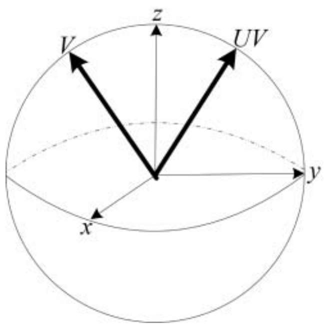


Figure 2.14. The unit sphere and the action of U on V.

If V is a vector on the unit sphere (in any dimension), then UV is also on the unit sphere. We shall see that a unitary matrix is a way of rotating the unit sphere.[7]

Exercise 2.6.8 Show that if U is a unitary matrix and V_1 and V_2 are in $\mathbb{C}^n$, then

$$d(UV_1, UV_2) = d(V_1, V_2), \tag{2.152}$$

i.e., U preserves distances. (An operator that preserves distances is called an **isometry**.) ■

What does unitary really mean? As we saw, it means that it preserves the geometry. But it also means something else: If U is unitary and $UV = V'$, then we can easily form $U^\dagger$ and multiply both sides of the equation by $U^\dagger$ to get $U^\dagger UV = U^\dagger V'$ or $V = U^\dagger V'$. In other words, because U is unitary, there is a related matrix that can "undo" the action that U performs. $U^\dagger$ takes the result of U's action and gets back the original vector. In the quantum world, all actions (that are not measurements) are "undoable" or "reversible" in such a manner.

Hermitian matrices and unitary matrices will be very important in our text. The Venn diagram shown in Figure 2.15 is helpful.

Exercise 2.6.9 Show that I_n and $-1 \cdot I_n$ are both hermitian and unitary. ■

Programming Drill 2.6.2 *Write a function that accepts a square matrix and tells if it is unitary.*

2.7 TENSOR PRODUCT OF VECTOR SPACES

At the conclusion of Section 2.2 we were introduced to the Cartesian product, which is one method of combining vector spaces. In this section, we study the tensor product, which is another, more important, method of combining vector spaces. If $\mathbb{V}$ describes one quantum system and $\mathbb{V}'$ describes another, then their tensor product describes both quantum systems as one. The tensor product is the fundamental building operation of quantum systems.

[7] These movements of the unit sphere are important in computer graphics.

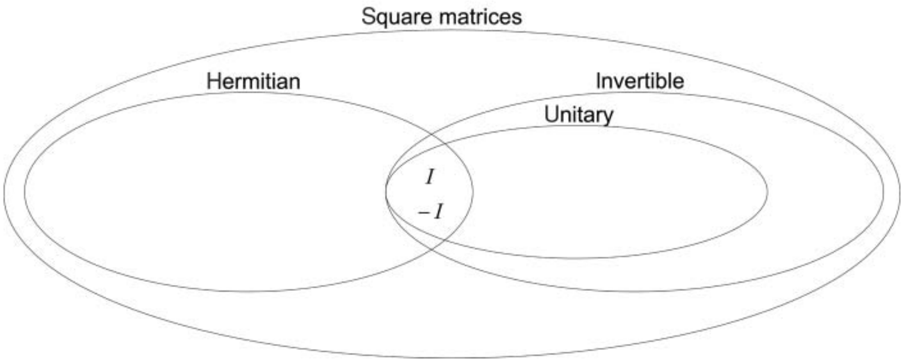


Figure 2.15. Types of matrices.

Reader Tip. A brief warning is in order. The tensor product of two vector spaces is perhaps one of the most difficult subjects in this chapter, as well as one of the most essential. Do not be intimidated if you do not understand it the first time you read it. Everyone has a hard time with tensor products. We also suggest that you read this section in conjunction with Sections 3.4 and 4.5. All these three sections deal with the tensor product from slightly different viewpoints. ♡

Given two vector spaces $\mathbb{V}$ and $\mathbb{V}'$, we shall form the **tensor product** of two vector spaces, and denote it $\mathbb{V} \otimes \mathbb{V}'$. The tensor product is generated by the set of "tensors" of all vectors:

$$\{V \otimes V' | V \in \mathbb{V} \text{ and } V' \in \mathbb{V}'\}, \tag{2.153}$$

where $\otimes$ is just a symbol. A typical element of $\mathbb{V} \otimes \mathbb{V}'$ looks like this:

$$c_0(V_0 \otimes V_0') + c_1(V_1 \otimes V_1') + \cdots + c_{p-1}(V_{p-1} \otimes V_{p-1}'), \tag{2.154}$$

where $V_0, V_1, \ldots, V_{p-1}$ are elements of $\mathbb{V}$ and $V_0', V_1', \ldots, V_{p-1}'$ are elements of $\mathbb{V}'$. We might write this as

$$\sum_{i=0}^{p-1} c_i(V_i \otimes V_i'). \tag{2.155}$$

The operations on this vector space are straightforward. For a given $\sum_{i=0}^{p-1} c_i(V_i \otimes V_i')$ and $\sum_{i=0}^{q-1} c_i'(W_i \otimes W_i')$, addition is simply the addition of summations, i.e.,

$$\sum_{i=0}^{p-1} c_i(V_i \otimes V_i') + \sum_{i=0}^{q-1} c_i'(W_i \otimes W_i'). \tag{2.156}$$

The scalar multiplication for a given $c \in \mathbb{C}$ is

$$c \cdot \sum_{i=0}^{p-1} c_i(V_i \otimes V_i') = \sum_{i=0}^{p-1} (c \times c_i)(V_i \otimes V_i'). \tag{2.157}$$

We impose the following important rewriting rules for this vector space:

(i) The tensor must respect addition in both $\mathbb{V}$ and $\mathbb{V}'$:

$$(V_i + V_j) \otimes V'_k = V_i \otimes V'_k + V_j \otimes V'_k, \tag{2.158}$$

$$V_i \otimes (V'_j + V'_k) = V_i \otimes V'_j + V_i \otimes V'_k. \tag{2.159}$$

(ii) The tensor must respect the scalar multiplication in both $\mathbb{V}$ and $\mathbb{V}'$:

$$c \cdot (V_j \otimes V'_k) = (c \cdot V_j) \otimes V'_k = V_j \otimes (c \cdot V'_k). \tag{2.160}$$

By following these rewriting rules and setting elements equal to each other, we form $\mathbb{V} \otimes \mathbb{V}'$.

Let us find a basis for $\mathbb{V} \otimes \mathbb{V}'$. Say, $\mathbb{V}$ has a basis $\mathcal{B} = \{B_0, B_1, \ldots, B_{m-1}\}$ and $\mathbb{V}'$ has a basis $\mathcal{B}' = \{B'_0, B'_1, \ldots, B'_{n-1}\}$. Given that every $V_i \in \mathbb{V}$ and $V'_i \in \mathbb{V}'$ can be written in a unique way for these bases, we can use the rewrite rules to "decompose" every element $\sum_{i=0}^{p-1} c_i(V_i \otimes V'_i)$ in the tensor product. This will give us a basis for $\mathbb{V} \otimes \mathbb{V}'$. In detail, the basis for $\mathbb{V} \otimes \mathbb{V}'$ will be the set of vectors

$$\{B_j \otimes B'_k \,|\, j = 0, 1, \ldots, m-1 \text{ and } k = 0, 1, \ldots, n-1\}. \tag{2.161}$$

Every $\sum_{i=0}^{p-1} c_i(V_i \otimes V'_i) \in \mathbb{V} \otimes \mathbb{V}'$ can be written as

$$c_{0,0}(B_0 \otimes B'_0) + c_{1,0}(B_1 \otimes B'_0) + \cdots + c_{m-1,n-1}(B_{m-1} \otimes B'_{n-1}). \tag{2.162}$$

The dimension of $\mathbb{V} \otimes \mathbb{V}'$ is the dimension of $\mathbb{V}$ times the dimension of $\mathbb{V}'$. (Remember that the dimension of $\mathbb{V} \times \mathbb{V}'$ is the dimension of $\mathbb{V}$ plus the dimension of $\mathbb{V}'$. So the tensor product of two vector spaces is usually a larger space than their Cartesian product.[8]) One should think of $\mathbb{V} \times \mathbb{V}'$ as the vector space whose states are the states of a system $\mathbb{V}$ *or* a system $\mathbb{V}'$ or both. $\mathbb{V} \otimes \mathbb{V}'$ is to be thought of as the vector space whose basic states are pairs of states, one from system $\mathbb{V}$ *and* one from the system $\mathbb{V}'$.

Given an element of $\mathbb{V}$

$$c_0 B_0 + c_1 B_1 + \cdots + c_{m-1} B_{m-1}, \tag{2.163}$$

and an element of $\mathbb{V}'$

$$c'_0 B'_0 + c'_1 B'_1 + \cdots + c'_{n-1} B'_{n-1}, \tag{2.164}$$

we can associate[9] the following element of $\mathbb{V} \otimes \mathbb{V}'$:

$$(c_0 \times c'_0)(B_0 \otimes B'_0) + (c_0 \times c'_1)(B_0 \otimes B'_1) + \cdots + (c_{m-1} \times c'_{n-1})(B_{m-1} \otimes B'_{n-1}). \tag{2.165}$$

Let us step down from the abstract highland and see what $\mathbb{C}^m \otimes \mathbb{C}^n$ actually looks like. $\mathbb{C}^m \otimes \mathbb{C}^n$ is of dimension mn and hence is isomorphic to $\mathbb{C}^{m \times n}$. What is important is how $\mathbb{C}^m \otimes \mathbb{C}^n$ is isomorphic to $\mathbb{C}^{m \times n}$. If E_j is an element of the canonical basis of each vector space, then we might identify $E_j \otimes E_k$ with $E_{j \times k}$. It is not hard to see

[8] But not always! Remember that $1 \times 1 < 1 + 1$ and $1 \times 2 < 1 + 2$, etc.

[9] It is important to notice that this "association" is not a linear map; it is something called a **bilinear map**.

from the association given in Equation (2.165) that the **tensor product of vectors** is defined as follows:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \otimes \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} a_0 \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} \\ a_1 \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} \\ a_2 \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} \\ a_3 \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} a_0b_0 \\ a_0b_1 \\ a_0b_2 \\ a_1b_0 \\ a_1b_1 \\ a_1b_2 \\ a_2b_0 \\ a_2b_1 \\ a_2b_2 \\ a_3b_0 \\ a_3b_1 \\ a_3b_2 \end{bmatrix}. \tag{2.166}$$

In general, $\mathbb{C}^m \times \mathbb{C}^n$ is much smaller than $\mathbb{C}^m \otimes \mathbb{C}^n$.

Example 2.7.1 For example, consider $\mathbb{C}^2 \times \mathbb{C}^3$ and $\mathbb{C}^2 \otimes \mathbb{C}^3 = \mathbb{C}^6$. Consider the vector

$$\begin{bmatrix} 8 \\ 12 \\ 6 \\ 12 \\ 18 \\ 9 \end{bmatrix} \in \mathbb{C}^6 = \mathbb{C}^2 \otimes \mathbb{C}^3. \tag{2.167}$$

It is not hard to see that this is simply

$$\begin{bmatrix} 2 \\ 3 \end{bmatrix} \otimes \begin{bmatrix} 4 \\ 6 \\ 3 \end{bmatrix}. \tag{2.168}$$

□

Example 2.7.2 In contrast to the above example,

$$\begin{bmatrix} 8 \\ 0 \\ 0 \\ 0 \\ 0 \\ 18 \end{bmatrix} \in \mathbb{C}^6 = \mathbb{C}^2 \otimes \mathbb{C}^3 \tag{2.169}$$

cannot be written as the tensor product of a vector from $\mathbb{C}^2$ and $\mathbb{C}^3$. In order to see this, consider the variables

$$\begin{bmatrix} x \\ y \end{bmatrix} \otimes \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} xa \\ xb \\ xc \\ ya \\ yb \\ yc \end{bmatrix}. \tag{2.170}$$

There are no solutions for the variable that will give you the required results. However, we can write the vector in Equation (2.169) as

$$\begin{bmatrix} 8 \\ 0 \\ 0 \\ 0 \\ 0 \\ 18 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 8 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 6 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix}. \tag{2.171}$$

This is a summation of two vectors. □

For reasons that are made clear in Sections 3.4 and 4.5, we shall call a vector that can be written as the tensor of two vectors **separable**. In contrast, a vector that cannot be written as the tensor of two vectors (but can be written as the nontrivial sum of such tensors) shall be called **entangled**.

Exercise 2.7.1 Calculate the tensor product $\begin{bmatrix} 3 \\ 4 \\ 7 \end{bmatrix} \otimes \begin{bmatrix} -1 \\ 2 \end{bmatrix}$. ■

Exercise 2.7.2 State whether $[5, 6, 3, 2, 0, 1]^T$ is a tensor product of smaller vectors from $\mathbb{C}^3$ and $\mathbb{C}^2$. ■

We will need to know not only how to take the tensor product of two vectors, but also how to determine the **tensor product of two matrices.**[10] Consider two matrices

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & a_{1,1} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} \\ b_{1,0} & b_{1,1} & b_{1,2} \\ b_{2,0} & b_{2,1} & b_{2,2} \end{bmatrix}. \tag{2.172}$$

From the association given in Equation (2.165), it can be seen that the tensor product $A \otimes B$ is the matrix that has every element of A, scalar multiplied with the entire matrix B. That is,

$$A \otimes B = \begin{bmatrix} a_{0,0} \cdot \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} \\ b_{1,0} & b_{1,1} & b_{1,2} \\ b_{2,0} & b_{2,1} & b_{2,2} \end{bmatrix} & a_{0,1} \cdot \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} \\ b_{1,0} & b_{1,1} & b_{1,2} \\ b_{2,0} & b_{2,1} & b_{2,2} \end{bmatrix} \\ a_{1,0} \cdot \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} \\ b_{1,0} & b_{1,1} & b_{1,2} \\ b_{2,0} & b_{2,1} & b_{2,2} \end{bmatrix} & a_{1,1} \cdot \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} \\ b_{1,0} & b_{1,1} & b_{1,2} \\ b_{2,0} & b_{2,1} & b_{2,2} \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} a_{0,0} \times b_{0,0} & a_{0,0} \times b_{0,1} & a_{0,0} \times b_{0,2} & a_{0,1} \times b_{0,0} & a_{0,1} \times b_{0,1} & a_{0,1} \times b_{0,2} \\ a_{0,0} \times b_{1,0} & a_{0,0} \times b_{1,1} & a_{0,0} \times b_{1,2} & a_{0,1} \times b_{1,0} & a_{0,1} \times b_{1,1} & a_{0,1} \times b_{1,2} \\ a_{0,0} \times b_{2,0} & a_{0,0} \times b_{2,1} & a_{0,0} \times b_{2,2} & a_{0,1} \times b_{2,0} & a_{0,1} \times b_{2,1} & a_{0,1} \times b_{2,2} \\ a_{1,0} \times b_{0,0} & a_{1,0} \times b_{0,1} & a_{1,0} \times b_{0,2} & a_{1,1} \times b_{0,0} & a_{1,1} \times b_{0,1} & a_{1,1} \times b_{0,2} \\ a_{1,0} \times b_{1,0} & a_{1,0} \times b_{1,1} & a_{1,0} \times b_{1,2} & a_{1,1} \times b_{1,0} & a_{1,1} \times b_{1,1} & a_{1,1} \times b_{1,2} \\ a_{1,0} \times b_{2,0} & a_{1,0} \times b_{2,1} & a_{1,0} \times b_{2,2} & a_{1,1} \times b_{2,0} & a_{1,1} \times b_{2,1} & a_{1,1} \times b_{2,2} \end{bmatrix}. \tag{2.173}$$

[10] It should be clear that the tensor product of two vectors is simply a special case of the tensor product of two matrices.

Formally, the tensor product of matrices is a function

$$\otimes : \mathbb{C}^{m\times m'} \times \mathbb{C}^{n\times n'} \longrightarrow \mathbb{C}^{mn\times m'n'} \tag{2.174}$$

and it is defined as

$$(A \otimes B)[j,k] = A[j/n, k/m] \times B[j \text{ Mod } n, k \text{ Mod } m]. \tag{2.175}$$

Exercise 2.7.3 Calculate

$$\begin{bmatrix} 3+2i & 5-i & 2i \\ 0 & 12 & 6-3i \\ 2 & 4+4i & 9+3i \end{bmatrix} \otimes \begin{bmatrix} 1 & 3+4i & 5-7i \\ 10+2i & 6 & 2+5i \\ 0 & 1 & 2+9i \end{bmatrix}. \tag{2.176}$$

■

Exercise 2.7.4 Prove that the tensor product is "almost" commutative. Take two 2-by-2 matrices A and B. Calculate $A \otimes B$ and $B \otimes A$. In general, although they are not equal, they do have the same entries, and one can be transformed to the other with a "nice" change of rows and columns. ■

Exercise 2.7.5 Let $A = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 3 & 2 \\ -1 & 0 \end{bmatrix}$, and $C = \begin{bmatrix} 6 & 5 \\ 3 & 2 \end{bmatrix}$. Calculate $A \otimes (B \otimes C)$ and $(A \otimes B) \otimes C$ and show that they are equal. ■

Exercise 2.7.6 Prove that the tensor product is associative, i.e., for arbitrary matrices A, B, and C,

$$A \otimes (B \otimes C) = (A \otimes B) \otimes C. \tag{2.177}$$

■

Exercise 2.7.7 Let $A = \begin{bmatrix} 2 & 3 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$. Calculate $(A \otimes B)^\dagger$ and $A^\dagger \otimes B^\dagger$ and show that they are equal. ■

Exercise 2.7.8 Prove that $(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger$. ■

Exercise 2.7.9 Let A, A', B, and B' be matrices of the appropriate sizes. Prove that

$$(A \star A') \otimes (B \star B') = (A \otimes B) \star (A' \otimes B'). \tag{2.178}$$

■

If A acts on V and B acts on V', then we define the action on their tensor product as

$$(A \otimes B) \star (V \otimes V') = A \star V \otimes B \star V'. \tag{2.179}$$

Such "parallel" actions will arise over and over again.

Programming Drill 2.7.1 *Write a function that accepts two matrices and constructs their tensor product.*

. . . .

References: There are plenty of good references for basic linear algebra. Many of the more elementary ones, like Gilbert and Gilbert (2004), Lang (1986), and Penney (1998), contain many examples and intuitive drawings. Complex vector spaces are discussed in, e.g., Nicholson (1994) and O'Nan (1976). The tensor product is found only in more advanced texts, such as Lang (1993).

A history of the development of the subject can be found in Crowe (1994).

5

Architecture

From the intrinsic evidence of his creation, the Great Architect of the Universe now begins to appear as a pure mathematician.

Sir James Jeans, *Mysterious Universe*

Now that we have the mathematical and physical preliminaries under our belt, we can move on to the nuts and bolts of quantum computing. At the heart of a classical computer is the notion of a bit and at the heart of quantum computer is a generalization of the concept of a bit called a qubit, which shall be discussed in Section 5.1. In Section 5.2, classical (logical) gates, which manipulate bits, are presented from a new and different perspective. From this angle, it is easy to formulate the notion of quantum gates, which manipulate qubits. As mentioned in Chapters 3 and 4, the evolution of a quantum system is reversible, i.e., manipulations that can be done must also be able to be undone. This "undoing" translates into reversible gates, which are discussed in Section 5.3. We move on to quantum gates in Section 5.4.

....

Reader Tip. Discussion of the actual physical implementation of qubits and quantum gates is dealt with in Chapter 11. ♡

....

5.1 BITS AND QUBITS

What is a **bit**?

Definition 5.1.1 *A* **bit** *is a unit of information describing a two-dimensional classical system.*

There are many examples of bits:

- A bit is electricity traveling through a circuit or not (or high and low).
- A bit is a way of denoting "true" or "false."
- A bit is a switch turned on or off.

All these examples are saying the same thing: a bit is a way of describing a system whose set of states is of size 2. We usually write these two possible states as 0 and 1, or F and T, etc.

As we have become adept at matrices, let us use them as a way of representing a bit. We shall represent 0 – or, better, the state $|0\rangle$ – as a 2-by-1 matrix with a 1 in the 0's row and a 0 in the 1's row:

$$|0\rangle = \begin{array}{c} \mathbf{0} \\ \mathbf{1} \end{array} \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{5.1}$$

We shall represent a 1, or state $|1\rangle$, as

$$|1\rangle = \begin{array}{c} \mathbf{0} \\ \mathbf{1} \end{array} \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{5.2}$$

Because these are two different representations (indeed orthogonal), we have an honest-to-goodness bit. We explore how to manipulate these bits in Section 5.2.

A bit can be either in state $|0\rangle$ or in state $|1\rangle$, which was sufficient for the classical world. Either electricity is running through a circuit or it is not. Either a proposition is true or it is false. Either a switch is on or it is off. But either/or is not sufficient in the quantum world. In that world, there are situations where we are in one state *and* in the other simultaneously. In the realm of the quantum, there are systems where a switch is both on *and* off at the same time. One quantum system can be in state $|0\rangle$ *and* in state $|1\rangle$ simultaneously. Hence we are led to the definition of a qubit:

Definition 5.1.2 *A* **quantum bit** *or a* **qubit** *is a unit of information describing a two-dimensional quantum system.*

We shall represent a qubit as a 2-by-1 matrix with complex numbers

$$\begin{array}{c} \mathbf{0} \\ \mathbf{1} \end{array} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}, \tag{5.3}$$

where $|c_0|^2 + |c_1|^2 = 1$. Notice that a classical bit is a special type of qubit. $|c_0|^2$ is to be interpreted as the probability that after measuring the qubit, it will be found in state $|0\rangle$. $|c_1|^2$ is to be interpreted as the probability that after measuring the qubit it will be found in state $|1\rangle$. Whenever we measure a qubit, it automatically becomes a bit. So we shall never "see" a general qubit. Nevertheless, they do exist and are the main characters in our tale. We might visualize this "collapsing" of a qubit to a bit as

$$\begin{tikzcd} & {[1,0]^T} \\ {[c_0,c_1]^T} \arrow[ur, rightsquigarrow, "|c_0|^2"] \arrow[dr, rightsquigarrow, "|c_1|^2"'] & \\ & {[0,1]^T} \end{tikzcd} \tag{5.4}$$

It is easy to see that the bits $|0\rangle$ and $|1\rangle$ are the canonical basis of $\mathbb{C}^2$. Thus, any qubit can be written as

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = c_0 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} + c_1 \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = c_0|0\rangle + c_1|1\rangle. \tag{5.5}$$

Exercise 5.1.1 Write $V = \begin{bmatrix} 3+2i \\ 4-2i \end{bmatrix}$ as a sum of $|0\rangle$ and $|1\rangle$. ■

Following the normalization procedures that we learned in Chapter 4 on page 109, any nonzero element of $\mathbb{C}^2$ can be converted into a qubit.

Example 5.1.1 The vector

$$V = \begin{bmatrix} 5+3i \\ 6i \end{bmatrix} \tag{5.6}$$

has norm

$$|V| = \sqrt{\langle V, V\rangle} = \sqrt{[5-3i, -6i] \begin{bmatrix} 5+3i \\ 6i \end{bmatrix}} = \sqrt{34+36} = \sqrt{70}. \tag{5.7}$$

So V describes the same physical state as the qubit

$$\frac{V}{\sqrt{70}} = \begin{bmatrix} \frac{5+3i}{\sqrt{70}} \\ \frac{6i}{\sqrt{70}} \end{bmatrix} = \frac{5+3i}{\sqrt{70}}|0\rangle + \frac{6i}{\sqrt{70}}|1\rangle. \tag{5.8}$$

After measuring the qubit $\frac{V}{\sqrt{70}}$, the probability of it being found in state $|0\rangle$ is $\frac{34}{70}$ and the probability of it being found in state $|1\rangle$ is $\frac{36}{70}$. □

Exercise 5.1.2 Normalize $V = \begin{bmatrix} 15 - 3.4i \\ 2.1 - 16i \end{bmatrix}$. ■

Let us look at several ways of denoting different qubits. $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ can be written as

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}. \tag{5.9}$$

Similarly, $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$ can be written as

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{5.10}$$

It is important to realize that

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{|1\rangle + |0\rangle}{\sqrt{2}}. \tag{5.11}$$

These are both ways of denoting $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$. In contrast,

$$\frac{|0\rangle - |1\rangle}{\sqrt{2}} \neq \frac{|1\rangle - |0\rangle}{\sqrt{2}}. \tag{5.12}$$

The left state is the vector $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$ and the right state is the vector $\begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$. However, the two states are related:

$$\frac{|0\rangle - |1\rangle}{\sqrt{2}} = (-1)\frac{|1\rangle - |0\rangle}{\sqrt{2}}. \tag{5.13}$$

How are qubits to be implemented? In Chapter 11, several different methods are explored. We simply state some examples of implementations for the time being:

- An electron might be in one of two different orbits around the nucleus of an atom (ground state and excited state).
- A photon might be in one of two polarized states.
- A subatomic particle might have one of two spin directions.

There will be enough quantum indeterminacy and quantum superposition effects within all these systems to represent a qubit.

Computers with only one bit of storage are not very interesting. Similarly, we will need quantum devices with more than one qubit. Consider a byte, or eight bits. A typical byte might be

$$01101011. \tag{5.14}$$

If we were to follow the preceding method of describing bits, we would represent the bits as follows:

$$\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}. \tag{5.15}$$

We learned previously that in order to combine quantum systems, one should use the tensor product; hence, we can describe the byte in Equation (5.14) as

$$|0\rangle \otimes |1\rangle \otimes |1\rangle \otimes |0\rangle \otimes |1\rangle \otimes |0\rangle \otimes |1\rangle \otimes |1\rangle. \tag{5.16}$$

As a qubit, this is an element of

$$\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2. \tag{5.17}$$

This vector space may be denoted as $(\mathbb{C}^2)^{\otimes 8}$. This is a complex vector space of dimension $2^8 = 256$. Because there is essentially only one complex vector space of this dimension, this vector space is isomorphic to $\mathbb{C}^{256}$.

We can describe our byte in yet another way: as a $2^8 = 256$ row vector

$$\begin{array}{c} \mathbf{00000000} \\ \mathbf{00000001} \\ \vdots \\ \mathbf{01101010} \\ \mathbf{01101011} \\ \mathbf{01101100} \\ \vdots \\ \mathbf{11111110} \\ \mathbf{11111111} \end{array} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}. \tag{5.18}$$

Exercise 5.1.3 Express the three bits 101 or $|1\rangle \otimes |0\rangle \otimes |1\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2$ as a vector in $(\mathbb{C}^2)^{\otimes 3} = \mathbb{C}^8$. Do the same for 011 and 111. ■

This is fine for the classical world. However, for the quantum world, in order to permit superposition, a generalization is needed: every state of an eight-qubit system can be written as

$$\begin{array}{c}\mathbf{00000000}\\\mathbf{00000001}\\\vdots\\\mathbf{01101010}\\\mathbf{01101011}\\\mathbf{01101100}\\\vdots\\\mathbf{11111110}\\\mathbf{11111111}\end{array}\begin{bmatrix}c_0\\c_1\\\vdots\\c_{106}\\c_{107}\\c_{108}\\\vdots\\c_{254}\\c_{255}\end{bmatrix}, \tag{5.19}$$

where $\sum_{i=0}^{255}|c_i|^2 = 1$. Eight qubits together is called a **qubyte**.

In the classical world, it is necessary to indicate the state of each bit of a byte. This amounts to writing eight bits. In the quantum world, a state of eight qubits is given by writing 256 complex numbers. As we stated in Section 3.4, this exponential growth was one of the reasons researchers started giving thought to the notion of quantum computing. If one wanted to emulate a quantum computer with a 64-qubit register, one would need to store $2^{64} = 18,446,744,073,709,551,616$ complex numbers. This is way beyond our current storage capability.

Let us practice writing two qubits in ket notation. A qubit pair can be written as

$$|0\rangle \otimes |1\rangle \quad \text{or} \quad |0 \otimes 1\rangle, \tag{5.20}$$

which means that the first qubit is in state $|0\rangle$ and the second qubit is in state $|1\rangle$. Because the tensor product is understood, we might also denote these qubits as $|0\rangle|1\rangle$, $|0, 1\rangle$, or $|01\rangle$. Yet another way to look at these two qubits as the 4-by-1 matrix is

$$\begin{array}{c}\mathbf{00}\\\mathbf{01}\\\mathbf{10}\\\mathbf{11}\end{array}\begin{bmatrix}0\\1\\0\\0\end{bmatrix}. \tag{5.21}$$

Exercise 5.1.4 What vector corresponds to the state $3|01\rangle + 2|11\rangle$? ■

Example 5.1.2 The qubit corresponding to

$$\frac{1}{\sqrt{3}}\begin{bmatrix}1\\0\\-1\\1\end{bmatrix} \tag{5.22}$$

can be written as

$$\frac{1}{\sqrt{3}}|00\rangle - \frac{1}{\sqrt{3}}|10\rangle + \frac{1}{\sqrt{3}}|11\rangle = \frac{|00\rangle - |10\rangle + |11\rangle}{\sqrt{3}}. \tag{5.23}$$

□

A general state of a two-qubit system can be written as

$$|\psi\rangle = c_{0,0}|00\rangle + c_{0,1}|01\rangle + c_{1,0}|10\rangle + c_{1,1}|11\rangle. \tag{5.24}$$

The tensor product of two states is not commutative:

$$|0 \otimes 1\rangle = |0\rangle \otimes |1\rangle = |0,1\rangle = |01\rangle \neq |10\rangle = |1,0\rangle = |1\rangle \otimes |0\rangle = |1 \otimes 0\rangle. \tag{5.25}$$

The left ket describes the state in which the first qubit is in state 0 and the second qubit is in state 1. The right ket indicates that first qubit is in state 1 and the second qubit is in state 0.

Let us briefly revisit the notion of entanglement again. If the system is in the state

$$\frac{|11\rangle + |00\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}|11\rangle + \frac{1}{\sqrt{2}}|00\rangle, \tag{5.26}$$

then that means that the the two qubits are entangled. That is, if we measure the first qubit and it is found in state $|1\rangle$ then we automatically know that the state of the second qubit is $|1\rangle$. Similarly, if we measure the first qubit and find it in state $|0\rangle$ then we know the second qubit is also in state $|0\rangle$.

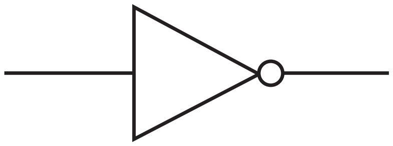
5.2 CLASSICAL GATES

Classical logical gates are ways of manipulating bits. Bits enter and exit logical gates. We will need ways of manipulating qubits and will study classical gates from the point of view of matrices. As stated in Section 5.1, we represent n input bits as a 2^n-by-1 matrix and m output bits as a 2^m-by-1 matrix. How should we represent our logical gates? When one multiplies a 2^m-by-2^n matrix with a 2^n-by-1 matrix, the result is a 2^m-by-1 matrix. In symbols:

$$(2^m\text{-by-}2^n) \star (2^n\text{-by-}1) = (2^m\text{-by-}1). \tag{5.27}$$

So bits will be represented by column vectors and logic gates by matrices.

Let us try a simple example. Consider the NOT gate.



NOT takes as input one bit, or a 2-by-1 matrix, and outputs one bit, or a 2-by-1 matrix. NOT of $|0\rangle$ equals $|1\rangle$ and NOT of $|1\rangle$ equals $|0\rangle$. Consider the matrix

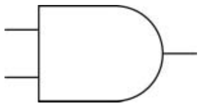
$$NOT = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{5.28}$$

This matrix satisfies

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \tag{5.29}$$

which is exactly what we want.

What about the other gates? Consider the AND gate. The AND gate is different from the NOT gate because AND accepts two bits and outputs one bit.



Because there are two inputs and one output, we will need a 2^1-by-2^2 matrix. Consider the matrix

$$AND = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.30}$$

This matrix satisfies

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{5.31}$$

We can write this as

$$AND|11\rangle = |1\rangle. \tag{5.32}$$

In contrast, consider another 4-by-1 matrix:

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{5.33}$$

We can write this as

$$AND|01\rangle = |0\rangle. \tag{5.34}$$

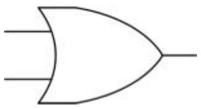
Exercise 5.2.1 Calculate AND$|10\rangle$. ■

What would happen if we put an arbitrary 4-by-1 matrix to the right of AND?

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3.5 \\ 2 \\ 0 \\ -4.1 \end{bmatrix} = \begin{bmatrix} 5.5 \\ -4.1 \end{bmatrix} \tag{5.35}$$

This is clearly nonsense. We are allowed only to multiply these classical gates with vectors that represent classical states, i.e., column matrices with a single 1 entry and all other entries 0. In the classical world, the bits are in only one state at a time and are described by such vectors. Only later, when we delve into quantum gates, will we have more room (and more fun).

The OR gate

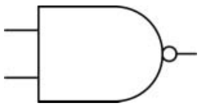


can be represented by the matrix

$$\text{OR} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}. \tag{5.36}$$

Exercise 5.2.2 Show that this matrix performs the OR operation. ■

The NAND gate

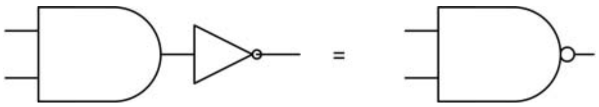


is of special importance because every logical gate can be composed of NAND gates. Let us try to determine which matrix would correspond to NAND. One way is to sit down and consider for which of the four possible input states of two bits (00, 01, 10, 11) does NAND output a 1 (answer: 00, 01, 10), and in which states does NAND output a 0 (answer: 11). From this, we realize that NAND can be written as

$$\text{NAND} = \begin{array}{c|cccc} & \mathbf{00} & \mathbf{01} & \mathbf{10} & \mathbf{11} \\ \hline \mathbf{0} & 0 & 0 & 0 & 1 \\ \mathbf{1} & 1 & 1 & 1 & 0 \end{array}. \tag{5.37}$$

Notice that the column names correspond to the inputs and the row names correspond to the outputs. 1 in the jth column and ith row means that on entry j the matrix/gate will output i.

There is, however, another way in which one can determine the NAND gate. The NAND gate is really the AND gate followed by the NOT gate.

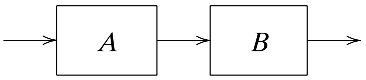


In other words, we can perform the NAND operation by first performing the AND operation and then the NOT operation. In terms of matrices we can write this as

$$NOT \star AND = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \star \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} = \text{NAND}. \tag{5.38}$$

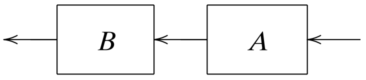
Exercise 5.2.3 Find a matrix that corresponds to NOR. ■

This way of thinking of NAND brings to light a general situation. When we perform a computation, we often have to carry out one operation followed by another.



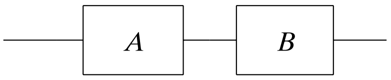
(5.39)

We call this procedure performing **sequential** operations. If matrix A corresponds to performing an operation and matrix B corresponds to performing another operation, then the matrix $B \star A$ corresponds to performing the operation sequentially. Notice that $B \star A$ looks like the reverse of our picture which has, from left to right, A and then B. Do not be alarmed by this. The reason for this is because we read from left to right and hence we depict processes as flowing from left to right. We could have easily drawn the above figure as



(5.40)

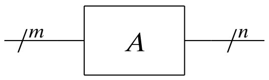
with no confusion.[1] We shall follow the convention that computation flows from left to right and omit the heads of the arrows. And so a computation of A followed by B shall be denoted



(5.41)

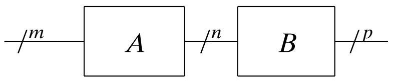
[1] If the text were written in Arabic or Hebrew, this problem would not even arise.

Let us be formal with the number of inputs and the number of outputs. If A is an operation with m input bits and n output bits, then we shall draw this as



(5.42)

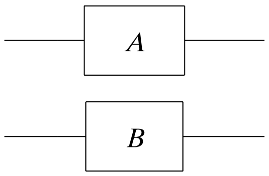
The matrix A will be of size 2^n-by-2^m. Say, B takes the n outputs of A as input and outputs p bits, i.e.,



(5.43)

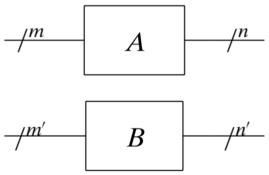
then B is represented by a 2^p-by-2^n matrix B, and performing one operation sequentially followed by another operation corresponds to $B \star A$, which is a $(2^p\text{-by-}2^n) \star (2^n\text{-by-}2^m) = (2^p\text{-by-}2^m)$ matrix.

Besides sequential operations, there are **parallel** operations as well.



(5.44)

Here we have A acting on some bits and B on others. This will be represented by $A \otimes B$ (see Section 2.7). Let us be exact with the number of inputs and the number of outputs.



(5.45)

A will be of size 2^n-by-2^m. B will be of size $2^{n'}$-by-$2^{m'}$. Following Equation (2.174) in Section 2.7, $A \otimes B$ is of size $2^n 2^{n'} = 2^{n+n'}$-by-$2^m 2^{m'} = 2^{m+m'}$.

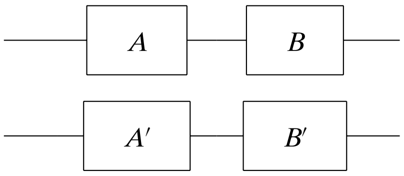
Exercise 5.2.4 In Exercise 2.7.4, we proved that $A \otimes B \cong B \otimes A$. What does this fact correspond to in terms of performing parallel operations on different bits? ■

Combinations of sequential and parallel operations gates/matrices will be called **circuits**. We will, of course, construct some really complicated matrices, but they will all be decomposable into the sequential and parallel compositions of simple gates.

Exercise 5.2.5 In Exercise 2.7.9, we proved that for matrices of the appropriate sizes A, A', B, and B' we have the following equation:

$$(B \otimes B') \star (A \otimes A') = (B \star A) \otimes (B' \star A'). \tag{5.46}$$

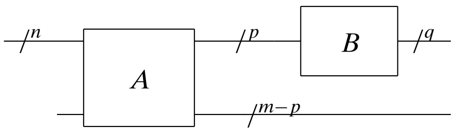
To what does this correspond in terms of performing different operations on different (qu)bits? (Hint: Consider the following figure.)



(5.47)

■

Example 5.2.1 Let A be an operation that takes n inputs and gives m outputs. Let B take $p < m$ of these outputs and leave the other $m - p$ outputs alone. B outputs q bits.



(5.48)

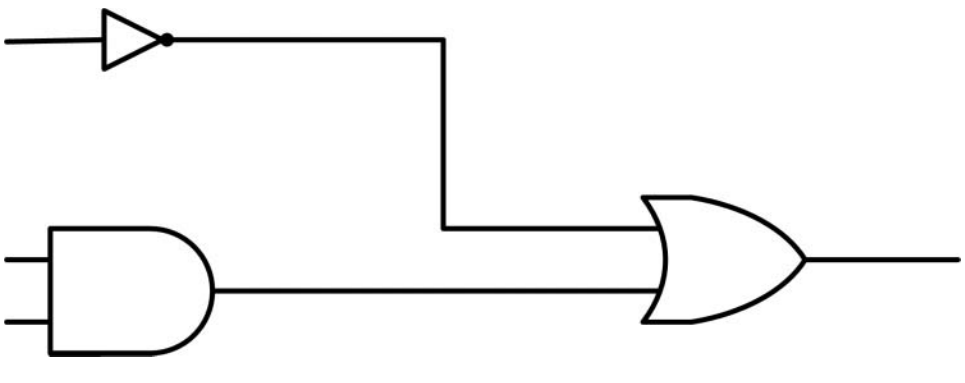
A is a 2^m-by-2^n matrix. B is a 2^q-by-2^p matrix. As nothing should be done to the $m - p$ bits, we might represent this as the 2^{m-p}-by-2^{m-p} identity matrix I_{m-p}. We do not draw any gate for the identity matrix. The entire circuit can be represented by the following matrix:

$$(B \otimes I_{m-p}) \star A. \tag{5.49}$$

□

Example 5.2.2 Consider the circuit.
This is represented by

$$\text{OR} \star (\text{NOT} \otimes \text{AND}). \tag{5.50}$$



Let us see how the operations look like as matrices. Calculating, we get

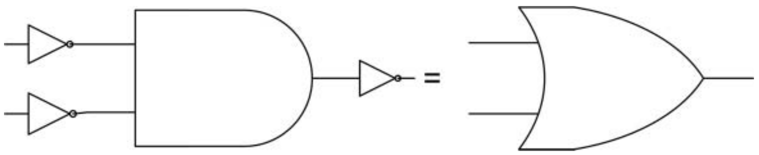
$$\text{NOT} \otimes \text{AND} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{5.51}$$

And so we get

$$\text{OR} \star (\text{NOT} \otimes \text{AND}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.52}$$

□

Let us see if we can formulate DeMorgan's laws in terms of matrices. One of DeMorgan's laws states that $\neg(\neg P \bigwedge \neg Q) = P \bigvee Q$. Here is a pictorial representation.



In terms of matrices this corresponds to

$$\text{NOT} \star \text{AND} \star (\text{NOT} \otimes \text{NOT}) = \text{OR}. \tag{5.53}$$

First, let us calculate the tensor product:

$$\text{NOT} \otimes \text{NOT} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{5.54}$$

This DeMorgan's law corresponds to the following identity of matrices:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \star \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \star \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}. \tag{5.55}$$

Exercise 5.2.6 Multiply out these matrices and confirm the identity. ■

Exercise 5.2.7 Formulate the other DeMorgan's law

$$\neg(\neg P \bigvee \neg Q) = P \bigwedge Q \tag{5.56}$$

in terms of matrices. ■

Exercise 5.2.8 Write the matrix that would correspond to a one-bit adder. A one-bit adder adds the bits x, y, and c (a carry-bit from an earlier adder) and outputs the bits z and c' (a carry-bit for the next adder). There are three inputs and two outputs, so the matrix will be of dimension 2^2-by-2^3. (Hint: Mark the columns as $000, 001, 010, \ldots, 110, 111$, where column, say, 101 corresponds to $x = 1, y = 0, c = 1$. Mark the rows as $00, 01, 10, 11$, where row, say, 10, corresponds to $z = 1, c' = 0$. When $x = 1, y = 0, c = 1$, the output should be $z = 0$ and $c' = 1$. So place a 1 in the row marked 01 and a 0 in all other rows.) ■

Exercise 5.2.9 In Exercise 5.2.8, you determined the matrix that corresponds to a one-bit adder. Check that your results are correct by writing the circuit in terms of classical gates and then converting the circuit to a big matrix. ■

5.3 REVERSIBLE GATES

Not all the logical gates that we dealt with in Section 5.2 will work in quantum computers. In the quantum world, all operations that are not measurements are reversible and are represented by unitary matrices. The AND operation is not reversible. Given an output of $|0\rangle$ from AND, one cannot determine if the input was $|00\rangle$, $|01\rangle$, or $|10\rangle$. So from an output of the AND gate, one cannot determine the input and hence AND is not reversible. In contrast, the NOT gate and the identity gates are reversible. In fact, they are their own inverses:

$$NOT \star NOT = I_2 \qquad I_n \star I_n = I_n. \tag{5.57}$$

Reversible gates have a history that predates quantum computing. Our everyday computers lose energy and generate a tremendous amount of heat. In the 1960s, Rolf Landauer analyzed computational processes and showed that erasing information, as opposed to writing information, is what causes energy loss and heat. This notion has come to be known as the **Landauer's principle**.

In order to gain a real-life intuition as to why erasing information dissipates energy, consider a tub of water with a wall separating the two sides as in Figure 5.1.

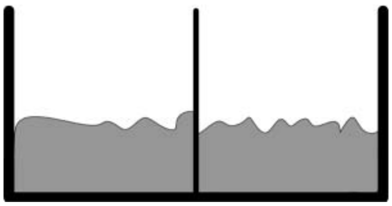


Figure 5.1. Tub with water in no state.

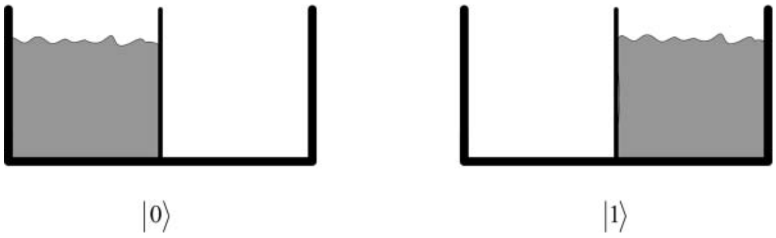


Figure 5.2. Tub with water in state $|0\rangle$ and state $|1\rangle$.

This tub is used as a way of storing a bit of information. If all the water is pushed to the left then the system is in state $|0\rangle$, and if all the water is pushed to the right then the system is in state $|1\rangle$, as in Figure 5.2.

What would correspond to erasing information in such a system? If there were a hole in the wall separating the 0 and 1 regions, then the water could seep out and we would not know what state the system would be in. One can easily place a turbine where the water is seeping out (see Figure 5.3) and generate energy. Hence, losing information means energy is being dissipated.

Notice, also, that writing information is a reversible procedure. If the tub is in no state and we push all the water to the left and set the water to state $|0\rangle$, all one needs to do is remove the wall and the water will go into both regions resulting in no state. This is shown in Figure 5.4. We have reversed the fact that information was written. In contrast, erasing information is not reversible. Start at state $|0\rangle$, and then remove the wall that separates the two parts of the tub. That is erasing the information. How could we return to the original state? There are two possible states to return to, as in Figure 5.5.

The obvious answer is that we should push all the water back to state $|0\rangle$. But the only way we know that $|0\rangle$ is the original state is if that information is copied to the brain. In that case, the system is both the tub and the brain, and we did not really erase the fact that state $|0\rangle$ was the original state. Our brain was still storing the information.

Let us reexamine this intuition by considering two people, Alice and Bob. If Alice writes a letter on an empty blackboard and then Bob walks into the room, he can then erase the letter that Alice wrote on the board and return the blackboard into its original pristine state. Thus, writing is reversible. In contrast, if there is a board with writing on it and Alice erases the board, then when Bob walks into the room he cannot write what was on the board. Bob does not know what was on the board before Alice erased it. So Alice's erasing was not reversible.

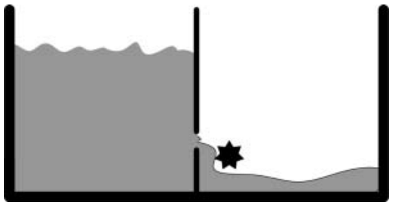


Figure 5.3. State $|0\rangle$ dissipating and creating energy.

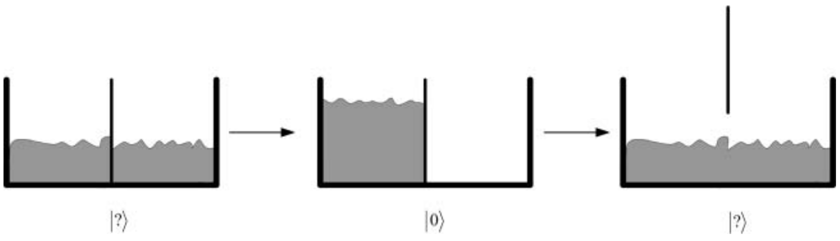
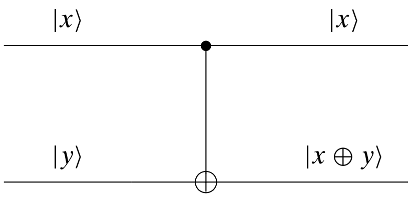


Figure 5.4. Reversibility of writing.

We have found that erasing information is an irreversible, energy-dissipating operation. In the 1970s, Charles H. Bennett continued along these lines of thought. If erasing information is the only operation that uses energy, then a computer that is reversible and does not erase would not use any energy. Bennett started working on reversible circuits and programs.

What examples of reversible gates are there? We have already seen that the identity gate and NOT gates are reversible. What else is there? Consider the following **controlled-NOT gate**:



(5.58)

This gate has two inputs and two outputs. The top input is the control bit. It controls what the output will be. If $|x\rangle = |0\rangle$, then the bottom output of $|y\rangle$ will be the same as the input. If $|x\rangle = |1\rangle$, then the bottom output will be the opposite. If we write the top qubit first and then the bottom qubit, then the controlled-NOT gate takes $|x, y\rangle$ to $|x, x \oplus y\rangle$, where $\oplus$ is the binary exclusive or operation.

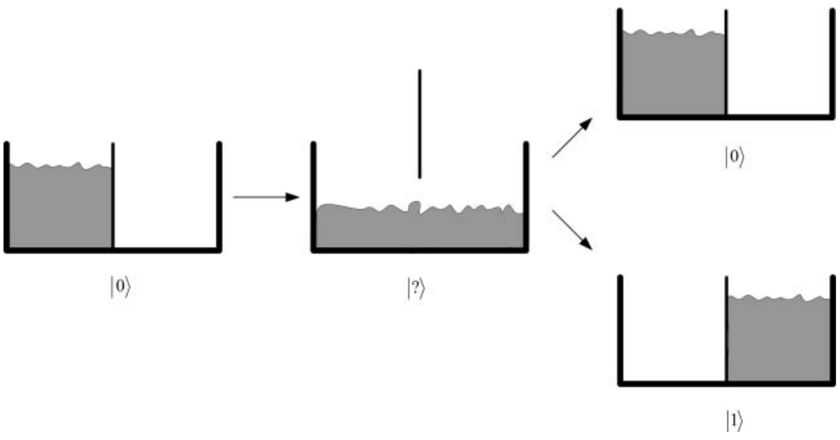
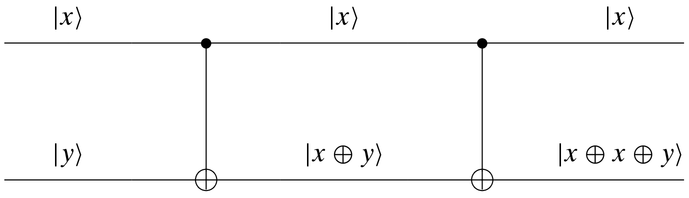


Figure 5.5. Irreversibility of erasing.

The matrix that corresponds to this reversible gate is

$$\begin{array}{c|cccc} & \mathbf{00} & \mathbf{01} & \mathbf{10} & \mathbf{11} \\ \mathbf{00} & 1 & 0 & 0 & 0 \\ \mathbf{01} & 0 & 1 & 0 & 0 \\ \mathbf{10} & 0 & 0 & 0 & 1 \\ \mathbf{11} & 0 & 0 & 1 & 0 \end{array}. \tag{5.59}$$

The controlled-NOT gate can be reversed by itself. Consider the following figure:

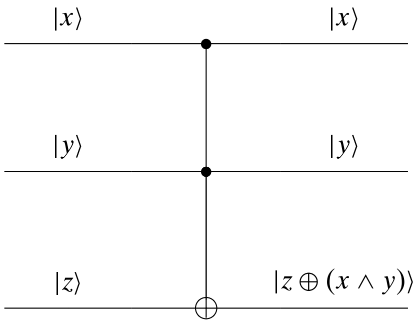


(5.60)

State $|x, y\rangle$ goes to $|x, x \oplus y\rangle$, which further goes to $|x, x \oplus (x \oplus y)\rangle$. This last state is equal to $|x, (x \oplus x) \oplus y\rangle$ because $\oplus$ is associative. Because $x \oplus x$ is always equal to 0, this state reduces to the original $|x, y\rangle$.

Exercise 5.3.1 Show that the controlled-NOT gate is its own inverse by multiplying the corresponding matrix by itself and arriving at the identity matrix. ■

An interesting reversible gate is the **Toffoli gate**:



(5.61)

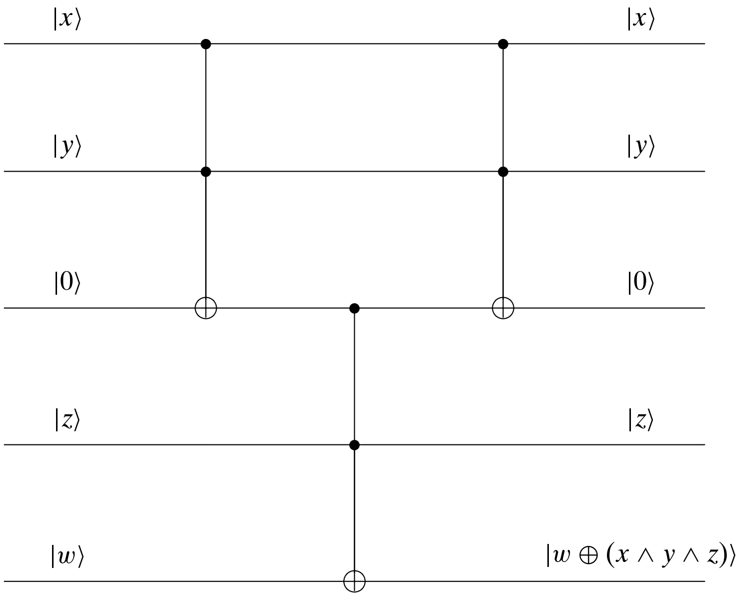
This is similar to the controlled-NOT gate, but with two controlling bits. The bottom bit flips only when *both* of the top two bits are in state $|1\rangle$. We can write this operation as taking state $|x, y, z\rangle$ to $|x, y, z \oplus (x \wedge y)\rangle$.

Exercise 5.3.2 Show that the Toffoli gate is its own inverse. ■

The matrix that corresponds to this gate is

$$\begin{array}{c|cccccccc|} & \mathbf{000} & \mathbf{001} & \mathbf{010} & \mathbf{011} & \mathbf{100} & \mathbf{101} & \mathbf{110} & \mathbf{111} \\ \hline \mathbf{000} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{001} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{010} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{011} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \mathbf{100} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \mathbf{101} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \mathbf{110} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \mathbf{111} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}. \qquad (5.62)$$

Example 5.3.1 The NOT gate has no controlling bit, the controlled-NOT gate has one controlling bit, and the Toffoli gate has two controlling bits. Can we go on with this? Yes. A gate with three controlling bits can be constructed from three Toffoli gates as follows:

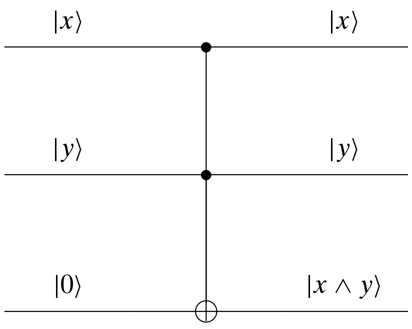


(5.63)

□

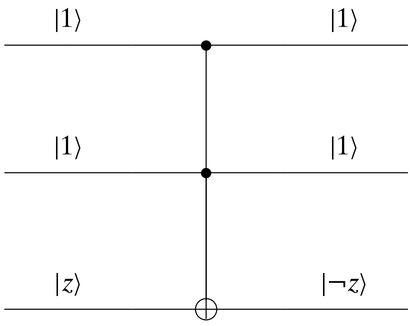
One reason why the Toffoli gate is interesting is that it is universal. In other words, with copies of the Toffoli gate, you can make any logical gate. In particular, you can make a reversible computer using only Toffoli gates. Such a computer would, in theory, neither use any energy nor give off any heat.

In order to see that the Toffoli gate is universal, we will show that it can be used to make both the AND and NOT gates. The AND gate is obtained by setting the bottom $|z\rangle$ input to $|0\rangle$. The bottom output will then be $|x \wedge y\rangle$.



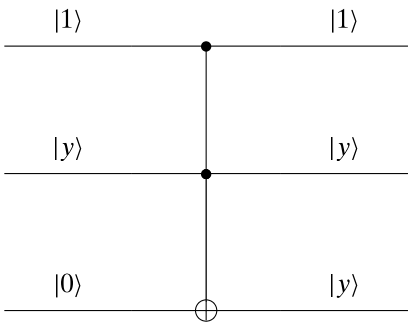
(5.64)

The NOT gate is obtained by setting the top two inputs to $|1\rangle$. The bottom output will be $|(1 \wedge 1) \oplus z\rangle = |1 \oplus z\rangle = |\neg z\rangle$.



(5.65)

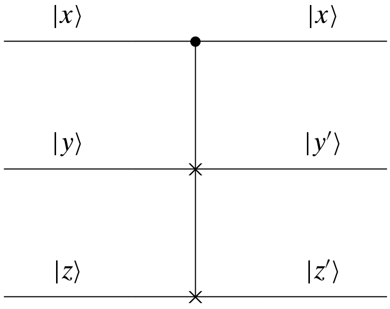
In order to construct all gates, we must also have a way of producing a fanout of values. In other words, a gate is needed that inputs a value and outputs two of the same values. This can be obtained by setting $|x\rangle$ to $|1\rangle$ and $|z\rangle$ to $|0\rangle$. This makes the output $|1, y, y\rangle$.



(5.66)

Exercise 5.3.3 Construct the NAND with one Toffoli gate. Construct the OR gate with two Toffoli gates. ■

Another interesting reversible gate is the **Fredkin gate**. The Fredkin gate also has three inputs and three outputs.



(5.67)

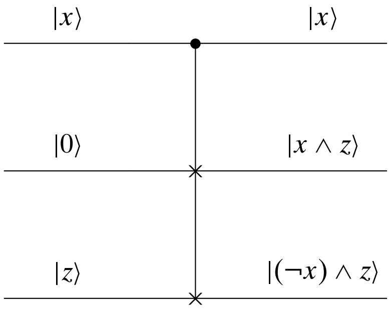
The top $|x\rangle$ input is the control input. The output is always the same $|x\rangle$. If $|x\rangle$ is set to $|0\rangle$, then $|y'\rangle = |y\rangle$ and $|z'\rangle = |z\rangle$, i.e., the values stay the same. If, on the other hand, the control $|x\rangle$ is set to $|1\rangle$, then the outputs are reversed: $|y'\rangle = |z\rangle$ and $|z'\rangle = |y\rangle$. In short, $|0, y, z\rangle \mapsto |0, y, z\rangle$ and $|1, y, z\rangle \mapsto |1, z, y\rangle$.

Exercise 5.3.4 Show that the Fredkin gate is its own inverse. ■

The matrix that corresponds to the Fredkin gate is

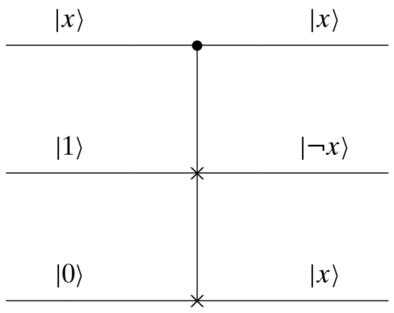
$$\begin{array}{c c}
 & \begin{array}{cccccccc} \mathbf{000} & \mathbf{001} & \mathbf{010} & \mathbf{011} & \mathbf{100} & \mathbf{101} & \mathbf{110} & \mathbf{111} \end{array} \\
\begin{array}{c} \mathbf{000} \\ \mathbf{001} \\ \mathbf{010} \\ \mathbf{011} \\ \mathbf{100} \\ \mathbf{101} \\ \mathbf{110} \\ \mathbf{111} \end{array} &
\left[\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right]
\end{array}. \qquad (5.68)$$

The Fredkin gate is also universal. By setting $|y\rangle$ to $|0\rangle$, we get the AND gate as follows:



(5.69)

The NOT gate and the fanout gate can be obtained by setting $|y\rangle$ to $|1\rangle$ and $|z\rangle$ to $|0\rangle$. This gives us



(5.70)

So both the Toffoli and the Fredkin gates are universal. Not only are both reversible gates; a glance at their matrices indicates that they are also unitary. In the next section, we look at other unitary gates.

5.4 QUANTUM GATES

Definition 5.4.1 *A* **quantum gate** *is simply an operator that acts on qubits. Such operators will be represented by unitary matrices.*

We have already worked with some quantum gates such as the identity operator I, the Hadamard gate H, the NOT gate, the controlled-NOT gate, the Toffoli gate, and the Fredkin gate. What else is there?

Let us look at some other quantum gates. The following three matrices are called **Pauli matrices** and are very important:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{5.71}$$

They occur everywhere in quantum mechanics and quantum computing.[2] Note that the X matrix is nothing more than our NOT matrix. Other important matrices that will be used are

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \quad \text{and} \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}. \tag{5.72}$$

Exercise 5.4.1 Show that each of these matrices are unitary. ■

Exercise 5.4.2 Show the action of each of these matrices on an arbitrary qubit $[c_0, c_1]^T$. ■

[2] Sometimes the notation σ_x, σ_y, and σ_z is used for these matrices.

Exercise 5.4.3 These operations are intimately related to each other. Prove the following relationships between the operations:

(i) $X^2 = Y^2 = Z^2 = I$,
(ii) $H = \frac{1}{\sqrt{2}}(X + Z)$,
(iii) $X = HZH$,
(iv) $Z = HXH$,
(v) $-1Y = HYH$,
(vi) $S = T^2$,
(vii) $-1Y = XYX$. ■

There are still other quantum gates. Let us consider a one-qubit quantum gate with an interesting name. The gate is called the **square root of NOT** and is denoted $\sqrt{\text{NOT}}$. The matrix representation of this gate is

$$\sqrt{\text{NOT}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}. \tag{5.73}$$

The first thing to notice is that this gate is not its own inverse, that is,

$$\sqrt{\text{NOT}} \neq \sqrt{\text{NOT}}^{\dagger}. \tag{5.74}$$

In order to understand why this gate has such a strange name, let us multiply $\sqrt{\text{NOT}}$ by itself:

$$\sqrt{\text{NOT}} \star \sqrt{\text{NOT}} = (\sqrt{\text{NOT}})^2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \tag{5.75}$$

which is very similar to the NOT gate. Let us put the qubits $|0\rangle$ and $|1\rangle$ through $\sqrt{\text{NOT}}$ gate twice. We get

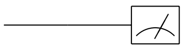
$$|0\rangle = [1, 0]^T \mapsto \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]^T \mapsto [0, 1]^T = |1\rangle \tag{5.76}$$

and

$$|1\rangle = [0, 1]^T \mapsto \left[-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]^T \mapsto [-1, 0]^T = -1|0\rangle. \tag{5.77}$$

Remembering that $|0\rangle$ and $-1|0\rangle$ both represent the same state, we are confident in saying that the square of $\sqrt{\text{NOT}}$ performs the same operation as the NOT gate, and hence the name.

There is one other gate we have not discussed: the measurement operation. This is not unitary or, in general, even reversible. This operation is usually performed at the end of a computation when we want to measure qubits (and find bits). We shall denote it as


(5.78)

There is a beautiful geometric way of representing one-qubit states and operations. Remember from Chapter 1, page 18, that for a given complex number $c = x + yi$ whose modulus is 1, there is a nice way of visualizing c as an arrow of length 1 from the origin to the circle of radius 1.

$$|c|^2 = c \times \overline{c} = (x + yi) \times (x - yi) = x^2 + y^2 = 1. \tag{5.79}$$

In other words, every complex number of radius 1 can be identified by the angle ϕ that the vector makes with the positive x axis.

There is an analogous representation of a qubit as an arrow from the origin to a three-dimensional sphere. Let us see how it works. A generic qubit is of the form

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle, \tag{5.80}$$

where $|c_0|^2 + |c_1|^2 = 1$. Although at first sight there are four real numbers involved in the qubit given in Equation (5.80), it turns out that there are only two actual degrees of freedom to the three-dimensional ball (as latitude and longitude on the Earth). Let us rewrite the qubit in Equation (5.80) in polar form:

$$c_0 = r_0\, e^{i\phi_0} \tag{5.81}$$

and

$$c_1 = r_1\, e^{i\phi_1}, \tag{5.82}$$

and so Equation (5.80) can be rewritten as

$$|\psi\rangle = r_0\, e^{i\phi_0}|0\rangle + r_1\, e^{i\phi_1}|1\rangle. \tag{5.83}$$

There are still four real parameters: r_0, r_1, ϕ_0, ϕ_1. However, a quantum physical state does not change if we multiply its corresponding vector by an arbitrary complex number (of norm 1, see Chapter 4, page 109). We can therefore obtain an equivalent expression for the qubit in Equation (5.80), where the amplitude for $|0\rangle$ is real, by "killing" its phase:

$$e^{-i\phi_0}|\psi\rangle = e^{-i\phi_0}(r_0\, e^{i\phi_0}|0\rangle + r_1\, e^{i\phi_1}|1\rangle) = r_0|0\rangle + r_1\, e^{i(\phi_1-\phi_0)}|1\rangle. \tag{5.84}$$

We now have only three real parameters, namely, r_0, r_1, and $\phi = \phi_1 - \phi_0$. But we can do better: using the fact that

$$1 = |c_0|^2 + |c_1|^2 = |r_0\, e^{i\phi_0}|^2 + |r_1\, e^{i\phi_1}|^2 = |r_0|^2|e^{i\phi_0}|^2 + |r_1|^2|e^{i\phi_1}|^2, \tag{5.85}$$

we get that

$$r_0^2 + r_1^2 = 1. \tag{5.86}$$

We can rename them as

$$r_0 = \cos(\theta) \quad \text{and} \quad r_1 = \sin(\theta). \tag{5.87}$$

Summing up, the qubit in Equation (5.80) is now in the canonical representation

$$|\psi\rangle = \cos(\theta)|0\rangle + e^{i\phi}\sin(\theta)|1\rangle, \tag{5.88}$$

with only two real parameters remaining.

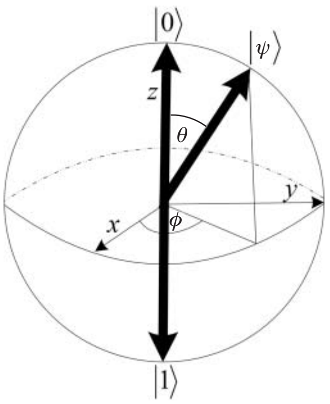


Figure 5.6. Bloch sphere.

What is the range of the two angles θ and ϕ? We invite you to show that $0 \leq \phi < 2\pi$ and $0 \leq \theta < \frac{\pi}{2}$ are enough to cover all possible qubits.

Exercise 5.4.4 Prove that every qubit in the canonical representation given in Equation (5.88) with $\theta > \frac{\pi}{2}$ is equivalent to another one where θ lies in the first quadrant of the plane. (Hint: Use a bit of trigonometry and change ϕ according to your needs.) ■

As only two real numbers are necessary to identify a qubit, we can map it to an arrow from the origin to the three-dimensional sphere of $\mathbb{R}^3$ of radius 1 known as the **Bloch sphere**, as shown in Figure 5.6.

Every qubit can be represented by two angles that describe such an arrow. The two angles will correspond to the latitude (θ) and the longitude (ϕ) needed to specify any position on Earth. The standard parametrization of the unit sphere is

$$x = \cos\phi \sin\theta, \tag{5.89}$$

$$y = \sin\phi \sin\theta, \tag{5.90}$$

$$z = \cos\theta. \tag{5.91}$$

where $0 \leq \phi \leq 2\pi$ and $0 \leq \theta \leq \pi$.

However, there is a caveat: suppose we use this representation to map our qubit on the sphere. Then, the points (θ, ϕ) and $(\pi - \theta, \phi + \pi)$ represent the same qubit, up to the factor -1. Conclusion: the parametrization would map the same qubit twice, on the upper hemisphere and on the lower one. To mitigate this problem, we simply double the "latitude" to cover the entire sphere at "half speed":

$$x = \cos\phi \sin 2\theta, \tag{5.92}$$

$$y = \sin 2\theta \sin\phi, \tag{5.93}$$

$$z = \cos 2\theta. \tag{5.94}$$

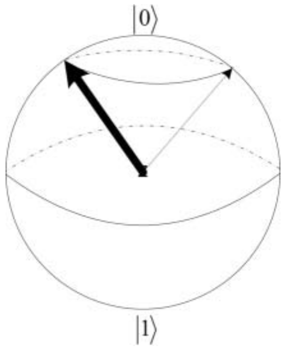
Let us spend a few moments familiarizing ourselves with the Bloch sphere and its geometry. The north pole corresponds to the state $|0\rangle$ and the south pole corresponds to the state $1\rangle$. These two points can be taken as the geometrical image of the good old-fashioned bit. But there are many more qubits out there, and the Bloch sphere clearly shows it.

The precise meaning of the two angles in Equation (5.88) is the following: ϕ is the angle that $|\psi\rangle$ makes from x along the equator and θ is half the angle that $|\psi\rangle$ makes with the z axis.

When a qubit is measured in the standard basis, it collapses to a bit, or equivalently, to the north or south pole of the Bloch sphere. The probability of which pole the qubit will collapse to depends exclusively on how high or low the qubit is pointing, i.e., to its *latitude*. In particular, if the qubit happens to be on the equator, there is a 50–50 chance of it collapsing to either $|0\rangle$ or $|1\rangle$. As the angle θ expresses the qubit's latitude, it controls its chance of collapsing north or south.

Exercise 5.4.5 Consider a qubit whose θ is equal to $\frac{\pi}{4}$. Change it to $\frac{\pi}{3}$ and picture the result. Then compute its likelihood of collapsing to the south pole after being observed. ■

Take an arbitrary arrow and rotate it around the z axis; in the geographical metaphor, you are changing its *longitude:*



Notice that the probability of which classical state it will collapse to is not affected. Such a state change is called a **phase change**. In the representation given in Equation (5.88), this corresponds to altering the phase parameter $e^{i\phi}$.

Before we move on, one last important item: just as $|0\rangle$ and $|1\rangle$ sit on opposite sides of the sphere, so an arbitrary pair of orthogonal qubits is mapped to antipodal points of the Bloch sphere.

Exercise 5.4.6 Show that if a qubit has latitude 2θ and longitude ϕ on the sphere, its orthogonal lives in the antipode $\pi - 2\theta$ and $\pi + \phi$. ■

That takes care of states of a qubit. What about the dynamics? The Bloch sphere is interesting in that every unitary 2-by-2 matrix (i.e., a one-qubit operation) can be visualized as a way of manipulating the sphere. We have seen in Chapter 2, page 66, that every unitary matrix is an isometry. This means that such a matrix maps qubits to qubits and the inner product is preserved. Geometrically, this corresponds to a rotation or an inversion of the Bloch sphere.

The X, Y, and Z Pauli matrices are ways of "flipping" the Bloch sphere 180° about the x, y, and z axes respectively. Remember that X is nothing more than the

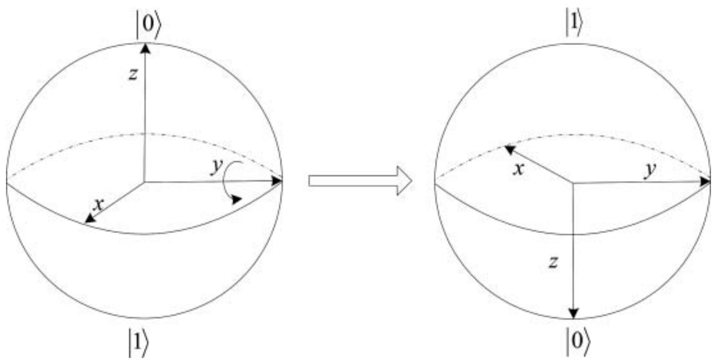


Figure 5.7. A rotation of the Bloch sphere at y.

NOT gate, and takes $|0\rangle$ to $|1\rangle$ and vice versa. But it does more; it takes everything above the equator to below the equator. The other Pauli matrices work similarly. Figure 5.7 shows the action of the Y operation.

There are times when we are not interested in performing a total 180° flip but just want to turn the Bloch sphere θ degrees along a particular direction.

The first such gates are the **phase shift** gates. It is defined as

$$R(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{\theta} \end{bmatrix}. \tag{5.95}$$

This gate performs the following operation on an arbitrary qubit:

$$\cos(\theta')|0\rangle + e^{i\phi}\sin(\theta')|1\rangle = \begin{bmatrix} \cos(\theta') \\ e^{i\phi}\sin(\theta') \end{bmatrix} \mapsto \begin{bmatrix} \cos(\theta') \\ e^{\theta}\, e^{i\phi}\sin(\theta') \end{bmatrix}. \tag{5.96}$$

This corresponds to a rotation that leaves the latitude alone and just changes the longitude. The new state of the qubit will remain unchanged. Only the phase will change.

There are also times when we want to rotate a particular number of degrees around the x, y, or z axis. These three matrices will perform the task:

$$R_x(\theta) = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}X = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \tag{5.97}$$

$$R_y(\theta) = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Y = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \tag{5.98}$$

$$R_z(\theta) = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Z = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}. \tag{5.99}$$

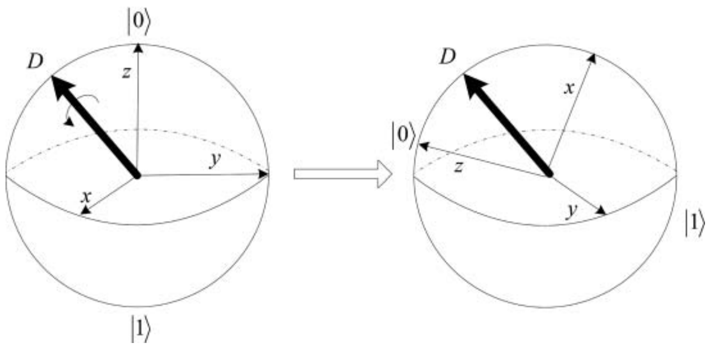


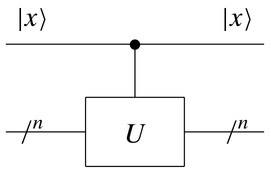
Figure 5.8. A rotation of the Bloch sphere at D.

There are rotations around axes besides the x, y, and z axes. Let $D = (D_x, D_y, D_z)$ be a three-dimensional vector of size 1 from the origin. This determines an axis of the Bloch sphere around which we can spin (see Figure 5.8). The rotation matrix is given as

$$R_D(\theta) = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}(D_xX + D_yY + D_zZ). \tag{5.100}$$

As we have just seen, the Bloch sphere is a very valuable tool when it comes to understanding qubits and one-qubit operations. What about n-qubits? It turns out there is a higher-dimensional analog of the sphere, but coming to grips with it is not easy. Indeed, it is a current research challenge to develop new ways of visualizing what happens when we manipulate several qubits at once. Entanglement, for instance, lies beyond the scope of the Bloch sphere (as it involves at least two qubits).

There are still other quantum gates. One of the central features of computer science is an operation that is done only under certain conditions and not under others. This is equivalent to an IF–THEN statement. If a certain (qu)bit is true, then a particular operation should be performed, otherwise the operation is not performed. For every n-qubit unitary operation U, we can create a unitary $(n + 1)$-qubit operation **controlled-**U or ${}^CU$.



(5.101)

This operation will perform the U operation if the top $|x\rangle$ input is a $|1\rangle$ and will simply perform the identity operation if $|x\rangle$ is $|0\rangle$.

For the simple case of

$$U = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \tag{5.102}$$

the controlled-U operation can be seen to be

$$
{}^{C}U = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix}. \tag{5.103}
$$

This same construction works for matrices larger than 2-by-2.

Exercise 5.4.7 Show that the constructed ${}^{C}U$ works as it should when the top qubit is set to $|0\rangle$ or set to $|1\rangle$. ■

Exercise 5.4.8 Show that if U is unitary, then so is ${}^{C}U$. ■

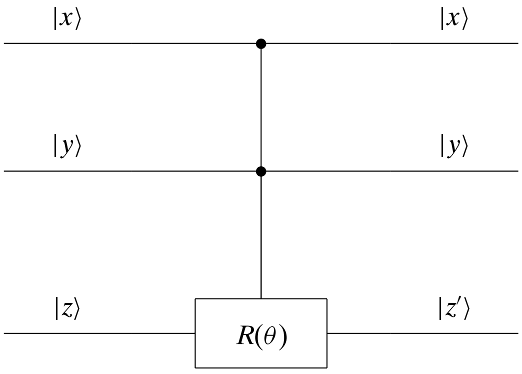
Exercise 5.4.9 Show that the Toffoli gate is nothing more than ${}^{C}({}^{C}\mathrm{NOT})$. ■

It is well known that every logical circuit can be simulated using only the AND gate and the NOT gate. We say that {AND, NOT} forms a set of **universal logical gates**. The NAND gate by itself is also a universal logical gate. We have also seen in Section 5.3 that both the Toffoli gate and the Fredkin gate are each universal logic gates. This leads to the obvious question: are there sets of quantum gates that can simulate all quantum gates? In other words, are there **universal quantum gates**? The answer is yes.[3] One set of universal quantum gates is

$$
\left\{ H,\ {}^{C}\mathrm{NOT},\ R\left(\cos^{-1}\left(\frac{3}{5}\right)\right)\right\}, \tag{5.104}
$$

that is, the Hadamard gate, the controlled-NOT gate, and this phase shift gate.

There is also a quantum gate called the **Deutsch gate**, $D(\theta)$, depicted as



(5.105)

[3] We must clarify what we mean by "simulate." In the classical world, we say that one circuit *Circ* simulates another circuit *Circ′* if for any possible inputs, the output for *Circ* will be the same for *Circ′*. Things in the quantum world are a tad more complicated. Because of the probabilistic nature of quantum computation, the outputs of a circuit are always probabilistic. So we have to reformulate what we mean when we talk about simulate. We shall not worry about this here.

which is very similar to the Toffoli gate. If the inputs $|x\rangle$ and $|y\rangle$ are both $|1\rangle$, then the phase shift operation $R(\theta)$ will act on the $|z\rangle$ input. Otherwise, the $|z'\rangle$ will simply be the same as the $|z\rangle$. When θ is not a rational multiple of π, $D(\theta)$ by itself is a universal three-qubit quantum gate. In other words, $D(\theta)$ will be able to mimic every other quantum gate.

Exercise 5.4.10 Show that the Toffoli gate is nothing more than $D(\frac{\pi}{2})$. ■

Throughout the rest of this text, we shall demonstrate many of the operations that can be performed with quantum gates. However, there are limitations to what can be done with them. For one thing, every operation must be reversible. Another limitation is a consequence of the the **No-Cloning Theorem.** This theorem says that it is impossible to clone an exact quantum state. In other words, it is impossible to make a copy of an arbitrary quantum state without first destroying the original. In "computerese," this says that we can "cut" and "paste" a quantum state, but we cannot "copy" and "paste" it. "Move is possible. Copy is impossible."

What is the difficulty? How would such a cloning operation look? Let $\mathbb{V}$ represent a quantum system. As we intend to clone states in this system, we shall "double" this vector space and deal with $\mathbb{V} \otimes \mathbb{V}$. A potential cloning operation would be a linear map (indeed unitary!)

$$C : \mathbb{V} \otimes \mathbb{V} \longrightarrow \mathbb{V} \otimes \mathbb{V}, \tag{5.106}$$

that should take an arbitrary state $|x\rangle$ in the first system and, perhaps, nothing in the second system and clone $|x\rangle$, i.e.,

$$C(|x\rangle \otimes 0) = (|x\rangle \otimes |x\rangle). \tag{5.107}$$

This seems like a harmless enough operation, but is it? If C is a candidate for cloning, then certainly on the basic states

$$C(|0\rangle \otimes |0\rangle) = |0\rangle \otimes |0\rangle \quad \text{and} \quad C(|1\rangle \otimes |0\rangle) = |1 \otimes |1\rangle. \tag{5.108}$$

Because C must be linear, we should have that

$$C((c_1|0\rangle + c_2|1\rangle) \otimes |0\rangle) = c_1|0\rangle \otimes |0\rangle + c_2|1\rangle \otimes |1\rangle, \tag{5.109}$$

for an arbitrary quantum state, i.e., an arbitrary *superposition* of $|0\rangle$ and $|1\rangle$. Suppose we start with $\frac{|x\rangle+|y\rangle}{\sqrt{2}}$. Cloning such a state would mean that

$$C\left(\frac{|x\rangle + |y\rangle}{\sqrt{2}} \otimes 0\right) = \left(\frac{|x\rangle + |y\rangle}{\sqrt{2}} \otimes \frac{|x\rangle + |y\rangle}{\sqrt{2}}\right). \tag{5.110}$$

However, if we insist that C is a quantum operation, then C must be linear,[4] and hence, must respect the addition and the scalar multiplication in $\mathbb{V} \otimes \mathbb{V}$. If C was

[4] Just a reminder: C being linear means that

$$C(|\phi\rangle + |\psi\rangle) = C(|\phi\rangle) + C(|\psi\rangle) \tag{5.111}$$

and

$$C(c|\phi\rangle) = cC(|\phi\rangle). \tag{5.112}$$

linear, then

$$\begin{aligned}C\left(\frac{|x\rangle + |y\rangle}{\sqrt{2}} \otimes 0\right) &= C\left(\frac{1}{\sqrt{2}}(|x\rangle + |y\rangle) \otimes 0\right) = \frac{1}{\sqrt{2}}C((|x\rangle + |y\rangle) \otimes 0)\\ &= \frac{1}{\sqrt{2}}(C(|x\rangle \otimes 0 + |y\rangle \otimes 0)) = \frac{1}{\sqrt{2}}(C(|x\rangle \otimes 0) + C(|y\rangle \otimes 0))\\ &= \frac{1}{\sqrt{2}}((|x\rangle \otimes |x\rangle) + (|y\rangle \otimes |y\rangle)) = \frac{(|x\rangle \otimes |x\rangle) + (|y\rangle \otimes |y\rangle)}{\sqrt{2}}.\end{aligned} \tag{5.113}$$

But

$$\left(\frac{|x\rangle + |y\rangle}{\sqrt{2}} \otimes \frac{|x\rangle + |y\rangle}{\sqrt{2}}\right) \neq \frac{(|x\rangle \otimes |x\rangle) + (|y\rangle \otimes |y\rangle)}{\sqrt{2}}. \tag{5.114}$$

So C is not a linear map,[5] and hence is not permitted.

In contrast to cloning, there is no problem **transporting** arbitrary quantum states from one system to another. Such a transporting operation would be a linear map

$$T : \mathbb{V} \otimes \mathbb{V} \longrightarrow \mathbb{V} \otimes \mathbb{V}, \tag{5.115}$$

that should take an arbitrary state $|x\rangle$ in the first system and, say, nothing in the second system, and transport $|x\rangle$ to the second system, leaving nothing in the first system, i.e.,

$$T(|x\rangle \otimes 0) = (0 \otimes |x\rangle). \tag{5.116}$$

We do not run into the same problem as earlier if we transport a superposition of states. In detail,

$$\begin{aligned}T\left(\frac{|x\rangle + |y\rangle}{\sqrt{2}} \otimes 0\right) &= T\left(\frac{1}{\sqrt{2}}(|x\rangle + |y\rangle) \otimes 0\right)\\ &= \frac{1}{\sqrt{2}}T((|x\rangle + |y\rangle) \otimes 0) = \frac{1}{\sqrt{2}}T((|x\rangle \otimes 0) + (|y\rangle \otimes 0))\\ &= \frac{1}{\sqrt{2}}(T(|x\rangle \otimes 0) + T(|y\rangle \otimes 0)) = \frac{1}{\sqrt{2}}((0 \otimes |x\rangle) + (0 \otimes |y\rangle))\\ &= \frac{(|0 \otimes (|x\rangle + |y\rangle)}{\sqrt{2}} = 0 \otimes \frac{(|x\rangle + |y\rangle)}{\sqrt{2}}.\end{aligned} \tag{5.117}$$

This is exactly what we would expect from a transporting operation.[6]

Fans of Star Trek have long known that when Scotty "beams" Captain Kirk down from the Starship Enterprise to the planet Zygon, he is transporting Captain Kirk to Zygon. The Kirk of the Enterprise gets destroyed and only the Zygon Kirk survives. Captain Kirk is not being cloned. He is being transported. (Would we really want many copies of Captain Kirk all over the Universe?)

[5] C is, however, a legitimate set map.

[6] In fact, we will show how to transport arbitrary quantum states at the end of Chapter 9.

The reader might see an apparent contradiction in what we have stated. On the one hand, we have stated that the Toffoli and Fredkin gates can mimic the fanout gate. The matrices for the Toffoli and Fredkin gates are unitary, and hence they are quantum gates. On the other hand, the no-cloning theorem says that no quantum gates can mimic the fanout operation. What is wrong here? Let us carefully examine the Fredkin gate. We have seen how this gate performs the cloning operation

$$(x, 1, 0) \qquad \mapsto \qquad (x, \neg x, x). \tag{5.118}$$

However, what would happen if the x input was in a superposition of states say, $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$, while leaving $y = 1$ and $z = 0$. This would correspond to the state

$$\begin{array}{cccccccc} \mathbf{000} & \mathbf{001} & \mathbf{010} & \mathbf{011} & \mathbf{100} & \mathbf{101} & \mathbf{110} & \mathbf{111} \end{array}^{T}$$
$$\left[\begin{array}{cccccccc} 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 \end{array}\right]. \tag{5.119}$$

Multiplying this state with the Fredkin gate gives us

$$\begin{array}{c|cccccccc} & \mathbf{000} & \mathbf{001} & \mathbf{010} & \mathbf{011} & \mathbf{100} & \mathbf{101} & \mathbf{110} & \mathbf{111} \\ \hline \mathbf{000} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{001} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{010} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{011} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \mathbf{100} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \mathbf{101} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \mathbf{110} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \mathbf{111} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{bmatrix}. \tag{5.120}$$

The resulting state is

$$\frac{|0, 1, 0\rangle + |1, 0, 1\rangle}{\sqrt{2}}. \tag{5.121}$$

So, whereas on a classical bit x, the Fredkin gate performs the fanout operation, on a superposition of states the Fredkin gate performs the following very strange operation:

$$\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}, 1, 0\right) \qquad \mapsto \qquad \frac{|0, 1, 0\rangle + |1, 0, 1\rangle}{\sqrt{2}}. \tag{5.122}$$

This strange operation is not a fanout operation. Thus, the no-cloning theorem safely stands.

Exercise 5.4.11 Do a similar analysis for the Toffoli gate. Show that the way we set the Toffoli gate to perform the fanout operation does not clone a superposition of states. ■

Reader Tip. The no-cloning theorem is of major importance in Chapter 9. ♡

References: The basics of qubits and quantum gates can be found in any textbook on quantum computing. They were first formulated by David Deutsch in 1989 (Deutsch, 1989).

Section 5.2 is simply a reformulation of basic computer architecture in terms of matrices.

The history of reversible computation can be found in Bennett (1988). The readable article (Landauer, 1991) by one of the forefathers of reversible computation is strongly recommended.

The no-cloning theorem was first proved in Dieks (1982) and Wootters and Zurek (1982).