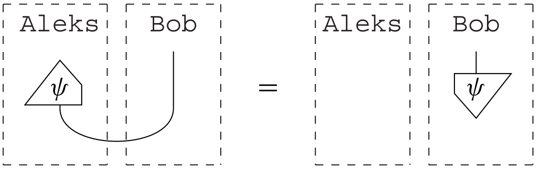
Transposes also have an operational interpretation. Consider the first equality of Proposition 4.30 by interpreting the two systems as distant locations in space, each controlled by an agent, say Aleks and Bob. Then, we have:



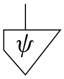
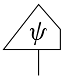
(4.17)

The equation says that, whenever two such systems are in a cup state, if Aleks applies f to his system, this will do the same thing as Bob applying the transpose of f to his system.

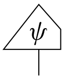
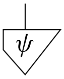
Now, suppose we consider (4.17) in the special case of states and effects:



Given that there is a bijective correspondence between:

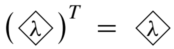
- Bob's states  and
- Aleks' effects ,

Aleks and Bob possess a pair of systems that are in *perfect correlation*. What does this mean? Recall that an effect can be interpreted as a successful test. Under this interpretation, we have the following property. For every state on Bob's system, there is a unique test on Aleks' system such that:

as soon as Aleks obtains  *Bob's system will be in state* 

A process that is equal to its own transpose is called *self-transposed*.

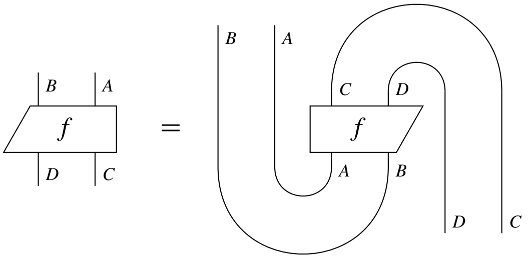
Example 4.31 Numbers are always self-transposed. By definition, the transpose bends all the input wires up and all the output wires down. Since a number has no wires, there's nothing to do:



(4.18)

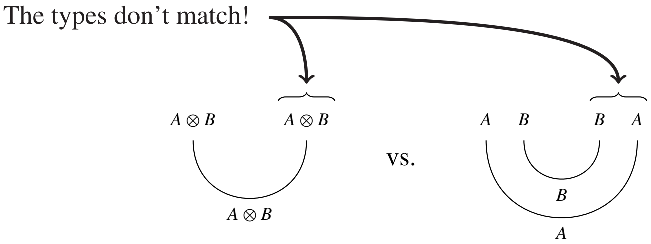
4.2.2 Transposition of Composite Systems

One has to be a bit careful when dealing with joint system-types. On the one hand, to be consistent with the '180° rotation' notation for transposition, we should define the transpose this way:



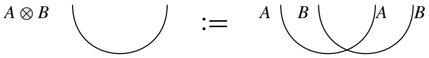
(4.19)

This would suggest that we should 'nest' the caps and cups inside of each other to define caps and cups for $A \otimes B$. However, there's a problem with defining $\cup_{A\otimes B}$ and $\cap_{A\otimes B}$ as above:

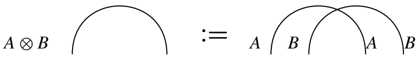


Remark* 4.32 This type mismatch vanishes if we introduce for every type A a *dual type* A^*, but this is at the cost of having two distinct types for the two systems involved in cups/caps, while in fact, these two types are often essentially the same. In Section* 4.6.2 we show how one can develop a theory of string diagrams with dual types.

We can avoid this type mismatch if we define cups/caps for $A \otimes B$ a bit differently, namely as 'criss-crossed' cups/caps:



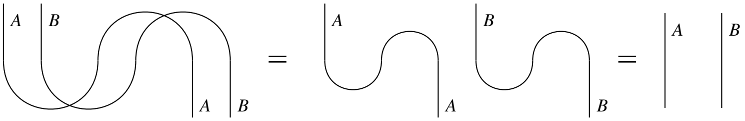
(4.20)



(4.21)

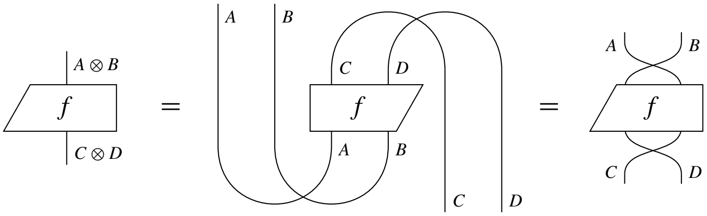
Proposition 4.33 The cup and cap defined in equations (4.20) and (4.21) satisfy the yanking equations (4.11).

Proof For the first yanking equation of (4.11) we have:



Proofs of the other two equations are similar. □

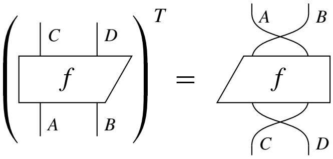
With these 'criss-crossed' cups/caps, we obtain an alternative notion of transposition, which introduces a twist for composite systems:



(4.22)

It is this twist that restores the matching of types, and since we define cups/caps on $A \otimes B$ in terms of cups/caps on A and B individually, there is no ambiguity when we apply this transpose on processes involving composite systems. But, at the same time, we lost a chunk of the elegant '180° rotation' notation for transposition.

In fact, it turns out that both versions of transposition can be useful. We will take the first version, indicated in (4.19), to be the default and continue to refer to this simply as 'the transpose'. The second version, involving the 'criss-crossed' cups and caps, will be referred to as the *algebraic transpose*, since it is in fact the one that is used in linear algebra (see Section 5.2.2). While the transpose is (still) denoted by 180° rotation, we will use the symbolic notation $(\ )^T$ to represent the algebraic transpose. So the equation:

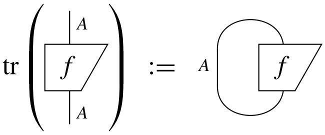


now relates the transpose to the algebraic transpose. Of course, when a box has at most one input/output wire, these two versions coincide. Unsurprisingly, a process that is equal to its algebraic transpose is called *algebraically self-transposed*.

4.2.3 The Trace and Partial Trace

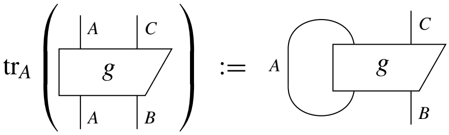
String diagrams give us a simple way of sending processes to numbers.

Definition 4.34 For a process f where the input type is the same as the output type, the *trace* is:

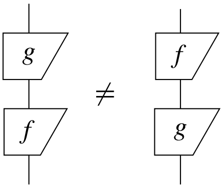


(4.23)

and for a process g with one of its inputs having the same type as one of its outputs, the *partial trace* is:



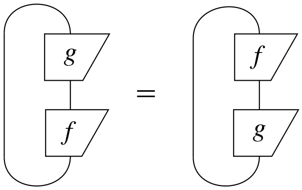
A funny property of the trace is that while in general:



the trace of the LHS and the RHS are in fact equal.

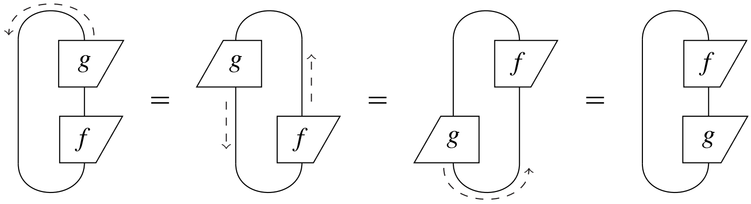
Proposition 4.35 (cyclicity of the trace) We have:

$$\mathrm{tr}(f \circ g) = \mathrm{tr}(g \circ f) \qquad \text{i.e.}$$

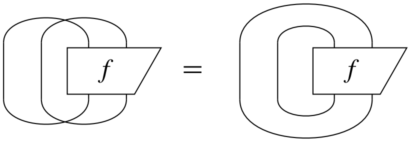


Proof This follows from the fact that the two diagrams are equal; i.e. the LHS can be deformed into the RHS without changing how the boxes are wired together. □

Alternatively, we can give a more step-wise proof of Proposition 4.35 using Proposition 4.30. While not strictly necessary, this other proof is nice because it takes the word ‘cyclicity’ quite literally, yielding a ferris wheel of boxes:



Remark 4.36 In the previous section we decided to distinguish the transpose (which involves nested cups/caps) from the algebraic transpose (which involves criss-crossed cups/caps). Since equation (4.23) also involves cups and caps, we might ask whether for composite systems we need to make a similar distinction between ‘nested’ and ‘criss-crossed’ trace. Luckily, we don’t:



Exercise 4.37 Show that there is only one trace; i.e. <u>any</u> cup/cap pair satisfying the yanking equations (4.11) defines the same trace via (4.23).

4.3 Reflecting Diagrams

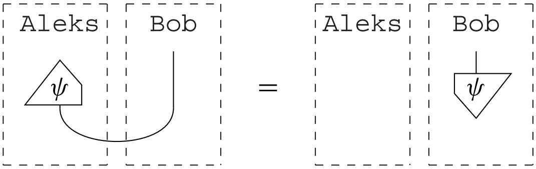
Having string diagrams at hand is already a substantial step towards the quantum world, since it guarantees the existence of non-separable states. They also give rise to some mathematical concepts that play a central role in quantum theory, such as transposition and trace.

We will now identify one more diagrammatic feature that makes string diagrams a lot more articulate, namely *vertical reflection* of diagrams. Vertical reflection allows us to define things like *adjoints, conjugates, inner products, unitarities*, and *positivity*, all of which are major players in any presentation of quantum theory.

Moreover, vertical reflection isn't just some extra degree of freedom that string diagrams happen to have, but has a clear operational meaning in terms of testability of states. This operational meaning will lead to a number of conditions that play an important role in quantum theory. These conditions are in fact quite standard in the literature, but they are usually just stated formally, without any conceptual justification.

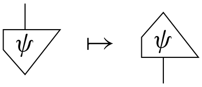
4.3.1 Adjoints

In the previous section we explained how the transpose acts graphically by rotating boxes 180°, and that operationally it captures the perfect correlation between Aleks' effects on one side of a cup state and Bob's states on the other:



But what about the relationship between Aleks' effects and Aleks' states?

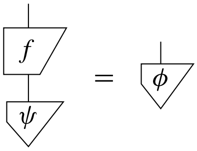
Recall from Section 3.4.1 that effects can be interpreted as testing a state for some property. Typically, we want to test whether a system is in a particular state. Therefore, we should have a means of relating a state ψ to the effect that tests a system for ψ. String diagrams do not yet tell us how to do this, so we extend our language by representing the effect testing for a state ψ as its vertical reflection:



(4.24)

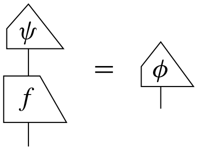
Instead of ‘the effect testing for a state ψ’ we’ll simply say *ψ’s adjoint*.

This reflection operation extends very naturally to all processes. If f transforms state ψ into a state ϕ:



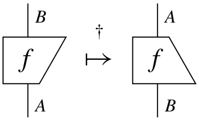
(4.25)

then *f’s adjoint* is the process that transforms ψ’s adjoint into ϕ’s adjoint:

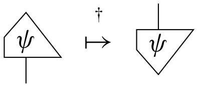


(4.26)

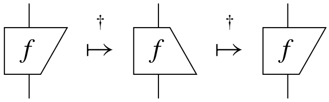
Note that, as in the case of states, we have depicted the adjoint of an arbitrary process as its vertical reflection:



As a result, equation (4.26) is just equation (4.25) upside-down. We use † to denote the operation sending f to its adjoint $f^\dagger$. As a special case, if we take the adjoint of the effect from (4.24), we get back to ψ:

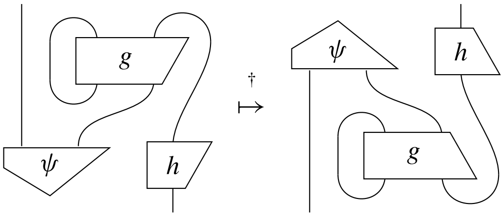


Hence, like the transpose, adjoints bijectively relate states and effects, and more generally, they bijectively relate processes from a type A to a type B to processes from B to A. Also like the transpose, this is an involution, which is plainly suggested by the diagrammatic notion:



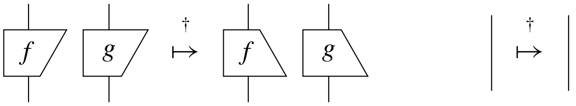
(4.27)

In the passage from (4.25) to (4.26), we saw one example where taking the adjoint of a process reflected its entire diagram. This actually applies to all diagrams, for example:



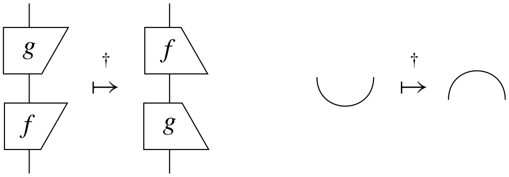
(4.28)

Equivalently (by Theorem 4.19), taking the adjoint preserves parallel composition and identities:



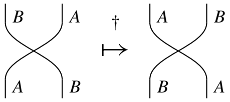
(4.29)

it reverses sequential composition and sends cups to caps:



(4.30)

and it reverses the direction of swaps:



(4.31)

Exercise 4.38 Using Exercise 3.38, prove:

$$0^{\dagger} = 0$$

So, we now know that adjoints are represented diagrammatically as vertical reflection. However, we haven't said how one should compute the adjoint of a process. The answer is: it depends on the theory. Since a process theory gives an interpretation for all diagrams (cf. Section 3.1.2), it must in particular give an interpretation for the vertical reflection of a diagram. This raises two questions:

1. Does there always exist such an interpretation?
2. Can there be more than one such interpretation?

The answer to the first question is yes, since one can always interpret vertical reflection as the algebraic transpose.

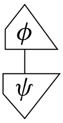
Exercise 4.39 Verify that the algebraic transpose provides a candidate interpretation for adjoints. In other words, show that it obeys (4.27) and (4.28). Explain why we pick the algebraic transpose instead of the transpose.

The algebraic transpose is, in some sense, a trivial interpretation for vertical reflection, since it doesn't add anything that string diagrams without adjoints didn't have already. Of course, if this were the only example, we wouldn't have bothered with adjoints in the first place. In the next chapter, we will encounter a very important non-trivial example of adjoints, which cannot be replaced by transposition. So the answer to the second question is also yes; there are multiple (trivial and non-trivial) ways to interpret the adjoint. The fact that there may be many candidate adjoints for a process theory raises yet another question:

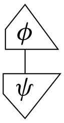
3. What are good interpretations for the adjoint?

Any interpretation of adjoints in a process theory should be involutive and reflect diagrams in the sense of (4.28). However, to be a 'good' adjoint, it should be consistent with the idea that it sends a state to the effect that tests for that state. This has some consequences, which we will discuss now.

When we test a state ψ for ϕ, there are two extremes:

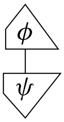


$= 0$ and

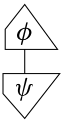


$=$

or equivalently:



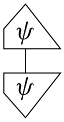
$= 0$ and



$= 1$

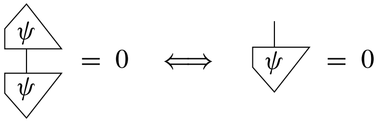
since '1' is just symbolic notation for the empty diagram. In the first case, we are saying that it is impossible to get a 'yes' outcome when testing ψ for ϕ. For the process theories we'll consider in this book, the second equation has one of two interpretations: it either means that a 'yes' outcome is 'possible' (in the case of **relations**) or 'certain' (in the case of most other process theories we will encounter in this book).

In light of the interpretation of the first equation, we would expect this:



$\neq 0$

That is, it should never be impossible to get a 'yes' outcome when testing a state ψ for itself. However, there is one thing we overlooked: ψ itself could be 0, i.e. the 'impossible state'. As we saw in Section 3.4.2, 0 absorbs everything, so in particular, 0 composed with 0 will always be ... you guessed it, 0. So, we can state the condition above a bit more carefully:

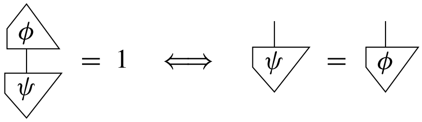


(4.32)

In words: if it is impossible to get ‘yes’ when testing ψ for itself, then ψ must already be impossible.

Next, we consider the other equation, i.e. when testing ψ for ϕ yields 1. If we interpret 1 as ‘possible’, this means that ψ and ϕ are not completely distinguishable by tests, but it doesn’t tell us much beyond that. In particular, it doesn’t tell us that they are equal (cf. Example 4.40).

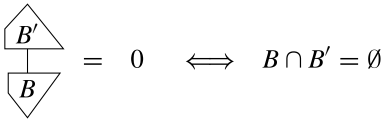
However, when 1 means ‘certain’, we can say something much stronger:



(4.33)

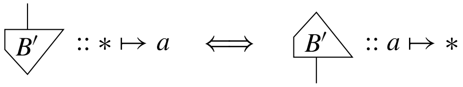
That is, we conclude (with certainty) that ‘ψ is ϕ’.

Example 4.40 In Example 3.36 we established that in the process theory of **relations** the states for a set A correspond to subsets $B \subseteq A$. We think of these as non-deterministic states, consisting of a set of possible, ‘actual’ states $b \in B$. Effects for A also correspond to subsets, and we think of these as testing whether the actual state of the system is one of the elements of B. If testing B for B' yields 0, this means that none of the elements $b \in B$ is in B'. That is:

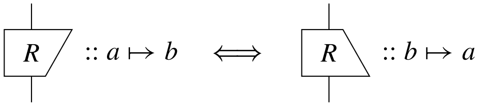


(4.34)

There is in fact only one effect B' that sends a state B to 0 if and only if $B \cap B'$ is empty. That is the *relational converse* of B':



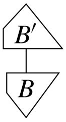
By means of (4.25) ⇔ (4.26), this lifts to any relation:



(4.35)

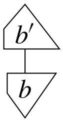
Consequently the adjoint of a relation coincides with its algebraic transpose (cf. Exercise 4.27).

Equation (4.34) arose as an instance of (4.32), so the first equation of ‘goodness’ uniquely fixes adjoints in the theory of **relations** to be the relational converse. But does it satisfy (4.33)? Should we expect it to? Since the number 1 in **relations** means 'possible' and not 'certain', the answer is no. In fact, there are many non-equal states where:



$$= 1 \qquad (4.36)$$

because this merely means that $B \cap B' \neq \emptyset$. However, there is one situation in **relations** where the number 1 does mean certain: when comparing 'actual' (i.e. deterministic) states. That is, for $b, b' \in A$:



$$= 1 \iff b = b' \qquad (4.37)$$

The following is a prime example of adjoints, which will play a role throughout the book and are crucially different from the transpose. We give it as a 'starred' example here, as it will already be familiar to some readers. If not, don't worry, it will be introduced properly in the next chapter.

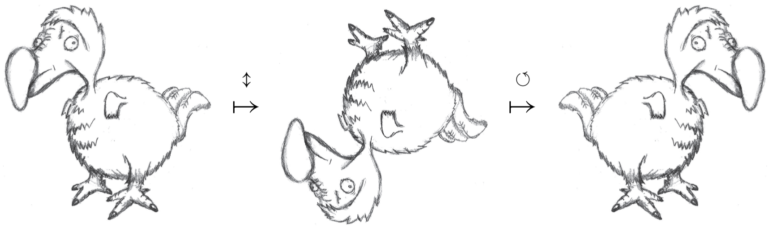
Example* 4.41 As the name suggests, adjoints in the process theory of **linear maps** are given by the *linear algebraic adjoint*, i.e. for a linear map f, the unique map $f^\dagger$ such that (in traditional notation):

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

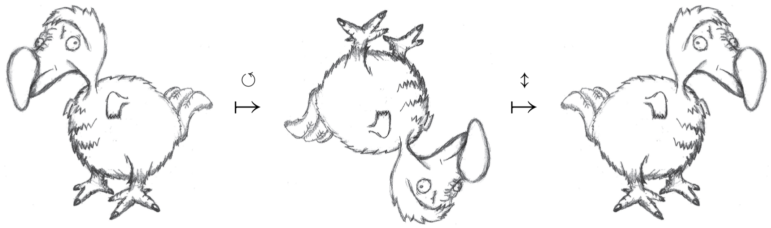
for all ψ, ϕ. In terms of matrices, this is the *conjugate-transpose*. These adjoints satisfy (4.32) and (4.33), where in the case of (4.33), we restrict to normalised states. We will see in Section 5.3.2 that, if we were to interpret adjoints using the transpose, both of these conditions will fail. So, the transpose is definitely not a good choice for adjoints in **linear maps**.

4.3.2 Conjugates

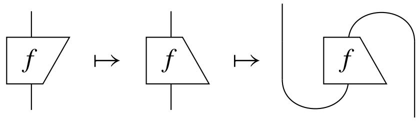
So, what happens when we combine adjoints with transposition? Geometry suggests that it shouldn't matter if we first reflect vertically, then rotate 180°:



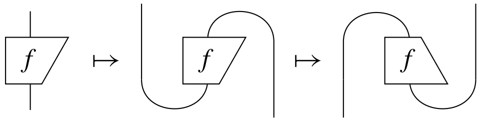
or if we first rotate 180°, then reflect vertically:



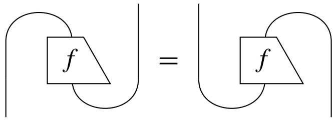
In either case, we get a horizontal reflection. Using the definition of transposition in terms of cups and caps, the transpose of the adjoint is:



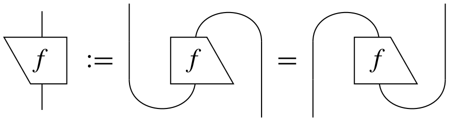
and the adjoint of the transpose is:



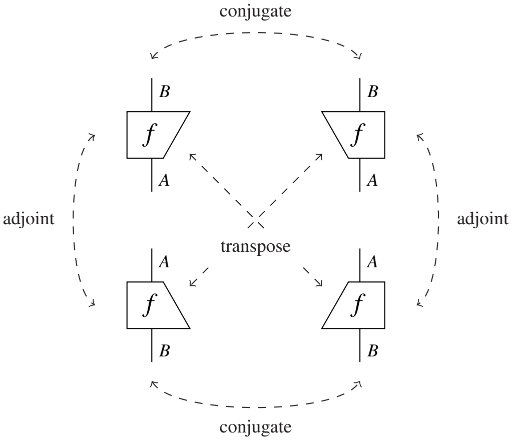
Hence these are equal, since the following is just a deformation of diagrams:



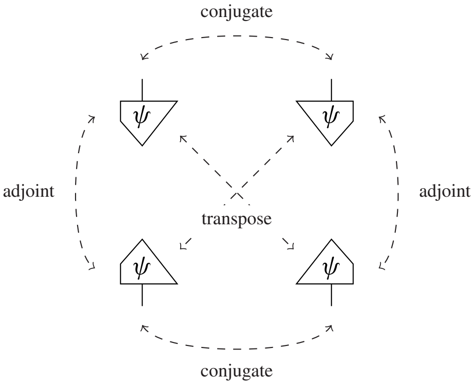
Definition 4.42 The *conjugate* of a process is the transpose of its adjoint (or equivalently, the adjoint of its transpose). Depicted graphically:



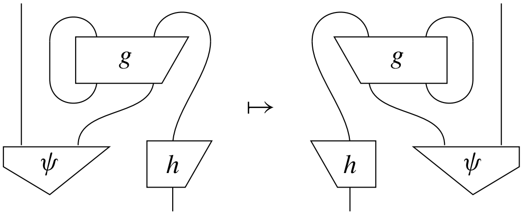
Just as in the case of adjoints and the transpose, if we conjugate twice, we return to where we started. So all together, boxes come in quartets:



It should now also be clear why we cut out a corner for states and effects:

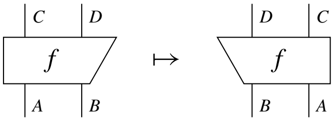


As with adjoints, conjugation mirrors entire diagrams, but horizontally rather than vertically:

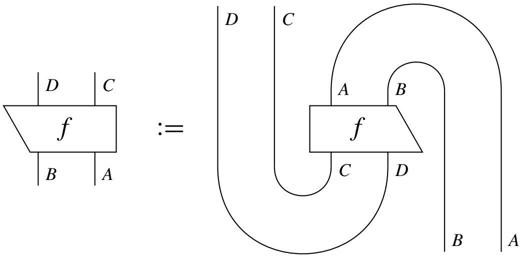


(4.38)

In particular, if a box has multiple inputs and outputs, it reverses their order:



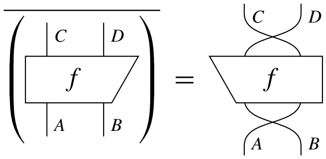
This feature of course comes from the fact that the transpose reverses input/output order, as we saw in Section 4.2.2:



As with the transpose, we will occasionally want to avoid this. In that case, we can use the *algebraic conjugate*, which is defined in terms of the algebraic transpose as:

$$\bar{f} := (f^T)^\dagger = (f^\dagger)^T$$

It keeps the order of inputs and outputs the same by introducing a twist in the wires relative to the normal conjugate:

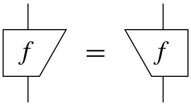


(4.39)

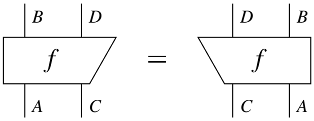
As in the case of the transpose, for single input/output wires the diagrammatic and the algebraic notions coincide.

Remark* 4.43 We will see in Chapter 5 that for **linear maps** the algebraic conjugate does exactly what one expects: it conjugates all matrix entries. So linear-algebraic adjoints, transposes, and conjugates correspond to adjoints, algebraic transposes, and algebraic conjugates, respectively.

Processes that are equal to their own conjugates will play an important role in this book. A process is said to be *self-conjugate* if:

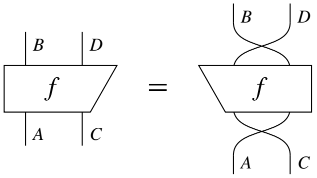


For joint systems, this becomes:



(4.40)

which of course only makes sense if $A = C$ and $B = D$, or more generally the input and output types are palindromes such as $A \otimes B \otimes A$ or $B \otimes C \otimes C \otimes B$. On the other hand, *algebraically self-conjugate* processes on joint systems look like this:



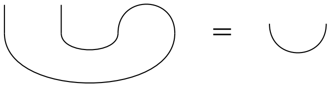
(4.41)

Remark* 4.44 Algebraically self-conjugate matrices are those whose entries are all real numbers.

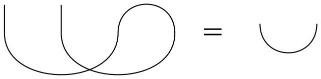
The horizontal reflection of a cup/cap is just a cup/cap again, so the following should not be surprising.

Proposition 4.45 Cups and caps are self-conjugate (and also algebraically self-conjugate).

Proof The conjugate of the cup is:

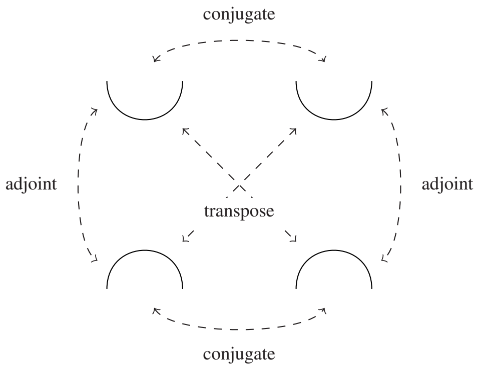


which is the same as the algebraic conjugate:



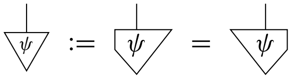
The calculation for caps is similar. □

Thus the ‘quartet’ for cups and caps is:

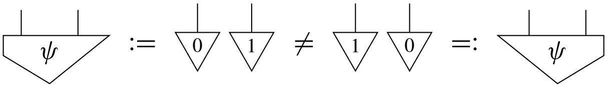


Exercise 4.46 Prove that all **relations** are algebraically self-conjugate. Of these, which relations are self-conjugate in the sense of (4.40)?

To emphasise that a state (or effect) is self-conjugate, we revert to our original notation using triangles:

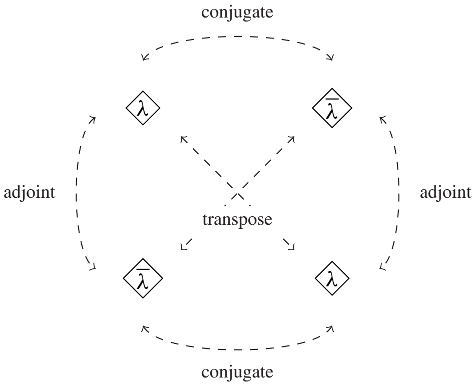


However, we should be a bit careful with compound systems. Since conjugation reflects diagrams, states consisting of multiple, self-conjugate states are not, in general, self-conjugate:



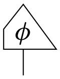
Example 4.47 States and effects for a single system in **relations** are always self-conjugate, so we will depict them as above. In particular, we will continue to depict the states from Example 3.36 as:

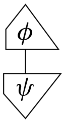
For numbers, since the transpose is trivial (cf. Example 4.31), the conjugate and the adjoint coincide. Thus the 'quartet' for numbers collapses to a 'couplet' of the number and its conjugate:



4.3.3 The Inner Product

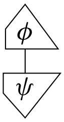
When we first introduced Dirac notation in Section 3.4.4, we mentioned that the diagrammatic language was missing one feature: the ability to turn a ket into a bra, and vice versa. This is exactly what the adjoint does for us, so we can now amend the rules **D2** and **D3** to use adjoints:

D2:  is written as $\langle \phi |$ and called a 'Dirac bra'

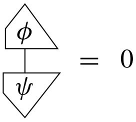
D3:  is written as $\langle \phi | \psi \rangle$ and called a 'Dirac braket'

The composition in **D3** is important enough to give it a special name.

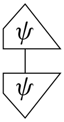
Definition 4.48 The *inner product* of states ψ and ϕ of the same type is:



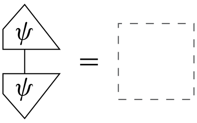
and these states are called *orthogonal* if:



The *squared norm* of a state ψ is the inner product of ψ with itself:

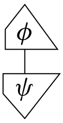


and a state ψ is called *normalised* if:



Remark 4.49 We say ‘squared norm’ because in linear algebra one typically takes the norm of ψ to be the square root of this quantity.

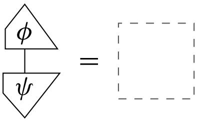
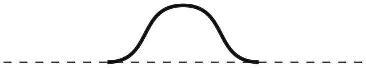
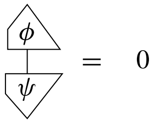
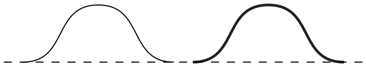
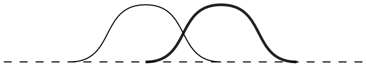
The meaning of the inner product immediately follows from our motivation for introducing adjoints. Since the adjoint of a state gives us the effect that tests for that state, we have:



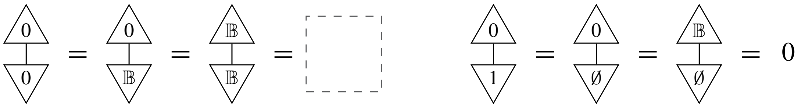
:= *‘testing state ψ for being state ϕ’*

In other words, the inner product computes ‘how much commonality’ states have, and orthogonality means that there is no commonality whatsoever. At the other end, we would expect to get 1 (i.e. the empty diagram) whenever we are testing a state for itself. This is true precisely when a state is normalised (which is a good reason to treat normalised states as the ‘default’).

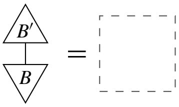
Another useful intuition for ‘measuring commonality’ is the following. When states are distributions of some sort (e.g. probability distributions), the inner product computes how much those distributions overlap:

		perfect overlap
		no overlap
none of the above		some overlap

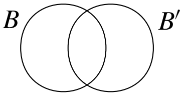
Example 4.50 In **relations** we have:



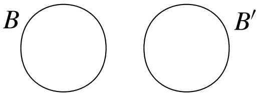
The first three inner products include cases of ‘perfect overlap’ as well as of ‘some overlap’, since both correspond to the same number. In **relations** overlap can be taken literally since:



means that the intersection of B and B' is non-empty:



On the other hand, the subsets in the last three inner products have no intersection:



The inner product and the corresponding squared norm are standard notions in linear algebra, and most of the defining properties of a linear algebraic inner product already follow from the language of string diagrams. Those readers familiar with inner products will recognise these properties more easily when denoting conjugation by $\overline{(\ )}$ and using Dirac bracket notation.

Proposition 4.51 The inner product:

1. is *conjugate symmetric*:

$$\overline{\langle\phi|\psi\rangle} = \langle\psi|\phi\rangle$$

2. preserves numbers in the second component:

$$\langle \phi | \lambda \cdot \psi \rangle = \lambda \cdot \langle \phi | \psi \rangle$$

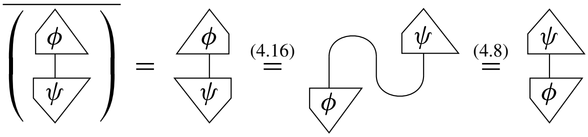
3. conjugates numbers in the first component:

$$\langle \lambda \cdot \phi | \psi \rangle = \overline{\lambda} \cdot \langle \phi | \psi \rangle$$

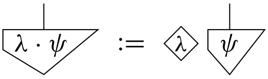
4. and is *positive definite*:

$$\langle \psi | \psi \rangle = 0 \quad \Leftrightarrow \quad |\psi\rangle = 0$$

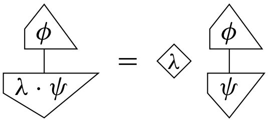
Proof For conjugate symmetry, we have:



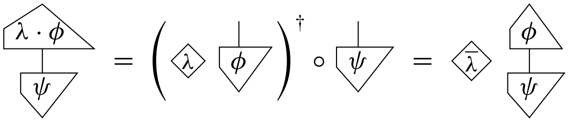
Next, setting:



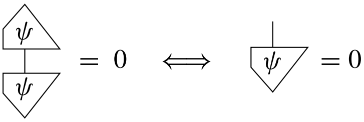
we straightforwardly have:



as well as:



At the end of Section 4.3.1 we moreover established:



as an instance of what makes for (good) adjoints. □

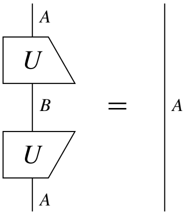
Remark 4.52 The reader may wonder what is 'positive' about 'positive definite'. This will become clear in Section 4.3.5.

Remark* 4.53 Readers familiar with inner products in linear algebra might have expected conditions 2 and 3 above to be 'linearity' and 'conjugate linearity'. The only thing missing is that the inner product should also preserve sums. This will naturally follow once we introduce sums of diagrams in Section 5.1.3.

4.3.4 Unitarity

As the inner product provides us with a measure of commonality/overlap, the natural follow-up step is to identify those processes that preserve this measure. That's what we do in this section.

Definition 4.54 A process U is an *isometry* if we have:

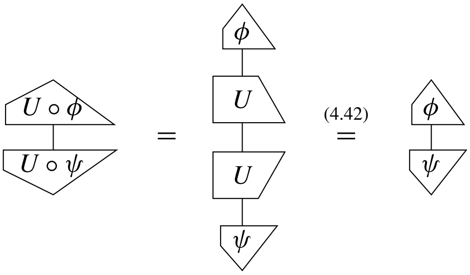


(4.42)

In other words, $U^\dagger$ satisfies one of the two equations from Definition 4.24, needed to be an inverse of U; i.e $U^\dagger$ is a *one-sided inverse* of U.

Proposition 4.55 Isometries preserve the inner product.

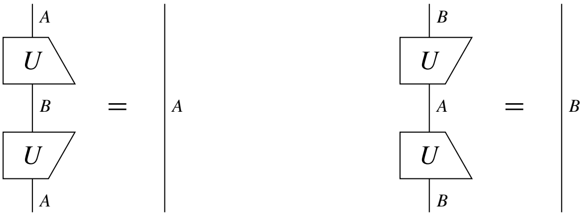
Proof We have:



□

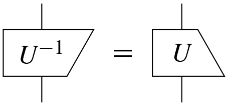
If $U^\dagger$ satisfies both inverse equations, we obtain the following notion:

Definition 4.56 A process U is *unitary* if we have:



(4.43)

We know from Exercise 4.25 that inverses are unique, so it immediately follows that, for a unitary process U:



From the diagrammatic definition of unitarity we also obtain the following proposition.

Proposition 4.57 Identities and swaps are unitary, and the sequential and parallel compositions of unitary processes are again unitary.

Here are some alternative characterisations of unitarity:

Proposition 4.58 For a process f the following are equivalent:

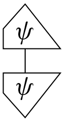
- f is unitary.
- f is an isometry and admits an inverse.
- $f^\dagger$ is an isometry and admits an inverse.

Exercise 4.59 Prove Proposition 4.58.

In Section 5.1.5, we'll meet several more equivalent ways to recognise isometries and unitaries.

4.3.5 Positivity

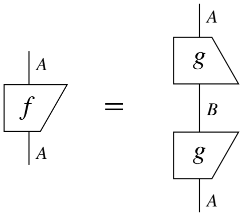
In Remark 4.52 we promised to explain the 'positive' part of 'positive definite'. Its turns out that the inner product of a state with itself:



(4.44)

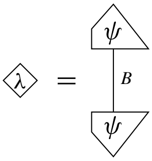
is a special case of a general notion of positivity that makes sense in any process theory.

Definition 4.60 A process f is *positive* if for some g we have:

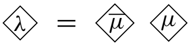


(4.45)

So, in fact, the number given by (4.44) is positive by definition. In fact, for many process theories:



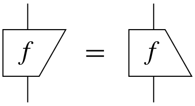
simplifies to



That is, we can always take B to be 'no wire'. Thus, as we will see in the next chapter, this does in fact exactly capture the usual notion of a positive number (i.e. a real number ≥ 0) for the theory of **linear maps**. However, this notion of positivity applies not just to numbers but to any process with the same input/output system.

From (4.45), it clearly follows that positive processes are invariant under vertical reflection.

Proposition 4.61 Positive processes are *self-adjoint*, that is:

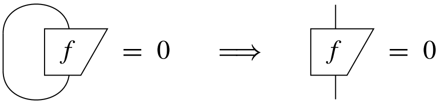


(4.46)

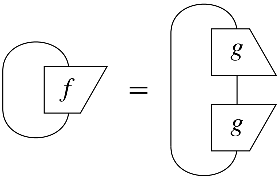
So in particular, positive numbers are self-adjoint, and hence, since the transpose of numbers is trivial, they are self-conjugate.

By positive definiteness, non-zero positive processes have non-zero trace. In other words, we can figure out whether a positive process is zero by computing its trace.

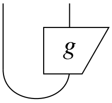
Proposition 4.62 For positive processes we have:



Proof If f is positive, then we have:

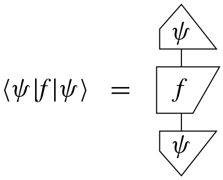


so the trace of f is the inner product of this state:

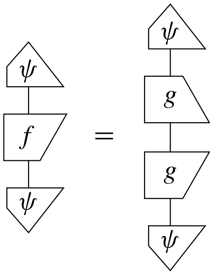


with itself. If this inner product is zero, then so too is the above state, by positive-definiteness. Thus g itself is 0, and hence so is f. □

This general definition of positive process also implies a notion of positivity familiar from linear algebra. Namely, if f is positive, the number:

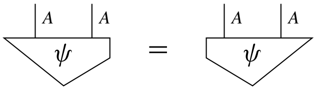


is positive for all ψ. This can be easily seen by expanding Definition 4.60:

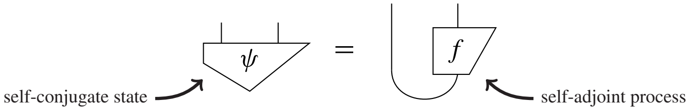


4.3.6 ⊗-Positivity

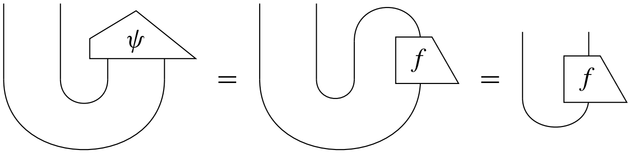
We started this chapter with the observation that, via process–state duality, we can relate ∘-separability of processes to ⊗-separability of bipartite states. Similarly, we can relate self-adjoint processes to self-conjugate states:



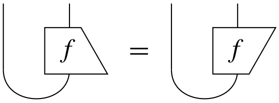
Proposition 4.63 A state ψ is self-conjugate if and only if the process f corresponding to it by process–state duality is self-adjoint:



Proof The conjugate of a bipartite state ψ is:



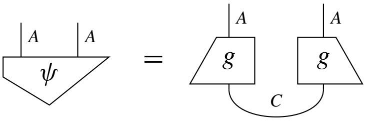
and this is equal to ψ itself if and only if:



that is, if and only if f is self-adjoint. □

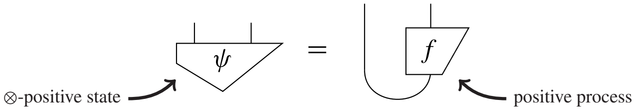
In the case of positivity we introduce a new name for its ⊗-counterpart.

Definition 4.64 A bipartite state ψ is $\otimes$*-positive* if for some g we have:



(4.47)

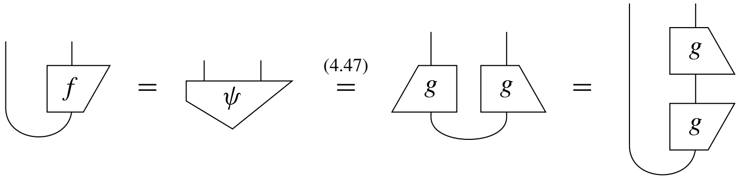
Proposition 4.65 A state ψ is $\otimes$-positive if and only if the process f corresponding to it by process–state duality is positive:



⊗-positive state

positive process

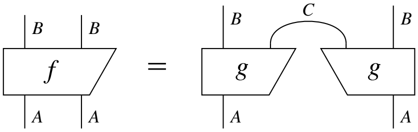
Proof When we express a bipartite state ψ in terms of the process f, then $\otimes$-positivity becomes:



for some process g, which is equivalent to positivity of f. □

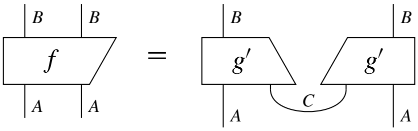
We can extend the definition of $\otimes$-positivity of states to processes.

Definition 4.66 A process f is $\otimes$*-positive* if for some g we have:



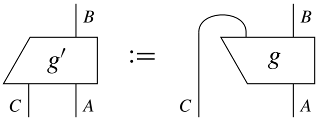
(4.48)

or equivalently, for some g' we have:

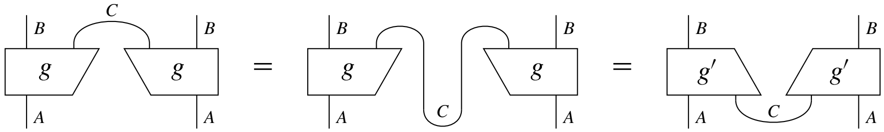


(4.49)

Setting:



one can indeed pass from equation (4.48) to (4.49) and vice versa:



As a consequence, it is easy to see that ⊗-positive states are a special case of ⊗-positive processes, where A is the trivial system.

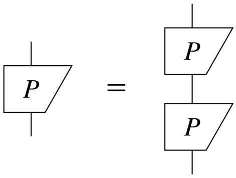
Exercise 4.67 Show that the sequential composition of two ⊗-positive processes is again a ⊗-positive process.

Example* 4.68 Some readers may be familiar with *density operators*, which are an example of positive 'processes'. Here we put processes in quotation marks, since in quantum theory density operators are used to represent states. Instead of density operators, in Chapter 6 we will use their ⊗-positive counterparts to represent quantum states. We will also use ⊗-positive processes, as opposed to *completely positive maps*, to represent quantum processes. The motivation for this is clear: states should be represented as states (not 'processes') and processes should be represented as processes (not 'super-processes', i.e. things that send processes to other processes).

4.3.7 Projectors

We extend our vocabulary about processes a bit more.

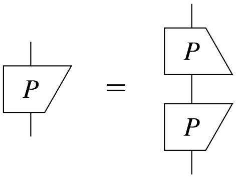
Definition 4.69 A *projector* is a process P that is positive and *idempotent*:



(4.50)

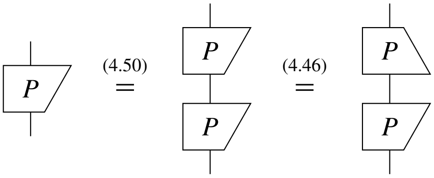
Proposition 4.70 For a process P, the following are equivalent:

(i) It is a projector.
(ii) It is self-adjoint and idempotent.
(iii) It satisfies:

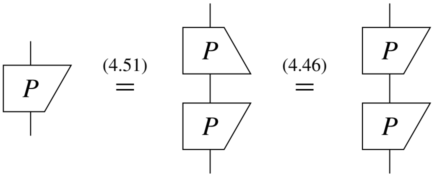


(4.51)

Proof (i ⇒ ii) follows from Proposition 4.61. For (ii ⇒ iii) we have:

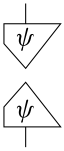


For (iii ⇒ i), it immediately follows from (4.51) that P is positive. Thus in particular, it is self-adjoint, so:



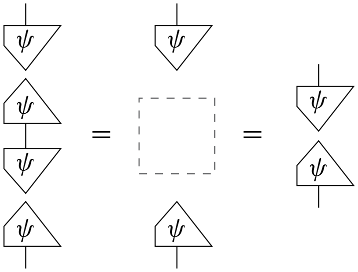
establishes idempotence. □

We can build projectors from any normalised state ψ as follows:



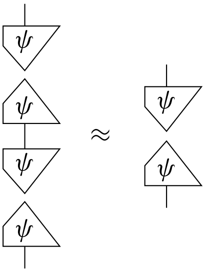
(4.52)

This is clearly positive, and for idempotence we have:



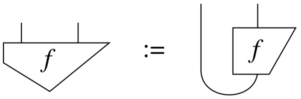
(4.53)

We will refer to these projectors as *separable projectors*. In fact, as long as $\psi \neq 0$, this recipe always yields a projector, up to a number (cf. Section 3.4.3):

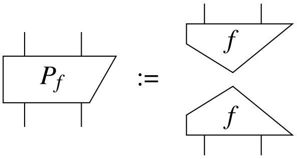


Remark* 4.71 In linear algebra, separable projectors are precisely the projections onto one-dimensional subspaces.

Now recall that any process f yields the following state, via process–state duality:

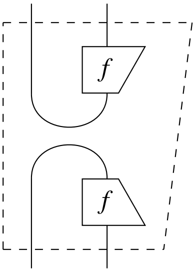


If the resulting state is normalised, we obtain a separable projector:



Of course, by process–state duality, any bipartite state can be written in terms of some process f, so we have the following.

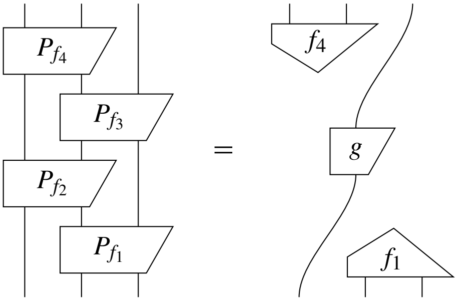
Corollary 4.72 In any process theory that admits string diagrams, every bipartite separable projector is of the form:



(4.54)

The following exercise concerns the composition of these bipartite separable projectors.

Exercise 4.73 Show that:



(4.55)

with:

$$g := f_3 \circ \bar{f}_4 \circ f_2^T \circ f_3^\dagger \circ f_1 \circ \bar{f}_2 \tag{4.56}$$

Can you generalise this particular computation into a more general statement about diagrams involving projectors of this kind?

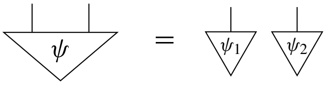
Note in particular that the order of the processes that make up g seems at first sight to be totally unrelated to the order of projectors in the LHS of (4.55). The resulting general statement on how these bipartite projectors compose was, in the early days of diagrammatic quantum reasoning, referred to as the *logic of entanglement*.

4.4 Quantum Features from String Diagrams

While not all process theories admit string diagrams, those that do have several important features in common, which may at first sight seem weird. We already saw how process–state duality assigns to each process a state that fully captures it, that the transpose assigns a converse to each process, and more generally, that we can freely interchange inputs and outputs of processes. We also just saw that bipartite projectors exhibit some pretty funky compositional behaviour. These features have no counterpart in a process theory that doesn't admit string diagrams, such as the world of **functions** (cf. Exercise 4.10). We will discuss some other simple consequences of string diagrams, which are sometimes (perhaps prematurely) branded as 'quantum weirdness', in the remainder of this chapter.

4.4.1 A No-Go Theorem for Universal Separability

One uses the terminology 'no-go theorem' to refer to a result that establishes the impossibility of something that we might assume to be true in the light of our everyday experiences. The first no-go theorem we'll meet states that if a non-trivial theory admits string diagrams, then its bipartite states cannot all be separable. Since a ⊗-separable state:



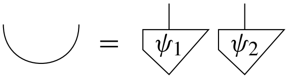
is described by describing the states ψ_1 and ψ_2 of the respective sub-systems, it follows that there must be states of composite systems that cannot be described merely by describing their parts.

To put this in everyday terms, suppose we have two things, like a plugstrip and a dodo. Then, it suffices to describe each thing individually to describe the whole system consisting of both things. In a ⊗-non-separable theory this is no longer true: the properties of the individual do not suffice to describe the properties of the whole. If this were the case for the dodo/plugstrip system, the properties of the two things would get all mixed up. For instance, the colour of the dodo's plumage could depend on whether the plugstrip has UK or EU sockets. While this situation appears to be nonsensical, there do exist some concepts in our daily world that are intrinsically ⊗-non-separable. One such concept is that of twins: twins are not defined by each member having a particular property, e.g. blond hair or being tall, but by the fact that whatever property one member has, the other one has that property as well. Such a concept could, for example, be modelled by the cups/caps in **relations** that we described in Exercise 4.16.

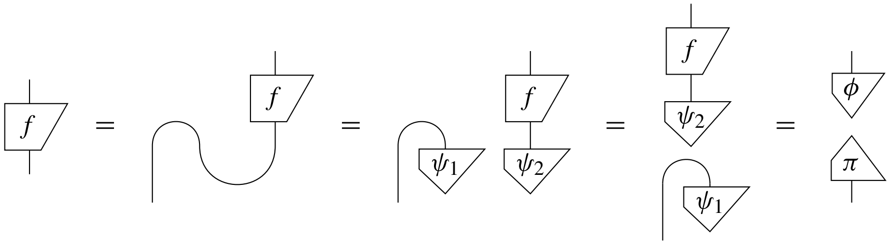
The way we establish our no-go theorem for universal separability was already alluded to in Remark 4.7. We show that if all bipartite states are ⊗-separable, then all processes are ∘-separable, so we are dealing with a process theory in which all processes are constant; i.e. nothing ever happens. As this is an absurd condition for any ‘reasonable’ process theory, all bipartite states cannot be ⊗-separable.

Proposition 4.74 If a theory is described by string diagrams, and all bipartite states are ⊗-separable, then all processes are ∘-separable.

Proof By assumption, the cup is ⊗-separable:

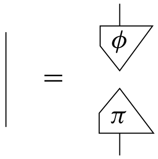


So, for any process f we have:



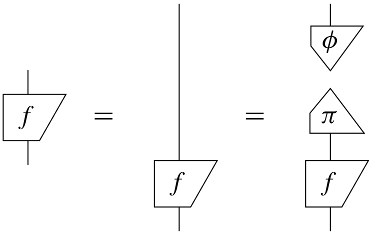
for state $\phi := f \circ \psi_2$ and effect $\pi := \psi_1^T$. □

If all processes are ⊗-separable, in particular, identities are ∘-separable:



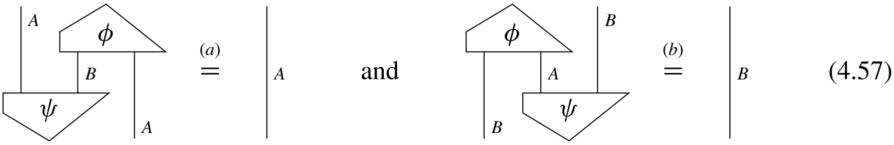
Hence, by looking close enough, one may discover that wires are actually not wires at all:

Conversely, we could as well have just shown that identities are $\circ$-separable, since then, any process is also $\circ$-separable:



Separable states inhabit one end of the spectrum of bipartite states. At the other end of that spectrum are those states that are like cups, in the sense that they satisfy yanking-like equations for some bipartite effect:

Definition 4.75 A bipartite state ψ is called *non-degenerate*, or *cup-like*, if there exists an effect ϕ such that:

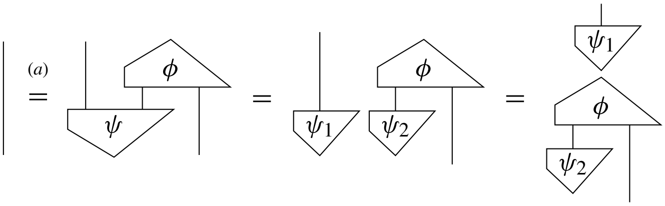


Similarly, an effect ϕ is called *non-degenerate*, or *cap-like*, if there exists a state ψ satisfying the above equations.

A single cup-like state already yields a no-go theorem on separability.

Proposition 4.76 Every non-degenerate state ψ in a non-trivial process theory must be $\otimes$-non-separable.

Proof If ψ is $\otimes$-separable, then:

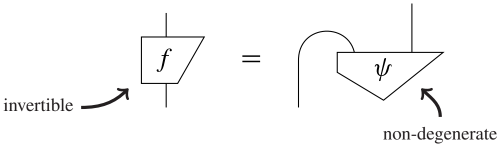


i.e. a plain wire is separable. Hence ψ cannot be separable. □

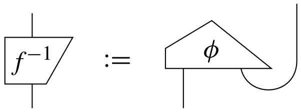
Remark 4.77 In Definition 4.75 we implicitly assumed that ψ was a ‘proper’ bipartite state. That is, neither A nor B is the trivial, ‘no wire’ type. Otherwise, ψ would of course be separable in a trivial manner.

In Section 4.3.6 we saw how certain kinds of processes translate via process–state duality into certain kinds of bipartite states. In the same vein, non-degenerate bipartite states arise via process–state duality from invertible processes.

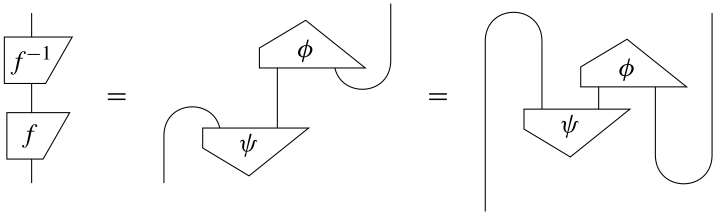
Proposition 4.78 A state ψ is non-degenerate if and only if the process f corresponding to it by process–state duality is invertible:



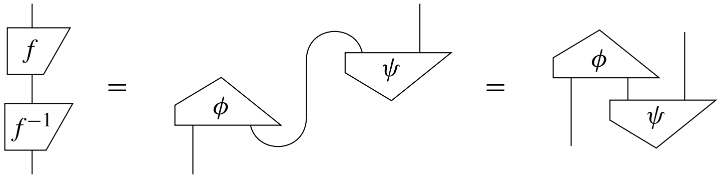
Proof Setting:



we have:



and:

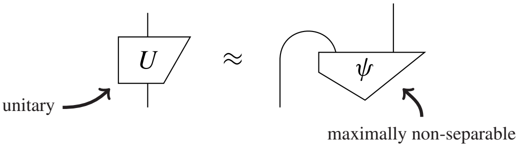


From these equations, it clearly follows that f and f^{-1} satisfy the inverse equations (4.13) if and only if ψ and ϕ satisfy the non-degeneracy equations (4.57). □

Example* 4.79 One often encounters non-degenerate effects like ϕ in linear algebra, under the name *non-degenerate bilinear form*. In quantum entanglement theory, non-degenerate states are called *states of full (Schmidt) rank*, or *SLOCC-maximal states*, for reasons that we explain in Section 13.3.2.

Unitary processes are special kinds of invertible processes, so there also are corresponding bipartite states that further specialise cup-like states.

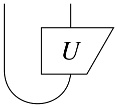
Definition 4.80 A state ψ is *maximally non-separable* if it corresponds to a unitary U by process–state duality, up to a number:



(4.58)

Note that we use $\approx$ to allow ψ to still be normalised even when the RHS of equation (4.58) is (almost) never normalised.

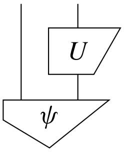
Exercise 4.81 What is the squared norm of:



when U is unitary? When is this state normalised?

The following exercise indicates the utility of unitary processes in inter-converting maximally non-separable states.

Exercise 4.82 Show that if one applies a unitary to one side of a maximally non-separable state:



that one again obtains a maximally non-separable state and that this unitary can always be chosen such that the resulting state is the cup (up to a number).

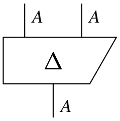
Example* 4.83 In quantum theory, maximally non-separable states as defined above will be called *maximally entangled states*, or *LOCC-maximal states*, for reasons that we will explain in Section 13.3.1.

4.4.2 Two No-Go Theorems for Cloning

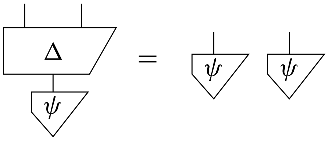
One feature that is characteristic of classical computation is that we can copy bits at will. For example, this book was once a PDF file on our computers, but it has since been copied and sent all over the place. While this seems like a totally obvious thing to do, 'copying' is no longer possible when we are dealing with states and processes in string diagrams! While at first this may seem really bad, this matches what goes on in the quantum world, and it also has some suprising benefits. For example, if someone wants to keep a secret, like the PIN code of a bank card, it becomes much harder in a world without copying for someone to steal the code without the card owner noticing. This principle forms the basis for quantum cryptography as well as many other quantum security protocols, as we will see in Section 9.2.6.

So far, most of the calculations we have done have been fairly trivial, and all the things that we proved were plainly obvious from the string diagram language. However, the next two theorems, and in particular the first one, will require some slightly more intricate diagram acrobatics.

By a *cloning process* for a system of type A we mean a process:

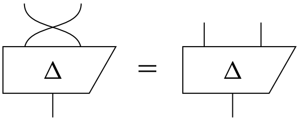


that turns an input state ψ into two copies of ψ:



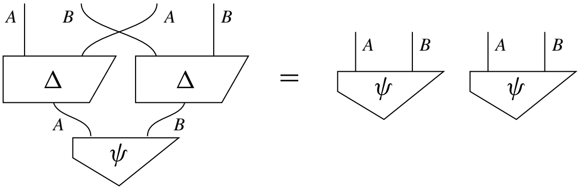
(4.59)

There are some obvious conditions one expects from such a process. For example, since it is producing two identical copies of a state as output, it should not matter if we interchange them:



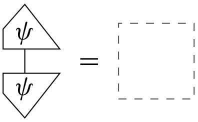
(4.60)

Furthermore, when we have two cloning processes, one for type A and one for type B, then we should be able to clone a state of type $A \otimes B$ just by cloning each system individually:



(4.61)

Finally, we will assume our process theory contains at least one normalised state. That is, at least one state ψ such that:

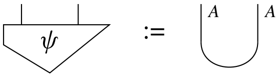


(4.62)

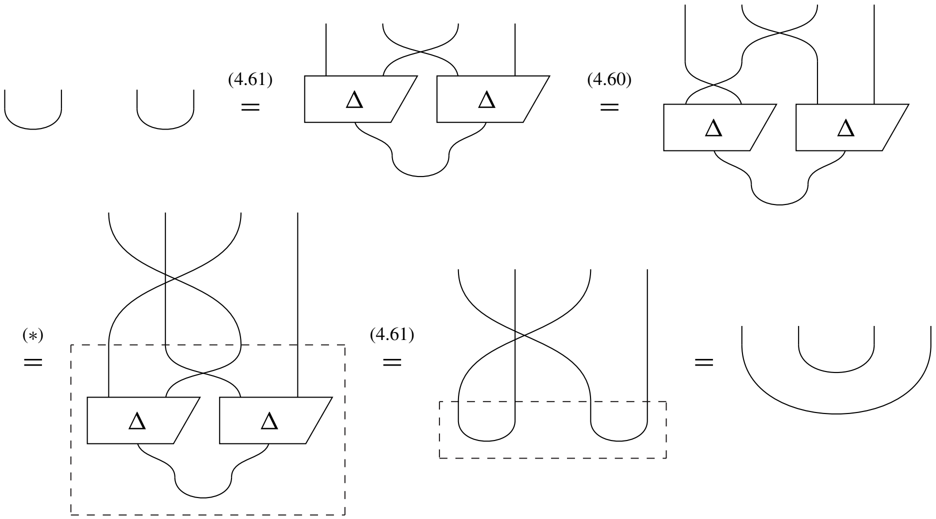
In essence, this a trivial assumption since otherwise there is nothing to be cloned anyway!

Theorem 4.84 Consider a process theory that admits string diagrams. If there is a cloning process of type A that satisfies (4.60), (4.61), and (4.62), then every process that has A as its input system has to be $\circ$-separable.

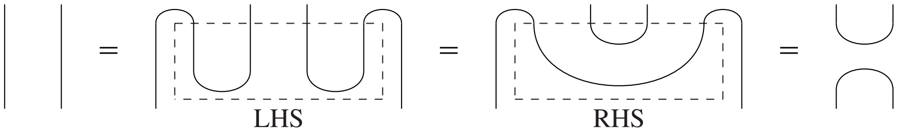
Proof Applying (4.61) for:



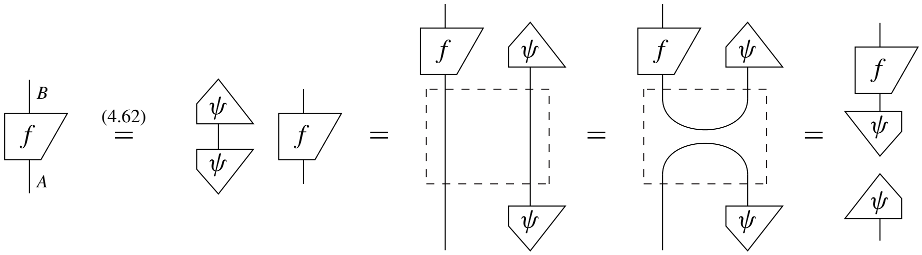
we have:



where all the wires have type A and $(*)$ is just a deformation of the diagram. Then converting the external outputs of the LHS and RHS into inputs:



we discover that an identity on a pair of systems is separable, which should already raise some eyebrows. Substituting the above equation into the dotted areas below:

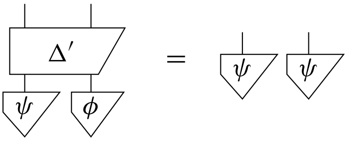


so we indeed obtain that any process f that has A as its input type is $\circ$-separable. □

Thus, in the light of our conception of 'trivial' outlined in Section 4.4.1, if there is a cloning process for a certain system-type A, then the process theory is trivial with respect to type A, and if every system-type admits a cloning process, then the process theory is trivial as a whole.

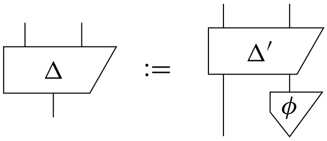
The assumption that really makes things go wrong is (4.61). If one 'thinks' in the language of string diagrams, it is pretty obvious that trying to copy a non-separable state by doing something to each of the subsystems isn't going to work.

Theorem 4.84 is quite different from the one that one finds in most textbooks. First, a cloning process is usually introduced as a process with two inputs and two outputs, whose second input is in some fixed state ϕ, which gets overwritten by the copied state ψ:



(4.63)

Of course, a one-input, two-output cloning process arises as follows:



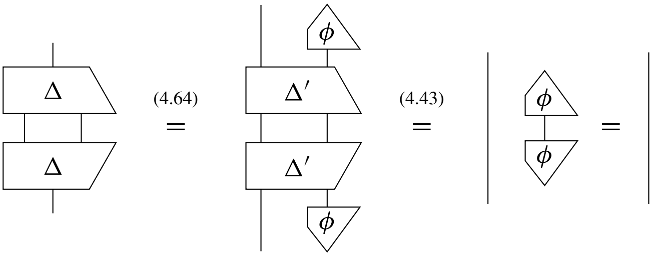
(4.64)

and we could then just as easily use (4.64) to prove Theorem 4.84.

Second, one usually assumes that the process Δ' in equation (4.63) is unitary, which in turn means we don't need assumptions (4.60) and (4.61).

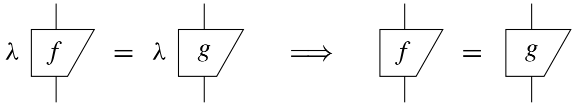
This is such a profound difference that it results in what should be considered as a different theorem, despite the fact that it aims to establish the same feature. The assumption of unitarity is motivated by quantum theory itself (cf. Section 6.2.6). In spite of the fact that this theorem relies on extra assumptions, it is still of interest because it additionally demonstrates precisely which sets of states *can* be jointly cloned by a single process, namely the *orthogonal* ones. This joint-clonility of orthogonal states is closely connected to the fact that they can be used to encode classical data within quantum systems, which we will exploit in Chapter 8 to give an elegant diagrammatic representation of classical data.

Instead of using Δ', we will use Δ defined as in equation (4.64). By unitarity of Δ', when ϕ is normalised, Δ must be an isometry:



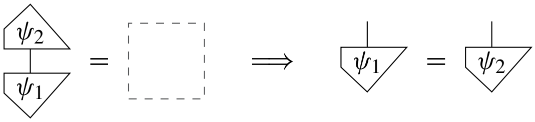
So we assume the existence of an isometry Δ satisfying the cloning equation (4.59). Of course the biggest extra assumption in the traditional no-cloning theorem is that we are dealing specifically with the theory of **quantum processes** (which of course, we haven't even defined yet!). However, this assumption is a bit overkill. All we really need is to assume a couple of things about the numbers of our theory:

(a) All non-zero numbers are *cancellable*, that is, if $\lambda \neq 0$:

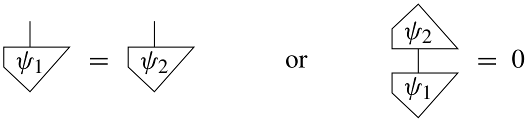


which is evidently the case for numbers representing either possibilities (cf. **relations**) or probabilities.

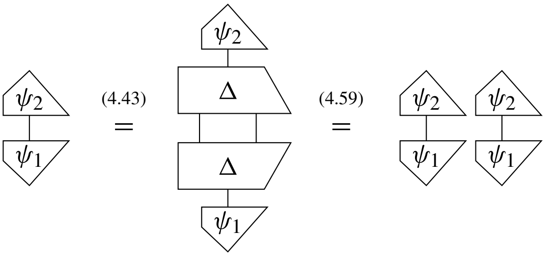
(b) The number 1 means ‘certain’, as discussed in Section 4.3.1. That is, for normalised ψ_1, ψ_2 we have:



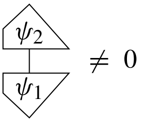
Theorem 4.85 In any process theory that admits string diagrams and satisfies conditions (a) and (b) above, if two normalised states ψ_1 and ψ_2 can both be cloned by an isometry Δ, then they must either be equal or orthogonal. That is, we either have:



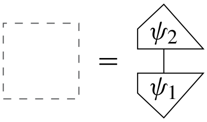
Proof First, note that:



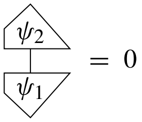
Next, consider two cases. If:



by assumption (a), we can cancel this number on both sides to obtain:



so by assumption (b) the states ψ_1 and ψ_2 are equal. On the other hand, if:



then ψ_1 and ψ_2 are orthogonal. □

Theorem 4.85 immediately yields the second no-cloning theorem.

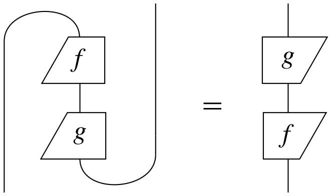
Corollary 4.86 Under the assumptions of Theorem 4.85, if a process theory has at least two states of type A that are neither equal nor orthogonal, then there is no cloning process of type A.

Remark 4.87 Note that the first no-cloning theorem applies to **relations**, whereas the second one doesn't, since it relies on condition (b). In fact, the proof of the first no-cloning theorem doesn't rely on adjoints at all, so in that sense is more general. However, the second no-cloning theorem does avoid assuming extra equations about the cloning device (notably equation (4.61) for cloning joint states), so it is not directly implied by Theorem 4.84.

Remark 4.88 There is a subtlety about the statement 'we can copy bits at will', namely, this assumes bits are *deterministic*, i.e. they have definite values. We won't discuss this any further now. A full discussion of this issue, as well as a more refined no-go theorem that avoids it, is given in Section 6.2.8.

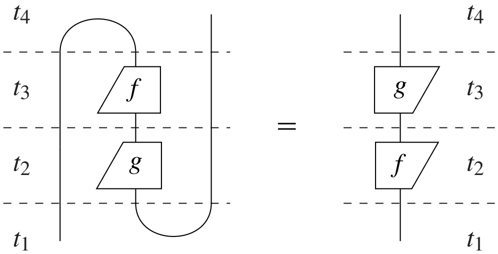
4.4.3 As If time Flows Backwards

Consider the following equal string diagrams:



(4.65)

Suppose we interpret the LHS and the RHS as processes happening at specific points in time $t_1, \ldots, t_4$:



(4.66)

then something strange happens. When considering the LHS we have:

t_2: something happens involving g;
t_3: something happens involving f;

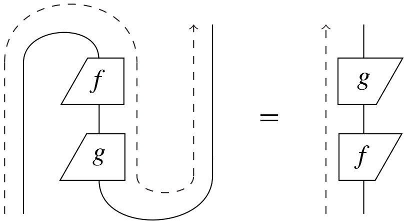
On the other hand, when considering the RHS:

t_2: something happens involving f;
t_3: something happens involving g.

While for the LHS the output of the g-labelled process is the input to the f-labelled process, for the RHS the output of the f-labelled process is the input to the g-labelled process. So the order in which the processes happen is reversed!

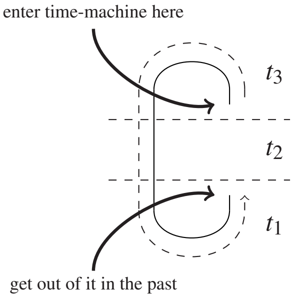
The reason that something like this can happen is that two equal diagrams may actually correspond to very different *operational scenarios*. An operational scenario is the manner in which a diagram is actually realised, e.g. by wiring together some devices in a lab. The LHS and RHS above have different operational readings, since the LHS involves three systems at times t_2 and t_3, while the RHS only involves one system at any time. The cups correspond with the creation of two systems, while caps correspond with the annihilation of two systems. As a result, we have a more complicated operational scenario in the LHS, which is then simplified in the RHS.

Despite the difference in the two operational readings, what actually happens, according to our theory, is the same. We might call this the *logical reading* of a string diagram. In this case, we can see that these diagrams have the same logical reading by tracing the logical flow of the diagram:



which passes first through the f-labelled box and then the g-labelled box.

In the case of the logical reading, the cup and the cap make it seem as if systems are actually travelling back in time! This has even led several researchers to propose a cup and a cap as a model for time travel:

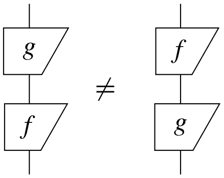


Does this mean that cups and caps provide a pathway to effectively building a time machine? Of course they don't! The main problem here is the fact that, as we shall see later, caps cannot be implemented with certainty, but only with some probability. The reason for this is what we will refer to as the *causality postulate* (see Section 6.4). In particular, if anyone proposes to you a trip in such a machine, we highly recommend you not to take it, since as we shall see in Section 6.4.4, you'd come out pretty scrambled up!

Exercise 4.89 Let $\mathbb{T} := \{0, 1, 2\}$ be a three-element set (the set of 'trits'). Define relations $f, g : \mathbb{T} \to \mathbb{T}$ as follows:

- $f :: \{0 \mapsto 1, 1 \mapsto 0, 2 \mapsto 2\}$;
- $g :: \{0 \mapsto 0, 1 \mapsto 2, 2 \mapsto 1\}$.

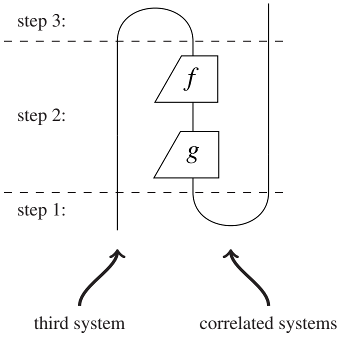
Note in particular that these relations do not commute:



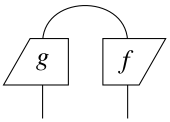
Let the cup and the cap be those of Exercise 4.16. Verify the 'time reversal' equation (4.65) by explicit composition of relations.

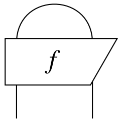
Exercise 4.89 demonstrates how the 'time reversal' can be realised by means of non-deterministic processes, in the following manner:

Step 1: Create non-deterministic perfect correlations for two systems.
Step 2: Apply two consecutive operations to one of the systems.
Step 3: Impose that the output matches the state of a third system.



In addition to string diagrams that simply have strange logical readings, we may have diagrams that have more than one such reading, or none at all, for example:



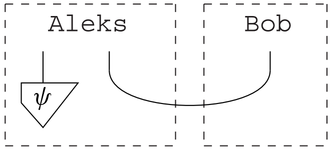


In the left diagram, is the ‘flow’ from g to f or from f to g? And who knows what is even happening in the diagram on the right? However, we can still make sense of such diagrams by thinking about wires less as ‘flow’, which has a definite direction, and more as ‘forcing the value at both ends to be the same’, which is a symmetric concept.

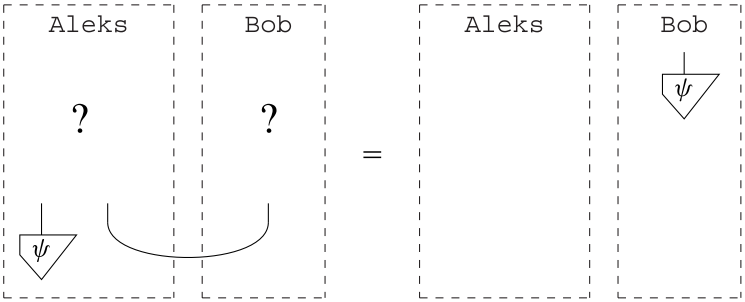
4.4.4 Teleportation

We’ll now meet one of the ‘killer apps’ of quantum theory for the first time: quantum teleportation. Perhaps surprisingly, the real ‘meat’ of quantum teleportation can already be described using just the diagrammatic concepts we’ve met so far. In Chapter 6, we will revisit this protocol and fill in the details, such as careful definitions of *quantum states* and *measurements*.

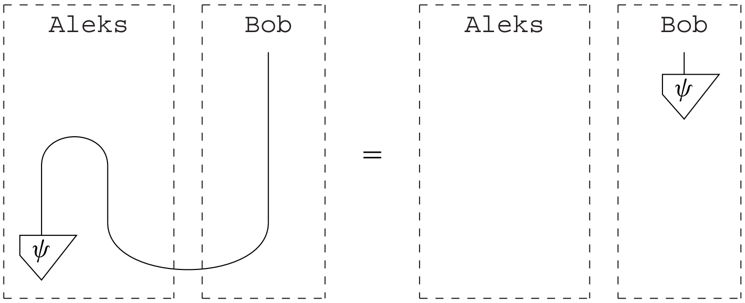
Here is the challenge. Aleks and Bob are far apart, Aleks possesses a system in a state ψ, and Bob needs this state. Suppose they also share another state: the cup state. So, we have this situation:



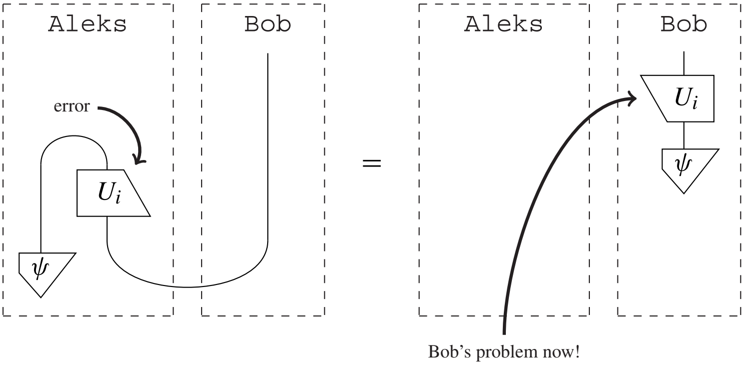
Starting from this arrangement, is there something Aleks and Bob can do in the regions marked ‘?’ below, that results in Bob obtaining ψ?



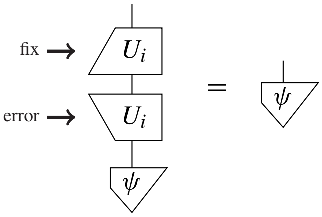
Here is a simple solution:



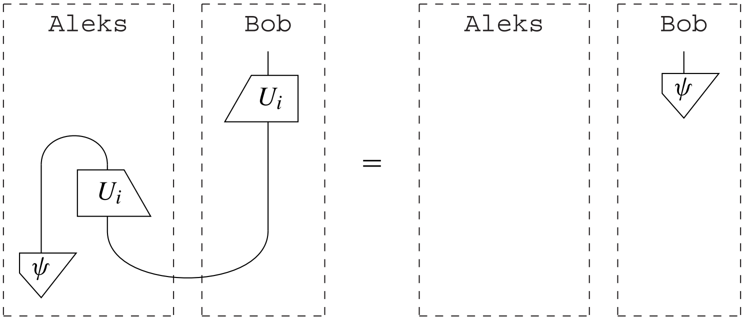
But is the cap (i.e. an effect) really a process that Aleks can 'do'? Well, not exactly. We'll see in Chapter 6 that effects arise as the result of quantum measurements. Now, a tricky issue about measurement is that Aleks might not get the effect he wants (i.e. the cap), but rather the cap with some (non-deterministic) error, which we can represent as a box:



Aleks doesn't know in advance which error he will get, just that it will be in some set $\{U_0, U_1, \ldots, U_{n-1}\}$. It could be the case that U_0 is the identity process, i.e. no error. If he's feeling lucky, he'll just hope this happens, and if it doesn't, he'll try the whole thing again with a new ψ and a new cup state. This is called *post-selection*. Thinking about it a bit more, Aleks realises this seems a bit wasteful. Moreover, if he only possesses one system in the state ψ, that state will be lost forever. Instead, Aleks decides to call up Bob and tell him to fix the error. Bob can achieve that as follows, in the particular case that U_i is a unitary process:

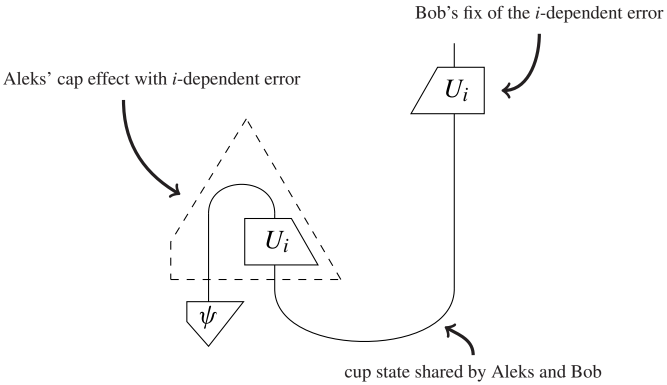


So, all Aleks needs to tell Bob is the value of $i \in \{0, \ldots, n-1\}$ so he knows which correction to perform. And poof! Bob now has ψ :



(4.67)

So this is it: quantum teleportation. In summary:


Bob's fix of the i-dependent error
Aleks' cap effect with i-dependent error
U_i
U_i
ψ
cup state shared by Aleks and Bob

(4.68)

One important thing to note is that it is crucial for Aleks to send Bob the value of i, otherwise Bob can't fix the error. In that case, Bob will only get noise, as we shall see in Section 6.4.4. This is why, for example, quantum teleportation is compatible with relativity theory, which forbids sending any signal faster than the speed of light. As a consequence, when we use the term 'teleportation', we don't really mean magically beaming something through space. Instead, the teleportation protocol uses one kind of information to send another kind of information. In this protocol, Aleks communicates some *classical data* to Bob, and as a result, Bob obtains *quantum data* (i.e. a quantum state). This is genuinely surprising, since, as we shall see in Chapter 7, the space of possible quantum states is infinitely larger than the number of possible classical values.

Exercise 4.90 Write the teleportation protocol of (4.67):

1. as a diagram formula, and
2. algebraically, using $\otimes$ and $\circ$.

We're not quite ready to fully describe quantum teleportation. However, there exists a totally classical analogue to quantum teleportation that can be described using **relations**.

Example 4.91 ('classical' teleportation) Suppose Aleks and Bob both have envelopes with a card inside that says either '0' or '1'. They don't know which it is, but they do know that they both have the <u>same</u> card. We can represent this non-deterministic state as the cup relation:

$$\cup :: \begin{cases} * \mapsto (0,0) \\ * \mapsto (1,1) \end{cases}$$

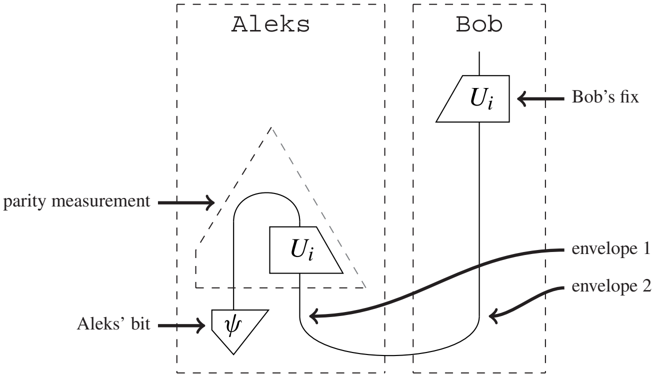
Now, suppose Aleks has a bit ψ that he wants to send to Bob. As before, they have a telephone, but unlike before, Aleks has a classical bit, so he could just look at it, ring up Bob, and tell him what it is. On the other hand, Aleks is a bit paranoid and doesn't want any potential eavesdroppers to get a hold of ψ. So instead of just telling ψ to Bob, he computes the parity of ψ together with the bit stored in his envelope. That is, he looks at both bits and tells Bob over the phone whether they are the same, which corresponds to the effect M_0, or different, which corresponds to the effect M_1:

$$M_0 :: \begin{cases} (0,0) \mapsto * \\ (1,1) \mapsto * \end{cases} \qquad M_1 :: \begin{cases} (0,1) \mapsto * \\ (1,0) \mapsto * \end{cases}$$

If they are the same, Bob knows that the card in his envelope is the bit ψ. If they are different, he knows that ψ is the negation of the bit in his envelope. In other words, based on the outcome of Aleks' parity measurement, Bob chooses a correction to apply to the bit in his envelope:

$$U_0 :: \begin{cases} 0 \mapsto 0 \\ 1 \mapsto 1 \end{cases} \qquad U_1 :: \begin{cases} 0 \mapsto 1 \\ 1 \mapsto 0 \end{cases}$$

Noting that the effect M_i can be rewritten in terms of a cap and U_i, we can stick this all together, giving us this picture:



which, of course, is just teleportation!

Remark 4.92 In the world of computer security, Example 4.91 is called *one-time pad encryption*. The bits in the envelopes correspond to the shared key, or 'pad'; the parity measurement *encrypts* Aleks' bit; and Bob's correction *decrypts* it. Above we noted that a teleportation protocol uses one kind of information to send another. This interpretation applies here as well. In this case, Aleks sends *public* (i.e. encrypted) data to Bob, and Bob receives *private* (i.e. unencrypted) data at the end. So, the analogy between classical and quantum teleportation goes like this:

	Aleks sends	Bob receives	Using a shared
One-time pad encryption:	public data	private data	encryption key
Quantum teleportation:	classical data	quantum data	quantum state

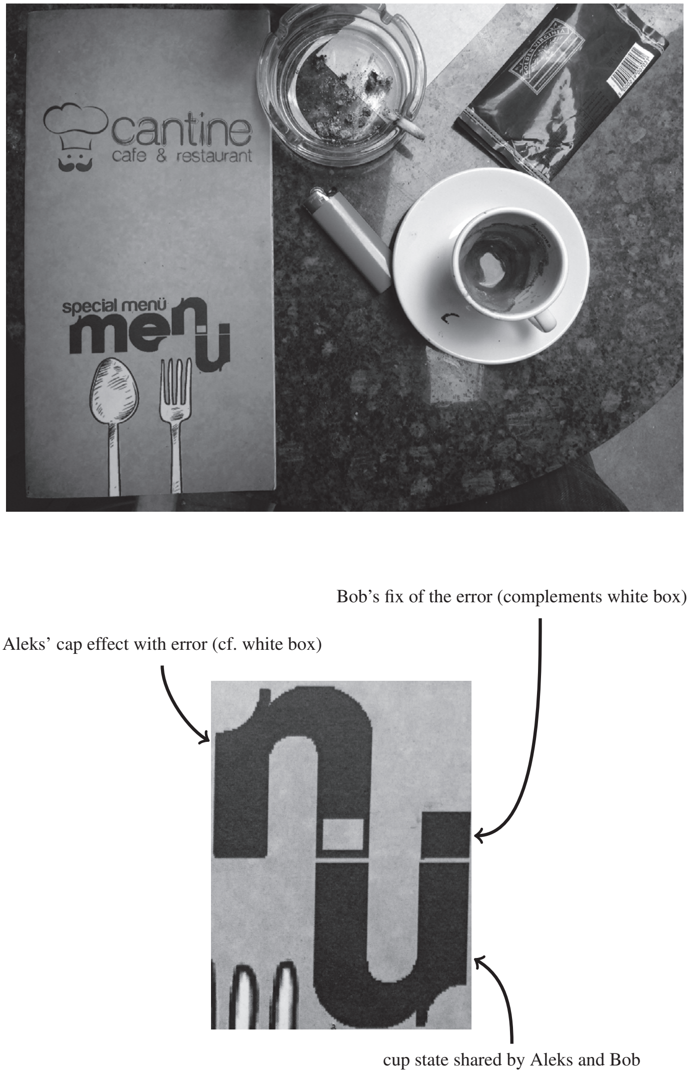
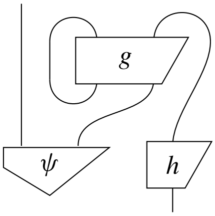

cantine
cafe & restaurant
special menü
menu
Bob’s fix of the error (complements white box)
Aleks’ cap effect with error (cf. white box)
cup state shared by Aleks and Bob

Figure 4.1 Teleportation is everywhere when you know how to recognise it. Here it is in the highly recommended Cafe Cantine in Istanbul’s Taksim square area.

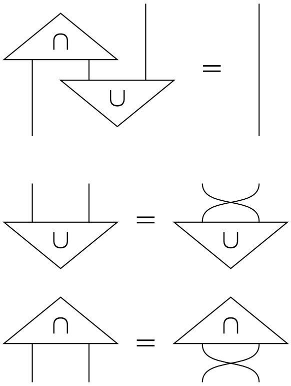
4.5 Summary: What to Remember

1. *String diagrams* realise what Schrödinger singled out as the characteristic trait of quantum theory: non-separability. They have the following two equivalent characterisations:

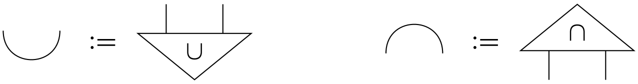
Diagrams in which wires may connect inputs to inputs and outputs to outputs, resulting in cup- and cap-shaped wires:



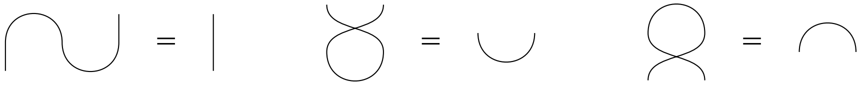
Circuits for which each type has a special state and a special effect, the *cup state* and *cap effect*, which satisfy equations:



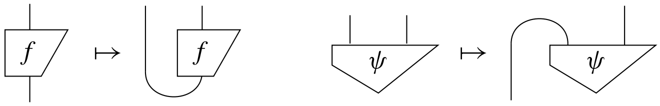
These two characterisations are related by setting:



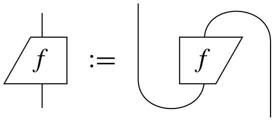
and then the defining equations of the cup state and the cap effect become the *yanking equations*:



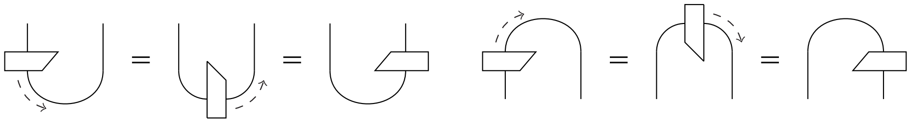
2. For string diagrams there is a bijective correspondence between processes and bipartite states, called *process–state duality*, which is realised as follows:



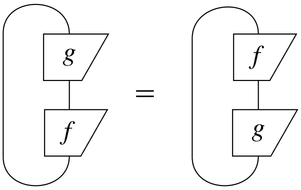
3. The string diagram language enables us to define the *transpose*, which we represent the transpose as a 180° rotation:



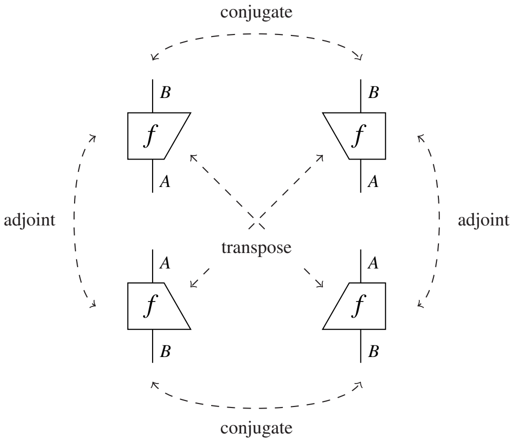
As a consequence, boxes can slide along cups and caps:



4. It also enables us to define the *trace* (see below), which gives us a way to assign numbers to processes and obeys *cyclicity* :

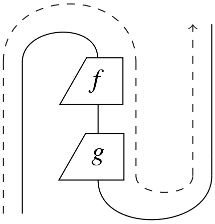


5. We also introduce vertical reflection of diagrams. *Adjoints* give an interpretation for vertical reflection within a process theory and associate to each state the effect that tests for that state. They also enable us to define an *inner product*, as well as *isometries*, *unitary* processes, *positive* processes, and *projectors* (see below for each of these).
6. Combining vertical reflection with transpose-rotation, we get a second, horizontal reflection, called the *conjugate*:

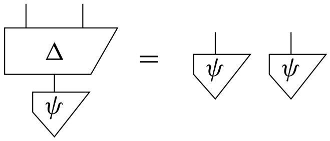


Conjugates enable us to define ⊗-*positive* bipartite states (see below).

7. We identified several ‘physical’ features of string diagrams:
   - In any non-trivial theory there always exist non-separable states, most notably the cup-state. This state establishes perfect correlations and can create the illusion of systems travelling back in time:

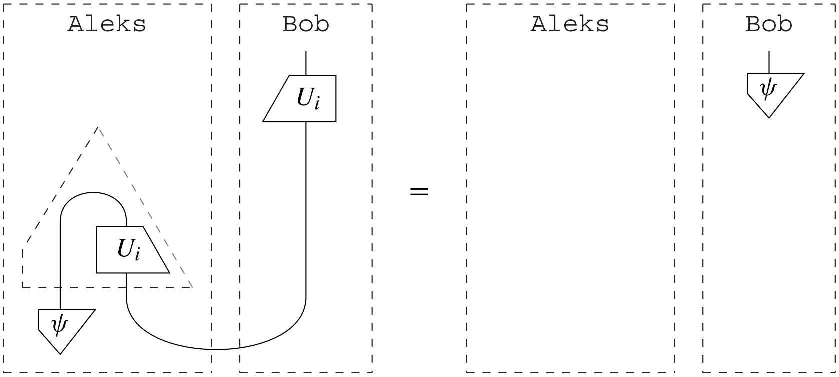


   - There does not exist an operation:

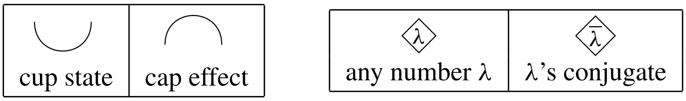


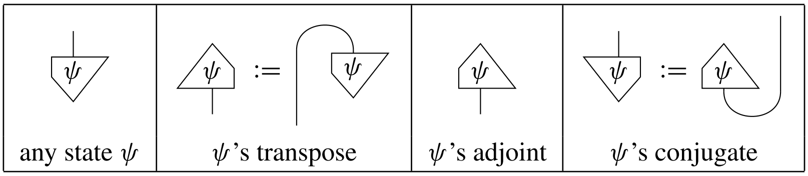
that clones all states ψ. In particular, the only states that can jointly be cloned by an isometry must be orthogonal.

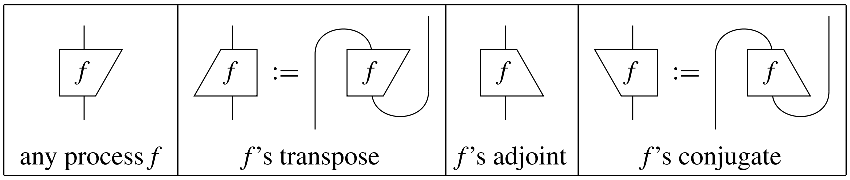
- We can describe *teleportation*, which boils down to this equation:

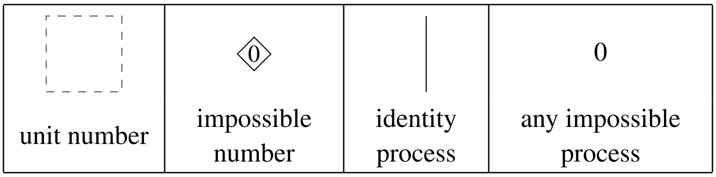


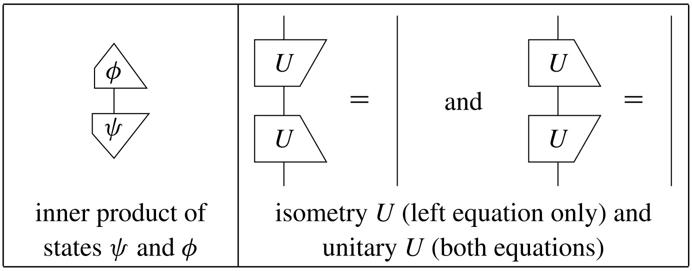
8. We introduced a plethora of diagrammatically defined concepts, which we summarise here for the reader's convenience:





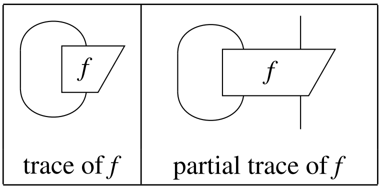






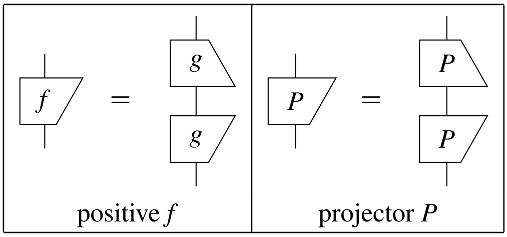
inner product of states ψ and ϕ

isometry U (left equation only) and unitary U (both equations)



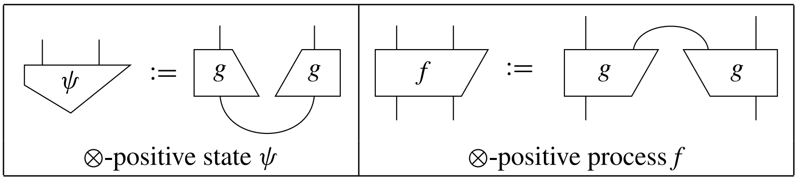
trace of f

partial trace of f



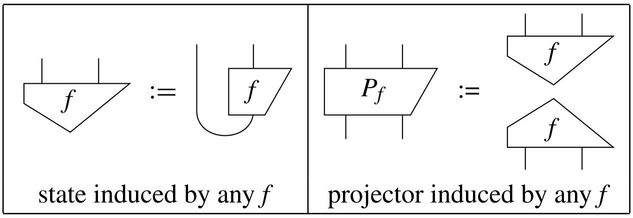
positive f

projector P



$\otimes$-positive state ψ

$\otimes$-positive process f



state induced by any f

projector induced by any f

4.6 Advanced Material*

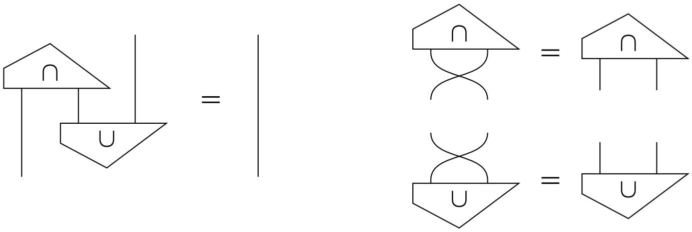
Following on from the advanced material in the previous chapter, we will now look at how the additional structure of string diagrams is defined in terms of abstract tensor systems, on the one hand, and symmetric monoidal categories, on the other. In the case of the latter, before we can do so we will need to refine our treatment of cups and caps a bit more.

4.6.1 String Diagrams in Abstract Tensor Systems*

String diagrams arise in abstract tensor systems just by adjoining special tensors $\cup^{AB}$ and $\cap_{AB}$ satisfying:

$$\cap_{AB}\cup^{BC} = \delta_A^C \qquad \begin{array}{c} \cap_{BA} = \cap_{AB} \\ \\ \cup^{BA} = \cup^{AB} \end{array}$$

Drawing these equations as pictures, these are just the normal identities we expect from caps and cups:



In tensorial language, the cups and caps are sometimes called *index-raising* and *index-lowering* operations, because composing other tensors with them raises or lowers an index (i.e. changes an input to an output or vice versa). This trick should already be familiar from Section 4.2:

$$f'^{BC}_{A} = f^{B}_{AC'}\cup^{C'C} \qquad\qquad f'^{B}_{AC} = f^{BC'}_{A}\cap_{C'C}$$

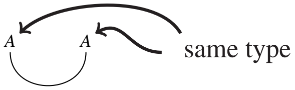
The tensor $\cap_{AB}$ plays a key role in *differential geometry*, which is a branch of mathematics that studies geometrical properties from inside curved surfaces. The cap is called the *metric* of a space. In that context, a more familiar notation for cups and caps is the following:

$$g^{\mu\nu} := \cup^{AB} \qquad\qquad g_{\mu\nu} := \cap_{AB}$$

where $\mu := A$ and $\nu := B$. Much like the inner product can be used to calculate lengths of vectors in plain old Cartesian space, the metric can be used to calculate lengths of paths in more general spaces, which can be curved and distorted. These spaces play a central role in general relativity, where the effects of gravity amount to distortions in spacetime (hence the use of ‘g’).

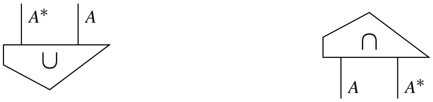
4.6.2 Dual Types and Self-Duality*

In order to simplify the presentation of caps and cups, we assume throughout this book that caps and cups satisfy the property of *self-duality*:

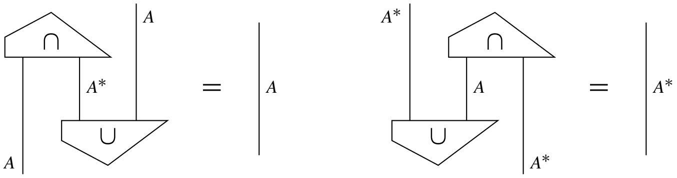


or, in other words, that all of our types are *self-dual*. We can relax the requirement that these types are the same. Rather than taking cups and caps as the main actors in the definition, the focus now moves to the ‘other type’ involved in cups and caps, and ‘having cups and caps’ becomes ‘having duals’.

Definition 4.93 For any type A, the type A^* is called the *dual type* of A if there exists a *cup state* and a *cap effect*:



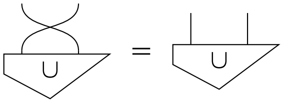
satisfying the *yanking equations*:



(4.69)

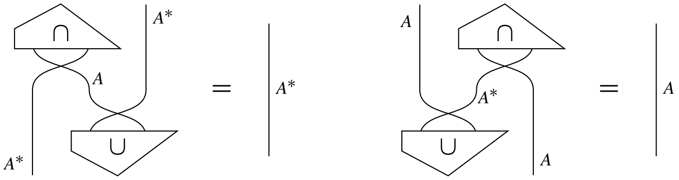
A type A is called *self-dual* if $A = A^*$.

Recalling Proposition 4.13 where we gave two equivalent presentations of the yanking equations, one can see that the equations above generalise version (ii). If A is self-dual and additionally satisfies:



(4.70)

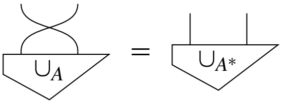
we say it is *coherently self-dual*. If $A \neq A^*$, this isn’t even a meaningful equation anymore, since the types on the LHS and RHS don’t match. However, just by deforming equations (4.69), we can see that the LHS above gives a cup for A^*. Pairing with the appropriate cap yields:



So, A is in fact a dual type for A^*. Thus we can let:

$$(A^*)^* := A$$

in which case (4.70) becomes:



(4.71)

When $A = A^*$, we have two ways to build a cup with the same type (either as $\cup_A$ or $\cup_{A^*}$), so (4.70) says that they should be equal.

For many examples in mathematics, it may indeed be more natural to treat a type as different from its dual type. An important example comes from **linear maps**, which are presented in Chapter 5. For those unfamiliar with vector spaces, linear maps, or the tensor product, it might be worth reading Chapter 5 before looking at the next example.

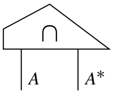
Example 4.94 For a finite-dimensional vector space A, let A^* be the *dual space* of A. That is, the elements $\xi \in A^*$ are themselves linear maps from A into $\mathbb{C}$. These form a vector space by letting addition and scalar multiplication act 'point-wise':

$$(\xi + \eta)(v) := \xi(v) + \eta(v) \qquad\qquad (\lambda \cdot \xi)(v) := \lambda\xi(v)$$

Furthermore, any basis $\{\phi_i\}_i$ in A fixes a *dual basis* $\{\widetilde{\phi_i}\}_i$ via:

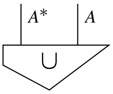
$$\widetilde{\phi}_i(\phi_j) := \delta_i^j$$

Now, we can show that A^* is the *dual type* of A, by defining a cup state and cap effect. The choice of cap effect is very natural; we just take the effect that evaluates $\xi \in A^*$ at the vector $v \in A$:



$$:: (v \otimes \xi) \mapsto \xi(v)$$

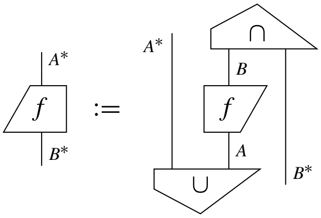
The cup state is given by summation over a basis:



$$:: 1 \mapsto \sum_i \widetilde{\phi}_i \otimes \phi_i$$

It is possible to check that this cup and cap satisfy (4.69), and one can also show that, unlike their self-dual variations, these cups and caps don't depend on the choice of basis $\{\phi_i\}_i$.

If we transpose $f : A \to B$ with respect to these new caps and cups, we get a new map $f^* : B^* \to A^*$:

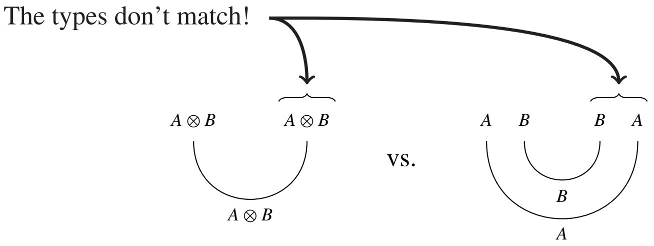


which is sometimes called the *linear operator transpose*. Concretely, f^* sends an element of B^* to an element of A^* by precomposing with f:

$$f^*(\xi) := \xi \circ f$$

which again doesn't depend on a choice of basis (unlike the normal transposition).

Even when it is possible to choose A to be self-dual, having the freedom to chose A^* can sometimes be helpful. For instance, consider the 'nested caps' problem from Section 4.2.2:

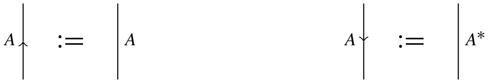


This problem goes away if we let:

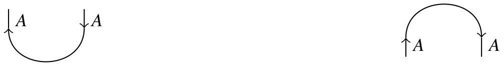
$$(A \otimes B)^* := B^* \otimes A^*$$

But we seem to have lost some notational niceness when $A \neq A^*$. Rather than representing caps and cups as a piece of wire, we seem forced to use explicit boxes for ∩ and ∪, otherwise we'll end up with wires that say one type at one end and another type at the other:

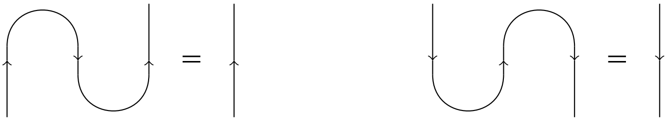
That doesn't look right! However, there is a very elegant way to deal with duals graphically. We just introduce a *direction* to the wires:



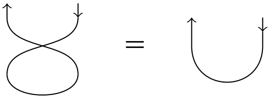
Wires of 'normal' types A are depicted as wires directed upwards (i.e. as 'stuff flowing forwards in time'), whereas wires of dual types A^* are depicted as wires directed downwards (i.e. as 'stuff flowing backwards in time'). In either case, we label the wire simply as A and use the direction to tell us whether the wire is A or A^*. This little tweak to the notation allows us to once again represent caps and cups as pieces of wire, but using directed wires this time:



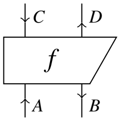
Equations (4.69) become:



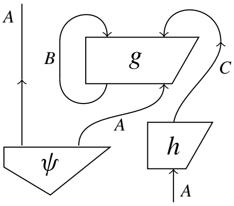
and (4.71) becomes:



Maps to and from duals are depicted as boxes connected to wires with the directions reversed. For example, a box with input type $A \otimes B^*$ and output type $C^* \otimes D$ looks like this:



Just as we were able to define string diagrams without referring to caps and cups (cf. Definition 4.18), we can define *directed string diagrams* as diagrams where we are allowed to connect any two wires, provided that the types and directions are compatible:



That is, if we connect an input to an output, the types should be the same. If we connect an input to an input (or an output to an output), the types should be dual to each other. For instance, the output of h with type C^* is connected to the output of g with type C.

4.6.3 Dagger Compact Closed Categories*

A compact closed category is a symmetric monoidal category where every object has a dual. We can say this in categorical language as follows.

Definition 4.95 A *compact closed category* is a symmetric monoidal category $\mathcal{C}$ such that for every object $A \in \mathrm{ob}(\mathcal{C})$, there exists another object $A^* \in \mathrm{ob}(\mathcal{C})$ and morphisms:

$$\epsilon_A : A \otimes A^* \to I \qquad\qquad \eta_A : I \to A^* \otimes A$$

such that:

$$(\epsilon_A \otimes 1_A) \circ (1_A \otimes \eta_A) = 1_A \qquad\qquad (1_{A^*} \otimes \epsilon_A) \circ (\eta_A \otimes 1_{A^*}) = 1_{A^*} \tag{4.72}$$

Remark 4.96 The adjective ‘closed’ means that for every two objects A and B there is a special object $[A \to B]$ whose states $\psi : I \to [A \to B]$ encode morphisms in $\mathcal{C}(A, B)$. For example, in the category that has sets as objects and functions as morphisms, $[A \to B]$ is the set of all functions from A to B, whereas in the category that has vector spaces as objects and linear maps as morphisms, $[A \to B]$ is the vector space of linear maps from A to B. ‘Compact closed’ means that these special objects take the form:

$$[A \to B] := A^* \otimes B$$

i.e. that the category has process–state duality.

We can also account for adjoints in category-theoretic terms.

Definition 4.97 A *dagger functor* for a symmetric monoidal category is an operation † that doesn’t alter objects:

$$A^\dagger := A$$

reverses morphisms:

$$(f : A \to B)^\dagger := f^\dagger : B \to A$$

is involutive:

$$(f^\dagger)^\dagger = f$$

and respects the symmetric monoidal category structure:

$$(g \circ f)^\dagger = f^\dagger \circ g^\dagger \qquad (f \otimes g)^\dagger = f^\dagger \otimes g^\dagger \qquad \sigma_{A,B}^\dagger = \sigma_{B,A}$$

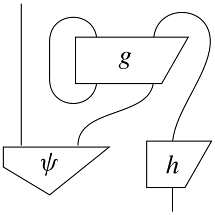
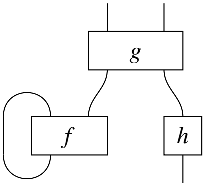
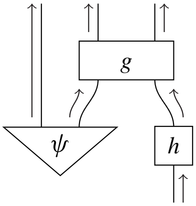
A *dagger compact closed category* is a compact closed category $\mathcal{C}$ with a dagger functor satisfying:

$$\epsilon_A^\dagger = \eta_{A^*}$$

Just as Theorem 3.49 gave the soundness/completeness for circuit diagrams with respect to symmetric monoidal categories, it is possible to show the same for string diagrams and dagger compact closed categories.

Theorem 4.98 String diagrams are sound and complete for dagger compact closed categories. That is, two morphisms f and g are provably equal using the equations of a dagger compact closed category if and only if they can be expressed as the same string diagram.

We can now extend our table from Section* 3.6.3.

String diagram	'General' diagram	Circuit diagram
 i.e. all to all	 i.e. outputs to inputs	 i.e. admits causal structure
ATS with g-metric	ATS	ATS 'without cycles'
†-compact closed	strict traced SMC	strict SMC

4.7 Historical Notes and References

The manner in which bipartite projectors compose, discussed in Section 4.3.7, is taken from Coecke (2003) and was the formal basis for the first formal graphical presentation of quantum teleportation. So while it is therefore fair to say that this paper kicked off the diagrammatic approach for depicting quantum processes, it was refused by a *Physical Review Letters* editor (without even consulting referees) on the basis of being too 'speculative'. The idea that these bipartite projectors should be interpreted in terms of cups, caps, and boxes as depicted in (4.54) first appeared in Abramsky and Coecke (2005, 2014a). Independently, a topology-based analysis of quantum teleportation was given in Kauffman (2005).

But in fact, cups and caps appeared much earlier, in Penrose's diagrammatic calculus for abstract tensor systems (Penrose, 1971), where they represented the metric tensors of spacetime geometry. However, the further development of string diagrams and obtaining an understanding of their meaning in terms of mathematical models mainly took place within the category theory community. Around the same time as Penrose's paper, string diagrams also appeared in Kelly's paper, which introduced compact closed categories (Kelly, 1972), which as we explained in Section* 4.6.3 are the category-theoretic version of process theories that admit string diagrams. The use of additional boxes besides cups and caps in the categorical context is due to Yetter (see e.g. Freyd and Yetter, 1989), who used the name 'coloured tangles'.

Process–state duality is known, in the context of quantum theory, as the Choi–Jamiołkowski isomorphism (Jamiołkowski, 1972; Choi, 1975). In fact, the isomorphisms presented respectively by Jamiołkowski and Choi were not equivalent. Choi's is the one that we will use throughout most of this book and is basis dependent. Jamiołkowski's is the basis-independent counterpart, resulting from the cups and caps of Example* 4.94.

Dagger compact closed categories were defined in Abramsky and Coecke (2004, 2005) in order to axiomatise the manner in which bipartite projectors compose, resulting in a presentation of basic quantum theory in category-theoretic terms. Around the same time, similar ideas were proposed by Baez (2006). Asymmetric boxes for representing transposes and adjoints are taken from Selinger (2007). As mathematical entities, dagger compact closed categories had already appeared in Baez and Dolan (1995) as a special case of a more general construct in n-categories. Theorem 4.98 is widely regarded as folklore, with a formal statement in Selinger (2007), citing an ‘implicit proof’ first occurring in (Kelly and Laplaza, 1980).

The notion of $\otimes$-positivity was introduced in Selinger (2007) as (abstract) complete positivity. The version of the no-cloning theorem presented in Theorem 4.85, which preceded the emergence of quantum information, was independently proved by Dieks (1982) and Wootters and Zurek (1982). The version presented in Theorem 4.84 that relies on the existence of cups rather exploiting unitarity is taken from Abramsky (2010).

The word ‘teleportation’ was first coined in 1931 by anomalistics writer Charles Fort (1931) in a book entitled *Lo!* consisting of two parts, one of which is entirely devoted to teleportation. However, the concept really took off with the role of the transporter in the adventures of the starship *Enterprise* in Gene Roddenberry’s (1966) *Star Trek*. Quantum teleportation was first proposed by Bennett et al. (1993), and its first experimental realisation was in Bouwmeester et al. (1997). A diagrammatic presentation of classical teleportation was in Coecke et al. (2008b), and a more detailed account is in Stay and Vicary (2013).

The observation related to time-reversal of Section 4.4.3 is taken from Coecke (2003, 2014a), and the results of this paper were experimentally simulated in Laforest et al. (2006). The inspiration for the fact that cups and caps can be used to simulate time travel also came from this paper and was first stated as such in Svetlichny (2009). This was later picked up again in Lloyd et al. (2011), which received a lot of media attention, with some headlines claiming that time travel had been realised. Much earlier, a different theory of quantum time travel had appeared in Deutsch (1991).

The best overview for a wide variety of monoidal categories and their associated notions of diagram is Selinger (2011b). If you want to read more about string diagrams in the context of physics, logic, and topology, we suggest Baez and Stay (2011), and in the context of physics and linguistics, Heunen et al. (2012a). To learn about a whole variety of (mostly) diagrammatic constructions used in higher-dimensional category theory, see Leinster (2004).

The quote at the beginning of this chapter is taken from Schrödinger (1935).