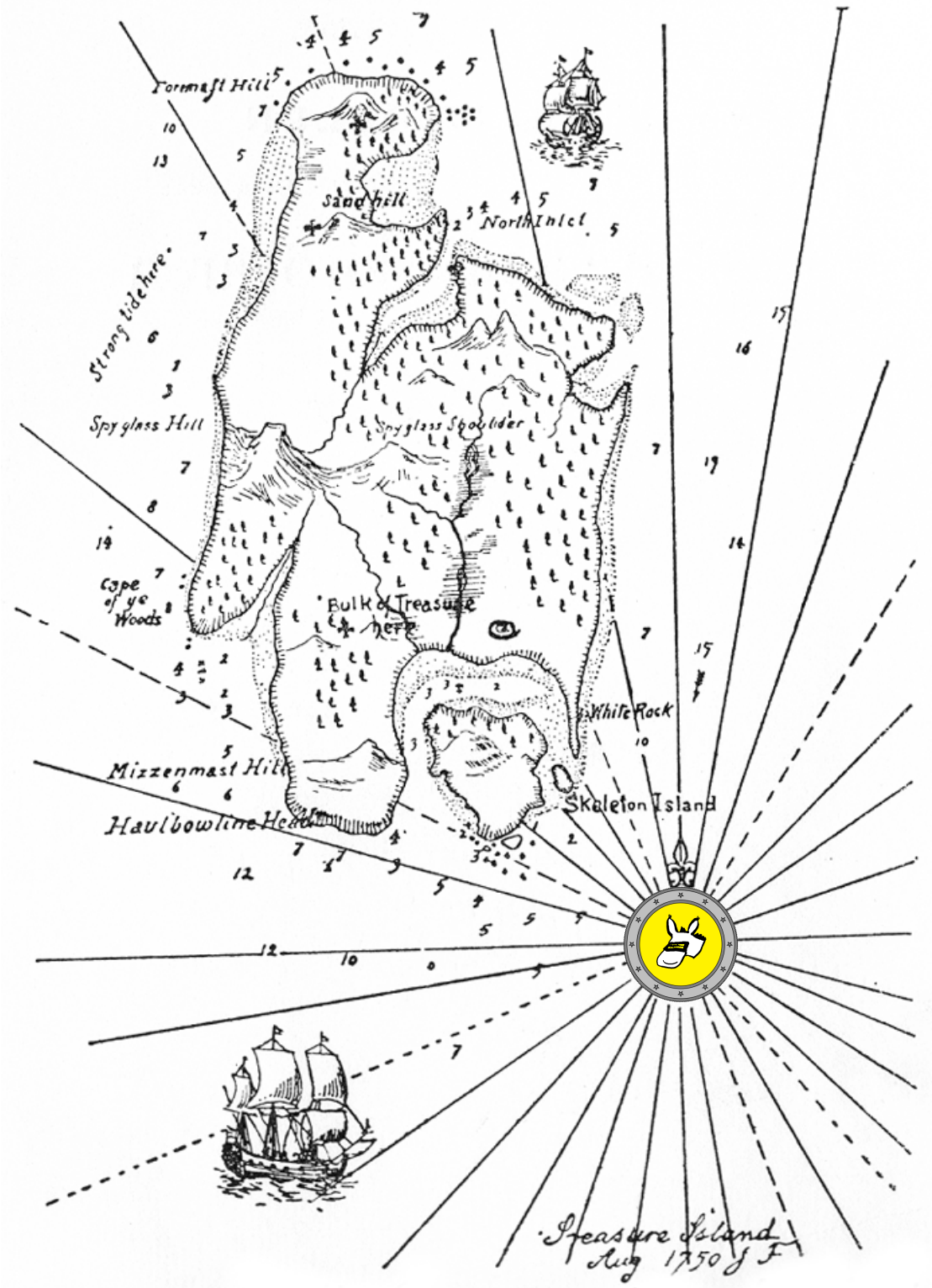
The Quantum Quest



Welcome to *The Quantum Quest*!

The goal of this five-week adventure is for you to learn the basics of quantum computing. By the end of it, you will understand what are quantum bits and quantum algorithms, and what are they good for. Along the way, you will become good friends with Alice and Bob who live in year 2058 and (just like you) also tinker with quantum computers after school. While scientists are still building actual quantum computers in their labs today, in 2058 quantum computers are everywhere, even in your pocket! However, as with all technology, not everyone is using it for good means, so you will have to help your two friends – Alice and Bob – to invent various tricks to protect themselves from the evil hacker Eve. Good luck on your adventure!

Truly quantumly yours,
Maris Ozols & Michael Walter

Reader's guide

Throughout these lecture notes, we have included plenty of exercises (in green boxes) and homeworks (in red boxes). The exercises are meant for you to test your understanding as you read along, and we provide solutions for all of them at the end of each quest. The homeworks are meant to be handed in each week. Some of the problems are marked as 'optional' – we think they are not strictly necessary for following the course, but they provide good additional practice. Some are also marked as 'challenging' – these are a bit harder then the rest!

Acknowledgments

We would like to thank to the following people for helping us run this web class: Doutzen Abma, Maya Bach, Valerie Bettaque, Amalia Böttger, Milo Camardese, Arjan Cornelissen, Bas Dirkse, Oliver Dorogi, Jari Egbers, Yassine Ferjani, Marten Folkertsma, Koen Groenland, Galina Pass, Philip Verduyn Lunel, Anurudh Peduri, Simon Schmidt, Quinten Tupker, Mees de Vries, Jordi Weggemans, Peter Ypma. We are also grateful to Craig Gidney, whose quantum simulator QUIRK we modified to build QUIRKY. Finally, we would like to thank all the enthusiastic students who participated in the web class.

The Quantum Quest

Maris Ozols and Michael Walter

November 2023

Contents

The Quantum Quest **1**

1 Quest 1: Maestro of probability **3**
1.1 Probabilistic bits . . . 3
1.1.1 Multiplying probabilities . . . 5
1.1.2 Adding probabilities . . . 5
1.1.3 Probability and computation . . . 6
1.2 Operations on a probabilistic bit . . . 7
1.2.1 Extending by linearity . . . 8
1.2.2 Random operations . . . 9
1.3 Measuring a probabilistic bit . . . 10
1.4 The QUIRKY simulator . . . 12
1.4.1 Getting started . . . 12
1.4.2 Making your own operations . . . 14
1.4.3 A mysterious operation . . . 14
1.5 Exercise solutions . . . 16

2 Quest 2: Conqueror of the qubit **19**
2.1 Quantum bits . . . 19
2.1.1 Probabilities versus amplitudes . . . 19
2.1.2 Qubit as a circle . . . 20
2.2 Measuring a quantum bit . . . 21
2.3 Simulating quantum bits with QUIRKY . . . 23
2.4 Operations on a quantum bit . . . 24
2.4.1 Rotations . . . 26
2.4.2 Composing quantum operations . . . 28
2.4.3 Reflections . . . 29
2.5 Distinguishing quantum states . . . 30
2.5.1 Another mysterious operation . . . 32
2.6 Interlude on physics (optional) . . . 33
2.6.1 Interference . . . 33
2.6.2 Polarization . . . 35
2.7 Exercise solutions . . . 37

3 Quest 3: Wizard of entanglement **41**
3.1 Two probabilistic bits . . . 41
3.1.1 Measuring both bits . . . 42
3.1.2 Local operations . . . 43
3.1.3 Measuring only one bit . . . 45
3.1.4 State of the other bit . . . 46
3.1.5 SWAP operation . . . 48
3.1.6 Controlled-NOT operation . . . 48
3.1.7 Product distributions . . . 50
3.1.8 Correlated distributions . . . 52
3.2 Two quantum bits . . . 54
3.2.1 Measuring two qubits . . . 56
3.2.2 Local operations . . . 56
3.2.3 Parallel operations . . . 58
3.2.4 Controlled operations . . . 60
3.2.5 Entangled states . . . 61
3.2.6 Entanglement and correlations . . . 63
3.2.7 The power of entanglement . . . 64
3.3 Exercise solutions . . . 67

4 Quest 4: Quantum composer **73**
4.1 Quantum circuits . . . 73
4.1.1 Many quantum bits . . . 73
4.1.2 Operations . . . 75
4.1.3 The most general quantum operations . . . 77
4.1.4 Circuit identities . . . 77
4.1.5 Measuring all qubits . . . 78
4.1.6 Measuring some of the qubits only . . . 79
4.2 Quantum surprises . . . 81
4.2.1 No cloning . . . 82
4.2.2 One-time pad . . . 83
4.2.3 Quantum teleportation . . . 84
4.2.4 A glance at quantum networks . . . 88
4.2.5 The uncertainty principle . . . 89
4.3 Exercise solutions . . . 92

5 Quest 5: Algorithm virtuoso **95**
5.1 Talking to oracles . . . 96
5.1.1 Reversible computation . . . 97
5.1.2 Bit oracles . . . 98
5.1.3 Sign oracles . . . 99
5.2 Quantum algorithms . . . 101
5.2.1 Deutsch's algorithm . . . 101
5.2.2 Hadamard transform and interference . . . 103
5.2.3 Deutsch-Jozsa algorithm . . . 106
5.2.4 Bernstein-Vazirani algorithm . . . 107
5.3 Searching with Grover . . . 109
5.3.1 Angle amplification . . . 111
5.4 Your quantum journey . . . 111
5.5 Exercise solutions . . . 112

Quest 1: Maestro of probability

Welcome to the first week of *The Quantum Quest* – you are at the very beginning of an exciting adventure!

Quantum computing is a fascinating topic that can easily capture your imagination. The main source of our fascination with it is the strangeness of the quantum world. However, it is also the main source of our confusion. Indeed, it is not hard to get entangled in quantum weirdness or lost in the exponentially big state space of a quantum computer. To avoid such trouble, you need to prepare yourself by first getting familiar with the world of probabilities. Once you have become a maestro of probability, you will be able to unlock the door to the quantum world as well!

The goal of the first quest is for you to learn about probabilities and probabilistic bits: what are the states of a probabilistic bit, what are the allowed operations on it, and how can we extract information from a probabilistic bit by a measurement? The main focus of the second quest will be quantum bits, which are very similar to probabilistic bits.

1.1 Probabilistic bits

Probabilities are used to quantify the likelihood of events – the more likely an event is, the larger its probability. An event that occurs with certainty has probability 1 (indeed, it has a 100% chance of happening) while an event that never occurs has probability 0.

As an example, we can think of tossing a coin. If you toss a coin and cover it with your hands without looking at it, there are two possible events – either the coin shows "heads" or it shows "tails". For a *fair* coin, the two events are equally likely to occur, so we assign a probability of $\frac{1}{2} = 0.50$ or 50% to both (this is represented by in Fig. 1.1). However, the coin could be biased and more likely to land on one side than the other. Depending on how biased it is, we can imagine a whole spectrum of possibilities: one extremely biased coin might always show "heads" while another might always show "tails" (see and on the left and right side of Fig. 1.1). The first coin shows "tails" with probability 0 (since it always shows "heads") while the second shows "tails" with probability 1.

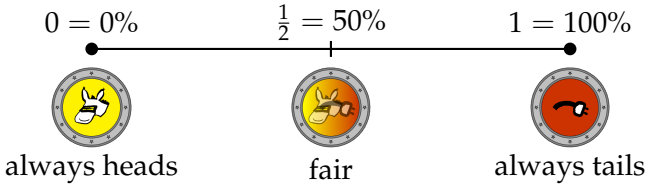


Figure 1.1: A probabilistic bit describing the state of a random donkey coin.

Since we don't want to worry about designing coins of different shapes and sizes, it is helpful to somehow abstract the information represented by a coin toss. We can do this by associating the outcomes "heads" and "tails" with the bit values 0 and 1, respectively. The coin toss can then be described by two probabilities: the probability p_0 that it is equal to 0 ("heads"), and the probability p_1 that it is equal to 1 ("tails"). Such a bit that attains its two possible values with some probabilities is called a probabilistic bit. Note that the two probabilities $p_0, p_1 \geq 0$ must necessarily add to one: $p_0 + p_1 = 1$. For example, if the coin is fair then $p_0 = p_1 = \frac{1}{2}$, as explained above.

Did you notice that it would be enough to specify just one of the probabilities p_0 or p_1 since either of them can be obtained from the other? For clarity we will always specify both and write

them down as a vector:

$$p = \begin{pmatrix} p_0 \\ p_1 \end{pmatrix}. \tag{1.1}$$

We call this vector the probability distribution or the state of the probabilistic bit. This vector notation is convenient not only for collecting all probabilities together in a nice table, but it will also provide a geometrical way to visualize a probabilistic bit. Moreover, it will help us to see the analogy between probabilistic and quantum bits.

We will call the states associated to 0 or "heads" and to 1 or "tails" deterministic, since in this case there is no uncertainty – is it fully determined which side of the coin is up. In Eq. (1.1), they correspond to probability distributions with $p_0 = 1$, $p_1 = 0$ and $p_0 = 0$, $p_1 = 1$, respectively. Since they will be used very often, it is convenient to introduce a shorthand notation for them:

$$[0] = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad [1] = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{1.2}$$

This notation might look confusing at first, but you can also think of $[0]$ and $[1]$ as and or as [heads] and [tails], if you wish.

These two states form a basis of all states, meaning that we can express any other state of a probabilistic bit as a linear combination of them:

$$\begin{pmatrix} p_0 \\ p_1 \end{pmatrix} = p_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + p_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = p_0[0] + p_1[1]. \tag{1.3}$$

Every state is a two-dimensional vector, which we can visualize in a two-dimensional coordinate system. Then the possible states of a probabilistic bit is precisely the line segment connecting the two points $[0]$ and $[1]$ which correspond to the two deterministic states of the bit (see Fig. 1.2).

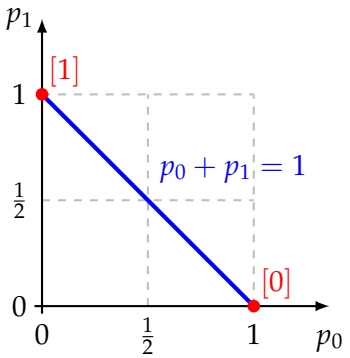


Figure 1.2: The blue line segment corresponds to the states of a probabilistic bit.

Exercise 1.1: Understanding the blue line segment

According to Fig. 1.2, the possible states of a probabilistic bit form a line segment. Take some time to investigate this. See if you can answer the following questions:

1. Why do the states of a probabilistic bit all lie on a line?
2. Why does this line end at the coordinate axes and does not go further?
3. Which point on the line segment corresponds to a fair coin?

1.1.1 Multiplying probabilities

If you toss two coins, what is the probability that both coins are "heads"? Assume the two coins are described by probabilistic bits

$$a = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \qquad \text{and} \quad b = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}, \tag{1.4}$$

where outcome 0 corresponds to "heads" and outcome 1 to "tails". Then the probability to get "heads" for coin a is a_0 while for coin b it is b_0. (We don't assume that the coins are fair, so these probabilities are not necessarily 50%.) The probability that both coins simultaneously show "heads" is given by *multiplying* the probabilities of the two individual events:

$$p_{00} = a_0 b_0. \tag{1.5}$$

Note that $p_{00} \leq a_0$ and $p_{00} \leq b_0$ since $a_0 \leq 1$ and $b_0 \leq 1$. This is intuitive, since getting "heads" simultaneously for both coins should be no more likely (and is typically less likely) than getting it for any of the coins individually. You can similarly compute the probabilities of all other combinations of heads and tails. We summarize all four cases in the following table:

$$\begin{aligned} p_{00} &= a_0 b_0, \qquad & p_{01} &= a_0 b_1, \\ p_{10} &= a_1 b_0, \qquad & p_{11} &= a_1 b_1. \end{aligned} \tag{1.6}$$

We call two events independent if they originate from two different sources, and the occurrence of one of them doesn't tell you anything about the occurrence of the other. Typically such situation is described using the word "and". For example, "the first coin is heads *and* the second coin is tails". We multiply probabilities if we want to know if two independent events occurred simultaneously.

Exercise 1.2: Multiplying probabilities

Alice is bored during her math class and starts looking at her digital watch. The second's counter on her watch can show values from 00 to 59. Assume that at some random point within the next minute Alice looks at the second's counter on her watch.

1. What is the probability that she sees 00?
2. What is the probability that the last digit is 0?
3. What is the probability that the first digit is 0?
4. Argue that the values of both digits are independent from each other. Verify your answer to question 1 by multiplying the probabilities from questions 2 and 3.

1.1.2 Adding probabilities

Consider now the following slightly more complicated problem. Assume again that you toss the coins a and b. What is the probability that both coins show the same outcome? There are two ways this could happen – either both coins are "heads" or both coins are "tails". We already know from Eq. (1.6) that the probabilities of these two individual events are

$$p_{00} = a_0 b_0, \qquad p_{11} = a_1 b_1.$$

Then the probability that one of these two events occurs is obtained by *adding* the probabilities:

$$p_{00} + p_{11} = a_0 b_0 + a_1 b_1. \tag{1.7}$$

You add probabilities when you want to group multiple outcomes of the same experiment together. Such combined events can usually be described using the word "or". For example, "both coins are heads *or* both coins are tails". Be careful: this only works when the two coins do not influence each other!

Exercise 1.3: Adding probabilities

Bob is also bored during the math class. He notices that Alice is staring at her watch so he takes a look at his watch too. Surprisingly, its second's counter shows 44, which seems very unlikely to Bob. What is the probability that both digits of the second's counter are the same if Bob takes a look at his watch at some random point within a minute?

Now that you know when to add and when to multiply probabilities, try to solve your first homework assignment!

Homework 1.1: Opposite coins

Alice tosses two coins called a and b, with probability distributions

$$a = \begin{pmatrix} 2/3 \\ 1/3 \end{pmatrix}, \qquad b = \begin{pmatrix} 3/4 \\ 1/4 \end{pmatrix}.$$

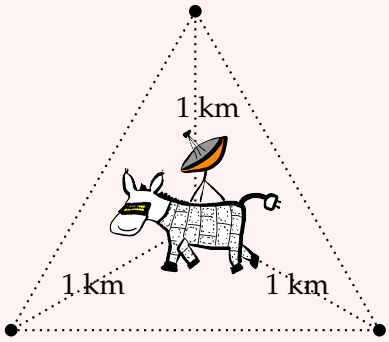
What is the probability that the two coins produce *opposite* outcomes?

1.1.3 Probability and computation

Is there any use of probabilistic bits in computation? It might seem at first that they are not particularly useful since the values 0 and 1 of a regular bit represent definite knowledge while a probabilistic bit represents *approximate* knowledge (or the *lack* of knowledge). Why should I waste the space on my computer to store probabilistic bits that reflect my lack of information if instead I could store the actual information that I have, even if it is incomplete? The advantage of probabilistic bits is that they represent partial knowledge more accurately – if you don't know something, it is better to admit it and make a random guess rather than pretend that you know the correct answer with certainty. This is illustrated in the problem below where Alice's donkey robot has to make a decision while lacking full information.

Homework 1.2: Alice's donkey

Problem: Alice wants to program her donkey robot so that it can walk on its own to a charging station and charge itself. There are three nearby stations, each at a distance of 1 km from the donkey, forming an equilateral triangle with the donkey located in its center. Alice's donkey has enough battery left to walk only 2.8 km.



Alice is going to upload a program to her donkey robot that tells it where to go, however she knows that her evil classmate Eve is trying to sabotage her. Since Eve can read everything that's transmitted over the WiFi, Eve can also see what program Alice is uploading. Because of this, once the program is uploaded, Alice will disconnect her donkey from the WiFi so that Eve cannot follow its movements. While the donkey walks, the only way for Eve to sabotage it is by hacking and disabling the charging stations towards which it has been programmed to go. However, Eve can shut down only two stations before her intrusion is detected. Since Eve cannot follow the donkey's movements,

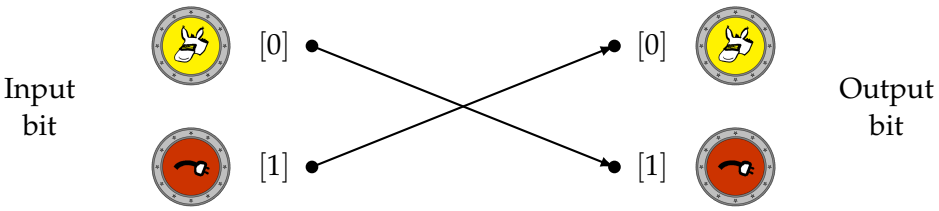
she has to decide which two charging stations to disable based only on Alice's program.

Questions:

1. How many charging stations can the donkey visit before its battery runs out?
2. Assume that Alice programs her donkey to visit stations in some predetermined order. Can Eve prevent it from reaching a working charging station? Remember that Eve has full access to Alice's program and so she knows in what order the donkey is programmed to visit the stations.
3. Assume that Alice programs the donkey to make its own random choices about where to go. (While Eve can see that this is what Alice has programmed, she cannot predict what choices the donkey will make once it starts to walk.) What randomized strategy should Alice upload to the donkey, and what hacking strategy should Eve use to counteract it? What is the probability that Alice's donkey successfully reaches a working charging station if both Alice and Eve are using optimal strategies?

1.2 Operations on a probabilistic bit

Once we have represented bits of information by vectors, we can represent operations on these bits by linear transformations and use tools from linear algebra. For example, consider the operation that exchanges "heads" and "tails" of a donkey coin:



We will denote this operation by NOT and write it down mathematically as follows:

(1.8)

Using the conventions from Eq. (1.2), we can also write

$$\text{NOT}\,[0] = [1], \qquad \text{NOT}\,[1] = [0]. \tag{1.9}$$

Note that we are writing NOT p as an abbreviation to NOT(p) – both simply mean that the operation NOT acts on a vector p.[1] We will generally write operations with capital letters to distinguish them from numbers and vectors.

Recall from Eq. (1.2) that the vectors $[0]$ and $[1]$ represent the deterministic states 0 and 1 of a probabilistic bit. Since the NOT operation exchanges these two vectors, it negates the value of the bit. This is precisely why we called it "NOT" – it represents the logical negation! One simple application of NOT is to enter some data in your computer. If all bits of your computer are initially set to 0, you can change some of them to 1 to enter data – this is often the first step of a computation.

How should we define the NOT operation on a probabilistic bit $p = \binom{p_0}{p_1}$? With probability p_0, the bit is zero and will get flipped to a one. With probability p_1, the bit is one and will get flipped to a zero. Thus, the effect of the NOT operation on a probabilistic bit is simply

[1] We could also write NOT $\cdot\, p$ since this action actually corresponds to matrix-vector multiplication.

$$\text{NOT}\begin{pmatrix} p_0 \\ p_1 \end{pmatrix} = \begin{pmatrix} p_1 \\ p_0 \end{pmatrix}. \tag{1.10}$$

In particular, this recovers Eq. (1.9) when $p_0 = 1$ (and $p_1 = 0$) or $p_0 = 0$ (and $p_1 = 1$). You can intuitively think of the NOT operation and Eq. (1.10) as follows: if you imagine a probabilistic bit as a coin that you have tossed but have not looked at yet, then the NOT operation corresponds to turning the coin around (again, without looking at it).

Exercise 1.4: Visualizing the NOT operation

Recall from Fig. 1.2 that all possible states of a probabilistic bit correspond to a line segment. Let us try to visualize how the NOT operation transforms this line segment.

1. Take an arbitrary[a] point with coordinates (p_0, p_1) on this segment. Where is it sent to by the NOT operation?
2. Where are the two endpoints of the line segment sent to?
3. Is there any point on the line segment that is sent to itself?

[a] When we say 'arbitrary', we mean that your calculation must work for any choice of p_0 and p_1. It is often best to write all the steps, including your final answer, in terms of p_0 and p_1, treating them as unknown numbers.

1.2.1 Extending by linearity

How should we define $M\left(\begin{smallmatrix} p_0 \\ p_1 \end{smallmatrix}\right)$ when M is an arbitrary operation on a bit? As before, we assume that we know how M acts on the two possible values of the bit, and we write $M[0]$ for the result of the operation when the bit is zero, and $M[1]$ for the result of the operation when the bit is one. (For the NOT operation, this was precisely what we did in Eq. (1.9).) Let us try to apply the same reasoning as above. With probability p_0, the bit is zero and so we obtain $M[0]$ by applying the operation M. With probability p_1, the bit is one and we obtain instead $M[1]$. Together, we see that we should define

$$M\begin{pmatrix} p_0 \\ p_1 \end{pmatrix} = p_0\, M[0] + p_1\, M[1], \tag{1.11}$$

where $p_0\, M[0]$ means that we are multiplying the vector $M[0]$ by the probability p_0.

Exercise 1.5: NOT on probabilistic bits

Show that when M is the NOT operation then Eq. (1.11) reproduces precisely Eq. (1.10).

Using Eq. (1.3), we can also write Eq. (1.11) in the following way:

$$M\Big(p_0\,[0] + p_1\,[1]\Big) = p_0\, M[0] + p_1\, M[1]. \tag{1.12}$$

Note that the difference between the two sides is in the order of operations: on the left-hand side we first take a linear combination and then act with M, while on the right-hand side we first act with M and only then take the linear combination. This equation looks very similar to the familiar rule $a(b+c) = ab + ac$ for numbers (the "distributive law").

If M is an operation on probabilistic bits that satisfies Eq. (1.12) then we say that M is linear. Our rule in Eqs. (1.11) and (1.12) for extending M from bits to probabilistic bits is called extending by linearity. We will often use the same idea also for quantum bits.

1.2.2 Random operations

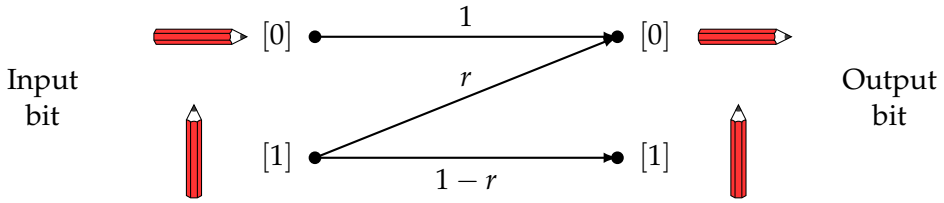
Note that, in our derivation of Eq. (1.11), we did not in fact assume that $M[0]$ and $M[1]$ were again one of the two deterministic states $[0]$ or $[1]$ of a bit (even though this happened to be the case for the NOT operation). This means that Eq. (1.11) works just as well when $M[0]$ or $M[1]$ are probabilistic bits! In this case we say that M is a random operation.

One of the simplest examples of a random operation is as follows. Imagine that you encode $[0]$ by placing a pencil horizontally on a table, and $[1]$ by balancing it vertically. If you gently hit the table with your hand, the pencil might fall down, changing its state from $[1]$ to $[0]$. However, if it was already laying down, its state $[0]$ will not change. Thus hitting the table *probabilistically resets* the pencil's state to $[0]$; the harder you hit, the more likely you are to reset it.

Mathematically, probabilistic reset is described by an operation $R(r)$ defined as

$$R(r)[0] = [0] = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad R(r)[1] = r[0] + (1-r)[1] = \begin{pmatrix} r \\ 1-r \end{pmatrix}, \tag{1.13}$$

where $r \in [0,1]$ is the *reset* probability. You can visualize the action of this operation as follows:



By linearity, we can extend this operation to all states:

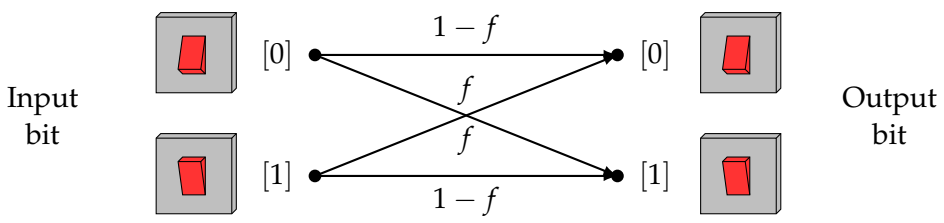
$$\begin{aligned} R(r)\begin{pmatrix} p_0 \\ p_1 \end{pmatrix} &= p_0\, R(r)[0] + p_1\, R(r)[1] \\ &= p_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + p_1 \begin{pmatrix} r \\ 1-r \end{pmatrix} = \begin{pmatrix} p_0 + p_1 r \\ p_1(1-r) \end{pmatrix}. \end{aligned}$$

As a special case of this equation, note that $R(0)$ does not change the state at all while $R(1)$ resets any state to $[0]$.

Another interesting example of a random operation is the probabilistic flip operation $F(f)$ that flips the input bit with probability f and leaves it alone with probability $1-f$:

$$F(f)[0] = (1-f)[0] + f[1], \qquad F(f)[1] = f[0] + (1-f)[1], \tag{1.14}$$

where $f \in [0,1]$ is the *flip* probability. More intuitively, imagine that $[0]$ and $[1]$ represent two states of a light switch on a wall. If you throw a pillow at the switch, you will successfully hit and flip it only with probability f, and with probability $1-f$ it will remain unchanged. You can visualize the action of $F(f)$ as follows:



The next exercise will help you to get more familiar with the probabilistic flip operation.

Exercise 1.6: Probabilistic flip

Recall that $F(f)$ denotes the probabilistic flip operation from Eq. (1.14).

1. Write down $F(f)\,[0]$ and $F(f)\,[1]$ as vectors.
2. For what value of f does $F(f)$ act as NOT? How can we prepare a probabilistic bit in an arbitrary state $\binom{p}{1-p}$ from $[0]$ by using F?
3. Extend $F(f)$ by linearity to probabilistic bits by computing $F(f)\binom{p_0}{p_1}$.
4. Let $\binom{p_0}{p_1}$ be an arbitrary probability distribution. Show that $F(1/2)\binom{p_0}{p_1} = \binom{1/2}{1/2}$.

The last part of this exercise shows that $F(1/2)$ always prepares the uniform distribution, no matter the input distribution. This can be used for simulating the toss of an unbiased coin. In fact, in the next exercise, you will show that by carefully adjusting the flip probability f you can also use $F(f)$ to *change* the bias of a given coin.

Homework 1.3: Chocolate coin

Today is Bob's birthday! Since he likes chocolate, Alice decides to make a chocolate coin for him. To make it more special, the shape of the coin should be such that if you spin it on the table, it would land on with probability $q = 5/15$ that represents Bob's birthday, May 15th. After experimenting with several different shapes, Alice manages to produce a chocolate coin with the right probability. Very excited, she leaves it on the table and runs to the shop to buy a nice birthday card.

Unfortunately, when she returns, Alice realizes that the coin was left in the sun and its edge has melted. After trying it out, Alice determines that the new probability of is $p = 4/15$. Since there is no time left to fix the problem, Alice writes on the birthday card that once the coin has landed, Bob should flip it around with probability f, and only then he will observe with the right probability q. Help Alice to determine the right value of f.

Hint: The quantities p, q, and f should satisfy $F(f)\binom{p}{1-p} = \binom{q}{1-q}$.

Exercise 1.7: Flip from reset and NOT (optional and challenging)

How can you build $F(f)$ using the $R(r)$ and NOT operations?

1.3 Measuring a probabilistic bit

If you toss a fair coin and immediately cover it, you have no idea on which side it landed. In this situation your knowledge about the state of the coin is described by the uniform distribution

$= \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} = \frac{1}{2}\,[0] + \frac{1}{2}\,[1].$ (1.15)

However, if you uncover the coin and see "heads", your knowledge is updated to

$= [0]$ (1.16)

because now you know with certainty that heads is up. We will refer to the process of uncovering a probabilistic coin to determine which side is up as measurement.[2]

Notice from Eqs. (1.15) and (1.16) that the state of the coin before and after the measurement is different. Indeed, after the measurement you are no longer ignorant about which side is up.

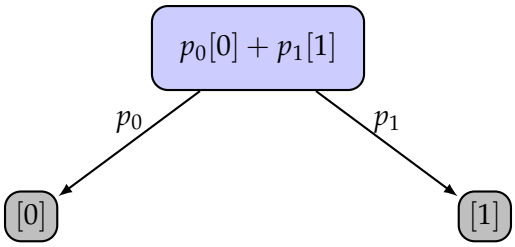
[2]We have borrowed this term from quantum computing where we will see that a similar procedure exists.

Now, imagine you cover the coin again after you have just measured it. What is its state now? Well, it clearly is still

$= [0]$ (1.17)

because you already know that "heads" is up. In fact, even if you measure (look at) the coin again, you will still see "heads". Similarly, if you got "tails" the first time you measured a random coin, you will keep getting "tails" no matter how many times you measure it again.

More generally, if you have a probabilistic bit described by the distribution $\binom{p_0}{p_1}$, measuring it yields the outcome 0 (or "heads") with probability p_0 and 1 (or "tails") with probability p_1:



(1.18)

The state of the probabilistic bit *after* the measurement is no longer $\binom{p_0}{p_1}$ but one of the basis states $[0] = \binom{1}{0}$ or $[1] = \binom{0}{1}$, depending on the measurement outcome. For example, if you got outcome 1 (or "tails"), the new state is $[1]$. In general a measurement *changes* the state!

An important point about measurements is that they *do not* let you extract the actual probabilities p_0 and p_1 – all you get as a measurement outcome is a single bit 0 or 1. Also, your original probabilistic bit $\binom{p_0}{p_1}$ is gone after the measurement, so you cannot measure it again. This is in fact quite intuitive. If we toss a coin exactly once then we get a single random outcome – but from this single outcome alone we cannot learn whether the coin was fair or biased!

However, suppose we toss the same coin a large number of times. In this case we would expect that the fraction of times that we obtain outcome 1 is roughly p_1. In other words,

$$\frac{N_1}{N} \approx p_1, \tag{1.19}$$

where N is the total number of measurements and N_1 the number of times that we obtained outcome 1. The more outcomes we have gathered, the better the approximation should get.[3]

This provides us with a procedure for estimating p_1. Of course, since $p_0 + p_1 = 1$, this also gives us an estimate of p_0. For example, Alice could use this procedure to estimate the probability that her biased coin in Homework 1.3 lands on and . You can now try this out for yourself.

Homework 1.4: Coin tossing

1. Find a coin and draw on its two sides 0 and 1 with a marker. Toss it 30 times and write down the outcomes in a table of the following form:

Number of tosses N	1	2	3	4	5	6	7	8	9	...	30
The N-th outcome	1	0	1	0	0	0	1	1	1	...	1

   (The gray outcomes are just an example. Replace them by the outcomes that you got.)

2. Estimate the probability of getting outcome 1 for your coin by using Eq. (1.19).

[3] How good is this estimate? One can show that, on average, the error is of the order of $1/\sqrt{N}$, so quickly goes to zero if we repeat the experiment many times.

3. It is interesting to see how the estimate changes as you increase the number of tosses N. For this, extend your table from part 1 by three more rows so that it looks as follows:

Number of tosses N	1	2	3	4	5	6	7	8	...	30
The N-th outcome	1	0	1	0	0	0	1	1	...	1
Cumulative sum N_1	1	1	2	2	2	2	3	4	...	16
Ratio N_1/N	1	1/2	2/3	2/4	2/5	2/6	3/7	4/8	...	16/30
Its numerical value	1.00	0.50	0.67	0.50	0.40	0.33	0.43	0.50	...	0.53

   The meaning of the rows is as follows: (1) the number N of tosses so far, (2) the outcome of the N-th toss, (3) the sum of the first N outcomes, (4) the estimate of the probability of getting outcome 1 based on the first N tosses, (5) the numerical value of this estimate. Feel free to use *Excel* or a similar program to make this table.

4. Plot the last row of your table as a function of the number of tosses N.

1.4 The QUIRKY simulator

The laws of probability can be somewhat counterintuitive. Fortunately, you can always try to simulate its behavior by using the computer you have at home (or the one in your pocket).[4]

In this course we will use a simulator called QUIRKY. QUIRKY is the little sibling of Quirk, a more featureful simulator developed by Craig Gidney at Google. Since Craig released his code under an open source license, we could adapt his simulator for our needs in this course. One of the best features of QUIRKY is that it runs right in your web browser – no installation required! Simply go to:

https://www.quantum-quest.org/quirky

and click on "Quest 1". Try it now – you can even open QUIRKY on your mobile phone! When you open QUIRKY for the first time, it should look like in Fig. 1.3.

1.4.1 Getting started

Let's go through the interface of QUIRKY step by step. At the top, you find a *menu bar* with some useful commands:

The 'Reset' button allows you to reset QUIRKY and start from scratch. The 'Undo' and 'Redo' buttons are self-explanatory. It is useful to know that you can even undo resetting the simulator. Press 'Share' to get some options for sharing your program with your friends. We will talk about the 'Make $R(r)$' button later.

Below the menu follows the *toolbox* that contains the basic operations that we learned so far:

For example, the first box, $\boxed{\oplus}$, is the NOT operation from §1.2, which sends the $[0]$ state to $[1]$ and vice versa. We will discuss the other two operations momentarily. Luckily you do not have to remember all this – simply hover your mouse over each box to see its description.

Below the toolbox comes the heart of the QUIRKY, the *probabilistic bit*:

[4] To some extent this is true even for quantum computers – but we are getting ahead of ourselves...

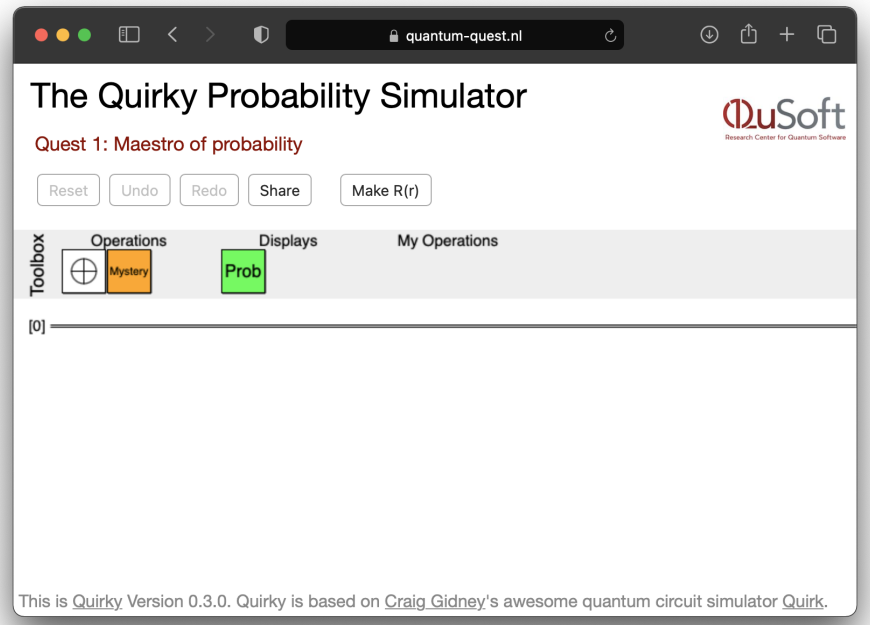


Figure 1.3: QUIRKY opened for the first time.

The double line or 'wire' corresponds to a bit, initialized in the state $[0]$. You can place operations simply by dragging and dropping them from the toolbox onto the wire. Try now to build the following simple computation in QUIRKY:[5]

How can we visualize the result of such a computation? Since we will in general be dealing with probabilities, what we are after is in fact a way of *displaying probabilities*. This is achieved by the green box labeled Prob in the 'Display' section of the toolbox. Let's add it to our computation and see what happens:

It looks like that measurement outcome will be 'one' 100% of the time (hover the box with your mouse to confirm our suspicion). Of course, this is precisely what we expect. The initial $[0]$ gets transformed to a $[1]$ state by the NOT operation, so that the outcome will always be 'one' according to the measurement rules in Eq. (1.18).

Exercise 1.8: Removing an operation

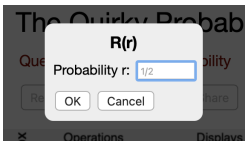
In QUIRKY, you can also remove operations simply by dragging and dropping them away from the wire and back to the toolbox. Remove the NOT operation from the computation, and confirm that the outcome is now 'zero' with certainty.

[5] If you read a digital version of these notes then you can simply click on any of the images to open QUIRKY in your browser. Everything you see in the image will then automatically be included! If this doesn't work just go to https://www.quantum-quest.org/quirky yourself.

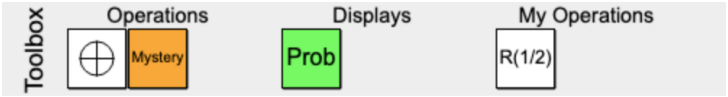
1.4.2 Making your own operations

So far, we only know how to create the $[0]$ and $[1]$ states using QUIRKY. To create an interesting probability distribution, we can use the reset operation $R(r)$ from §1.2.2. Since there are infinitely many such operations (one for each choice of r), we could not add them all to the toolbox. Instead, you can add your own reset operations to the toolbox!

Let's practice by adding an operation that resets with probability $r = \frac{1}{2} = 50\%$. To start, select 'Make $R(r)$' in the menu bar. A new window appears where you can input an angle:



Enter `1/2`, and confirm by pressing the button. Congratulations! You have successfully added the $R(1/2)$ operation to the toolbox, which now looks as follows:



To test our new rotation, let us build the following computation in QUIRKY:

Let's quickly see that this outcome makes sense. We started with the $[0] = \binom{1}{0}$ state. The NOT operation flips the bit into the $[1] = \binom{0}{1}$ state. By Eq. (1.13), the operation $R(1/2)$ resets a bit in state $[1]$ with probability $\frac{1}{2}$, that is, it changes the state to

$$R(1/2)\,[1] = \frac{1}{2}\,[0] + \frac{1}{2}\,[1] = \binom{1/2}{1/2} = \binom{50\%}{50\%}.$$

This is precisely what QUIRKY told us.

In the following exercise you will use QUIRKY to carry out a more complicated experiment.

Homework 1.5: Resetting twice

1. Build the following sequence of operations using QUIRKY: First prepare the state $[1]$, then reset with probability $\frac{1}{4}$, then reset with probability $\frac{2}{3}$. Use the probability display in QUIRKY to determine the probability of the measurement outcomes.

2. Argue that the answer given by QUIRKY is correct.

1.4.3 A mysterious operation

We still have not discussed the mysterious orange box. Let us call this operation M. How can we figure what is going on inside the box? As a first step, let us consider the problem of determining $M\,[0]$, that is, the result of applying the mysterious operation M to a bit in state $[0]$. In QUIRKY, this corresponds to the following setup:

How can we read off $M\,[0]$? At this point it is good to remind ourselves that when random bits appear in nature we *cannot* simply look at them and read off their probabilities. Instead, as explained in §1.3, we have to perform many measurements (e.g., toss a coin many times) and estimate the probabilities from the outcomes. The advantage of using a simulator like QUIRKY is that we do not have to play by these rules – we can use the probability display to determine the state:

Thus, we find that

$$M\,[0] = 0.2\,[0] + 0.8\,[1].$$

Now it is your turn!

Homework 1.6: Mystery time

1. Determine the state $M\,[1]$.
2. Do $M\,[0]$ and $M\,[1]$ specify the random operation M completely?
   If yes, write down a formula for $M\left(\begin{smallmatrix}1/2\\1/2\end{smallmatrix}\right)$ and verify it in QUIRKY. If not, explain why.

In the coming weeks we will make the jump from ordinary bits to quantum bits, and learn how to compute with them in increasingly sophisticated ways. QUIRKY will serve as our trusty tool, gaining new capabilities as we move along. You are warmly encouraged to use it to investigate the theory that you will learn, as well as to help you solve your homework problems.

1.5 Exercise solutions

Solution to Exercise 1.1

1. Since one of the two events has to occur, the two probabilities necessarily add to 1. This means that $p_0 + p_1 = 1$. If you write this as $p_1 = 1 - p_0$ you recognize this as the equation of a line with slope minus one.
2. If the line would go further one of the probabilities would become negative. Since probabilities cannot be negative, we must require that $p_0 \geq 0$ and $p_1 \geq 0$, which is equivalent to saying that the line segment must end at the coordinate axes.
3. This is the midpoint, where $p_0 = p_1 = \frac{1}{2}$.

Solution to Exercise 1.2

1. The counter can have 60 different values. The probability to see any of them is $\frac{1}{60}$.
2. The last digit has 10 different values. The probability to see any of them is $\frac{1}{10}$.
3. The first digit has 6 different values. The probability to see any of them is $\frac{1}{6}$.
4. If you see only the first digit, the last digit could have any of the 10 possible values with equal probability. Similarly, if you see only the last digit, the first digit can have any of the 6 possible values with equal probability. Hence, the values of the two digits are independent. You can verify that the probability to see 00 is indeed $\frac{1}{60}$ by multiplying the probabilities of each digit to be 0:

$$\frac{1}{6} \cdot \frac{1}{10} = \frac{1}{60}.$$

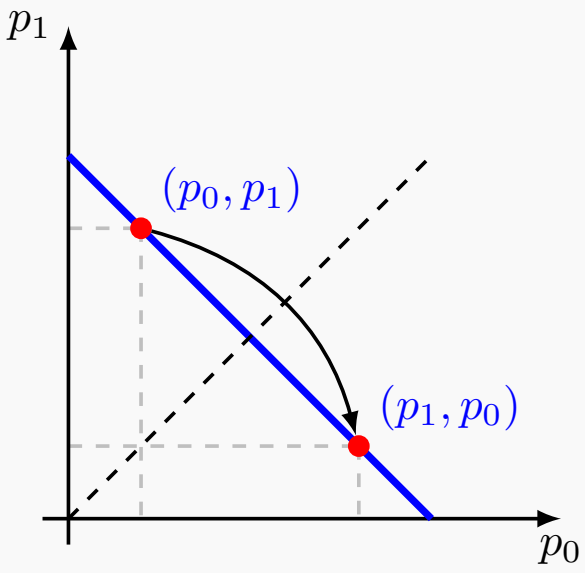
Solution to Exercise 1.3

There are six possible cases when both digits are the same (from 00 to 55). Each of these cases occurs with probability $\frac{1}{60}$. We can group them together in a single event whose probability is the sum of the probabilities of the six individual events:

$$\underbrace{\frac{1}{60} + \frac{1}{60} + \frac{1}{60} + \frac{1}{60} + \frac{1}{60} + \frac{1}{60}}_{\text{6 terms}} = \frac{6}{60} = \frac{1}{10}.$$

Solution to Exercise 1.4

1. Note from Eq. (1.10) that the point (p_0, p_1) is sent to (p_1, p_0). In other words, the two coordinates of the point are exchanged. Here is an example of how this looks like:



   You can thus think of the NOT operation as the reflection around the dashed line that goes right in the middle between the two coordinate axes.

2. According to Eq. (1.2), the two end-points $(1,0)$ and $(0,1)$ of the line segment correspond to the two deterministic states $[0]$ and $[1]$. Recall from Eq. (1.9) that the NOT operation exchanges them.

3. A point with coordinates (p_0, p_1) remains fixed by the NOT operation if $(p_0, p_1) = (p_1, p_0)$, meaning that $p_0 = p_1$. Since $p_0 + p_1 = 1$, we find that $p_0 = p_1 = 1/2$ which corresponds to the point $(1/2, 1/2)$. This is the only point that remains fixed.

Solution to Exercise 1.5

Using Eqs. (1.9) and (1.11),

$$\text{NOT}\begin{pmatrix} p_0 \\ p_1 \end{pmatrix} = p_0\,\text{NOT}\begin{pmatrix} 1 \\ 0 \end{pmatrix} + p_1\,\text{NOT}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = p_0\begin{pmatrix} 0 \\ 1 \end{pmatrix} + p_1\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} p_1 \\ p_0 \end{pmatrix},$$

which is Eq. (1.10).

Solution to Exercise 1.6

1. Using the definition of $F(f)$ in Eq. (1.14),

$$F(f)\,[0] = (1-f)\,\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right) + f\,\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right) = \left(\begin{smallmatrix}1-f\\f\end{smallmatrix}\right),$$
$$F(f)\,[1] = f\,\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right) + (1-f)\,\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right) = \left(\begin{smallmatrix}f\\1-f\end{smallmatrix}\right).$$

2. $F(f)$ flips the bit with certainty when $f = 1$, so NOT $= F(1)$. To prepare an arbitrary state $\left(\begin{smallmatrix}p\\1-p\end{smallmatrix}\right)$ from $[0]$ we need to choose $f = 1-p$. Indeed, we see from the first equation above that

$$F(1-p)\,[0] = \begin{pmatrix}1-(1-p)\\1-p\end{pmatrix} = \begin{pmatrix}p\\1-p\end{pmatrix}.$$

3. Following Eq. (1.11),

$$F(f)\,\begin{pmatrix}p_0\\p_1\end{pmatrix} = p_0\begin{pmatrix}1-f\\f\end{pmatrix} + p_1\begin{pmatrix}f\\1-f\end{pmatrix} = \begin{pmatrix}p_0(1-f)+p_1 f\\p_0 f + p_1(1-f)\end{pmatrix}.$$

4. Substituting $f = 1/2$ in the previous equation we get

$$F(1/2)\,\begin{pmatrix}p_0\\p_1\end{pmatrix} = \begin{pmatrix}p_0/2+p_1/2\\p_0/2+p_1/2\end{pmatrix} = \frac{1}{2}\begin{pmatrix}p_0+p_1\\p_0+p_1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}1\\1\end{pmatrix}.$$

Solution to Exercise 1.7

We distinguish two cases:

- $\frac{1}{2} \le f \le 1$: In this case, $0 \le \frac{1-f}{f} \le 1$. We claim that the flip operation $F(f)$ can be built by first applying $R(\frac{1-f}{f})$, then NOT, and finally $R(1-f)$. Indeed:

$$R(1-f)\,\mathrm{NOT}\,R(\tfrac{1-f}{f})\,[0] = R(1-f)\,\mathrm{NOT}\,[0] = R(1-f)\,[1] = (1-f)\,[0] + f\,[1]$$

and

$$\begin{aligned}R(1-f)\,\mathrm{NOT}\,R(\tfrac{1-f}{f})\,[1] &= R(1-f)\,\mathrm{NOT}\left(\tfrac{1-f}{f}\,[0] + (1-\tfrac{1-f}{f})\,[1]\right)\\ &= R(1-f)\left((1-\tfrac{1-f}{f})\,[0] + \tfrac{1-f}{f}\,[1]\right)\\ &= (1-\tfrac{1-f}{f})\,[0] + \tfrac{1-f}{f}\left((1-f)\,[0] + f\,[1]\right)\\ &= f\,[0] + (1-f)\,[1].\end{aligned}$$

- $0 \le f \le \frac{1}{2}$: This case can be reduced to the first, since $F(f)$ is the same as first applying $F(1-f)$ and then NOT.

Quest 2: Conqueror of the qubit

Now that you have mastered probabilities and probabilistic bits, you are ready to learn about quantum bits. Quantum bits are very similar to probabilistic bits – indeed, all you have to do is replace probabilities by quantum amplitudes. This week you will learn about the states of a quantum bit, the allowed operations, and the measurement for extracting information out of it. You will also have a chance to try out a new *quantum* version of QUIRKY.

2.1 Quantum bits

Bits are the basic units of information in our current-day computers. To realize a bit, you need a physical object that can be in one of two reliably distinguishable states, such as a coin with two sides or a capacitor storing electric charge at two different possible voltage levels.[6] The behavior of such objects (and hence the bits they encode) can be described by physical theories such as mechanics (for coins) or electromagnetism (for capacitors).

However, for really tiny[7] objects these theories no longer apply and you have to use a more fundamental theory called quantum mechanics. For example, an electron has a property called spin, which (just like a coin) can take one of two values – up or down – and hence can be used to store a bit. However, unlike a coin, electron's spin may not be just in one of these two states but also in a "superposition" of both! Intuitively, this is somewhat similar to a probabilistic bit that can also be in an intermediate state between 0 and 1.

However, there is a subtle difference between probabilities and "superpositions" (see §2.6.1 on interference). As we will see, the laws of quantum mechanics lead to a much more fundamental notion of information than a bit – a quantum bit or qubit. To distinguish the usual bits from their more exotic quantum friends, we will call the usual bits classical.

We will define quantum bits using a simple mathematical model and not worry about how their strange behavior should be interpreted but instead ask the question: "What can it be used for?". Similarly, we will also not worry about how they can be implemented physically or what kind of physical objects can be used to store them. However, if you are curious about this, we briefly discuss in §2.6.2 how polarization of light can be used to represent a qubit.

2.1.1 Probabilities versus amplitudes

Quantum bits are very similar to probabilistic bits. There are only two major differences:

1. probabilities are replaced by amplitudes (which can also be negative),
2. amplitudes are squared during the measurement (while probabilities are not).

We will explain these differences in more detail shortly, but let us first describe the possible states of a qubit. Recall how we used the two sides of a coin to denote the two possible deterministic states of a probabilistic bit (see Fig. 1.1)? In quantum computing, these two states are commonly denoted by $|0\rangle$ and $|1\rangle$ to distinguish them from the classical bits $[0]$ and $[1]$. Just like with probabilistic bits, a general qubit state $|\psi\rangle$ is then a linear combination or superposition of these two deterministic states:

$$|\psi\rangle = \psi_0 |0\rangle + \psi_1 |1\rangle . \tag{2.1}$$

[6] This is essentially the way that bits are represented in your computer, mobile phone, etc.

[7] By "really tiny" we mean *really, really tiny*! If you would put electrons next to each other in a line, the number of electrons you would need to reach the length of 1 cm is similar to the number of pages you would need to put on top of each other to reach the Moon.

Here, the greek letter ψ (pronounce "psi") is the name of the qubit state (just like we named the probabilistic bit p). The brackets $|\cdot\rangle$ form a so-called "ket" that indicates that we are dealing with a quantum state. For comparison, recall from Eq. (1.3) that an arbitrary probabilistic bit p can be written as

$$p = p_0[0] + p_1[1]. \tag{2.2}$$

Note that Eq. (2.1) looks identical to this, except the probabilities p_0 and p_1 are replaced by the amplitudes ψ_0 and ψ_1, and the classical notation $[0]$ and $[1]$ is replaced by the quantum notation $|0\rangle$ and $|1\rangle$! However, there is one major difference: while the probabilities in Eq. (2.2) are subject to

$$p_0, p_1 \geq 0 \qquad \text{and} \qquad p_0 + p_1 = 1, \tag{2.3}$$

the amplitudes are subject to

$$\psi_0^2 + \psi_1^2 = 1. \tag{2.4}$$

In particular, this implies $\psi_0^2 \leq 1$ and $\psi_1^2 \leq 1$, and hence $\psi_0, \psi_1 \in [-1, 1]$. In contrast, the constraints from Eq. (2.3) on probabilities imply that $p_0, p_1 \in [0, 1]$. The crucial difference is that amplitudes are actually allowed to be negative while probabilities are not![8]

Just like with probabilistic bits, it is convenient to represent qubit states by vectors. In complete analogy with Eq. (1.2), we represent the deterministic qubit states $|0\rangle$ and $|1\rangle$ by the two basis vectors:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

A general quantum state $|\psi\rangle$ from Eq. (2.1) is then represented as

$$|\psi\rangle = \psi_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \psi_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \psi_0 \\ \psi_1 \end{pmatrix}.$$

2.1.2 Qubit as a circle

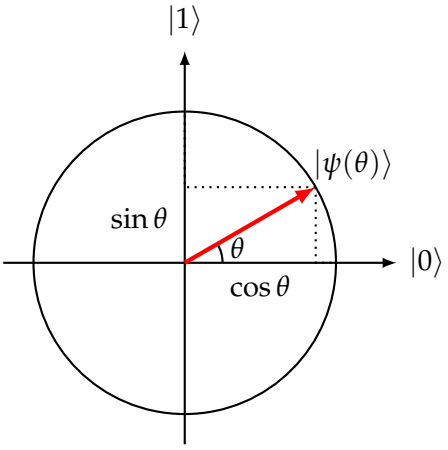


Figure 2.1: Qubit state $|\psi(\theta)\rangle$ as a point on the unit circle.

Notice how Eq. (2.4) for qubit amplitudes is reminiscent of the equation $x^2 + y^2 = 1$ of a circle. Let us work out this correspondence in more detail because it will help us to visualize quantum bits and to understand them on a more intuitive level.

A convenient way to parametrize the qubit amplitudes is by letting

[8] In fact, amplitudes are even allowed to be so-called *complex numbers*. We will not need them in this course, but you're encouraged to browse the web to learn more about this.

$$\psi_0 = \cos\theta, \qquad \psi_1 = \sin\theta$$

for some angle $\theta \in [0, 2\pi)$. In fact, it will often be convenient to allow the angle θ to be an arbitrary real number (which is fine provided we keep in mind that any two angles that differ by 2π result in the same amplitudes). Since $\cos^2\theta + \sin^2\theta = 1$, we are guaranteed to automatically satisfy Eq. (2.4). With this choice, a general qubit state looks as follows:

$$|\psi(\theta)\rangle = \cos\theta\,|0\rangle + \sin\theta\,|1\rangle = \begin{pmatrix}\cos\theta\\ \sin\theta\end{pmatrix}. \tag{2.5}$$

One can visualize this as a unit vector in two dimensions that starts at the origin of the plane and has angle θ with the horizontal $|0\rangle$ axis (see Fig. 2.1). In particular, $|0\rangle = |\psi(0)\rangle$ and $|1\rangle = |\psi(\frac{\pi}{2})\rangle$. The set of all qubit states then correspond to a unit circle centered at the origin. In contrast, recall from Fig. 1.2 that the set of all states of a probabilistic bit is a line segment connecting the points $\binom{1}{0}$ and $\binom{0}{1}$ located on the two coordinate axes. The two sets are compared in Fig. 2.2.

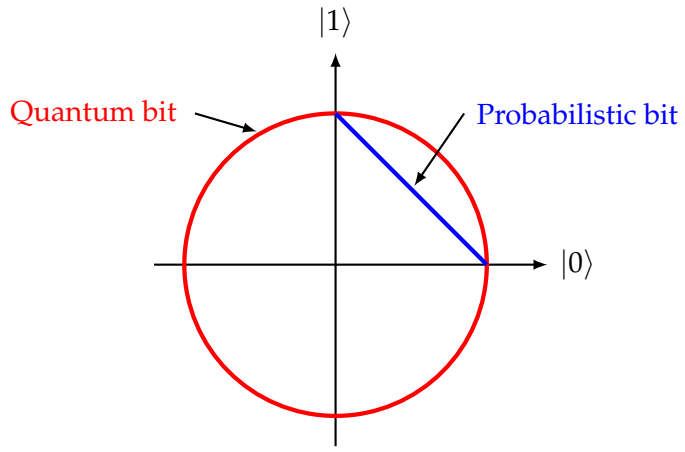


Figure 2.2: State spaces of a probabilistic bit (blue) and quantum bit (red).

Exercise 2.1: States on the circle

Consider the following two states of a qubit:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \qquad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

Where do these two states lie on the circle? What angles θ do they correspond to?

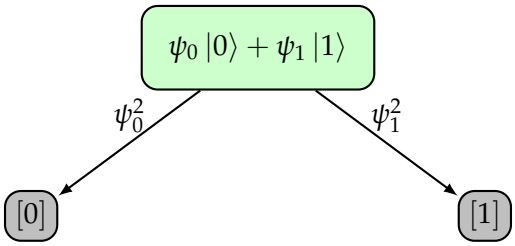
2.2 Measuring a quantum bit

We now know that any qubit state is of the form $|\psi(\theta)\rangle$. Let's say you get your hands on a state $|\psi(\theta)\rangle$ and you would like to know the value of θ. Unfortunately, quantum mechanics does not allow you to learn it! This seems like a big problem – what is a quantum computer good for if you cannot get out the answer? Well, not so fast! Recall from Eq. (1.18) that the same was true for probabilistic bits as well – if you measure a probabilistic bit with distribution $\binom{p_0}{p_1}$, you do not learn p_0 or p_1. All you get is a single bit of information: 0 with probability p_0 and 1 with probability p_1.

The quantum measurement is very similar and is described by what is known as *Born rule*. If you have a qubit in state $\binom{\psi_0}{\psi_1} = \psi_0\,|0\rangle + \psi_1\,|1\rangle$ and you measure it, you also obtain just a single bit: you get 0 or 1 with probabilities

$$p_0 = \psi_0^2, \qquad p_1 = \psi_1^2. \tag{2.6}$$

While the square might seem surprising, note that $p_0 + p_1 = \psi_0^2 + \psi_1^2 = 1$. Thus, the square is precisely what guarantees that $\binom{p_0}{p_1}$ is a valid probability distribution, so the above rule makes sense! After the measurement, the qubit is gone and all you are left with is a single bit containing the measurement outcome you observed. In other words, the measurement process converts a qubit into a regular bit whose value is determined probabilistically by Eq. (2.6):

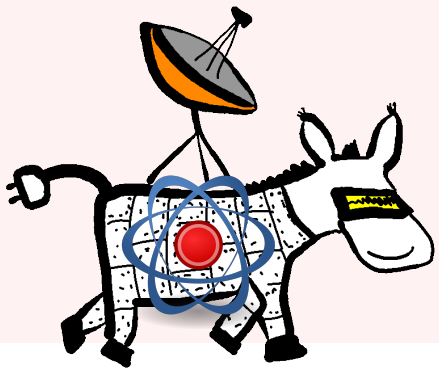


(2.7)

As you can see from Eqs. (1.18) and (2.7), the measurement rule for probabilistic bits and qubits is very similar. In both cases, the original state is gone and all you are left with is a single bit whose value depends probabilistically on the original state you measured. (In particular, if you measure the state more than once then you will always get the same outcome as the first time – so repeated measurements do not give any additional information about what the original state was.) The only difference is that for qubits you need to square the amplitudes to get the probabilities, as in Eq. (2.6), while for probabilistic bits you get them directly and hence don't need to square anything. While this may seem like a small difference, it does have significant impact on the allowed states, as the amplitudes of a qubit are allowed to be negative whilst the probabilities of a probabilistic bit are always positive (see Fig. 2.2).

Well, actually there is another, even more subtle difference. Namely, that nobody can predict the outcome of a quantum measurement in advance. This is subtle because it seems that the same should be true also for probabilistic bits. What is the difference? In short, the answer is that probabilistic bits appear random because of our lack of knowledge about their state, while quantum bits are random even if we know all there is to know about their state. For example, imagine that your friend tosses a fair coin and immediately covers it once it lands. You would normally describe the state of such coin as uniformly random, see Eq. (1.15). However, if you were filming the coin with a high-speed camera, you might be able to accurately predict on which side it landed from your footage. In this sense, the randomness of probabilistic bits has to do with our ignorance. For quantum bits, however, randomness arises on a more fundamental level. Whatever prior knowledge we may have, it is in general *impossible* to perfectly predict the outcome of a quantum measurement. On the flipside, this means that the outcomes of quantum measurements can be used as a good source of randomness!

Homework 2.1: Generating a random bit quantumly



Problem: Alice's donkey robot is running low on power again and needs to find its way to a charging station. Unfortunately, this time Eve's hacking skills have improved – she has figured out how to hack the donkey's random number generator and reprogram it so that it generates any sequence of numbers she wants! Luckily, Alice is aware of this since Eve bragged about it on a hacker forum recently. To counteract Eve's evil plan, Alice decided to install a miniature single-

qubit quantum computer inside her donkey robot. Using the inherent unpredictability of quantum measurement outcomes, Alice wants to generate uniformly random bits that Eve cannot guess.

Question: Alice is able to produce any qubit state $|\psi(\theta)\rangle$, and she wants to generate a uniformly random bit by measuring it.

1. When measuring the state $|\psi(\theta)\rangle$, what is the probability to get measurement outcome 0? What is the probability for measurement outcome 1?
2. Alice wants to find an angle θ such that both probabilities equal $1/2$. What θ should she choose? (There might be more than one option!)

2.3 Simulating quantum bits with QUIRKY

The laws of quantum computing are quite strange and most of us don't have a quantum computer to experiment with. Fortunately, QUIRKY has gained new powers since last week and now allows us to simulate a *quantum* bit![9] To begin, go to:

https://www.quantum-quest.org/quirky

and click on "Quest 2". Your web browser will look similarly to Fig. 2.3.

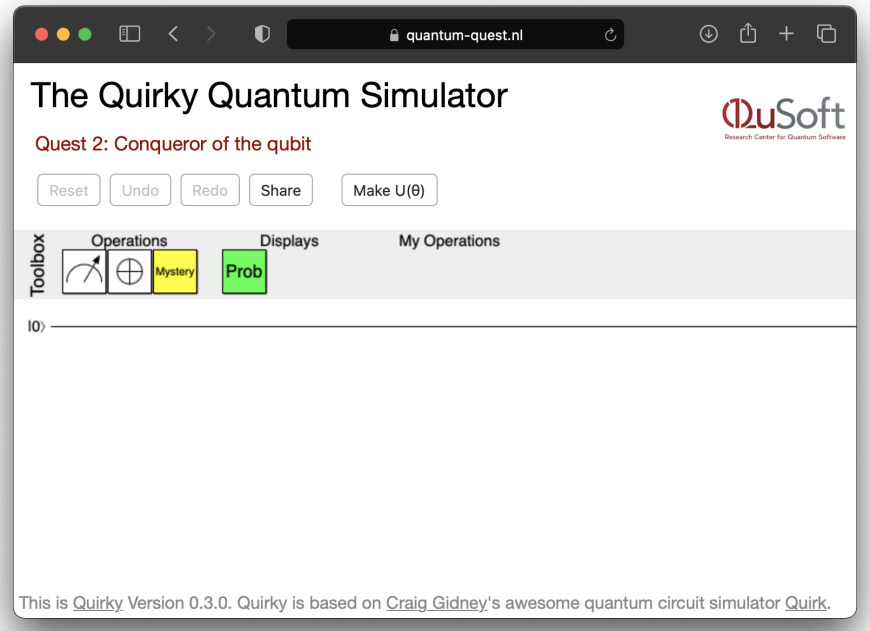


Figure 2.3: QUIRKY for Quest 2.

The key difference to last week that the 'wire' now corresponds to a *quantum bit*, which is initialized in the state $|0\rangle$.

[9] Why do we want to build quantum computers at all if we can simulate them so nicely on existing computers? The reason is that while simulators like QUIRKY work well when all you have is a handful of of quantum bits, they quickly break down as the number of quantum bits increases. We will see why this is so on page 74 in §4.

Just like last week, the *toolbox* contains operations that we can apply by dragging and dropping them from the toolbox onto the wire:

The first box, , allows us to measure a quantum bit. Let us go ahead and build the following simple quantum computation in QUIRKY:

You will notice that upon the single line turned into a double line. In QUIRKY, single lines refer to quantum bits, and double lines refer to ordinary or 'classical' bits. Indeed, we know from §2.2 that when we measure a quantum bit we get an outcome that is either zero or one with some probabilities, that is, a probabilistic bit.

To view the probabilities of outcomes, we can use the probability display Prob that we already know from last week. Let's add it to our computation and see what happens:

It looks like that the measurement outcome will be 'zero' 100% of the time (hover the box with your mouse to confirm our suspicion). Of course, this is precisely what we expect. When we measure $|0\rangle$, the outcome will always be 'zero' according to the measurement rules in Eq. (2.7).

In the remainder of this chapter we will discuss the other boxes in the toolbox.

2.4 Operations on a quantum bit

Before we measure a state we might want to do some operation on it. But what kind of operations can we do on a qubit? For example, when we start our quantum computer, its quantum bit will always be in state $|0\rangle$, so we need to apply an operation to create some interesting state $|\psi(\theta)\rangle$. Whatever the operation is, it should produce another qubit state as an output. In other words, it should map the qubit state space to itself. Recall from Fig. 2.1 that this state space corresponds to a circle, so we are looking for ways of mapping the circle to itself.

Let us first consider the NOT operation, which we can define in exactly the same way as in Eq. (1.9) for probabilistic bits:

$$\text{NOT}\,|0\rangle = |1\rangle\,, \qquad \text{NOT}\,|1\rangle = |0\rangle\,.$$

How can we extend NOT to arbitrary qubit states? Just like we did for probabilistic bits in §1.2.1, we will use the idea of linearity. If an operation M is defined on $|0\rangle$ and $|1\rangle$ then we can define it on an arbitrary qubit state by

$$M\Big(\psi_0\,|0\rangle + \psi_1\,|1\rangle\Big) = \psi_0\,M\,|0\rangle + \psi_1\,M\,|1\rangle\,. \tag{2.8}$$

We can also write out Eq. (2.8) explicitly using the vector notation:

$$M\begin{pmatrix}\psi_0\\ \psi_1\end{pmatrix} = M\left(\psi_0\begin{pmatrix}1\\0\end{pmatrix} + \psi_1\begin{pmatrix}0\\1\end{pmatrix}\right) = \psi_0\,M\begin{pmatrix}1\\0\end{pmatrix} + \psi_1\,M\begin{pmatrix}0\\1\end{pmatrix}. \tag{2.9}$$

As mentioned earlier, in mathematics an operation M that satisfies this condition is called linear, and extending an operation in this way is called extending "by linearity". The key point is that

if M is linear and we know how it acts on $|0\rangle$ and on $|1\rangle$, we can deduce how it acts on arbitrary qubit states!

In Eq. (2.8), we only considered the vectors $|0\rangle$ and $|1\rangle$. However it is more generally true that

$$M\Big(a\,|\psi\rangle + b\,|\phi\rangle\Big) = a\,M\,|\psi\rangle + b\,M\,|\phi\rangle \tag{2.10}$$

for arbitrary vectors $|\psi\rangle$, $|\phi\rangle$ and numbers a, b. Can you see how (2.10) follows from (2.8)?

The laws of quantum mechanics guarantee that any linear operation M is a possible qubit operation – provided it sends the entire qubit state space to itself! By this we mean that every qubit state (point on the circle) is mapped to some qubit state (point on the circle).

In the case of the NOT operation, the result of extending by linearity is

$$\text{NOT}\Big(\psi_0\,|0\rangle + \psi_1\,|1\rangle\Big) = \psi_0\,|1\rangle + \psi_1\,|0\rangle\,, \quad \text{or} \quad \text{NOT}\begin{pmatrix}\psi_0\\ \psi_1\end{pmatrix} = \begin{pmatrix}\psi_1\\ \psi_0\end{pmatrix}. \tag{2.11}$$

Note that Eqs. (2.8), (2.9) and (2.11) look exactly like Eqs. (1.10) and (1.12) – except that now ψ_0 and ψ_1 can also be negative. In terms of Fig. 2.2, the NOT operation amounts to a *reflection* about the 45 degree axis (this is true also for probabilistic bits). This is visualized in Fig. 2.4. Clearly, NOT maps the qubit state space (circle) to itself. Thus, the NOT operation is a valid operation on a quantum bit.

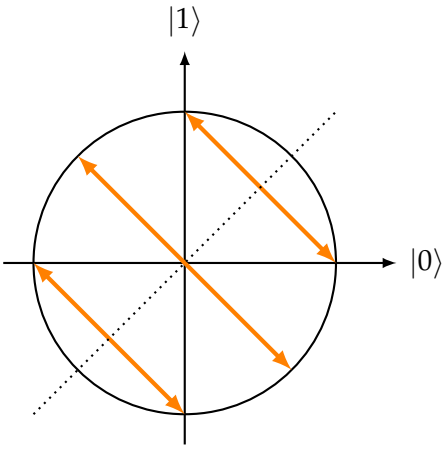


Figure 2.4: The NOT operation on a qubit, defined in Eq. (2.11), amounts to a reflection about the 45 degree (or $\pi/4$) axis (dotted).

In QUIRKY, the NOT operation on qubits looks like just like the NOT operation on bits, namely $\boxed{\oplus}$. Try now to build the following quantum computation:

Now it looks like the measurement outcome will be *'one'* 100% of the time. Indeed, the initial $|0\rangle$ gets transformed to a $|1\rangle$ state by the NOT operation, so that the outcome will always be 'one' according to the measurement rules in Eq. (2.7).

We can similarly define qubit operations by considering reflections through other axes. For example, the Z operation defined by

$$Z\,|0\rangle = |0\rangle\,, \qquad Z\,|1\rangle = -\,|1\rangle\,, \tag{2.12}$$

corresponds to a reflection about the horizontal $|0\rangle$-axis. Indeed, if we extend Z by linearity then it acts on an arbitrary qubit state as

$$Z\begin{pmatrix}\psi_0\\ \psi_1\end{pmatrix} = \begin{pmatrix}\psi_0\\ -\psi_1\end{pmatrix},$$

which certainly maps qubit states to qubit states.

Homework 2.2: Z operation

Consider the following two states of a qubit:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \qquad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

1. Compute $Z|+\rangle$ and $Z|-\rangle$.
2. Visualize the Z-operation graphically on the circle, as in Fig. 2.4.

Exercise 2.2: Linearity is not enough (optional)

Consider the operation MAD obtained by extending $\text{MAD}|0\rangle = |0\rangle$ and $\text{MAD}|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ by linearity. Find a state $|\psi\rangle$ such that $\text{MAD}|\psi\rangle$ is not a valid qubit state. Thus, MAD is *not* a valid operation on qubits!

2.4.1 Rotations

So far, we only know how to create the $|0\rangle$ and $|1\rangle$ states using QUIRKY. Quantum computing would not be much fun if those were our only options! To create more interesting states, we need to come up with other quantum operations.

One natural such operation is to *rotate* the circle by some fixed angle. Let us denote the rotation by an angle θ by $U(\theta)$. You can always assume that the angle is in $[0, 2\pi)$. Since $|0\rangle = |\psi(0)\rangle$ and $|1\rangle = |\psi(\frac{\pi}{2})\rangle$, this operation acts on the basis vectors as follows (see Fig. 2.5):

$$U(\theta)|0\rangle = |\psi(\theta)\rangle, \qquad U(\theta)|1\rangle = |\psi(\theta + \tfrac{\pi}{2})\rangle. \tag{2.13}$$

We can also write out this definition explicitly in the vector notation:

$$U(\theta)\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}\cos\theta\\ \sin\theta\end{pmatrix}, \qquad U(\theta)\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}-\sin\theta\\ \cos\theta\end{pmatrix}, \tag{2.14}$$

where we used $\cos(\theta + \frac{\pi}{2}) = -\sin\theta$ and $\sin(\theta + \frac{\pi}{2}) = \cos\theta$.

As before, we will use linearity to extend $U(\theta)$ from the basis vectors to arbitrary qubit states. In the following exercise, you will show that the resulting operation $U(\theta)$ indeed acts as a rotation on qubit states. In particular, this means that it sends qubit states to qubit states, so $U(\theta)$ is an allowed operation on qubits!

Exercise 2.3: Qubit rotation

1. Compute $U(\alpha)\begin{pmatrix}\psi_0\\ \psi_1\end{pmatrix}$ by using Eqs. (2.8) and (2.13).
2. Use the definition of $|\psi(\theta)\rangle$ in Eq. (2.5) to verify that, for all angles α and β,

$$U(\alpha)|\psi(\beta)\rangle = |\psi(\alpha + \beta)\rangle. \tag{2.15}$$

This means that $U(\theta)$ acts as a rotation on arbitrary qubit states $|\psi(\beta)\rangle$.

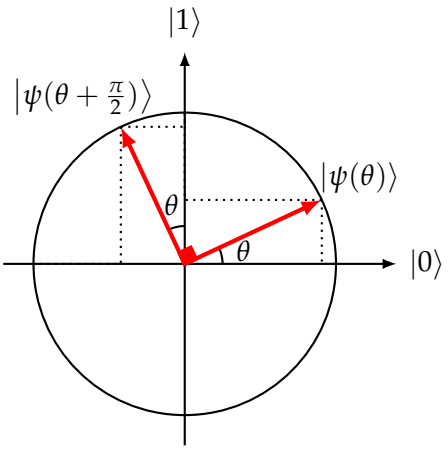


Figure 2.5: States $|0\rangle$ and $|1\rangle$ rotated by angle θ, see Eq. (2.13).

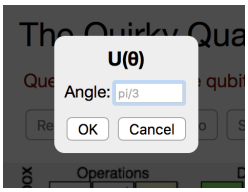
Hint: The trigonometric angle sum and difference formulas might be useful:

$$\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta, \qquad \cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta. \tag{2.16}$$

Note that a rotation by 90 degrees (i.e., $\pi/2$) is not the same as a reflection. Indeed, while both the NOT operation and the $U(\pi/2)$ rotation send $|0\rangle$ to $|1\rangle$, they act differently on $|1\rangle$:

$$\text{NOT}\,|1\rangle = |0\rangle\,, \qquad U(\pi/2)\,|1\rangle = -\,|0\rangle\,.$$

How can we rotate a quantum bit in QUIRKY? Since there are infinitely many rotations operations $U(\theta)$, we could not add them all to the toolbox. Instead, you can add your own rotations to the toolbox! Let's practice by adding a rotation by 30°. To start, select 'Make $U(\theta)$' in the menu bar. A new window appears where you can input an angle:



Enter `pi/6`, which corresponds to 30°, and confirm by pressing the button. Congratulations! You have successfully added the $U(\pi/6)$ rotation to the toolbox, which now looks as follows:

To test our new rotation, let us build the following computation in QUIRKY:

Let's quickly see that this outcome makes sense. We started with the $|0\rangle = \binom{1}{0}$ state. By Eq. (2.14), any rotation $U(\theta)$ sends $|0\rangle$ to $|\psi(\theta)\rangle = \binom{\cos(\theta)}{\sin(\theta)}$. In our case $\theta = \pi/6$, and

$$|\psi(\pi/6)\rangle = \begin{pmatrix}\cos(\pi/6)\\ \sin(\pi/6)\end{pmatrix} = \begin{pmatrix}\sqrt{3}/2\\ 1/2\end{pmatrix}.$$

Using the quantum measurement rule in Eq. (2.7), we conclude that the probability of obtaining outcome 1 is

$$p_1 = \left(\frac{1}{2}\right)^2 = \frac{1}{4} = 25\%,$$

which is exactly what QUIRKY told us. In the following exercise you will use QUIRKY to similarly test the effect of the rotation $U(\theta)$ on the other basis vector, $|1\rangle$.

Homework 2.3: Testing the 30° rotation

1. Build the following sequence of operations using QUIRKY: First prepare the qubit state $|1\rangle$, then rotate by the same angle $\pi/6$, and finally measure the qubit.
2. Use the probability display in QUIRKY to determine the probability of the measurement outcomes. Argue that the answer given by QUIRKY is correct.
3. Modify your solution to the first question such that the probability of measurement outcome 1 is 42 percent.

2.4.2 Composing quantum operations

We can always compose two given qubit operations M and N to obtain a new qubit operation. Indeed, if $|\psi\rangle$ is the input state and we first apply M, we get $M(|\psi\rangle) = M|\psi\rangle$. If we then apply N, resulting state is $N(M|\psi\rangle)$. We will denote this composite operation by NM, so that

$$NM|\psi\rangle = N(M|\psi\rangle).$$

Be careful not to confuse the order of the two operations. If the composite operation is NM, this means that M is applied first and N is applied second! This is because M stands next to $|\psi\rangle$, so it must be the first one to act on the state.

Exercise 2.4: Linearity of a composed operation (optional)

Verify that NM is again linear.

Hint: Use Eq. (2.10).

We can similarly compose three and more qubit operations. We will write ONM and so on. In particular, we may obtain new qubit operations by composing rotations and reflections. We will discuss this in §2.4.3 below.

It is interesting to observe that all qubit operations that we discussed so far are invertible. This means that for any operation M there exists another operation, which we write as M^{-1}, such that when we first apply M and then M^{-1} (or the other way around) the state of the qubit is unchanged. [10] In formulas, we can write

$$M^{-1}M = MM^{-1} = I, \tag{2.17}$$

where I is the identity operation which has the "trivial" property

$$I|0\rangle = |0\rangle, \qquad I|1\rangle = |1\rangle \tag{2.18}$$

(We could have also defined I as $U(0)$, the rotation by an angle of zero.) Therefore $I|\psi\rangle = |\psi\rangle$ holds for any state $|\psi\rangle$ when extended by linearity.

[10] This notation and Eq. (2.17) will remind you of the following: If x is a nonzero number then $x^{-1} = \frac{1}{x}$ is its inverse, which means that $xx^{-1} = x^{-1}x = 1$.

As an example, let's look at the operation U. It is geometrically clear that if we first rotate by β and then by $-\beta$ then the quantum bit is unchanged. To see this more formally, we only need to use Eq. (2.15) twice:

$$U(-\beta)U(\beta)\,|\psi(\alpha)\rangle = U(-\beta)\,|\psi(\alpha+\beta)\rangle = |\psi(\alpha+\beta-\beta)\rangle = |\psi(\alpha)\rangle$$

and similarly if we first rotate by $-\beta$ and then by β. This means that the inverse operation $U(\beta)^{-1}$ is simply $U(-\beta)$:

$$U(\beta)^{-1} = U(-\beta).$$

Similarly, since the NOT operation amounts to a reflection, it is clear that applying it twice leaves the qubit state invariant. Indeed, from Eq. (1.9)

$$\text{NOT}\,\text{NOT}\,|0\rangle = \text{NOT}\,|1\rangle = |0\rangle \quad \text{and} \quad \text{NOT}\,\text{NOT}\,|1\rangle = \text{NOT}\,|0\rangle = |1\rangle\,.$$

By linearity, this means that $\text{NOT}\,\text{NOT}\,|\psi\rangle = |\psi\rangle$ for any state $|\psi\rangle$, so NOT is not only invertible but its own inverse, i.e., $\text{NOT}^{-1} = \text{NOT}$.

Exercise 2.5: Inverse of a composed operation

Show that if M and N are invertible then so is NM. Express the inverse $(NM)^{-1}$ of the composed operation in terms of the individual inverses N^{-1} and M^{-1}.

In fact, one can show that any linear operation that sends the qubit state space to itself is necessarily invertible. Indeed, this is the case for rotations $U(\theta)$ and will also be the case for reflections $V(\theta)$ that we will discuss next. This is in contrast to operations on probabilistic bits where, for example, the probabilistic flip operation $F(1/2)$ maps any state to the uniform distribution $\binom{1/2}{1/2}$ (see Exercise 1.6) and hence is not invertible.

2.4.3 Reflections

Any qubit operation is either a rotation or a reflection. We are already familiar with the most general rotation, $U(\theta)$, defined in Eq. (2.13). However, in terms of reflections we have encountered only two of them so far: Z and NOT, see Eqs. (2.11) and (2.12). But what does the most general reflection look like?

One way of obtaining any reflection is by taking some fixed reflection (say, the NOT reflection) and composing it with suitable rotations so that the axis of the reflection is adjusted by the right amount. In the following exercise, you will show how to obtain the Z reflection from the NOT reflection in two different ways.

Homework 2.4: Z from NOT

Let Z, NOT, and $U(\theta)$ be the qubit operations defined in Eqs. (2.11) to (2.13).

1. Find an angle θ such that $Z = U(\theta)\,\text{NOT}\,U(-\theta)$.
2. Find an angle θ such that $Z = \text{NOT}\,U(\theta)$.

Can you visualize these two sequences of transformations on the circle?

Hint: Take a look at Fig. 2.4 and the figure you drew for Homework 2.2.

It turns out that you can in fact obtain *any* reflection by using a similar trick. The most general reflection is of the form

$$V(\theta) = \text{NOT}\,U(\theta) = U(-\theta)\,\text{NOT}. \tag{2.19}$$

For example, a very useful operation is the Hadamard transformation that acts on the basis states as follows (see Fig. 2.6):

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle\,, \qquad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle\,. \tag{2.20}$$

It is obtained as the following special case of the general reflection:

$$H = V(\pi/4). \tag{2.21}$$

In summary, any qubit operation is either a rotation $U(\theta)$ or a reflection $V(\theta)$, for some angle θ.

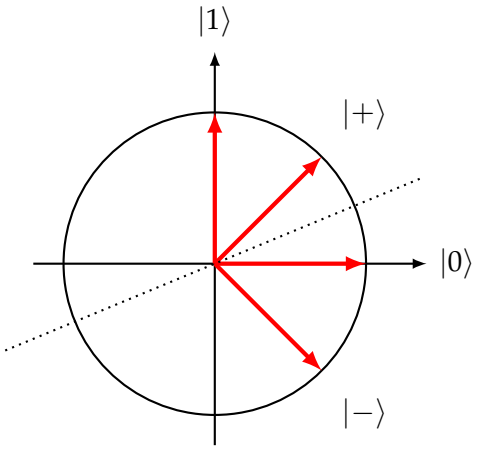


Figure 2.6: The Hadamard operation H on a qubit amounts to a reflection about the 45/2 degree (or $\pi/8$) axis (dotted). Depicted are also the states $|0\rangle$, $|1\rangle$, $|+\rangle$, and $|-\rangle$ from Eq. (2.20).

2.5 Distinguishing quantum states

Alice is watching a running competition for robot donkeys, and she notes down whether her favourite donkey wins: a 1 when this happens, and a 0 otherwise. She could also encode this information into a qubit: in the most general case, she prepares the state $|\psi(\theta_0)\rangle$ in situation 0 (no win), or a state $|\psi(\theta_1)\rangle$ in the case 1 (her favourite donkey wins). Alice can make these states by simply applying $U(\theta_0)$ or $U(\theta_1)$ to $|0\rangle$, as in Eq. (2.13). Now, assume Alice gives this qubit to Bob. Can Bob guess what bit value (0 or 1) was encoded, based purely on this one qubit? Would Bob's odds improve if he can first perform a rotation or reflection? You can practice this idea in the following exercise.

Exercise 2.6: Plus and minus

Imagine that you are given a qubit that is one of the following two states:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \qquad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

You want to guess which state it is. You can apply some rotation and then measure. What rotation should you apply and with what probability can you guess the correct state?

If you want to represent different bit values by different quantum states, you should be careful to not use $|\psi(\theta)\rangle$ and $|\psi(\theta+\pi)\rangle$ since these states cannot be distinguished.

Exercise 2.7: Indistinguishable states

Show that the two states $|\psi(\theta)\rangle$ and $|\psi(\theta+\pi)\rangle = -|\psi(\theta)\rangle$ cannot be distinguished in any way. That is, no matter what qubit operation you apply and then measure the resulting state, the measurement outcomes in both cases will always have the same probabilities.

It is quite interesting to compare Exercises 2.6 and 2.7. When two states differ by an overall minus sign, they are completely indistinguishable, as in Exercise 2.7. For all practical purposes, the two vectors $\pm|\psi(\theta)\rangle$ describe the *same* state. In contrast, 'relative' minus signs as in Exercise 2.6 are important and can even lead to states that can be perfectly distinguished!

In the following homework problem you can figure out the optimal way of distinguishing two *arbitrary* quantum states.

Homework 2.5: Telling two states apart

Let θ, θ' be two angles. For simplicity, assume that $-\frac{\pi}{2} \leq \theta \leq \theta' \leq \frac{\pi}{2}$. Suppose that Eve hands you a single qubit, either in state $|\psi(\theta)\rangle$ or in state $|\psi(\theta')\rangle$, with 50% probability each. (For example, she could toss a fair coin to decide which of the two states to give to you.) Your task is to identify which of the two states you received. In a few steps you will find an optimal procedure:

1. First apply a rotation $U(\phi)$ by some angle ϕ. Which two possible states will you obtain?

2. Next, measure the qubit and interpret the outcome as follows: If the outcome is 0 then your guess is that you were handed the state $|\psi(\theta)\rangle$, otherwise your guess is $|\psi(\theta')\rangle$. What is the probability of correctly identifying the state that was given to you? Write down a formula in terms of θ, θ' and ϕ.

   Hint: First compute the success probability assuming you are given the first state, then the success probability assuming you are given the second state, and then remember that in reality you get one of the two states with 50% probability each.

3. You still have the freedom of choosing the rotation angle ϕ in a clever way. What is the success probability as a function of θ and θ' if you choose ϕ optimally?

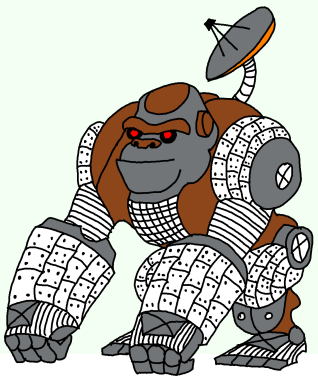
   Hint: Try using the trigonometric identities from Eq. (2.16). In particular, from these you can show that

$$\sin^2\alpha = \frac{1}{2}\big(1-\cos(2\alpha)\big), \qquad \cos^2\alpha = \frac{1}{2}\big(1+\cos(2\alpha)\big). \tag{2.22}$$

   If you are stuck, you can also use Wolfram Alpha.

Exercise 2.8: Broken leg and arm (challenging)

Problem: Alice and Bob like to explore the wilderness surrounding their town. For this purpose they have built two large gorilla robots that can navigate through rough terrain while carrying them comfortably on their backs. This is not a good day for Bob since his robot accidentally falls off a cliff! Luckily, Bob survives the fall with only a few bruises, but his robot gets damaged pretty badly: one arm, one leg, and its communications device are all broken. Bob does not have any spare parts for legs and arms, but at least he manages to fix his communications



device for a brief moment. Unfortunately, it can send only one bit or one qubit before it stops working. Bob would like to communicate to Alice which leg (left or right) and which arm (also, left or right) of his robot is broken so that she could take off the corresponding one from her robot and send it down to him. Alice can send him just one limb (either leg or arm) because both robots need to be able to walk back home (which they can still do on three limbs). The situation is made more complicated by the fact that Alice does not have all the required tools for removing an arbitrary limb from her robot. Bob recalls that Alice took with her either the tools for legs or for arms (and not both), but he cannot remember which.

There are four possible combinations of which leg and which arm of Bob's robot broke – you can assume that each of them happened with probability $1/4$. Similarly, there are two types of limbs Alice can remove from her robot (she has either tools for removing legs or arms) and you can assume that she took the right tools for each with probability $1/2$.

Questions:

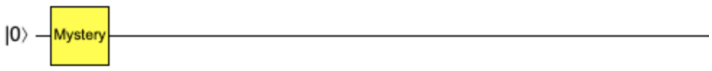
1. If Bob can send only one bit to Alice, how should he decide on its value depending on which of the four ways his robot could be broken? How should Alice interpret his message and decide on whether to send the left or the right limb? (Recall that Alice can send only legs or only arms, and Bob does not know whether it is legs or arms.) If they both use an optimal strategy, with what probability will Alice interpret Bob's message correctly and send the right limb for his robot?

2. What if Bob can instead send one quantum bit? Depending on his situation, he can choose one of four states and, depending on her situation, Alice can apply one of two rotations before measuring it. What is their optimal joint strategy and with what probability does it succeed?

You can assume that Alice and Bob know how to interpret each other's messages, since they have discussed beforehand what to do if this particular emergency situation ever happens.

2.5.1 Another mysterious operation

We still have not discussed the yellow box in QUIRKY. Unlike last week's mystery operation, which operated on bits, this week's mystery box operates on quantum bits. Let us call this mysterious quantum operation M. How can we figure what is going on inside the box? As a first step, let us consider the problem of determining $M|0\rangle$. In QUIRKY, we can create this state by the following setup:



The problem of determining an unknown state is called quantum state tomography, since we want to reconstruct an unknown quantum state 'from the outside', by performing various measurements. It is a fundamental task that experimentalists are faced with every day, when they want to make sure that quantum state that they prepared in the laboratory is the state that they intended to create!

We can already get substantial information by performing a measurement operation on the unknown state. To see this, let us write

$$M|0\rangle = \begin{pmatrix} \psi_0 \\ \psi_1 \end{pmatrix}.$$

If we perform a measurement then by Eq. (2.6) we obtain outcome 1 with probability ψ_1^2. What this means is that if we repeat the above experiment a large number of times then we would expect that the fraction of times that we obtain outcome 1 is roughly ψ_1^2. This is completely analogously to how one can estimate a coin by tossing it many times and counting the number of heads and tails, as we discussed last week in §1.3. This provides us with a procedure for estimating ψ_1^2. In QUIRKY, we can simply use the probability display after the measurement to determine the probability of outcome 1:

Thus, we find that $\psi_1^2 \approx 11.7\%$. Since $M|0\rangle$ is a unit vector, we can also infer that $\psi_0^2 = 1 - \psi_1^2 \approx 88.3\%$. However, amplitudes can be negative, so this only determines ψ_0 and ψ_1 up to signs! Now, remember from Exercise 2.7 that $|\psi\rangle$ and $-|\psi\rangle$ are indistinguishable, so we can only hope to determine $|\psi\rangle = M|0\rangle$ up to an overall sign. Thus, we are left with two possibilities:

$$\pm\begin{pmatrix}\sqrt{88.3\%}\\ \sqrt{11.7\%}\end{pmatrix}, \quad \pm\begin{pmatrix}\sqrt{88.3\%}\\ -\sqrt{11.7\%}\end{pmatrix}$$

Note that this situation is very similar to Exercise 2.6, where we had to decide between $|+\rangle$ and $|-\rangle$. In the last homework problem, you will clarify the situation and reveal the inner workings of the mystery box.

Homework 2.6: Mystery time

1. How can you decide which of the two options is the case? Use QUIRKY to determine the quantum state $M|0\rangle$ up to sign.
2. Similarly, determine the quantum state $M|1\rangle$ up to sign.
3. *Bonus question:* Do steps 1 and 2 specify the quantum operation M completely? If yes, write down a formula for M. If not, how can you learn M?

2.6 Interlude on physics (optional)

Our main focus in *The Quantum Quest* is on the *mathematics* of quantum computing. However, since small quantum computers have already been built in laboratories around the world, it is also useful to know a little bit about the *physics* of quantum computing. What kind of physical effects make the quantum computers work?

2.6.1 Interference

One of the most important physical effects used in quantum computing is interference – the interaction between overlapping waves or oscillations. One way you can observe interference is by looking at water waves created by two boats that are passing each other by, or by simultaneously throwing two rocks in a still lake. When the waves amplify each other, we call the interference *constructive*, and when they cancel each other out, we call it *destructive* (see Fig. 2.7).

Interference also occurs in other contexts, for example with sound waves. A familiar example might be the destructive interference in noise-cancelling headphones. They operate by capturing the background noise and playing it back to you, but with an opposite direction of oscillation. When this recorded sound overlaps with the original noise, they cancel each other out: $1 - 1 = 0$. If the headphones would not invert the direction of oscillation but instead play

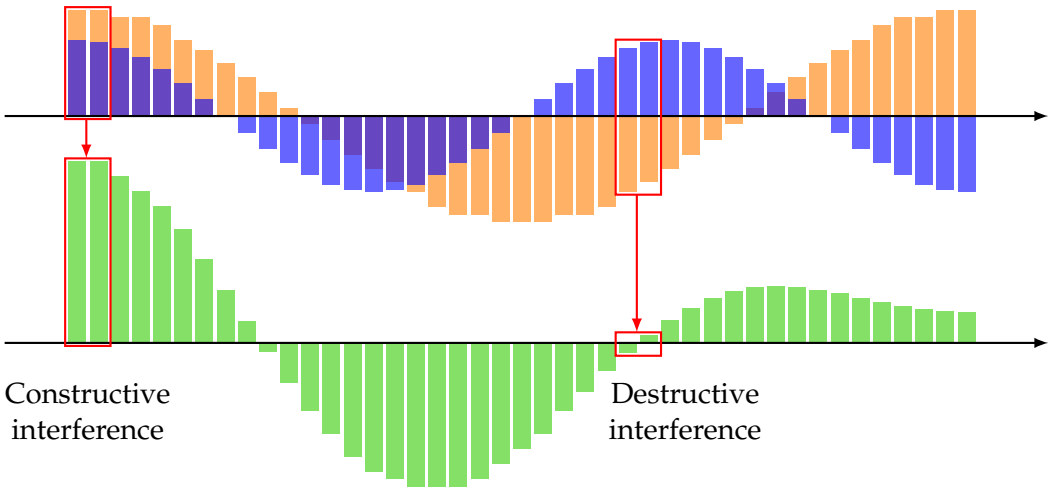


Figure 2.7: Interference of two waves: at every location the amplitudes of the blue and orange wave add to produce the green wave. When both amplitudes have the same sign, the interference is *constructive* and we get an even larger amplitude. When the interfering amplitudes have different signs, the interference is *destructive* and we get a much smaller amplitude.

the sound back to you as it is, you would hear a much louder noise: $1 + 1 = 2$. This would turn your headphones into a hearing aid!

An important way in which quantum computation differs from probabilistic computation is that it can make use of *both* types of interference – constructive and destructive – while probabilistic computation can use only constructive interference. To illustrate this mathematically, recall the probabilistic flip operation $F(1/2)$ and the Hadamard operation H from Eqs. (1.14) and (2.20):

$$\begin{aligned} F(1/2)\,[0] &= \frac{1}{2}\,[0] + \frac{1}{2}\,[1], \qquad & H\,|0\rangle &= \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle\,, \\ F(1/2)\,[1] &= \frac{1}{2}\,[0] + \frac{1}{2}\,[1], \qquad & H\,|1\rangle &= \frac{1}{\sqrt{2}}\,|0\rangle - \frac{1}{\sqrt{2}}\,|1\rangle\,. \end{aligned} \tag{2.23}$$

Apart from the square roots, the two operations are almost identical. However, notice that $F(1/2)\,[1]$ has a plus sign while $H\,|1\rangle$ has a minus sign. This may seem like a small difference but it can have big consequences.

Let us consider the action of these two operations on the uniform distribution $\frac{1}{2}\,[0] + \frac{1}{2}\,[1]$ and its quantum analogue, the plus state $|+\rangle = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle$. The probabilistic flip operation $F(1/2)$ acts on the uniform distribution as follows:

$$\begin{aligned} F(1/2)\left(\frac{1}{2}\,[0] + \frac{1}{2}\,[1]\right) &= \frac{1}{2}\,F(1/2)\,[0] + \frac{1}{2}\,F(1/2)\,[1] \\ &= \frac{1}{2}\left(\frac{1}{2}\,[0] + \frac{1}{2}\,[1]\right) + \frac{1}{2}\left(\frac{1}{2}\,[0] + \frac{1}{2}\,[1]\right) \\ &= \left(\frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2}\right)[0] + \left(\frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2}\right)[1] \\ &= \frac{1}{2}\,[0] + \frac{1}{2}\,[1], \end{aligned}$$

where we used linearity, Eq. (2.23), and collected the probabilities at states $[0]$ and $[1]$. Note how the probabilities at $[1]$ from both terms amplify each other, resulting in the final probability of $1/2$. This is very intuitive: if you flip a uniformly random bit then it stays uniformly random.

However, consider now the action of the Hadamard operation H on the plus state $|+\rangle$:

$$\begin{aligned} H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) &= \frac{1}{\sqrt{2}}H|0\rangle + \frac{1}{\sqrt{2}}H|1\rangle \\ &= \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) \\ &= \left(\frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2}}\right)|0\rangle + \left(\frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2}}\right)|1\rangle \\ &= |0\rangle\,. \end{aligned}$$

The calculation is almost identical but the outcome is very different – the amplitudes at $|1\rangle$ completely cancel each other and we are left only with $|0\rangle$. Such destructive interference is impossible with probabilistic bits because probabilities are always positive – they can only amplify each other but can never cancel.

While probabilistic and quantum bits are quite similar, this example illustrates how they can behave differently thanks to destructive interference. Many of the quantum surprises that you will encounter in further weeks are in some way a consequence of this phenomenon. The ability to perform destructive interference is exactly what gives a quantum computer an advantage over classical computers – it allows the quantum computer to output just the right answer, while the wrong answers are cancelled out by destructive interference. As we will see in more detail in §5.2, this is often achieved by the Hadamard gate, which therefore plays a central role in many quantum algorithms.

2.6.2 Polarization

Now that we are mathematically familiar with qubit states and operations, it would be nice to connect them with something physical.

One of the simplest ways of representing a qubit physically is by polarization of light. Light is an electromagnetic wave that propagates through space in a straight line. This wave oscillates in a direction perpendicular to one in which it travels. Note that there are several possible such directions – a wave that travels forward can oscillate from left to right or from top to bottom. These horizontal and vertical modes of oscillation can be used to represent the two basis states of a qubit:

$$|\leftrightarrow\rangle = \begin{pmatrix}1\\0\end{pmatrix}, \qquad |\updownarrow\rangle = \begin{pmatrix}0\\1\end{pmatrix}.$$

More generally, we can use a wave that oscillates at an angle θ with the horizontal axis to represent the state

$$|\psi(\theta)\rangle = \cos\theta\,|\leftrightarrow\rangle + \sin\theta\,|\updownarrow\rangle = \begin{pmatrix}\cos\theta\\\sin\theta\end{pmatrix}.$$

For example, diagonally polarized light that oscillates at 45° angle between the vertical and horizontal directions represents the state $|\psi(\pi/4)\rangle = |+\rangle$. Note that in this representation the direction of oscillation of the electromagnetic wave agrees with the direction of the vector we used in Fig. 2.1 of §2.1.2 to represent a qubit state on a circle.

To prepare one of these states, we can simply pass a beam of light through a polarizer, like the one in your sunglasses or in 3D glasses at the cinema. A polarizer lets through only some part of the wave – that whose direction of oscillation is compatible with the direction of the polarizer. To prepare the state $|\psi(\theta)\rangle$, we can simply tilt the polarizer at angle θ from the horizontal axis. For example, Fig. 2.8 depicts how to prepare the states $|0\rangle$, $|1\rangle$, and $|+\rangle$.

An interesting feature of representing qubit states by polarized light is that the states $|\psi(\theta)\rangle$ and $|\psi(\theta+\pi)\rangle$ are prepared using the same procedure – tilting the polarizer at an angle θ. This

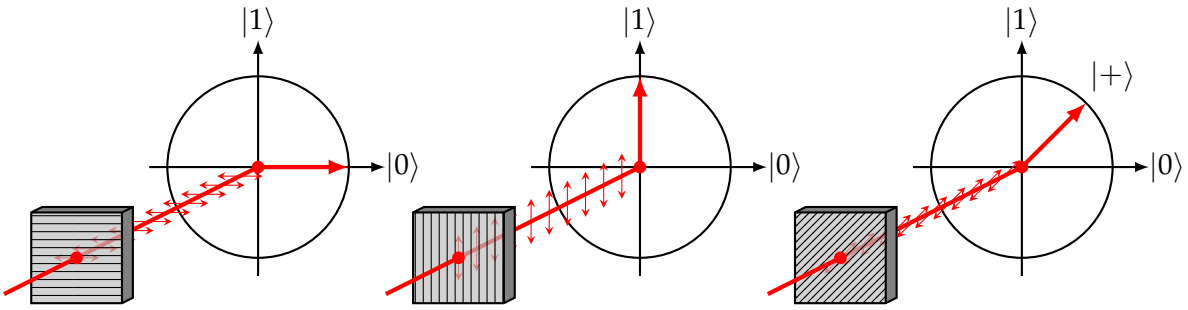


Figure 2.8: Horizontally, vertically, and diagonally polarized light can be used to represent the qubit states $|0\rangle$, $|1\rangle$, and $|+\rangle$.

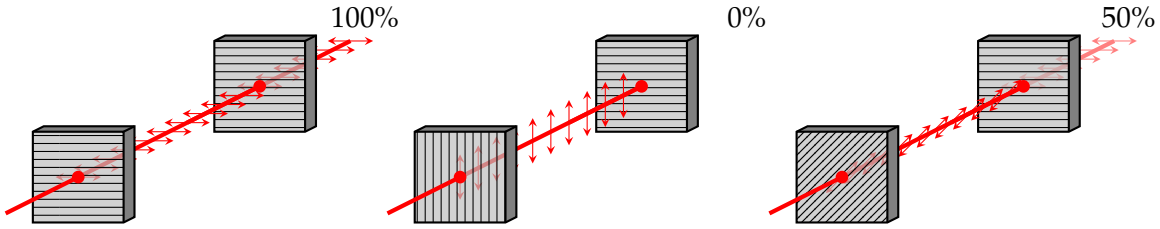


Figure 2.9: The amount of horizontal polarization in light can be determined by passing it through a horizontal polarizer and then measuring its brightness. For horizontally, vertically, and diagonally polarized light this results in 100%, 0%, and 50% brightness, which coincide with the probabilities of observing outcome 0 when measuring the states $|0\rangle$, $|1\rangle$, and $|+\rangle$.

means that these two states must be identical! Hence polarization gives an intuitive explanation for why the states $|\psi\rangle$ and $-|\psi\rangle$ should be indistinguishable (see Exercise 2.7).

Another nice feature of viewing qubits as polarized light is that we can easily visualize measurement. Assume we want to measure the state $|\psi(\theta)\rangle$ to determine the probability of the outcome 0. If the state is provided to us as a beam of light, polarized at angle θ, we can simply pass it through a horizontal polarizer and see how much light gets through – if the brightness has decreased to 70%, the probability of outcome 0 is 70%. In particular, if the input beam was horizontally polarized, all of it will get through, while if it was vertically polarized, none of it will get through. Passing a diagonally polarized beam of light through a horizontal polarizer will result in a 50% decrease in brightness (see Fig. 2.9).

Exercise 2.9: Polarization experiment

If you happen to have a pair of polarized sunglasses at home, you can put them on and take a look at the screen of your phone or computer. Usually screens emit polarized light (whose direction of polarization depends on device). When you tilt your head sideways, you should see the screen becoming lighter or darker. Can you explain why this is the case?

Polarization of light is just one example of how a qubit could be implemented in laboratory. Another example is the *location* of the light-carrying particle called photon – since a photon behaves according to the laws of quantum mechanics, it can simultaneously be at two locations in superposition. If we call these locations 0 and 1, the photon's state corresponds to a qubit. There are many other options: the current in a superconducting circuit can simultaneously flow in both directions, an electron can simultaneously occupy two orbitals around an atom, and so on. In short, any quantum mechanical system that can be in two distinct states can also be in their superposition, hence it can potentially be used as a physical representation of a qubit.

2.7 Exercise solutions

Solution to Exercise 2.1

Note that

$$|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = |\psi(\pi/4)\rangle\,, \qquad |-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} = |\psi(-\pi/4)\rangle\,.$$

Thus the angles are $\theta = \pm\pi/4$, and the two states are located 45 degrees upwards and downwards from $|0\rangle$, respectively.

Solution to Exercise 2.2

Let $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, which is a valid quantum state. Since MAD is obtained by extending by linearity, Eq. (2.8) implies that

$$\begin{aligned}\mathrm{MAD}\,|\psi\rangle &= \mathrm{MAD}\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{\sqrt{2}}\mathrm{MAD}\,|0\rangle + \frac{1}{\sqrt{2}}\mathrm{MAD}\,|1\rangle \\ &= \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{2}(|0\rangle + |1\rangle) = \left(\frac{1}{\sqrt{2}} + \frac{1}{2}\right)|0\rangle + \frac{1}{2}|1\rangle\,.\end{aligned}$$

But

$$\left(\frac{1}{\sqrt{2}} + \frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 = 1 + \frac{1}{\sqrt{2}} \neq 1,$$

so $\mathrm{MAD}\,|\psi\rangle$ is not a valid qubit state.

Solution to Exercise 2.3

1. $U(\alpha)$ acts on an arbitrary state as follows:

$$\begin{aligned}U(\alpha)\begin{pmatrix}\psi_0\\\psi_1\end{pmatrix} &= U(\alpha)\left(\psi_0\begin{pmatrix}1\\0\end{pmatrix} + \psi_1\begin{pmatrix}0\\1\end{pmatrix}\right) = \psi_0\begin{pmatrix}\cos\alpha\\\sin\alpha\end{pmatrix} + \psi_1\begin{pmatrix}-\sin\alpha\\\cos\alpha\end{pmatrix} \\ &= \begin{pmatrix}\psi_0\cos\alpha - \psi_1\sin\alpha\\\psi_0\sin\alpha + \psi_1\cos\alpha\end{pmatrix}.\end{aligned}$$

2. Since $|\psi(\beta)\rangle = \begin{pmatrix}\cos\beta\\\sin\beta\end{pmatrix}$,

$$\begin{aligned}U(\alpha)\,|\psi(\beta)\rangle &= U(\alpha)\begin{pmatrix}\cos\beta\\\sin\beta\end{pmatrix} = \begin{pmatrix}\cos\beta\cos\alpha - \sin\beta\sin\alpha\\\cos\beta\sin\alpha + \sin\beta\cos\alpha\end{pmatrix} = \begin{pmatrix}\cos(\alpha+\beta)\\\sin(\alpha+\beta)\end{pmatrix} \\ &= |\psi(\alpha+\beta)\rangle\,.\end{aligned}$$

Solution to Exercise 2.4

Take an arbitrary state $\psi_0 |0\rangle + \psi_1 |1\rangle$, first use the linearity of M and then the linearity of N:

$$\begin{aligned} NM(\psi_0 |0\rangle + \psi_1 |1\rangle) &= N\big(M(\psi_0 |0\rangle + \psi_1 |1\rangle)\big) \\ &= N\big(\psi_0 M |0\rangle + \psi_1 M |1\rangle\big) = \psi_0 NM |0\rangle + \psi_1 NM |1\rangle . \end{aligned}$$

In the last step, we used Eq. (2.10).

Solution to Exercise 2.5

We have $(NM)^{-1} = M^{-1}N^{-1}$, since for any $|\psi\rangle$ we have

$$M^{-1}N^{-1}NM |\psi\rangle = M^{-1}(N^{-1}N(M |\psi\rangle)) = M^{-1}(M |\psi\rangle) = M^{-1}M |\psi\rangle = |\psi\rangle$$

and

$$NMM^{-1}N^{-1} |\psi\rangle = N(MM^{-1}(N^{-1} |\psi\rangle)) = N(N^{-1} |\psi\rangle) = NN^{-1} |\psi\rangle = |\psi\rangle .$$

Solution to Exercise 2.6

Apply $U(-\pi/4)$ and measure. You can guess the state with certainty!

Solution to Exercise 2.7

We saw above that any combination M of rotations and reflections is linear. Thus if $M |\psi(\theta)\rangle = |\psi(\theta')\rangle = \binom{\cos\theta'}{\sin\theta'}$, then $M\big(-|\psi(\theta)\rangle\big) = -|\psi(\theta')\rangle = \binom{-\cos\theta'}{-\sin\theta'}$. In view of Eq. (2.6), the probabilities p_0 and p_1 of measurement outcomes are the same for both states.

Solution to Exercise 2.9

Tilting your head changes the angle between the polarizer in your sunglasses and the direction in which the electromagnetic light waves emitted by your screen oscillate. Since the amount of light that can pass through a polarizer depends on this angle, the screen will appear either brighter or darker. Similarly, changing the angle θ will change the probability that measuring the state $|\psi(\theta)\rangle$ will produce the outcome 0.

Solution to Exercise 2.8

Ideally, Bob would like to send 2 bits indicating which leg and which arm is broken. However, Alice only cares about one of these bits since she has only one type of tools with her.

1. Let us denote the two possible values of each bit by L (left) and R (right). One strategy Bob can use is to send the "majority" of his two bits. Namely, he can use the following encoding: $LL \mapsto L$, $RR \mapsto R$. The remaining two cases he can encode arbitrarily, say, $LR \mapsto L$, $RL \mapsto R$. Alice's strategy is simply to send the limb corresponding to Bob's message (the left limb if she received L and the right limb if she received R). This works with probability

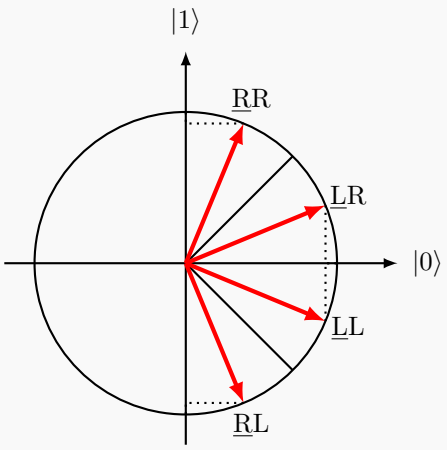
$$\frac{1}{4}\left(1+1+\frac{1}{2}+\frac{1}{2}\right) = \frac{3}{4} = 0.75, \tag{2.24}$$

   where the four terms inside the brackets are the probabilities that Alice makes the right decision for each of the four Bob's situations.

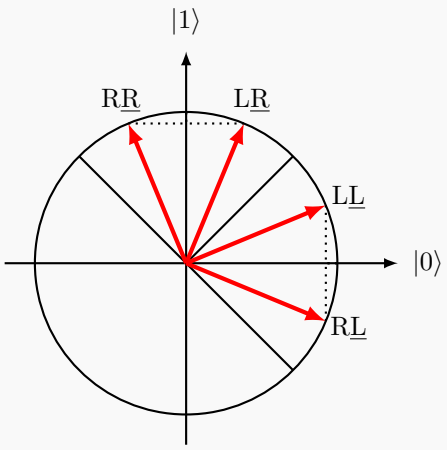
2. Bob can send the following qubit state, depending on what limbs are broken (i.e., LL means left leg and left arm, LR means left leg and right arm, etc.)

$$\begin{aligned}
|LL\rangle &= \cos(\pi/8)\,|0\rangle - \sin(\pi/8)\,|1\rangle \\
|LR\rangle &= \cos(\pi/8)\,|0\rangle + \sin(\pi/8)\,|1\rangle \\
|RR\rangle &= \cos(3\pi/8)\,|0\rangle + \sin(3\pi/8)\,|1\rangle \\
|RL\rangle &= \cos(3\pi/8)\,|0\rangle - \sin(3\pi/8)\,|1\rangle
\end{aligned}$$

   To recover the leg bit, Alice just measures. To recover the arm bit, Alice applies $U(\pi/4)$ and then measures. (Note that she cannot recover both bits since the original state is no longer around after the measurement.) They will succeed with probability $\cos^2(\pi/8) = \frac{1}{2} + \frac{1}{2\sqrt{2}} \approx 0.85$. This is better than in the first scenario!



Extracting the leg bit



Extracting the arm bit

Quest 3: Wizard of entanglement

In the previous Quests, we discussed all about how a single probabilistic bit and a single quantum bit behave. This week, you will learn what happens when you have two of them. We first discuss two bits and learn in which states they can be and how they can be *correlated*. We then move on to consider two quantum bits and what it means for them to be *entangled*.

3.1 Two probabilistic bits

When you have two coins placed on a table, they can be in one out of four configurations:

These correspond to the four possible bit strings[11]: 00, 01, 10, 11.

If you have a pair of probabilistic bits, their state is described by a probability distribution over these four possible deterministic states. In other words, their state is specified by four numbers $p_{00}, p_{01}, p_{10}, p_{11} \geq 0$ such that $p_{00} + p_{01} + p_{10} + p_{11} = 1$. Just like in the case of a single bit, we could write this down as a vector

$$\begin{pmatrix} p_{00} \\ p_{01} \\ p_{10} \\ p_{11} \end{pmatrix}. \tag{3.1}$$

While accurate, this representation is quite clumsy because keeping track of which probability is assigned to which bit configuration can be hard (does p_{10} or p_{01} go first?!), and it will become only harder once we have more than two bits. It is much more convenient to use a notation that lets us directly keep track of the probabilities assigned to particular bit strings. We therefore extend the notation we introduced in Eq. (1.3) for a single probabilistic bit and write the above two-bit state as follows:

$$p_{00}[00] + p_{01}[01] + p_{10}[10] + p_{11}[11]. \tag{3.2}$$

If you wish, you can always relate this notation back to the 4-vector notation in Eq. (3.1) using the following dictionary that generalizes Eq. (1.2):

$$[00] = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad [01] = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \qquad [10] = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \qquad [11] = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{3.3}$$

One advantage of this notation in the case of multiple bits is that we can simply omit entries that are zero. For example, we can simply write

$$\frac{1}{2}[00] + \frac{1}{2}[11]$$

instead of the more clumsy

$$\frac{1}{2}[00] + 0\,[01] + 0\,[10] + \frac{1}{2}[11].$$

With this notation it is also much easier to describe measurements and operations on several probabilistic bits. We can explore two probabilistic bits using QUIRKY, which has again gained new powers since last week. To begin, go to:

[11] A 'string' means a sequence of symbols (in this case: a sequence of bit values). It will be our favourite notation when we are dealing with multiple coins or bits.

https://www.quantum-quest.org/quirky

and click on "Quest 3" and select "Two Bits". Your web browser will look similarly to Fig. 3.1. Note that we now have *two* wires, corresponding to two bits, which are initialized in state [00]. Perhaps surprisingly, the first bit corresponds to the *bottom* wire and the second bit to the *top* wire. In addition, there is one new box: [•] (but the mystery box is gone). We will discuss the meaning of this box later in this chapter.

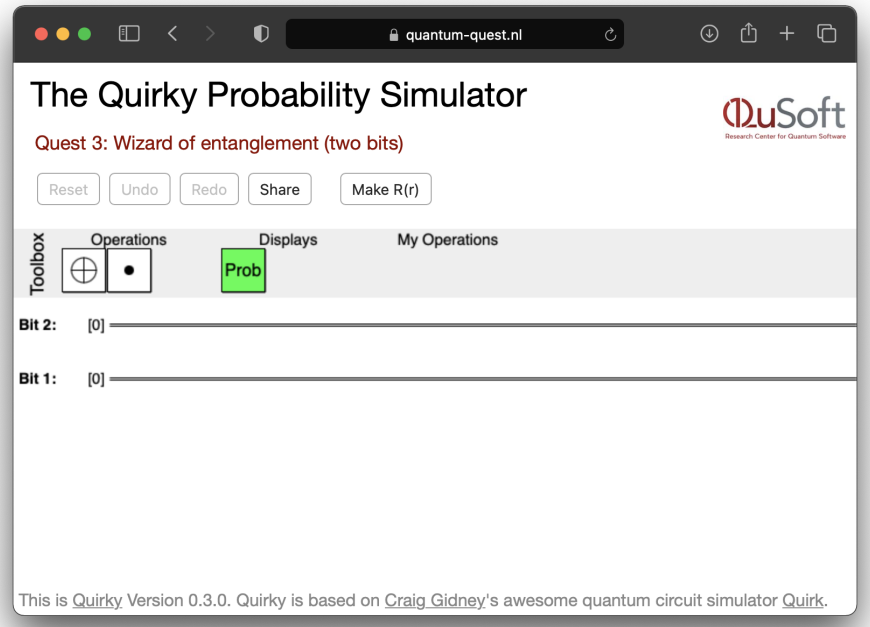
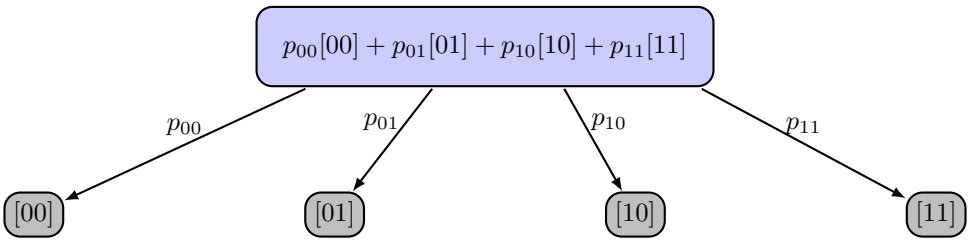


Figure 3.1: QUIRKY for Quest 3.

3.1.1 Measuring both bits

Measuring (or "looking at") two probabilistic bits works the same way as Eq. (1.18) for a single bit. You get one out of four possible outcomes (00, 01, 10, or 11) with the corresponding probability:



(3.4)

The state of both bits after the measurement is no longer a distribution over four possibilities but rather just a single option that corresponds to the measurement outcome you observed. To emphasize this difference, we use light blue for probabilistic bits and gray for deterministic ones after the measurement. How can we measure both bits in QUIRKY? We simply use the probability display, like so:

Note that, by default, the probability display is connected to *both* wires, so it shows the probabilities for *both* bits. The order of probabilities is as in the 4-vector notation in Eq. (3.1). You don't have to remember the order, though. Simply hover the table with your mouse cursor to remind yourself (thanks, Craig!).

For example, if the two probabilistic bits are in state

$$\frac{1}{2}[00] + \frac{1}{2}[11], \tag{3.5}$$

then we obtain as measurement outcomes either 00 or 11, each with 50% probability. Notice that this state is special – if we see that the measurement outcome of the first bit is 0, we immediately learn that the outcome from the second bit also has to be 0, and similarly for when either of the two outcomes is 1. Since the measurement outcomes of both bits are always equal, we call two bits in state (3.5) perfectly correlated. We will see below how such states can be created.

3.1.2 Local operations

When you have two or more probabilistic bits, you can act on them in many different ways. In particular, you can act on all of them together with a global operation or only on one or a few at a time with a local operation. Let us consider local operations first.

Recall the NOT operation from §1.2 that flips a bit. What happens if we have two bits and apply NOT only on the first one? In that case the first bit should be flipped while the second should remain the same. This means the *local* NOT operation on the first bit, which we will denote by NOT_1, acts as follows:

$$\mathrm{NOT}_1\,[00] = [10], \quad \mathrm{NOT}_1\,[01] = [11], \quad \mathrm{NOT}_1\,[10] = [00], \quad \mathrm{NOT}_1\,[11] = [01]. \tag{3.6}$$

Similarly, if we apply NOT only on the second bit, the resulting NOT_2 operation acts as follows:

$$\mathrm{NOT}_2\,[00] = [01], \quad \mathrm{NOT}_2\,[01] = [00], \quad \mathrm{NOT}_2\,[10] = [11], \quad \mathrm{NOT}_2\,[11] = [10]. \tag{3.7}$$

What we just described are local NOT operations on deterministic bits. How should we extend them to probabilistic bits? Recall from §1.2.1 that any operation that is fully specified on deterministic bits can be extended *by linearity* to probabilistic bits. For example, NOT_2 acts on two probabilistic bits as follows:

$$\begin{aligned}
&\mathrm{NOT}_2\,(p_{00}[00] + p_{01}[01] + p_{10}[10] + p_{11}[11]) \\
&\quad = p_{00}[01] + p_{01}[00] + p_{10}[11] + p_{11}[10] \\
&\quad = p_{01}[00] + p_{00}[01] + p_{11}[10] + p_{10}[11],
\end{aligned}$$

where in the first step we used Eq. (3.7) and in the second step we just reordered the terms to sort the binary strings. You can also write this in the 4-vector notation, but it is somewhat less intuitive:

$$\mathrm{NOT}_2 \begin{pmatrix} p_{00} \\ p_{01} \\ p_{10} \\ p_{11} \end{pmatrix} = \begin{pmatrix} p_{01} \\ p_{00} \\ p_{11} \\ p_{10} \end{pmatrix}. \tag{3.8}$$

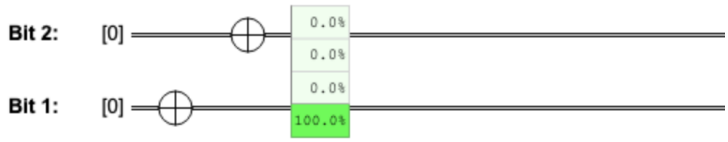
Exercise 3.1: NOT_1 in the 4-vector notation (optional)

Similar to Eq. (3.8), write the action of NOT_1 on two probabilistic bits in the 4-vector notation.

To apply a single-bit operation in QUIRKY, we drop the corresponding box onto eiter the first or the second wire. For example, the following sequence prepares the [10] state and shows the outcome probabilities when measuring both bits:

This makes sense since the bottom wire in QUIRKY corresponds to the first bit.

Similarly, if we first flip one bit and then the other, the result is the [11] state:



Clearly, the order in which we apply the two NOT operations does not matter. This means that we can also apply them *in parallel*:

We can in the same way apply random operations to one of the bits. For example, suppose we subject the first bit to the operation $R(r)$ that resets a bit with probability r (Eq. (1.13)). Since $R(r)[0] = [0]$, we have that

$$R(r)_1[00] = [00], \qquad R(r)_1[01] = [01]. \tag{3.9}$$

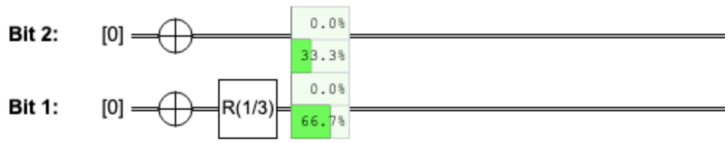
And since $R(r)[1] = r[0] + (1-r)[1]$, we have that

$$R(r)_1[10] = r[00] + (1-r)[10], \qquad R(r)_1[11] = r[01] + (1-r)[11]. \tag{3.10}$$

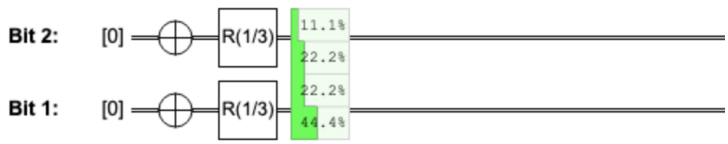
For example, if we prepare the state [11] and apply $R(1/3)$ to the first bit, we obtain

$$R(1/3)_1[11] = \frac{1}{3}[01] + \frac{2}{3}[11], \tag{3.11}$$

as confirmed by QUIRKY:



Here is an even more interesting example, which you can tackle in the exercise below:



(3.12)

Homework 3.1: $R(r)$ on the second bit

1. Write down formulas for $R(r)_2$ analogously to Eqs. (3.9) and (3.10).

2. Explain why QUIRKY gives the correct answer in (3.12).

3.1.3 Measuring only one bit

If you have two probabilistic bits and you measure only one of them, what are the probabilities to get each of the two outcomes? Our notation is particularly convenient for figuring this out. Consider again a general probabilistic two-bit state

$$p_{00}[00] + p_{01}[01] + p_{10}[10] + p_{11}[11].$$

To find the probability of outcome 0 when measuring some bit, you simply sum together the probabilities of all terms where the bit you are measuring is in the desired state 0; similarly for the outcome 1.

For example, the probability of observing outcome 1 when measuring the *first* bit is

$$p_{10} + p_{11}, \tag{3.13}$$

corresponding to the probabilities in Eq. (3.4) that lead to [10] and [11], which are the two bit strings that start with 1. Similarly, the probability of observing outcome 0 when measuring the *second* bit is

$$p_{00} + p_{10},$$

corresponding to the probabilities that lead to the two bit strings that end with zero, [00] and [10]. It is easy to compute this if you arrange the four probabilities in a 2×2 square, as in Fig. 3.2.

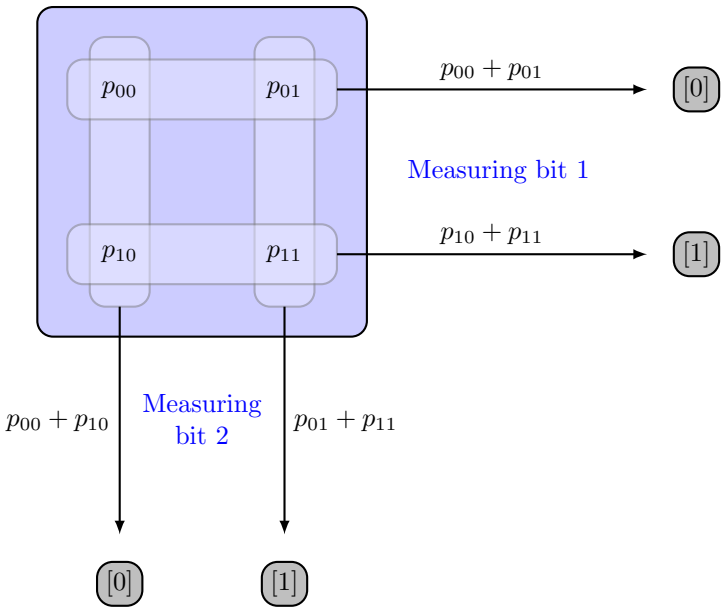
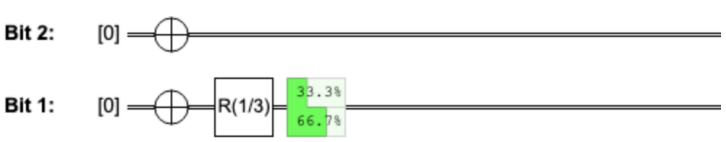
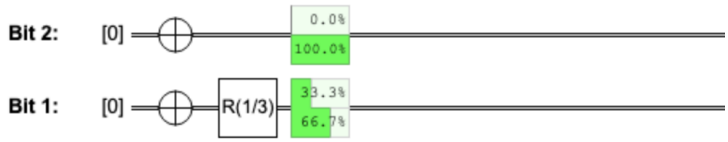


Figure 3.2: Probabilities of measurement outcomes when measuring only one out of two probabilistic bits.

We can also use QUIRKY to display the probabilities when measuring a single bit. Simply resize the probability display so that it only covers a single wire, like so:



In fact, we can view both the probabilities of outcomes when measuring only the first bit and the probabilities of outcomes when measuring only the second bit at the same time:



Note that the result is very intuitive. Since the two bits are never "correlated", it is clear that the first bit should be in state $\frac{1}{3}[0] + \frac{2}{3}[1]$ and the second bit in state $[1]$.

3.1.4 State of the other bit

After the measurement, the bit that was measured is in a deterministic state that corresponds to the measurement outcome (just as when measuring a single bit – see Eq. (1.18)). But how about the other bit that was not measured? Its state after the measurement in general will not be deterministic. For example, if the initial state of the two probabilistic bits is

$$\frac{1}{2}[10] + \frac{1}{2}[11] \tag{3.14}$$

and the first bit is measured, the probability of observing 1 is $1/2 + 1/2 = 1$. In other words, the first bit in Eq. (3.14) is deterministic and the two probabilities actually describe only the second bit. Hence, you can intuitively think of this state as the result of "combining" together two separate probabilistic bits: $[1]$ and $\frac{1}{2}[0] + \frac{1}{2}[1]$ (we will talk more about how to combine two probabilistic bits in §3.1.7). Therefore it makes sense that the state of the second bit after the measurement should be uniformly random, namely

$$\frac{1}{2}[0] + \frac{1}{2}[1].$$

More generally, suppose that we start with two probabilistic bits in an arbitrary state

$$p_{00}[00] + p_{01}[01] + p_{10}[10] + p_{11}[11] \tag{3.15}$$

and measure the first bit. The state of the remaining bit will generally depend on the measurement outcome. For example, if the outcome was 1, to find the state of the second bit we first collect all terms in Eq. (3.15) where the first bit has value 1:

$$p_{10}[10] + p_{11}[11].$$

Then, we ignore the first bit since we already know that its value is 1:

$$p_{10}[0] + p_{11}[1].$$

Finally, since these two probabilities may not sum to one, we divide them by their sum $p_{10} + p_{11}$:

$$\frac{p_{10}}{p_{10} + p_{11}}[0] + \frac{p_{11}}{p_{10} + p_{11}}[1]. \tag{3.16}$$

This is the probability distribution on the second bit when the first bit is measured and yields outcome 1 (see the right side of Fig. 3.3). The remaining cases are also summarized in Fig. 3.3.

To see that these rules make sense, let us verify that measuring the first bit and then the second bit gives the same probabilities as directly measuring both bits. For example, the probability of obtaining 1 from the first bit and 0 from the second bit should simply be p_{10}. Indeed, according to Eq. (3.13), we obtain the outcome 1 from the first bit with probability

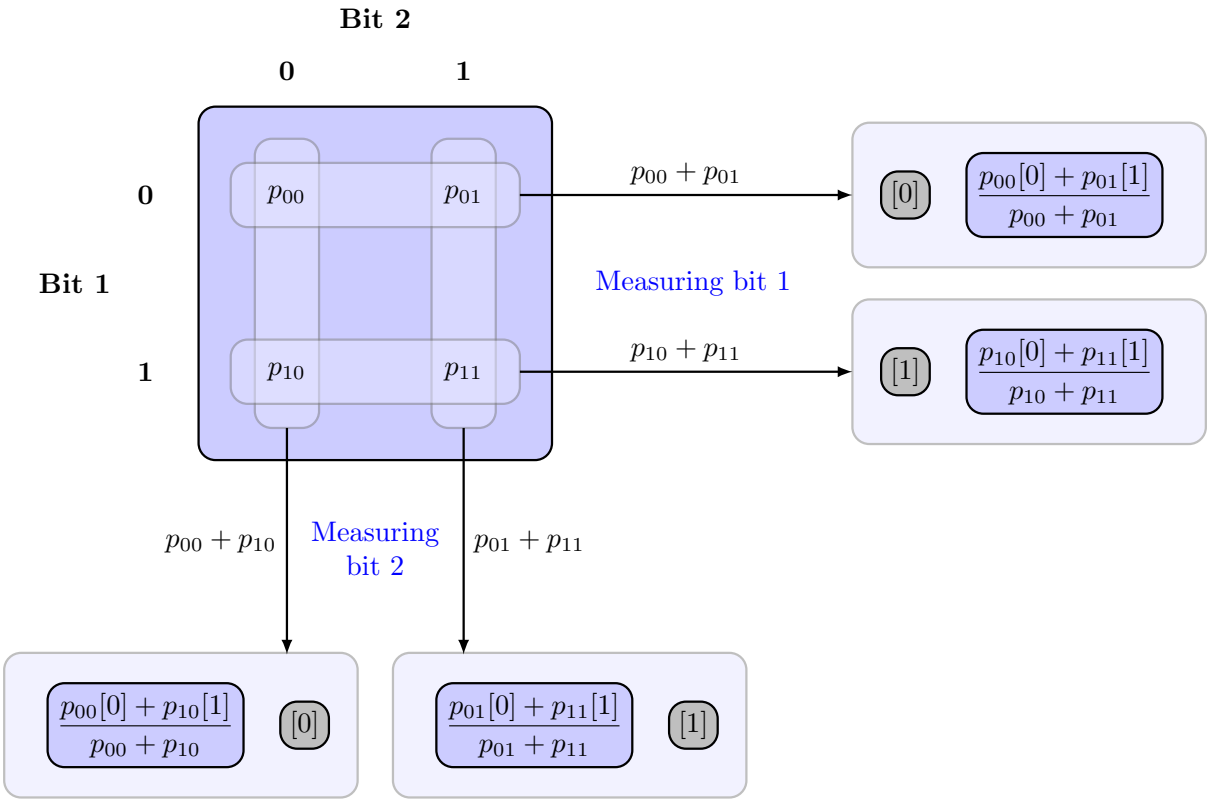


Figure 3.3: Outcome probabilities and the state of the remaining bit when measuring one out of two probabilistic bits. After the measurement, the bit that was measured becomes deterministic (gray) while the other bit stays probabilistic (light blue).

$p_{10} + p_{11}$ and, given this outcome, Eq. (3.16) asserts that we obtain the outcome 0 from the second bit with probability $p_{10}/(p_{10} + p_{11})$. Thus the total probability of first obtaining 1 from the first bit and then 0 from the second bit is

$$(p_{10} + p_{11}) \times \frac{p_{10}}{p_{10} + p_{11}} = p_{10},$$

which is exactly what we want according to Eq. (3.4). The other cases can be verified similarly.

Exercise 3.2: Guessing Alice's coin

Problem: Alice has three coins called u, q, r with the following probability distributions:

$$u = \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix}, \qquad q = \begin{pmatrix} 3/4 \\ 1/4 \end{pmatrix}, \qquad r = \begin{pmatrix} 1/3 \\ 2/3 \end{pmatrix}.$$

Alice performs the following sequence of coin tosses:

1. She tosses the coin u.
2. Depending on the outcome, she tosses one of the other two coins:
   (0) if u produced 0, she tosses q;
   (1) if u produced 1, she tosses r.
3. Alice tells her friend Bob the outcome (0 or 1) of the coin she tossed in step 2 (but she does not tell whether it was produced by tossing the coin q or r).

In this situation there are *two* probabilistic bits: Alice' first coin toss and Alice' second coin toss (which is exactly the same as Bob's probabilistic bit).

Questions:

1. What is the probability distribution of Alice' two coin tosses?
2. What is the probability distribution when Bob measures his probabilistic bit?
3. Given that Bob measures his bit and obtains outcome 0, is it more likely that Alice' first coin shows 0 or 1? How about if his outcome is 1?

3.1.5 SWAP operation

Now that we know how to manipulate individual probabilistic bits, we would also like to apply operations on several bits at a time. One of the simplest such operations is the SWAP operation that interchanges two bits:

$$\begin{aligned}\text{SWAP}\,[00] &= [00],\\ \text{SWAP}\,[01] &= [10],\\ \text{SWAP}\,[10] &= [01],\\ \text{SWAP}\,[11] &= [11].\end{aligned}$$

SWAP effectively amounts to exchanging the strings $[01]$ and $[10]$ while leaving the other two strings alone. The above equations can be written more concisely as follows:

$$\text{SWAP}\,[a,b] = [b,a], \tag{3.17}$$

for all $a, b \in \{0,1\}$, where we use a comma to separate the two bits. As usual, we can extend SWAP from deterministic to probabilistic bits by linearity:

$$\begin{aligned}&\text{SWAP}\,\left(p_{00}[00] + p_{01}[01] + p_{10}[10] + p_{11}[11]\right)\\ &\quad= p_{00}[00] + p_{01}[10] + p_{10}[01] + p_{11}[11]\\ &\quad= p_{00}[00] + p_{10}[01] + p_{01}[10] + p_{11}[11].\end{aligned}$$

Exercise 3.3: SWAP in the 4-vector notation (optional)

Write down the action of SWAP on two probabilistic bits in the 4-vector notation.

3.1.6 Controlled-NOT operation

NOT and SWAP operations together can be used to change individual bits and to reorder them in a different order. However, they don't cause the bits to interact with one another. What we would like is another more sophisticated operation that can change the value of one bit depending on the value of another. The simplest and most important such operation is CNOT or the controlled-NOT operation. It flips the target bit if the control bit is set to 1.

When the first bit is the control and the second bit is the target, we will write this operation as $\text{CNOT}_{1\to 2}$. In this case:

$$\begin{aligned}\text{CNOT}_{1\to 2}\,[00] &= [00],\\ \text{CNOT}_{1\to 2}\,[01] &= [01],\\ \text{CNOT}_{1\to 2}\,[10] &= [11],\\ \text{CNOT}_{1\to 2}\,[11] &= [10].\end{aligned} \tag{3.18}$$

This amounts to exchanging the strings $[10]$ and $[11]$ while leaving the other two strings alone.

Another way of thinking about $\mathrm{CNOT}_{1\to 2}$ is as an addition: it adds the two bits (modulo 2) and stores the result in the second bit. This can be summarized very concisely as follows:

$$\mathrm{CNOT}_{1\to 2}\,[a,b] = [a, a \oplus b], \tag{3.19}$$

for any $a, b \in \{0,1\}$, where "$\oplus$" denotes *addition modulo* 2 and we use a comma to separate the values of the two bits. The laws for addition modulo 2 are as follows:

$$0 \oplus 0 = 1 \oplus 1 = 0, \qquad 0 \oplus 1 = 1 \oplus 0 = 1. \tag{3.20}$$

Sometimes "$\oplus$" is also called XOR or the exclusive OR operation, since the result is 1 when exactly one out of the two bits are 1. As usual, we can extend $\mathrm{CNOT}_{1\to 2}$ from deterministic to probabilistic bits by linearity.

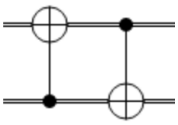
We will also be interested in controlled-NOT operations where the *second* bit is the control and the *first* is the target. In analogy, we will denote this operation by $\mathrm{CNOT}_{2\to 1}$.

Exercise 3.4: Swapping control and target

1. Write down a formula similar to Eq. (3.19) for $\mathrm{CNOT}_{2\to 1}$.
2. How can $\mathrm{CNOT}_{2\to 1}$ be implemented using SWAP and $\mathrm{CNOT}_{1\to 2}$?

In QUIRKY, you can build a controlled-NOT operation by the following two-step process. First, drag the $\boxed{\bullet}$ box onto one of the wires – this marks the wire as the control bit. Next, drag the $\boxed{\oplus}$ box onto the corresponding location on the other wire, marking it as the target bit. A connection will emerge to indicate that you have successfully created a controlled-NOT operation. For example, if we choose the bottom bit as the control and the top bit as the target, we obtain the following result:

Remembering that the bottom (!) bit is the first bit and the top bit is the second bit, we see that this corresponds to the $\mathrm{CNOT}_{1\to 2}$ operation. Here is a more complicated example, where we first apply $\mathrm{CNOT}_{1\to 2}$ and then $\mathrm{CNOT}_{2\to 1}$:



Homework 3.2: SWAP from CNOTs

Problem: It is almost midnight but Bob is still tinkering with his prototype probabilistic bit computer that he wants to present at school the next day. He is so preoccupied with calibrating the randomness generator that he forgot to feed his parrot Ziggy. To attract Bob's attention, Ziggy knocks off Bob's coffee cup and spills the coffee all over his custom-made keyboard for activating different operations. Bob is horrified since the SWAP key is no longer working! Luckily, the CNOT key still works.

Question: How can Bob implement the SWAP operation using only controlled-NOT operations? (If you want to prove that two operations are the same, you only have to show this for the basis states because of extension by linearity.)

Hint: You should use three CNOT operations.

3.1.7 Product distributions

Let us now talk in more detail about the states of a two-bit system. Suppose, for example, that we have two probabilistic bits, $q = q_0[0] + q_1[1]$ and $r = r_0[0] + r_1[1]$. How can we build a state of a two-bit system from this? While we did not explicitly put it this way, we already discussed this question in §1.1.1, where we defined the probability of two events to occur simultaneously by the product of the probabilities of the two individual events. For example, the probability that the bits q and r are in state $[00]$ is $q_0 r_0$. Similarly, the probability of them being in the state $[01]$ is $q_0 r_1$. If we account for all four possibilities, we get

$$q_0 r_0[00] + q_0 r_1[01] + q_1 r_0[10] + q_1 r_1[11]. \tag{3.21}$$

In other words, the four probabilities p_{ab} of the combined state

$$p_{00}[00] + p_{01}[01] + p_{10}[10] + p_{11}[11]$$

are given by the formula

$$p_{ab} = q_a r_b. \tag{3.22}$$

You can verify that the outcome probabilities when measuring the first bit are given by q, while the outcome probabilities when measuring the second bit are given by r (you can do this using the rule from Fig. 3.2 and the fact that $r_0 + r_1 = 1$ and $q_0 + q_1 = 1$). However, our two-bit state has an additional special property: the values of the two bits are independent from each other. This means that when we observe either of the two bits we do not learn any information about the other bit. You can verify this in the next exercise:

Exercise 3.5: Independent bits (optional)

Suppose that we measure the first of the two bits in the state (3.21) and denote the obtained outcome by $a \in \{0, 1\}$. Show that the state on the second bit is r, independently of the measurement outcome a on the first bit. In other words, putting the two bits together and then measuring the first bit did not affect the state of the second bit at all (as it should be)!

Let us introduce some notation that makes the special structure of the state more clear. Namely, let us use "$\otimes$" to denote the operation of putting two probabilistic bits together and considering them as a single system consisting of two bits:

$$q \otimes r = \big(q_0[0] + q_1[1]\big) \otimes \big(r_0[0] + r_1[1]\big) \tag{3.23}$$

The symbol "$\otimes$" is called the tensor product or *Kronecker product*. How can we convert this strange expression to an actual distribution on two bits, like in Eq. (3.21)? First, note that for deterministic bits the operation "$\otimes$" reduces to simply concatenating strings. For example,

$$[0] \otimes [1] = [01]. \tag{3.24}$$

This makes sense since having a bit in state $[0]$ and another one in state $[1]$ is the same as having two bits in the state $[01]$. As always, to extend this rule to probabilistic bits, we ask our good

friend linearity for help! By linearity, we can expand both terms in Eq. (3.23) and then apply the concatenation rule:

$$\begin{aligned}&(q_0[0]+q_1[1])\otimes(r_0[0]+r_1[1])\\&=q_0r_0([0]\otimes[0])+q_0r_1([0]\otimes[1])+q_1r_0([1]\otimes[0])+q_1r_1([1]\otimes[1])\\&=q_0r_0[00]+q_0r_1[01]+q_1r_0[10]+q_1r_1[11].\end{aligned}$$

Note that we have recovered the distribution in Eq. (3.21). In other words, we have the following identity between (3.23) and (3.21):

$$(q_0[0]+q_1[1])\otimes(r_0[0]+r_1[1])=q_0r_0[00]+q_0r_1[01]+q_1r_0[10]+q_1r_1[11]. \tag{3.25}$$

This means that the way we defined the tensor product operation "$\otimes$" is indeed consistent with our earlier argument that the probability distribution of a two-bit system is obtained by multiplying the probabilities of individual bits, see Eq. (3.22).

Note that Eq. (3.25) is very similar to the distributive law for addition and multiplication:

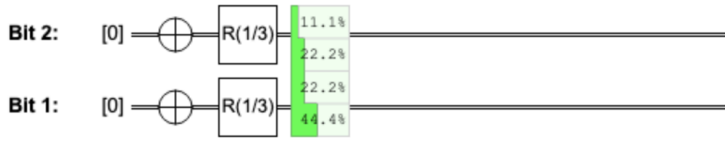
$$(a+b)(c+d)=ac+ad+bc+bd.$$

The only difference is that instead of numbers we have vectors and instead of multiplication we have the concatenation rule $[a]\otimes[b]=[a,b]$. A major difference between concatenation and multiplication is that the order of elements is important in concatenation. Generally, $[a,b]\neq[b,a]$ since concatenating $[a]$ and $[b]$ is not the same as concatenating $[b]$ and $[a]$. By the way, you can check that using the vector notation, you can write the tensor product as follows:

$$\begin{pmatrix}q_0\\q_1\end{pmatrix}\otimes\begin{pmatrix}r_0\\r_1\end{pmatrix}=\begin{pmatrix}q_0\begin{pmatrix}r_0\\r_1\end{pmatrix}\\q_1\begin{pmatrix}r_0\\r_1\end{pmatrix}\end{pmatrix}=\begin{pmatrix}q_0r_0\\q_0r_1\\q_1r_0\\q_1r_1\end{pmatrix},$$

where the second expression is a block vector whose both entries are vectors.

The tensor product gives a quick way of understanding what is going on in (3.12), which we repeat here for convenience:



Note that we prepare each bit independently in the state $\frac{1}{3}[0]+\frac{2}{3}[1]$. Therefore, the joint state of both bits is

$$\left(\frac{1}{3}[0]+\frac{2}{3}[1]\right)\otimes\left(\frac{1}{3}[0]+\frac{2}{3}[1]\right)=\frac{1}{9}[00]+\frac{2}{9}[01]+\frac{2}{9}[10]+\frac{4}{9}[11]=\begin{pmatrix}1/9\\2/9\\2/9\\4/9\end{pmatrix}\approx\begin{pmatrix}11.1\%\\22.2\%\\22.2\%\\44.4\%\end{pmatrix},$$

in agreement with QUIRKY.

Homework 3.3: Tensor product

Find two probabilistic bits q and r such that

$$q\otimes r=0.48[00]+0.32[01]+0.12[10]+0.08[11].$$

The tensor product also allows us to write more compact formulas for local operations. Namely, if M is an operation on one bit then

$$M_1([a] \otimes [b]) = M[a] \otimes [b], \qquad M_2([a] \otimes [b]) = [a] \otimes M[b]. \tag{3.26}$$

This agrees with the formulas discussed in §3.1.2.

Since the two-bit distributions described above are obtained by taking a product of two one-bit distributions, $p = q \otimes r$, they are called product states or product distributions. As you saw in Exercise 3.5, product distributions naturally model a situation when two bits arise independently, such as when tossing two coins. But is every two-bit distribution a product distribution? Interestingly, this is not the case, as we will see in the following section.

3.1.8 Correlated distributions

Some two-bit distributions are *not* product distributions – they cannot be written as in Eq. (3.21) or Eq. (3.23), no matter what values of q_a and r_b you choose. We will say that a distribution is correlated if it is not a product distribution. One example of a correlated distribution is

$$\frac{1}{2}[00] + \frac{1}{2}[11]. \tag{3.27}$$

To see that this is not a product distribution, suppose we measure one of the bits. The outcome a will be completely random, i.e., either 0 or 1 with probability 50% each. However, once we know the outcome a, the state of the remaining bit is completely determined – measuring it would yield the same outcome with probability 100%. Hence the state of the remaining bit is $b = a$, which depends on the outcome a of the measurement performed on the other bit. We saw in Exercise 3.5 that this cannot be the case for a product distribution. Thus, we have proved that Eq. (3.27) describes a correlated state. In fact, the two bits are *perfectly* correlated since both measurement outcomes are completely random but always identical ($a = b$). Because of this property we say that Eq. (3.27) describes a pair of perfectly correlated random bits.

Correlated distributions arise naturally through some kind of interaction. For example, suppose you toss a fair coin, write the outcome on a piece of paper, hide the paper in an envelope, and pass the envelope to a friend. From the perspective of your friend (who knows the preparation procedure but not what is written on the paper inside the envelope), the state of your coin (bit 1) and of the piece of paper inside their envelope (bit 2) is described by

$$\frac{1}{2} \otimes \boxed{\text{heads}} + \frac{1}{2} \otimes \boxed{\text{tails}}$$

which is nothing but an amusing way of writing the two-bit state in Eq. (3.27).

How can we create correlated states in QUIRKY? Local operations alone are not enough since those can only create product states. However, we can use the controlled-NOT operation to make the two bits interact, as in the following QUIRKY computation:

Why does this work? You can figure this out in the following exercise:

Exercise 3.6: Creating perfectly correlated random bits

Explain why the above QUIRKY computation prepares the state $\frac{1}{2}[00] + \frac{1}{2}[11]$.

To check whether an arbitrary two-bit distribution

$$p = p_{00}[00] + p_{01}[01] + p_{10}[10] + p_{11}[11]$$

corresponds to a product or a correlated state, you can simply compute the following quantity:

$$\Delta(p) = p_{00}p_{11} - p_{01}p_{10}. \tag{3.28}$$

If $\Delta(p) = 0$ then p is a product distribution; otherwise it is correlated. For example, for the state in Eq. (3.27) we find that

$$\Delta\left(\frac{1}{2}[00] + \frac{1}{2}[11]\right) = \frac{1}{2}\cdot\frac{1}{2} - 0\cdot 0 = \frac{1}{4} \neq 0,$$

which confirms that it is indeed correlated and not a product state. If you are curious why this simple condition works, you can read the explanation below. But since it is not essential for understanding the rest, you can also feel free to skip it.

To prove that $\Delta(p) = 0$ is equivalent to p being a product state, we need to show two things. First, let us show that if p is a product state then $\Delta(p) = 0$. Indeed, if $p = q \otimes r$ then

$$\Delta(p) = q_0 r_0 q_1 r_1 - q_0 r_1 q_1 r_0 = 0.$$

How about the converse claim – could we have $\Delta(p) = 0$ even when p is not a product state? It turns out that this is not possible! To prove this, let us assume that $\Delta(p) = 0$ and show that $p = q \otimes r$, for some probabilistic bits q and r. Let us choose these two bits as follows:

$$\begin{aligned} q &= q_0[0] + q_1[1] = (p_{00} + p_{01})[0] + (p_{10} + p_{11})[1], \\ r &= r_0[0] + r_1[1] = (p_{00} + p_{10})[0] + (p_{01} + p_{11})[1]. \end{aligned} \tag{3.29}$$

Note from Fig. 3.3 that q and r are simply the distributions of outcomes that we would obtain if we measured the first or the second bit, respectively. Let us now verify that this choice of q and r indeed produces the state p:

$$\begin{aligned} q \otimes r &= \big((p_{00} + p_{01})[0] + (p_{10} + p_{11})[1]\big) \otimes \big((p_{00} + p_{10})[0] + (p_{01} + p_{11})[1]\big) \\ &= (p_{00} + p_{01})(p_{00} + p_{10})[00] + \cdots \\ &= (p_{00}p_{00} + p_{00}p_{10} + p_{01}p_{00} + \boldsymbol{p_{01}p_{10}})[00] + \cdots \\ &= (p_{00}p_{00} + p_{00}p_{10} + p_{01}p_{00} + \boldsymbol{p_{00}p_{11}})[00] + \cdots \\ &= p_{00}(p_{00} + p_{10} + p_{01} + p_{11})[00] + \cdots \\ &= p_{00}[00] + \cdots \\ &= p. \end{aligned}$$

Here we first used Eq. (3.25), then we multiplied out the product, next we used $\Delta(p) = 0$ to replace $\boldsymbol{p_{01}p_{10}}$ by $\boldsymbol{p_{00}p_{11}}$ (see Eq. (3.28)) , and finally we simplified $p_{00} + p_{01} + p_{10} + p_{11} = 1$, since p is a probability distribution. The three terms that we abbreviated by "…" can be treated similarly. Can you fill in the details and verify one of them yourself?

We saw before in Exercise 3.5 that p cannot be a product state if measuring the first bit "disturbs" the state of the second bit. In fact, the converse is also true, as you can show in the following homework.

Homework 3.4: Independence implies product (optional)

Assume that p is an arbitrary two-bit probability distribution such that the state of the second bit does not depend on the outcome of measuring the first bit. Show that such p is a product distribution. You can do this in two steps:

1. The measurement on the first bit can either produce outcome 0 or 1. Use Fig. 3.3 to compare the remaining state on the second bit in these two cases and show the following identities:

$$\frac{p_{00}}{p_{00}+p_{01}} = \frac{p_{10}}{p_{10}+p_{11}}, \qquad \frac{p_{01}}{p_{00}+p_{01}} = \frac{p_{11}}{p_{10}+p_{11}}.$$

2. Use these equations to show that $\Delta(p) = 0$ from Eq. (3.28).

3.2 Two quantum bits

The way we can describe two quantum bits is very similar to how we described two probabilistic bits. The main difference is that we have amplitudes instead of probabilities, meaning that they can be negative and are normalized in a different way (see §2.1.1). A general two-qubit quantum state looks as follows:

$$|\psi\rangle = \psi_{00}|00\rangle + \psi_{01}|01\rangle + \psi_{10}|10\rangle + \psi_{11}|11\rangle \tag{3.30}$$

where $\psi_{ij} \in [-1, 1]$ and

$$\psi_{00}^2 + \psi_{01}^2 + \psi_{10}^2 + \psi_{11}^2 = 1.$$

We write $|a, b\rangle$ instead of $[a, b]$ to make it clear that we are now dealing with quantum bits and not probabilistic bits. Just like we did in Eq. (3.3) for probabilistic bits, we can identify the four basis states $|a, b\rangle$ with the four basis vectors

$$|00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \qquad |01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \qquad |10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \qquad |11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}. \tag{3.31}$$

Similarly to Eq. (3.1), the general two-qubit state in Eq. (3.30) can be represented by a 4-vector

$$\begin{pmatrix}\psi_{00}\\\psi_{01}\\\psi_{10}\\\psi_{11}\end{pmatrix}.$$

However, it quickly becomes cumbersome to manipulate such vectors. They are also not so easy to visualize anymore, so we will mostly work with the $|a, b\rangle$ notation in Eq. (3.30).

Another advantage of this notation is that it is much easier to combine quantum systems. Recall from §3.1.7 that we used the tensor product operation "$\otimes$" to combine two independent probabilistic bits into a joint two-bit system. The same operation (except with $[a]$ replaced by $|a\rangle$) works also for qubits. In particular, just like we combined the basis vectors of probabilistic bits in Eq. (3.24), we can also do this for qubits:

$$|0\rangle \otimes |0\rangle = |00\rangle, \qquad |0\rangle \otimes |1\rangle = |01\rangle, \qquad |1\rangle \otimes |0\rangle = |10\rangle, \qquad |1\rangle \otimes |1\rangle = |11\rangle. \tag{3.32}$$

Notice that this corresponds to simply concatenating the bit strings. This gives you an alternative way to view the four two-qubit basis vectors in Eq. (3.31).

We can extend the tensor product operation to combine two arbitrary one-qubit states $|\alpha\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$ and $|\beta\rangle = \beta_0 |0\rangle + \beta_1 |1\rangle$ in the following way:

$$\begin{aligned} |\alpha\rangle \otimes |\beta\rangle &= (\alpha_0 |0\rangle + \alpha_1 |1\rangle) \otimes (\beta_0 |0\rangle + \beta_1 |1\rangle) \\ &= \alpha_0\beta_0 |00\rangle + \alpha_0\beta_1 |01\rangle + \alpha_1\beta_0 |10\rangle + \alpha_1\beta_1 |11\rangle . \end{aligned} \tag{3.33}$$

Two-qubit states of this form are called product states. In §3.2.3, we will discuss how to construct these states using quantum operations. Importantly, just like for two probabilistic bits, not all two-qubit states are product states.

Exercise 3.7: Tensor product and product states

Recall the states $|+\rangle$ and $|-\rangle$ from Exercise 2.1.

1. Write $|+\rangle \otimes |-\rangle$ in the same form as Eq. (3.30).
2. Is the state $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$ a product state?

In the following, we will discuss the rules for measuring and manipulating two quantum bits. While these rules will follow in complete analogy to the case of two probabilistic bits, we will find some new and surprising phenomena along the way. Happily, this week QUIRKY is also able to help us explore the world of two quantum bits! To begin, go to:

https://www.quantum-quest.org/quirky

and click on "Quest 3" and then on "Two Qubits". Your web browser will look similarly to Fig. 3.4. As compared to last week's QUIRKY, we now have two *quantum bits*, which are initialized in state $|00\rangle$. In addition, there are three new boxes: $\boxed{Z}$, $\boxed{H}$, and $\boxed{\bullet}$ (and again the mystery box is gone). We will discuss those in the remainder of this chapter.

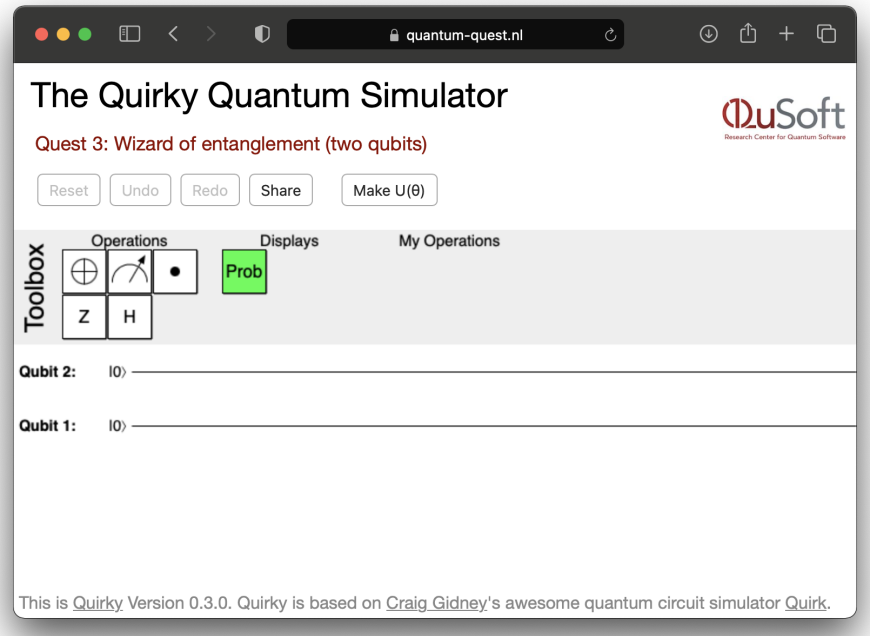
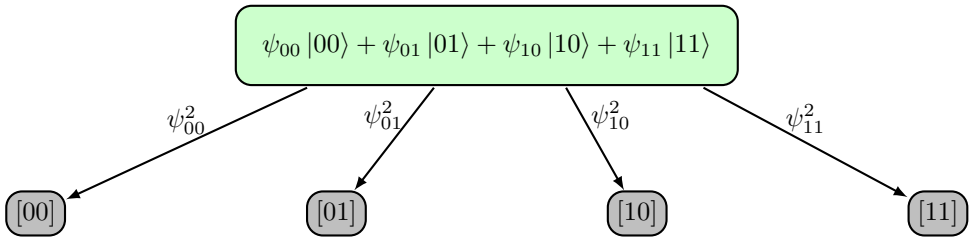


Figure 3.4: QUIRKY for Quest 3.

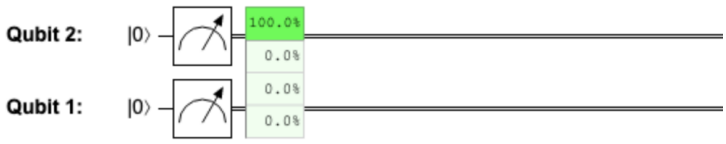
3.2.1 Measuring two qubits

Measurement for two-qubit states is defined in a very similar way to Eq. (2.7) for one-qubit states, except as an outcome you get two bits with the following probabilities:



(3.34)

Here we use light green for quantum bits and gray for the resulting two deterministic bits after the measurement. How can we measure both qubits in QUIRKY? We simply add two measurements, one for each qubit, and use the probability display just like before:



3.2.2 Local operations

If you have two qubits, a *local operation* acts only on one of them. Any single-qubit operation can be used as a local operation that acts on one out of two qubits, just like we did for probabilistic bits in §3.1.2. For example, if you recall the single-bit NOT operation in Eq. (1.9) and how we turned it into local NOT operations in Eqs. (3.6) and (3.7), we can do the exact same thing for the single-*qubit* NOT. The resulting local quantum NOT operations are very similar:

$$\begin{aligned}
&\text{NOT}_1\,|00\rangle = |10\rangle\,, \quad \text{NOT}_1\,|01\rangle = |11\rangle\,, \quad \text{NOT}_1\,|10\rangle = |00\rangle\,, \quad \text{NOT}_1\,|11\rangle = |01\rangle\,,\\
&\text{NOT}_2\,|00\rangle = |01\rangle\,, \quad \text{NOT}_2\,|01\rangle = |00\rangle\,, \quad \text{NOT}_2\,|10\rangle = |11\rangle\,, \quad \text{NOT}_2\,|11\rangle = |10\rangle\,.
\end{aligned}$$

The only difference is that we have replaced the bit notation $[ab]$ with the qubit notation $|ab\rangle$.

As an application of this, let's say we want to build all four possible two-qubit basis states out of $|00\rangle$. This can always be done by some sequence of NOT operations:

$$|00\rangle = |00\rangle\,, \quad |01\rangle = \text{NOT}_2\,|00\rangle\,, \quad |10\rangle = \text{NOT}_1\,|00\rangle\,, \quad |11\rangle = \text{NOT}_2\,\text{NOT}_1\,|00\rangle\,.$$

Note that in the last case we could have done the two NOT operations in the opposite order since either sequence corresponds to negating both bits. In other words, $\text{NOT}_2\,\text{NOT}_1 = \text{NOT}_1\,\text{NOT}_2$.

To apply a local operation in QUIRKY, we drop the corresponding box onto eiter the first or the second wire. For example, the following sequence prepares the $|10\rangle$ state and shows the outcome probabilities when measuring both qubits:

This makes sense since the bottom wire in QUIRKY corresponds to the first qubit.

This is most simple to state using the tensor product notation from Eq. (3.33) and linearity. If U is an arbitrary single-qubit operation then we define U_1 as the two-qubit operation that acts on any basis vector $|a,b\rangle = |a\rangle \otimes |b\rangle$, where $a, b \in \{0,1\}$, as follows:

$$U_1\,|a,b\rangle = U|a\rangle \otimes |b\rangle\,. \tag{3.35}$$

Just to be clear, the right-hand side means $(U|a\rangle) \otimes |b\rangle$, i.e., the tensor product of the state $U|a\rangle$ and the state $|b\rangle$. This is intuitive, as it simply means that we apply U to the first quantum bit and leave the second quantum bit alone.

To apply U_1 to an arbitrary two-qubit state, we extend this prescription *by linearity*. As in the first week, this means that we first expand $|\psi\rangle$ in the form of Eq. (3.30) and then apply the operation to each basis vector. That is,

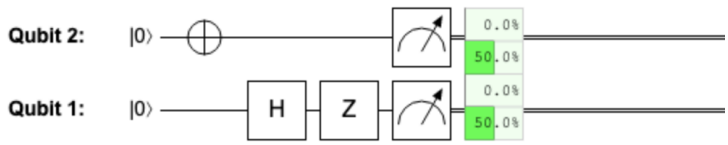
$$U_1|\psi\rangle = \psi_{00} U_1|00\rangle + \psi_{01} U_1|01\rangle + \psi_{10} U_1|10\rangle + \psi_{11} U_1|11\rangle$$

and now we can use Eq. (3.35) for each of the four terms. We similarly define U_2 by

$$U_2|a,b\rangle = |a\rangle \otimes U|b\rangle \tag{3.36}$$

and extend it by linearity. This is analogous to the formulas in Eq. (3.26) for ordinary bits.

In addition to the NOT operation, two important quantum operations that we will use all the time are the Z operation from Eq. (2.12) and the Hadamard operation from Eq. (2.20). Since they are so important, we gave them their own boxes in QUIRKY, namely $\boxed{Z}$ and $\boxed{H}$. For example, the following sequence of operations applies a NOT on the second (!) qubit, then a Hadamard on the first qubit, and finally a Z, again on the first qubit:



(3.37)

What is the state that we get in this way? The mathematical expression for this state is

$$Z_1 H_1 \mathrm{NOT}_2 |00\rangle\,. \tag{3.38}$$

Note that, in contrast to the graphical depiction (3.37), the input state $|00\rangle$ in this expression is on the right-hand side. This causes the order of operations to appear reversed. However, both (3.37) and Eq. (3.38) describe the same process – the first operation applied to $|00\rangle$ is NOT_2, then H_1, and finally Z_1. The only difference between (3.37) and Eq. (3.38) is the convention for depicting the direction of time: it goes from left to right in (3.37) and from right to left in Eq. (3.38). Unfortunately these two mismatching conventions are standard in quantum computing so there is nothing we can do about it. You simply have to be careful when translating QUIRKY pictures to equations and vice versa!

Exercise 3.8: Is QUIRKY right?

Compute the two-qubit state in Eq. (3.38) (that is, right before the measurement in (3.37)). Compute the probability of measurement outcomes and compare your result with QUIRKY.

Last week, in §2.4.3, we dicussed that any operation on a single qubit is either a rotation $U(\theta)$ or a reflection $V(\theta) = \mathrm{NOT}\, U(\theta)$. Since we can construct arbitrary rotations in QUIRKY (see §2.4.1 of last week's lecture notes), we can therefore apply arbitrary local operations on either of the qubits using QUIRKY.

In fact, the rules of Eqs. (3.35) and (3.36) do not only work for basis vectors, but in fact for arbitrary product states. That is, if $|\alpha\rangle$ and $|\beta\rangle$ are arbitrary single-qubit states then

$$U_1(|\alpha\rangle \otimes |\beta\rangle) = U|\alpha\rangle \otimes |\beta\rangle\,, \tag{3.39}$$

$$U_2(|\alpha\rangle \otimes |\beta\rangle) = |\alpha\rangle \otimes U|\beta\rangle\,. \tag{3.40}$$

Exercise 3.9: Local operations on product states (optional)

Can you verify Eq. (3.39) or (3.40)?

3.2.3 Parallel operations

If we apply one operation on the first qubit and another on the second qubit then the order of these two operations does not matter. That is, if U and V are arbitrary single-qubit operations then

$$U_1 V_2 = V_2 U_1. \tag{3.41}$$

We can verify this intuitive fact by using Eqs. (3.39) and (3.40). For every basis state $|a,b\rangle$ where $a,b \in \{0,1\}$,

$$U_1 V_2 |a,b\rangle = U_1 (|a\rangle \otimes V|b\rangle) = U|a\rangle \otimes V|b\rangle = V_2 (U|a\rangle \otimes |b\rangle) = V_2 U_1 |a,b\rangle ,$$

so Eq. (3.41) follows by linearity.

In fact, since the two operations act on different qubits, we could even do them in parallel! This suggests introducing a new notation for the operation in Eq. (3.41):

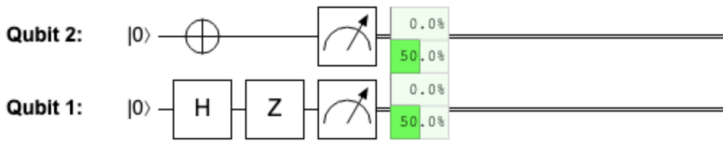
$$U \otimes V.$$

We are re-using the tensor product symbol that we originally introduced for combining two independent single-qubit states into a two-qubit state. The same meaning extends also to quantum operations: $U \otimes V$ denotes the combined operation that consists of applying U and V to two different subsystems of a larger system. This notation is particularly convenient since it interplays nicely with the original tensor product for states:

$$(U \otimes V)(|\alpha\rangle \otimes |\beta\rangle) = U|\alpha\rangle \otimes V|\beta\rangle . \tag{3.42}$$

This equation simply says that if you have two independent states and you apply an operation that acts independently on each of the two states, then you just end up applying each of the two operations on the corresponding state. We will call $U \otimes V$ a tensor product of two quantum operations or a parallel operation.

QUIRKY allows us to apply local quantum operations in parallel. For example, we could have written the sequence of operations in Eq. (3.37) as

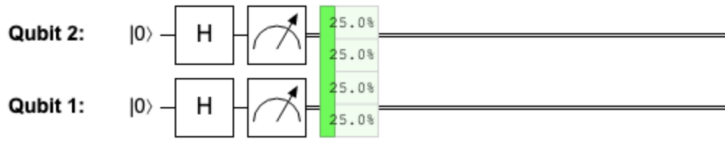


where the NOT operation and the Hadamard operation are now applied in parallel.

Let us discuss another example. What happens if we apply H to both qubits? This operation is denoted by $H \otimes H$ and according to Eqs. (2.20) and (3.42) acts as follows:

$$\begin{aligned}
(H \otimes H)|00\rangle &= (H|0\rangle) \otimes (H|0\rangle) \\
&= \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \\
&= \frac{1}{2}|0\rangle \otimes |0\rangle + \frac{1}{2}|0\rangle \otimes |1\rangle + \frac{1}{2}|1\rangle \otimes |0\rangle + \frac{1}{2}|1\rangle \otimes |1\rangle \\
&= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle).
\end{aligned}$$

The resulting state is called the uniform superposition on two qubits since it contains each of the two-qubit basis vectors with an equal amplitude. If we perform a measurement on this state we obtain all four outcomes with equal probability. We can readily verify this using QUIRKY:



Exercise 3.10: Constructing a product state

1. Write $\frac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle)$ as a tensor product of two single-qubit states.
2. How can you construct this state by applying a sequence of local operations to $|00\rangle$?
3. Implement the sequence of operations from step 2 in QUIRKY.

Eq. (3.42) shows that if we apply a parallel operation to a product state then we get another product state. In fact, starting from $|00\rangle$ we can get any product state in this way. This gives an intuitive interpretation to the product states: they are exactly those states that we can get by applying an arbitrary parallel operation to two qubits initialized in state $|00\rangle$.

Homework 3.5: Product states from parallel operations

Show that every product state can be constructed by applying some parallel quantum rotations (i.e., an operation of the form $U(\theta) \otimes U(\phi)$) to $|00\rangle$. Recall from Eq. (2.13) that $U(\theta)$ denotes a rotation by angle θ.

Hint: Recall from Eq. (2.5) that the most general one-qubit state is of the from $|\psi(\theta)\rangle$.

We close our discussion of local operations with some general remarks. First, it is not hard to check that

$$(U \otimes V)(U' \otimes V') = UU' \otimes VV', \tag{3.43}$$

for any four single-qubit operations U, U', V, V'. Can you draw a picture to visualize what is going on here?

We can now make use of the identity operation I from Eq. (2.18) which acts as $I|\psi\rangle = |\psi\rangle$ (this trivially extends to I_1 and I_2 on two qubits). It is rather useless on its own, but can be conveniently used in tensor product notation. For example, it allows us to write

$$U_1 = U \otimes I \qquad \text{and} \qquad V_2 = I \otimes V,$$

which makes it very clear that, e.g., U_1 applies the U-operation on the first qubit and nothing on the second qubit. For example, the identity $U_1V_2 = V_2U_1$ from Eq. (3.41) now becomes

$$(U \otimes I)(I \otimes V) = U \otimes V = (I \otimes V)(U \otimes I) \tag{3.44}$$

which is rather intuitive.

You might wonder if the order of operations actually matters if we apply them to the *same* qubit? If we consider two rotations then it follows from Eq. (2.15) that their order is not important since

$$U(\theta)U(\theta') = U(\theta + \theta') = U(\theta' + \theta) = U(\theta')U(\theta).$$

However, if U and V are two arbitrary single-qubit operations (in particular, one of them is a reflection) then their composition generally depends on the order (see Exercise 3.11 below). That is,

$$UV \neq VU.$$

This issue of course persists also when you have another qubit around, but are still acting with both operations on the same qubit. For example,

$$(U \otimes I)(V \otimes I) = UV \otimes I \neq VU \otimes I = (V \otimes I)(U \otimes I). \tag{3.45}$$

We can also write this as $U_1 V_1 \neq V_1 U_1$ (and similarly when we apply both operations to the second qubit).

To understand intuitively the difference between Eqs. (3.44) and (3.45), imagine that U means "putting on a sock" and V means "putting on a shoe". Clearly, when U and V are applied to the same foot, you will get different results depending on the order! However, if you apply U and V to different feet (e.g., you use $U \otimes I$ and $I \otimes V$) then you will get the same result irrespectively of the order of the two operations. Either way, if you want to be properly dressed, you should apply $(U \otimes U)$ and then $(V \otimes V)$.

Exercise 3.11: The order is important

Show that $HZ \neq ZH$.

3.2.4 Controlled operations

To go beyond product states, we need an operation that allows the two quantum bits to interact. As before (see Eq. (2.18) for probabilistic bits), we will use a *controlled-NOT* operation for this, which we define in complete analogy to Eqs. (3.18) and (3.19):

$$\begin{aligned} \mathrm{CNOT}_{1\to 2}\ |00\rangle &= |00\rangle\,, \\ \mathrm{CNOT}_{1\to 2}\ |01\rangle &= |01\rangle\,, \\ \mathrm{CNOT}_{1\to 2}\ |10\rangle &= |11\rangle\,, \\ \mathrm{CNOT}_{1\to 2}\ |11\rangle &= |10\rangle\,, \end{aligned} \tag{3.46}$$

or, more concisely,

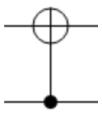
$$\mathrm{CNOT}_{1\to 2}\ |a,b\rangle = |a, a\oplus b\rangle\,. \tag{3.47}$$

Thus, on basis states, the operation $\mathrm{CNOT}_{1\to 2}$ toggles the second qubit controlled on the value of the first qubit. We can also define an operation $\mathrm{CNOT}_{2\to 1}$ which uses the second qubit as the control and the first as the target, i.e.,

$$\mathrm{CNOT}_{2\to 1}\ |a,b\rangle = |a\oplus b, b\rangle\,. \tag{3.48}$$

As usual, we extend these formulas by linearity to arbitrary two-qubit states.

In QUIRKY, you can build a controlled-NOT operation for quantum bits in the same way as you learned for ordinary bits – see §3.1.6 in case you don't remember. For example, the $\mathrm{CNOT}_{1\to 2}$ operation for quantum bits looks just like before:



Many of the things that we proved for probabilistic bits are still true for quantum bits. E.g., your solution to Homework 3.2 will just as well allow you to swap two quantum bits! Another example of this is the fact that doing the same controlled-NOT operation twice amounts to doing nothing. For example, for $\mathrm{CNOT}_{1\to 2}$ this is the case because

$$\mathrm{CNOT}_{1\to 2}\,\mathrm{CNOT}_{1\to 2}\,|a,b\rangle = \mathrm{CNOT}_{1\to 2}\,|a, a\oplus b\rangle = |a, a\oplus a\oplus b\rangle = |a,b\rangle$$

since $a \oplus a = 0$ for any $a \in \{0,1\}$. As a consequence, the controlled-NOT operation is the inverse of itself:

$$\mathrm{CNOT}_{1\to 2}^{-1} = \mathrm{CNOT}_{1\to 2} \tag{3.49}$$

where M^{-1} denotes the inverse of operation M (see §2.4.2).

If you play around a bit with QUIRKY, you may have noticed that you can combine the $\boxed{\bullet}$ with arbitrary single-qubit operations, not just with the NOT operation. Indeed, we can define a controlled-U operation for any single-qubit operation U. They are denoted by $CU_{1\to 2}$ and $CU_{2\to 1}$, depending on which qubit is the control and which is the target. For example, $CU_{1\to 2}$ is defined as follows on the four basis states:

$$\begin{aligned}
CU_{1\to 2}\,|00\rangle &= |0\rangle \otimes |0\rangle\,, \\
CU_{1\to 2}\,|01\rangle &= |0\rangle \otimes |1\rangle\,, \\
CU_{1\to 2}\,|10\rangle &= |1\rangle \otimes U\,|0\rangle\,, \\
CU_{1\to 2}\,|11\rangle &= |1\rangle \otimes U\,|1\rangle\,.
\end{aligned}$$

You can quickly verify that for $U = \mathrm{NOT}$ we recover our definition of $\mathrm{CNOT}_{1\to 2}$ from before.

3.2.5 Entangled states

In Eq. (3.33) we used the tensor product to build a two-qubit state from two single-qubit states. In §3.2.3, we saw that these *product states* are exactly those states that can be constructed by applying local quantum operations to $|00\rangle = |0\rangle \otimes |0\rangle$ (a product state itself). Now, there also exist two-qubit states that are *not* product states. We call these states entangled, and as we will see they are very important for quantum computing.

How can we determine if a state is a product state or not? Even though quantum states are specified by amplitudes and not by probabilities, we can use the same method that we used in §3.1.8 for detecting whether a probability distribution is correlated. Given a two-qubit state in the form (3.30), we first use Eq. (3.28) to compute

$$\Delta(|\psi\rangle) = \psi_{00}\psi_{11} - \psi_{01}\psi_{10}. \tag{3.50}$$

Then, $|\psi\rangle$ is a product state if and only if $\Delta(|\psi\rangle) = 0$. In this way, entangled states are analogous to correlated probability distributions.

A simple but important example of an entangled two-qubit state is

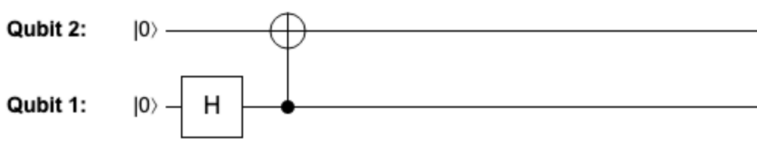
$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle\,. \tag{3.51}$$

This state is analogous to a pair of perfectly correlated random bits from Eq. (3.27). It is entangled since

$$\Delta(|\Phi^+\rangle) = \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} - 0 \cdot 0 = \frac{1}{2} \neq 0.$$

We call $|\Phi^+\rangle$ the maximally entangled state of two qubits (although neither the reason for this name nor the peculiar notation are very clear at this point).

How can we build entangled states? Just like when we wanted to build correlated states of two bits, we can use the controlled-NOT operation to make two quantum bits interact. For example, the following sequence of operations prepares the maximally entangled state $|\Phi^+\rangle$:



Let us quickly verify this:

$$\mathrm{CNOT}_{1\to 2}\,(H\otimes I)\,|00\rangle = \mathrm{CNOT}_{1\to 2}\left(\frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|10\rangle\right) = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle\,.$$

Note that applying the CNOT directly to $|00\rangle$, or for that matter to any other basis state, would have *not* worked (see Eq. (3.46)).

Homework 3.6: Another entangled state

1. Verify that the state $|\psi\rangle = \frac{1}{2}\,|01\rangle + \frac{\sqrt{3}}{2}\,|10\rangle$ is entangled.
   Hint: Compute Eq. (3.50).
2. Find a sequence of operations in QUIRKY that prepares the state $|\psi\rangle$.
   Hint: You may need to create a suitable rotation.
3. What are the probabilities of measurement outcomes when measuring both qubits of $|\psi\rangle$? Use QUIRKY to confirm your result.

The maximally entangled state in Eq. (3.51) is one of a family of four states, called the Bell states. The Bell states are defined as follows:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle\,, \tag{3.52}$$

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}\,|00\rangle - \frac{1}{\sqrt{2}}\,|11\rangle\,, \tag{3.53}$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}\,|01\rangle + \frac{1}{\sqrt{2}}\,|10\rangle\,, \tag{3.54}$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}\,|01\rangle - \frac{1}{\sqrt{2}}\,|10\rangle\,. \tag{3.55}$$

They are named after John Steward Bell, who was one of the first to recognize the remarkable features of quantum entanglement. How can we create these four Bell states? We saw above that we can obtain $|\Phi^+\rangle$ by applying a Hadamard and a CNOT operation to the basis state $|00\rangle$. It is an easy exercise to verify that the other three Bell states can be constructed similarly, that is, by applying the same sequence of operations to the other three basis states. In other words, if we define

$$U_{\text{Bell}} = \mathrm{CNOT}_{1\to 2}\,(H\otimes I) \tag{3.56}$$

then

$$|\Phi^+\rangle = U_{\text{Bell}}\,|00\rangle\,, \qquad |\Phi^-\rangle = U_{\text{Bell}}\,|10\rangle\,,$$
$$|\Psi^+\rangle = U_{\text{Bell}}\,|01\rangle\,, \qquad |\Psi^-\rangle = U_{\text{Bell}}\,|11\rangle\,.$$

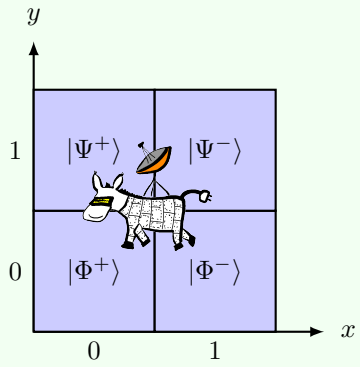
Exercise 3.12: Preparing Bell states

Draw how you would create the other three Bell states in QUIRKY: $|\Phi^-\rangle$, $|\Psi^+\rangle$, and $|\Psi^-\rangle$.

Exercise 3.13: Discriminating Bell states

Alice's donkey robot got lost while on an exploration mission! It wants to urgently let Alice know of its position so that she can come to its rescue. The donkey is located in one of the four neighborhoods surrounding the school. To communicate which one, the donkey sends a two-qubit quantum message $|x,y\rangle$, where $x\in\{0,1\}$ tells the x coordinate and $y\in\{0,1\}$

tells the y coordinate of its location:



Unfortunately, Alice's evil classmate Eve is jamming the signal and what Alice receives instead is one of the four Bell states as shown above. Help Alice to correctly decode the signal and locate her donkey robot! That is, find a sequence of operations that maps each of the four Bell states back onto the corresponding basis state $|x, y\rangle$.

3.2.6 Entanglement and correlations

Given the similarity between entangled states and correlated probability distributions, you may wonder how these two notions are related. To compare them, let us discuss more generally the relation between quantum states and probability distributions.

To start, suppose we have a single-qubit state $|\psi\rangle = \psi_0 |0\rangle + \psi_1 |1\rangle$ and we measure it. Then we know from §2.2 that we get as outcome a bit that is either zero or one, with probabilities ψ_0^2 and ψ_1^2. We can model this as a probability distribution

$$\psi_0^2[0] + \psi_1^2[1].$$

Intuitively, this models the situation where we measured the qubit but we did not actually look at the outcome (if we did, we would not have a probabilistic bit but a deterministic one that is either in state zero or in state one).

The same logic works just as well for two qubits. If we measure a two-qubit state $|\psi\rangle = \psi_{00} |00\rangle + \psi_{01} |01\rangle + \psi_{10} |10\rangle + \psi_{11} |11\rangle$, we can describe the measurement outcomes by the probability distribution

$$p = \psi_{00}^2[00] + \psi_{01}^2[01] + \psi_{10}^2[10] + \psi_{11}^2[11]. \tag{3.57}$$

For example, if we prepare and measure the maximally entangled state $|\Phi^+\rangle$, we obtain a perfectly correlated pair of random bits in Eq. (3.27). We can verify this using QUIRKY:

Clearly the same is true if we measure the Bell state $|\Phi^-\rangle$ instead. (How about the other two Bell states $|\Psi^+\rangle$ or $|\Psi^-\rangle$? Measuring either of them produces perfectly *anti-correlated* bits, described by the probability distribution $\frac{1}{2}[01] + \frac{1}{2}[10]$.)

The preceding example was not an accident. In fact, the probability distribution p in Eq. (3.57) that is obtained by measuring a two-qubit quantum state can be correlated only if the

corresponding quantum state $|\psi\rangle$ is entangled. To see this, assume that $|\psi\rangle$ is a product state, so that $\Delta(|\psi\rangle) = 0$. Then p is a product distribution since

$$\begin{aligned}\Delta(p) &= p_{00}p_{11} - p_{01}p_{10} = \psi_{00}^2\psi_{11}^2 - \psi_{01}^2\psi_{10}^2 \\ &= (\psi_{00}\psi_{11} - \psi_{01}\psi_{10})\,(\psi_{00}\psi_{11} + \psi_{01}\psi_{10}) \\ &= \Delta(|\psi\rangle)\,(\psi_{00}\psi_{11} + \psi_{01}\psi_{10}) = 0.\end{aligned} \tag{3.58}$$

This proves the claim that correlated measurement outcomes imply presence of entanglement in the measured state.

Note that generally quantum states are at least as useful as probabilistic bits because any probability distribution can be obtained by measuring an appropriately chosen quantum state. That is, given any probability distribution p we can always find a quantum state $|\psi\rangle$ whose measurement outcomes are distributed according to p. For example, for a two-bit distribution p we can simply choose

$$|\psi\rangle = \sqrt{p_{00}}\,|00\rangle + \sqrt{p_{01}}\,|01\rangle + \sqrt{p_{10}}\,|10\rangle + \sqrt{p_{11}}\,|11\rangle\,.$$

In particular, this means that entanglement is generally at least as useful as probabilistic correlations since any correlated distribution over two probabilistic bits can be produced by measuring some entangled two-qubit state.

3.2.7 The power of entanglement

In fact, entangled quantum bits are truly more powerful than probabilistic bits! In the following we will only have a glimpse, but we will see many more examples in the coming weeks.

Let us illustrate the power of entanglement by a little story which follows up on Exercise 3.13. There, you helped Alice decode the position of her donkey robot, which was in one out of four locations, labeled by two bits $a, b \in \{0, 1\}$. Alice does not have time to pick up the donkey, so she wants to send the position onwards to Bob. Unfortunately, Alice is running low on her quantum mobile phone plan, so she can only send a single qubit to Bob. But sending a single quantum bit will certainly *not* be enough to perfectly communicate two bits. We know this because the only way for Bob to extract information is by measuring the qubit – but from the measurement he only learns a single bit and then the original state of the qubit is gone.

Happily enough, Alice and Bob had the foresight of sharing a maximally entangled state $|\Phi^+\rangle$. By this we mean that Alice is in possession of the first qubit and Bob is in possession of the second qubit. Can we help Alice send the donkey's location (i.e., the two bits a and b) by transmitting a *single* qubit? This can indeed be done and the key ingredient is the following:

Exercise 3.14: Transforming one Bell state into any other

Show that Alice can, by applying local operations on her qubit alone, transform the maximally entangled state $|\Phi^+\rangle$ into any of the three other Bell states $|\Phi^-\rangle$, $|\Psi^+\rangle$, and $|\Psi^-\rangle$.

It is now clear how Alice and Bob can solve the challenge. Assume that Alice and Bob have beforehand agreed on a mapping between the four possible values of the two bits $[ab]$ and the four Bell states (say, $[00]$ corresponds to $|\Phi^+\rangle$, $[01]$ corresponds to $|\Psi^+\rangle$, and so on). Using Exercise 3.14, Alice first applies an operation on her part of the maximally entangled state to turn the state into the Bell state that corresponds to the donkey's location. Next, she sends her quantum bit over to Bob. Now Bob has both quantum bits in his possession and he knows that they are in one of the four Bell states. Thus, he can simply apply the same operations as in Exercise 3.13 and measure the two qubits to figure out the donkey robot's location.

The procedure we have just described is known as the superdense coding protocol, since we are transmitting two deterministic bits by sending a single quantum bit (at the cost of using up one maximally entangled state shared between Alice and Bob). Could Alice and Bob have

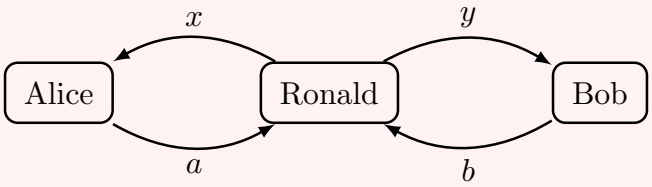
just as well used probabilistic bits instead of an entangled state to solve this challenge (e.g., a pair of perfectly correlated bits)? Interestingly, this is not the case. Thus, superdense coding demonstrates that quantum entanglement provides an advantage for communication tasks.

In the following homework problem you can encounter another famous situation in which quantum entanglement provides an unambiguous advantage. The problem may perhaps look a bit daunting – but this is mostly because we could not resist telling a little story around it. As usual, do not hesitate to ask any questions on Discord!

Homework 3.7: An entangled game (challenging)

Alice and Bob are bored in class, so they ask their quantum mechanics teacher Ronald to pose them a challenging puzzle. After only a brief pause, Ronald explains to them an interesting game. The goal of the game is for Alice and Bob to cooperate as well as possible (they are *not* playing against each other). However, they are *not* allowed to communicate with each other while the game is going on! The rules of the game are as follows:

- To start, Ronald secretly tosses two fair coins. He tells Alice the result of the first coin toss (bit x) and Bob the result of the second coin toss (bit y). We call these input bits.
- After receiving the bits, Alice and Bob each have to answer with a bit of their own (bits a and b).
- Alice and Bob win the game under the following condition: if $x = y = 1$ then they win the game if $a \neq b$; otherwise, they win the game if $a = b$.



x	y	Winning condition
0	0	$a = b$
0	1	$a = b$
1	0	$a = b$
1	1	$a \neq b$

Before the game starts, Alice and Bob briefly get together to discuss their strategy. First, they consider simply applying two functions $f, g : \{0,1\} \to \{0,1\}$ on their input bits x and y and compute their answers as follows: $a = f(x)$ and $b = g(y)$.

1. Show that in this case they can win the game with probability 75%, but no higher.

Next, they consider coordinating their play using shared randomness. Bob suggests using more complicated functions f and g with an extra binary argument and to compute their answers as follows: $a = f(x, r)$ and $b = g(y, s)$, where r and s are two random bits jointly described by some two-bit probability distribution.

2. Show that they still cannot win with probability higher than 75%, no matter what the functions f and g are and what the joint distribution on the bits r and s is.

It begins to dawn on our protagonists that surely Ronald had a quantum-mechanical strategy in mind. Alice has an ingenious idea and proposes that they should share a maximally entangled state $|\Phi^+\rangle$ before the game starts. She proposes that upon receiving their bits, she will rotate her qubit by some angle θ_x (that depends on her input bit x) and measure it to obtain her answer a, while Bob should instead rotate by some other angle ω_y (depending on his input bit y) and then measure to obtain his answer b.

3. Write down the state after Alice and Bob applied their rotations in the form (3.30). Confirm that the probability of winning the game is

$$\frac{1}{4}\left(\cos^2(\theta_0 - \omega_0) + \cos^2(\theta_0 - \omega_1) + \cos^2(\theta_1 - \omega_0) + \sin^2(\theta_1 - \omega_1)\right).$$

Hint: Use the trigonometric formulas from Eqs. (2.16) and (2.22).

Alice and Bob quickly figure out that $\theta_0 = 0$, $\theta_1 = \pi/4$, and $\omega_0 = \pi/8$ are good choices, but they are struggling with the last angle and time is quickly running out.

4. Find an angle ω_1 such that they win the game with probability higher than 75%.

In the first two parts of the homework, you showed that any strategy using classical bits cannot win the game with probability higher than 75%. This is an example of a *Bell inequality*. The version above goes back to John Clauser, Michael Horne, Abner Shimony, and Richard Holt, so the game that Alice and Bob play with Ronald is often called the *CHSH game*. The quantum mechanical strategy that you discovered in the homework *violates* this Bell inequality. It is an empirical fact that Bell inequalities can indeed be violated by quantum mechanics. This was first demonstrated by Alain Aspect in the 80s and recently confirmed under very stringent conditions in a beautiful experiment by Ronald Hanson and his team at TU Delft.

Interestingly, Alice and Bob can not only play the CHSH game better using a shared maximally entangled state, but they can in fact convince Ronald that they must be using some quantum tricks to play the game so well. Indeed, since using only probabilistic strategies they cannot win the game with more than 75% probability, if they somehow succeed winning it more often then the only possible explanation for this is that they must be using something more powerful. In fact, using some further tricks they can even convince Ronald that they must be using the maximally entangled state and applying the rotations with the particular angles. Effectively, this means that they can prove to Ronald in an irrefutable way that they have small quantum computers that can manipulate single qubits and share entanglement. This is a very important observation, since it lets you use the CHSH game to verify that somebody genuinely has built a quantum computer! Indeed, if two of your classmates would claim that they have each built a quantum computer in their garages, you could simply ask them both to play together this game against you. If they manage to win with probability that is significantly larger than 75%, you would know that they indeed are not lying to you and they have real quantum computers. But if they cannot win it with more than 75% then you would easily refute their claims. Isn't this amazing?

These types of verification procedures is something that researchers around the world are currently actively working on. This is a very important problem because various large companies, such as IBM, Google, and Microsoft are trying to build a quantum computer. If they claim they have built one, you would probably believe them more than your classmates. However, it is still great if you can actually verify this!

3.3 Exercise solutions

Solution to Exercise 3.2

1. Let us denote the probability distributions of Alice' two coin flips by p. We use Fig. 3.3 to figure out the probabilities. We know that Alice first coin flip is distributed according to u, which means that if we measure the first bit then we should get both outcomes with probability $1/2$:

$$p_{00} + p_{01} = \frac{1}{2}, \quad p_{10} + p_{11} = \frac{1}{2}.$$

Next, we know that if the first coin flip gave outcome 0 then the state of the second bit should be described by the coin q. Thus we must have

$$\frac{p_{00}[0] + p_{01}[1]}{p_{00} + p_{01}} = \frac{3}{4}[0] + \frac{1}{4}[1],$$

from which we deduce that $p_{00} = \frac{3}{8}$ and $p_{01} = \frac{1}{8}$. Similarly, if the first coin flip gave outcome 1 then the state of the second bit is described by coin r, so

$$\frac{p_{10}[0] + p_{11}[1]}{p_{00} + p_{01}} = \frac{1}{3}[0] + \frac{2}{3}[1],$$

hence $p_{10} = \frac{1}{6}$ and $p_{11} = \frac{2}{6}$. Together,

$$p = \frac{3}{8}[00] + \frac{1}{8}[01] + \frac{1}{6}[10] + \frac{2}{6}[11],$$

2. The value of Bob's bit is the same as the outcome of Alice' second coin flip, so we simply follow Fig. 3.3 to compute the probability of outcomes when measuring the second bit. Thus, the probability that Bob's outcome is 0 is

$$p_{00} + p_{10} = \frac{3}{8} + \frac{1}{6} = \frac{13}{24}$$

and the probability that his outcome is 1 is $1 - \frac{13}{24} = \frac{11}{24}$. We can write this as the probability distribution

$$\frac{13}{24}[0] + \frac{11}{24}[1].$$

3. Again we use Fig. 3.3. If the second bit is measured and the result is outcome 0 then the state of the first bit is given by

$$\frac{p_{00}[0] + p_{10}[1]}{p_{00} + p_{10}} = \frac{9}{13}[0] + \frac{4}{13}[1].$$

Thus it is more likely that Alice' first coin flip shows 0.

Similarly, if the result of measuring the second bit is 1 then the state of the first bit is described by

$$\frac{p_{01}[0] + p_{11}[1]}{p_{01} + p_{11}} = \frac{3}{11}[0] + \frac{8}{11}[1].$$

In this case, it is more likely that Alice' first coin flip shows 1.

Solution to Exercise 3.1

$$\mathrm{NOT}_1 \begin{pmatrix} p_{00} \\ p_{01} \\ p_{10} \\ p_{11} \end{pmatrix} = \begin{pmatrix} p_{10} \\ p_{11} \\ p_{00} \\ p_{01} \end{pmatrix}.$$

Solution to Exercise 3.3

$$\mathrm{SWAP} \begin{pmatrix} p_{00} \\ p_{01} \\ p_{10} \\ p_{11} \end{pmatrix} = \begin{pmatrix} p_{00} \\ p_{10} \\ p_{01} \\ p_{11} \end{pmatrix}.$$

Solution to Exercise 3.4

1. $\mathrm{CNOT}_{2\to1}[a,b] = [a \oplus b, b] = [b \oplus a, b]$.
2. This can be done by first applying a SWAP, then a $\mathrm{CNOT}_{1\to2}$, and finally another SWAP. Indeed, using Eqs. (3.17) and (3.19),

$$\begin{aligned} &\mathrm{SWAP}(\mathrm{CNOT}_{1\to2}(\mathrm{SWAP}[a,b])) = \mathrm{SWAP}(\mathrm{CNOT}_{1\to2}[b,a]) \\ &= \mathrm{SWAP}[b, b \oplus a] = [b \oplus a, b]. \end{aligned}$$

Solution to Exercise 3.5

Using Fig. 3.3, the state of the second bit after the measurement can be computed as

$$\frac{p_{a0}[0] + p_{a1}[1]}{p_{a0} + p_{a1}} = \frac{q_a r_0[0] + q_a r_1[1]}{q_a r_0 + q_a r_1} = \frac{r_0[0] + r_1[1]}{r_0 + r_1} = r_0[0] + r_1[1] = r,$$

where we cancelled q_a and used $r_0 + r_1 = 1$.

Solution to Exercise 3.6

The state before the controlled-NOT operation is

$$\left(\frac{1}{2}[0] + \frac{1}{2}[1]\right) \otimes [0] = \frac{1}{2}[00] + \frac{1}{2}[10].$$

After applying the controlled-NOT operation, we obtain

$$\mathrm{CNOT}_{1\to2}\left(\frac{1}{2}[00] + \frac{1}{2}[10]\right) = \frac{1}{2}[00] + \frac{1}{2}[11].$$

Solution to Exercise 3.7

1.

$$\begin{aligned}|+\rangle \otimes |-\rangle &= \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) \\ &= \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle .\end{aligned}$$

2.

$$\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \otimes |+\rangle .$$

So this is indeed a product state.

Solution to Exercise 3.8

We start with $|00\rangle$. After the NOT on the second qubit, we get $|01\rangle$. The Hadamard on the first qubit transforms this into $|+\rangle \otimes |1\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|11\rangle$ and with the final Z operation we obtain the following two-qubit state right before the measurement:

$$\frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|11\rangle .$$

Thus, we get either 01 or 11, with probability 50% each.

Solution to Exercise 3.9

We only show how to verify Eq. (3.39) (the other equation can be derived completely analogously). For this, let us write $|\alpha\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ and $|\beta\rangle = \beta_0|0\rangle + \beta_1|1\rangle$. Then,

$$\begin{aligned}U_1(|\alpha\rangle \otimes |\beta\rangle) &= U_1(\alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle) \\ &= \alpha_0\beta_0 U_1|00\rangle + \alpha_0\beta_1 U_1|01\rangle + \alpha_1\beta_0 U_1|10\rangle + \alpha_1\beta_1 U_1|11\rangle \\ &= \alpha_0\beta_0 U|0\rangle \otimes |0\rangle + \alpha_0\beta_1 U|0\rangle \otimes |1\rangle + \alpha_1\beta_0 U|1\rangle \otimes |0\rangle + \alpha_1\beta_1 U|1\rangle \otimes |1\rangle \\ &= \alpha_0 U|0\rangle \otimes (\beta_0|0\rangle + \beta_1|1\rangle) + \alpha_1 U|1\rangle \otimes (\beta_0|0\rangle + \beta_1|1\rangle) \\ &= \alpha_0 U|0\rangle \otimes |\beta\rangle + \alpha_1 U|1\rangle \otimes |\beta\rangle \\ &= (\alpha_0 U|0\rangle + \alpha_1 U|1\rangle) \otimes |\beta\rangle \\ &= U|\alpha\rangle \otimes |\beta\rangle\end{aligned}$$

by Eq. (3.33), the definition of U_1, and linearity.

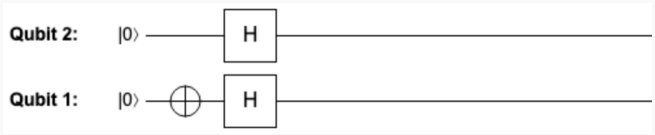
Solution to Exercise 3.10

1. The state can be written as

$$\frac{1}{2}\left(|00\rangle + |01\rangle - |10\rangle - |11\rangle\right) = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right).$$

2. This is equal to

$$H|1\rangle \otimes H|0\rangle = H\,\mathrm{NOT}\,|0\rangle \otimes H|0\rangle = (H\,\mathrm{NOT} \otimes H)\,|00\rangle.$$

3. 

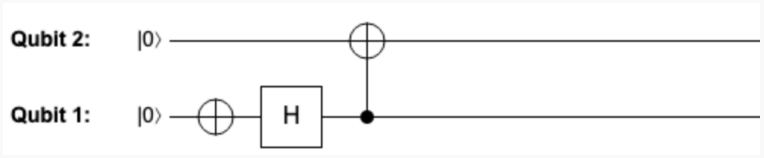
Solution to Exercise 3.11

To show that $HZ \neq ZH$ it is enough to verify that they give different results when applied to the $|0\rangle$ state. Indeed:

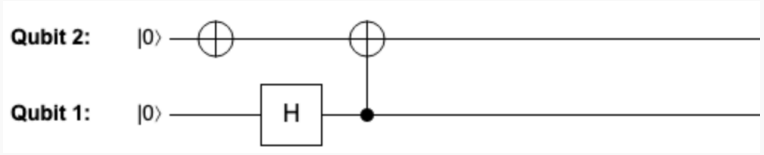
$$HZ|0\rangle = H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$
$$ZH|0\rangle = Z\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

Solution to Exercise 3.12

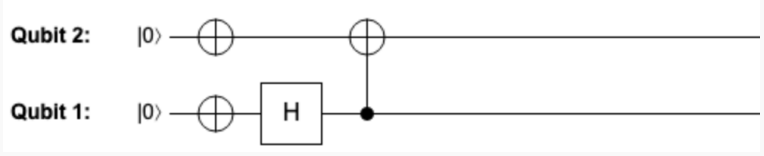
- $|\Phi^-\rangle$:



- $|\Psi^+\rangle$:



- $|\Psi^-\rangle$:



Solution to Exercise 3.13

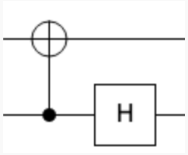
Note that this is the same as inverting the operation U_{Bell} in Eq. (3.56). Recall that $U_{\text{Bell}} = \text{CNOT}_{1\to 2}\,(H \otimes I)$. This is a composite operation, so recall from Exercise 2.5 that

$$U_{\text{Bell}}^{-1} = (H \otimes I)^{-1}\,\text{CNOT}_{1\to 2}^{-1} = (H^{-1} \otimes I)\,\text{CNOT}_{1\to 2}^{-1}.$$

Recall from Eq. (3.49) that $\text{CNOT}_{1\to 2}$ is the inverse of itself, i.e., $\text{CNOT}_{1\to 2}^{-1} = \text{CNOT}_{1\to 2}$. It is also true that $H^{-1} = H$ (this holds for any reflection, in fact). This implies that we can undo U_{Bell} by applying the same two operations but in reverse order:

$$U_{\text{Bell}}^{-1} = (H \otimes I)\,\text{CNOT}_{1\to 2}.$$

That is, we first apply $\text{CNOT}_{1\to 2}$ and then H on the first qubit, as in:



Solution to Exercise 3.14

Note from Eqs. (3.52) to (3.55) that the Bell states differ only by bit flips and signs. Recall that we can flip a bit using a local NOT operation and that we can introduce some signs by applying local Z operations, where Z is defined in Eq. (2.12). To create $|\Phi^-\rangle$, Alice can simply apply a Z operation on her qubit:

$$(Z \otimes I)\,|\Phi^+\rangle = (Z \otimes I)\left(\frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle\right) = \frac{1}{\sqrt{2}}\,|00\rangle - \frac{1}{\sqrt{2}}\,|11\rangle = |\Phi^-\rangle\,.$$

To create $|\Psi^+\rangle$, Alice instead applies an NOT operation on her qubit:

$$(\text{NOT} \otimes I)\,|\Phi^+\rangle = (\text{NOT} \otimes I)\left(\frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle\right) = \frac{1}{\sqrt{2}}\,|10\rangle + \frac{1}{\sqrt{2}}\,|01\rangle = |\Psi^+\rangle\,.$$

To create $|\Psi^-\rangle$, Alice applies first an NOT operation and then a Z operation on her qubit:

$$(Z\,\text{NOT} \otimes I)\,|\Phi^+\rangle = (Z \otimes I)\left(\frac{1}{\sqrt{2}}\,|10\rangle + \frac{1}{\sqrt{2}}\,|01\rangle\right) = -\frac{1}{\sqrt{2}}\,|10\rangle + \frac{1}{\sqrt{2}}\,|01\rangle = |\Psi^-\rangle\,.$$

Quest 4: Quantum composer

Last week, we discussed the states of two quantum bits and which operations you can perform with them. We also learned that there exist *entangled* quantum states and that they can be used to communicate more efficiently (e.g., by transmitting two bits when sending only a single qubit) and to outperform deterministic or probabilistic strategies at a certain game. This week, we will allow ourselves an arbitrary number of qubits and start composing quantum circuits consisting of many operations and use them to solve interesting problems.

4.1 Quantum circuits

When you have many qubits and you are applying lots of operations on them, you can easily lose track of what is going on. Just like a composer would write sheet music to neatly arrange the instructions for each musician of an orchestra telling them exactly what note to play at any given time, so do quantum circuits provide a nice pictorial description of what operation should be applied to any qubit at any given time. Different operations are like different notes and different qubits are like different musicians in an orchestra!

Unlike in an orchestra, however, some quantum operations affect two (or sometimes even three!) qubits at a time. This would be like asking the guitar player to strum the violin while the violinist is bowing the guitar! Such interactions between musicians (or qubits) can be a lot of fun and produce a much richer and complicated sound. However, they can be hard to perform in practice (as you can imagine, the violinist might have to run across the stage to reach the guitar player!). In addition, during this hectic collaboration between the two musicians, the violinist's hair might get entangled with guitar player's large earrings forcing them to play the rest of the song together.

The more interactions you perform between the qubits, the more entangled they will usually get. At the end of the song, typically all qubits end up being entangled with each other in such a way that it is completely impossible to tell them apart. The only way to sort them out and get them back to any reasonable state is by measuring them! Indeed, this is how a general quantum computation proceeds. More formally, a *quantum circuit* consists of three ingredients:

1. an initial state, which is typically $|0\rangle$ on each of the qubits,

2. a sequence of quantum operations, where each operation typically acts on a few qubits at a time (usually, one or two qubits),

3. measurements to read the information out (typically we measure all qubits).

We can represent quantum circuits graphically using the same pictures that we already know from QUIRKY. When discussing quantum circuits, operations and measurements are often called quantum gates (e.g., 'Hadamard gate' instead of 'Hadamard operation'). Let us now look more closely at the three ingredients and see how they play out in a large orchestra of qubits.

4.1.1 Many quantum bits

The rules that we learned so far for one and two qubits generalize very naturally to quantum system of many qubits. For example, an arbitrary state of three qubits can be written as follows:

$$\begin{aligned} |\psi\rangle = {} & \psi_{000}\,|000\rangle + \psi_{001}\,|001\rangle + \psi_{010}\,|010\rangle + \psi_{011}\,|011\rangle \\ & + \psi_{100}\,|100\rangle + \psi_{101}\,|101\rangle + \psi_{110}\,|110\rangle + \psi_{111}\,|111\rangle\,, \end{aligned} \tag{4.1}$$

where $\psi_{ijk} \in [-1, 1]$ and the squares of these amplitudes again sum to one, i.e.,

$$\psi_{000}^2 + \psi_{001}^2 + \psi_{010}^2 + \psi_{011}^2 + \psi_{100}^2 + \psi_{101}^2 + \psi_{110}^2 + \psi_{111}^2 = 1.$$

Note that, in total, there are $8 = 2^3$ amplitudes, one for each bitstring of three bits. We can also think of $|\psi\rangle$ as a vector with eight entries:

$$|\psi\rangle = \begin{pmatrix} \psi_{000} \\ \psi_{001} \\ \psi_{010} \\ \psi_{011} \\ \psi_{100} \\ \psi_{101} \\ \psi_{110} \\ \psi_{111} \end{pmatrix}.$$

More generally, a state of n qubits can be specified by using 2^n amplitudes $\psi_{a_1,\ldots,a_n}$, one for each bitstring of n bits:

$$|\psi\rangle = \psi_{00\ldots00}\,|00\ldots00\rangle + \psi_{00\ldots01}\,|00\ldots01\rangle + \ldots + \psi_{11\ldots11}\,|11\ldots11\rangle \tag{4.2}$$

Again, each amplitude $\psi_{a_1,\ldots,a_n}$ should be in $[-1, 1]$ and their squares should sum to one:

$$\psi_{00\ldots00}^2 + \psi_{00\ldots01}^2 + \ldots + \psi_{11\ldots11}^2 = 1 \tag{4.3}$$

If some of the amplitudes $\psi_{a_1,\ldots,a_n}$ are zero, we can simply leave them out. For example, the five-qubit quantum state

$$\frac{1}{\sqrt{2}}\left(|00000\rangle + |11111\rangle\right)$$

has 32 amplitudes, of which 30 are zero.

Since there are 2^n amplitudes, we can also think of $|\psi\rangle$ as a vector in a 2^n-dimensional space. What does Eq. (4.3) mean geometrically? For a single qubit, we saw in §2.1.2 that the states correspond to points on the unit circle, i.e., two-dimensional vectors of length one. By the Pythagorean theorem, it is true in any dimension that the sum of squares of a vector's all entries is the square of its length. Thus, Eq. (4.3) means geometrically that $|\psi\rangle$ corresponds to a vector of *length one* or a *unit vector* in a 2^n-dimensional space.

Note that the number of amplitudes grows very rapidly with the number of qubits. This explains why it quickly becomes impossible to directly store quantum states on a classical computer. For example, to represent a quantum state of $n = 300$ qubits, one would need *more amplitudes than there are atoms in the observable universe*! Because of this, you cannot have more than 10 qubits in QUIRKY because we don't want your web browser to run out of memory!

As in Eq. (3.33), we can use the tensor product "$\otimes$" to combine quantum states on any number of qubits. If we have two basis states, we define their tensor product simply by concatenating the bitstrings. Generalizing the two-qubit case from Eq. (3.32),

$$|a_1, \ldots, a_n\rangle \otimes |b_1, \ldots, b_m\rangle = |a_1, \ldots, a_n, b_1, \ldots, b_m\rangle\,. \tag{4.4}$$

For example,

$$|101\rangle \otimes |01\rangle = |10101\rangle\,.$$

In general, if $|\alpha\rangle$ is an arbitrary quantum state of n qubits and $|\beta\rangle$ is an arbitrary quantum state of m qubits, then their tensor product or combined state is a state of $n + m$ qubits. To compute

this state we simply "multiply out" by using the distributivity law and then apply Eq. (4.4) for each term. For example, the tensor product of two maximally entangled states is the following:

$$\begin{aligned}
&\left|\Phi^+\right\rangle \otimes \left|\Phi^+\right\rangle \\
&= \left(\frac{1}{\sqrt{2}}\left|00\right\rangle + \frac{1}{\sqrt{2}}\left|11\right\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}\left|00\right\rangle + \frac{1}{\sqrt{2}}\left|11\right\rangle\right) \\
&= \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\left|00\right\rangle \otimes \left|00\right\rangle + \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\left|00\right\rangle \otimes \left|11\right\rangle + \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\left|11\right\rangle \otimes \left|00\right\rangle + \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\left|11\right\rangle \otimes \left|11\right\rangle \\
&= \frac{1}{2}\left|0000\right\rangle + \frac{1}{2}\left|0011\right\rangle + \frac{1}{2}\left|1100\right\rangle + \frac{1}{2}\left|1111\right\rangle .
\end{aligned}$$

Exercise 4.1: Tensoring Bell states

Compute the tensor product $\left|\Phi^-\right\rangle \otimes \left|\Psi^-\right\rangle$ of the two Bell states in Eqs. (3.53) and (3.55).

4.1.2 Operations

What are the quantum operations that we can perform if we have multiple qubits? For one, we can apply any one-qubit or two-qubit operation discussed in Sections 2 and 3 to any chosen qubits of a many-qubit state. This works as in §3.2.2.

For example, if U is a single-qubit operation, that is, a rotation or a reflection, then we can define a quantum operation, denoted U_1, that corresponds to applying U to the first qubit of an n-qubit state. It is defined as follows on the basis states:

$$U_1 \left|a_1, \ldots, a_n\right\rangle = U\left|a_1\right\rangle \otimes \left|a_2, \ldots, a_n\right\rangle .$$

Note that the tensor product combines the single-qubit state $U\left|a_1\right\rangle$ with the $(n-1)$-qubit basis state $\left|a_2, \ldots, a_n\right\rangle$ to form a state of n qubits, as we desire. As usual, we extend U_1 by linearity to general quantum states of n qubits. We similarly define the quantum operations U_2, U_3, etc. that correspond to applying U to the second, third, etc. qubit.

Exercise 4.2: Applying a single-qubit operation

Compute the result of applying the Hadamard operation on the second qubit of the three-qubit state $\left|\Phi^+\right\rangle \otimes \left|1\right\rangle$. In other words, compute $H_2\left(\left|\Phi^+\right\rangle \otimes \left|1\right\rangle\right)$. Write your result in the form of Eq. (4.1).

We can similarly figure out how a two-qubit operation can be applied to selected two qubits out of n. We will mostly be interested in controlled-NOT operations: $\mathrm{CNOT}_{k\to l}$ with $k \neq l$ is the operation that flips the l-th qubit (the target qubit) depending on the value of the k-th qubit (the control qubit). Mathematically, its action on basis states is as follows:

$$\mathrm{CNOT}_{k\to l}\left|a_1, \ldots, a_l, \ldots, a_n\right\rangle = \left|a_1, \ldots, a_l \oplus a_k, \ldots, a_n\right\rangle ,$$

and we extend this prescription by linearity to arbitrary n-qubit states. For example, the controlled-NOT operation $\mathrm{CNOT}_{1\to 3}$ is defined as follows for four-qubit basis states:

$$\mathrm{CNOT}_{1\to 3}\left|a_1, a_2, a_3, a_4\right\rangle = \left|a_1, a_2, a_3 \oplus a_1, a_4\right\rangle .$$

What does all this look like in QUIRKY? Let's go to

https://www.quantum-quest.org/quirky

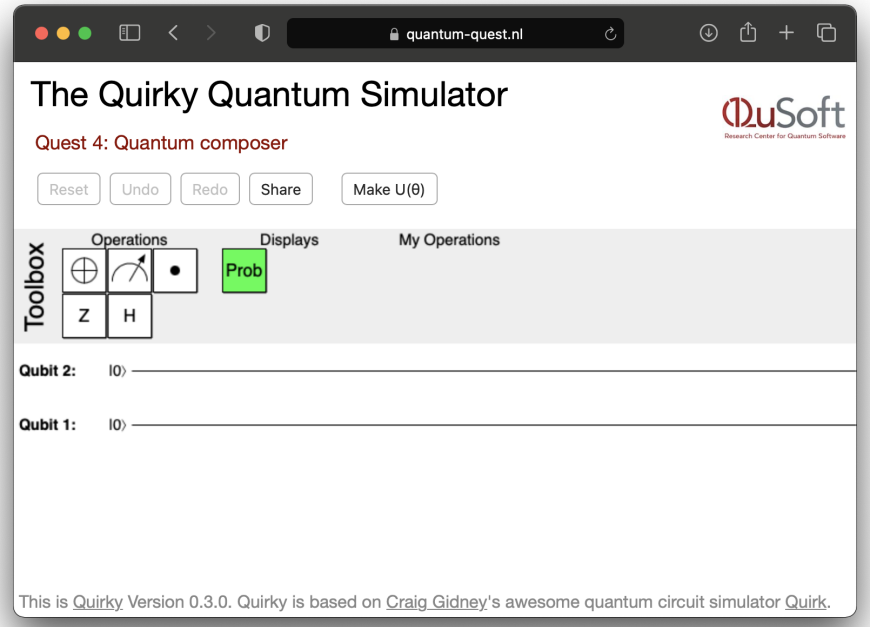


Figure 4.1: QUIRKY for Quest 4.

and click on "Quest 4" to find out. Your web browser will look similarly to Fig. 4.1.

Hold on, it seems like QUIRKY looks exactly the same as last week!? However, as soon as you pick up an operation in the toolbox, a new wire will appear at the bottom – allowing you to act on an additional qubit. (Of course, we have limited the number of qubits to some reasonable number that your classical computer is happy to simulate!) Why don't you try this now and create a $\text{CNOT}_{1\to 3}$ operation, as in the following picture?

When quantum operations act on separate qubits, we can perform them in parallel. As in §3.2.3, we re-use the tensor product symbol for this. If U is a quantum operation on n qubits and V a quantum operation on m qubits then we can define a quantum operation $U \otimes V$ on $(n+m)$ qubits which corresponds to performing both operations in parallel. On basis states,

$$(U \otimes V)\,|a_1,\ldots,a_n,b_1,\ldots,b_m\rangle = U\,|a_1,\ldots,a_n\rangle \otimes V\,|b_1,\ldots,b_m\rangle\,, \tag{4.5}$$

and we extend this by linearity to arbitrary states. It follows as a consequence of Eq. (4.5) that

$$(U \otimes V)(|\alpha\rangle \otimes |\beta\rangle) = U\,|\alpha\rangle \otimes V\,|\beta\rangle\,,$$

but only if $|\alpha\rangle$ is an n-qubit state and $|\beta\rangle$ and m-qubit state! In the following exercise, the two tensor product symbols are *not* aligned in this way, so you *cannot* use this rule!

Exercise 4.3: Misaligned tensor products

Consider the three-qubit state $(\text{CNOT}_{2\to 1} \otimes I)(|0\rangle \otimes |\Phi^-\rangle)$.

1. How can you build this state using QUIRKY?

2. Write out the state in the form of Eq. (4.1).

We can use tensor product several times to iteratively build up larger and larger quantum operations. Here are three examples for various numbers of qubits:

1. $I \otimes I \otimes U \otimes I$ is the same four-qubit operation as U_3,
2. $I \otimes \text{CNOT}_{1\to 2} \otimes I \otimes I$ is the controlled-NOT operation $\text{CNOT}_{2\to 3}$ for five qubits,
3. $Z \otimes I \otimes X$ is the quantum operation that applies Z on the first qubit and, in parallel, X on the third qubit (we could also write this as either $Z_1 X_3$ or $X_3 Z_1$).

4.1.3 The most general quantum operations

What are the most general operations that we can apply to quantum states on n qubits? In fact, any operation that has the following three properties:

1. it is linear,
2. it sends quantum states to quantum states,
3. it is invertible

is a valid quantum operation!

Exercise 4.4: Toffoli

Define the Toffoli operation on three qubits by

$$T|a,b,c\rangle = |a,b,c \oplus ab\rangle$$

on basis states (ab is the product of the two bits $a, b \in \{0,1\}$, and $\oplus$ was defined in Eq. (3.20)), and extend it by linearity to arbitrary three-qubit states. Show that T sends quantum states to quantum states, and that T is invertible.

Note: T inverts the third bit of the basis vector if and only if the first two bits are both set to one – so it is like a 'doubly controlled' NOT operation.

Exercise 4.4 shows that the Toffoli operation is a valid quantum operation of three qubits. Interestingly, it is actually possible to write T as a sequence of one- and two-qubit operations. In fact, this is possible for *any* quantum operation of n qubits – but we will not do it in this class since it takes an experienced quantum composer to understand how this can be done!

4.1.4 Circuit identities

When you are dealing with a very complicated quantum circuit, it is good to know some tricks for simplifying things. Such tricks can not only make it easier to understand what the circuit does but also make the circuit more efficient and thus faster to execute on a quantum computer. Let us look at a few simple examples of such tricks involving just a single qubit.

Recall from §2.4.3 that each single-qubit operation is either a rotation or a reflection. It is convenient to represent the action of these operations visually by recalling from §2.1.2 that qubit states form a circle. One immediate observation you can make is that if any fixed reflection is applied twice on the same qubit, then you end up back with the original state. Indeed, this is visually clear since reflecting about the same axis twice restores everything as it was (for example, you can see this in Fig. 2.4 for the NOT operation). In particular, this is the case for the Hadamard operation H, which we know is a reflection because of Eq. (2.21). Let us verify this geometric intuition by doing a small calculation, and also check that the Hadamard gate lets you convert between the NOT and Z gates.

Exercise 4.5: Z and NOT

Recall from Eq. (2.20) that the Hadamard gate H acts as follows:

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

1. Verify that applying H again would get you back to $|0\rangle$ and $|1\rangle$. That is, check that

$$H|+\rangle = |0\rangle\,, \qquad H|-\rangle = |1\rangle\,.$$

2. Verify that $HZH = \text{NOT}$ where Z is defined in Eq. (2.12).
3. Verify that $H\,\text{NOT}\,H = Z$.

Another interesting question to ask is what happens if you apply two arbitrary rotations or reflections consecutively? We know that what you get should again be either a rotation or a reflection. But which one is it and what is the new angle? Two consecutive rotations are simply the same as a single rotation by the sum of the two angles, i.e., $U(\varphi_2)U(\varphi_1) = U(\varphi_1 + \varphi_2)$. The following exercise gives you a rule for simplifying two consecutive reflections to a single *rotation*.

Exercise 4.6: Reflections and rotations (optional)

Verify that a product of two reflections is a rotation. That is, show that

$$V(\theta_2)V(\theta_1) = U(\theta),$$

for some angle θ. Can you express the angle θ in terms of θ_1 and θ_2?

Hint: Use Eq. (2.19) and use that $U(\varphi_2)U(\varphi_1) = U(\varphi_1 + \varphi_2)$.

You can work out the remaining two cases that involve one rotation and one reflection on your own, and check that they both result in a reflection $V(\theta)$, for some angle θ.

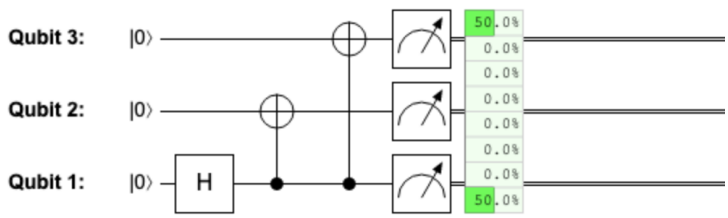
4.1.5 Measuring all qubits

Once we are done with applying operations that transform our quantum bits, we would like to get some information out. As before, the only way of doing this is by measuring the qubits.

What are the rules for measuring a quantum state of n qubits? If we measure all n qubits then we obtain n bits as the outcome, i.e., a bitstring $a_1 \ldots a_n$. As before, the probability of obtaining any particular outcome $a_1 \ldots a_n$ is the squared amplitude:

$$p_{a_1,\ldots,a_n} = \psi^2_{a_1,\ldots,a_n}. \tag{4.6}$$

In QUIRKY, we can measure all qubits as before, by adding one measurement box for each qubit. To view the probabilities of the measurement outcomes, we can add a probability display – but we have to make sure to resize it suitably so that it applies to all wires. Try reproducing the following example:



This sequence of QUIRKY operations prepares the state

$$\frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|111\rangle \tag{4.7}$$

and measures all three qubits. Can you see how this works?

4.1.6 Measuring some of the qubits only

Naturally, we can also measure only a subset of the qubits. (We didn't even discuss this for two qubits last week, since there was already plenty to talk about.) For example, assume that you have a three-qubit quantum state $|\psi\rangle$, like the one in Eq. (4.1), but instead of measuring all three qubits you measure only the first qubit. What is the probability p_a of obtaining outcome $a \in \{0,1\}$? It is simply the sum of all probabilities in Eq. (4.6) that correspond to a string that starts with a:

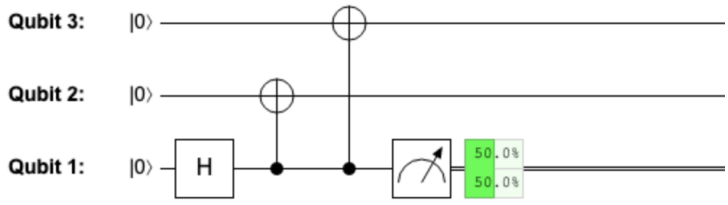
$$p_a = \psi_{a00}^2 + \psi_{a01}^2 + \psi_{a10}^2 + \psi_{a11}^2. \tag{4.8}$$

For example, if we measure the first qubit of the state +

$$\frac{1}{\sqrt{8}}|000\rangle + \sqrt{\frac{2}{8}}|010\rangle + \sqrt{\frac{5}{8}}|111\rangle$$

then we obtain outcome 0 with probability $1/8 + 2/8 = 3/8$.

We can measure individual qubits using QUIRKY simply by adding only a single measurement on the wire that we are interested in. To view the probability of measurement outcomes, drag a probability display onto the circuit. Why don't you give it a try right now? For example, the following sequence of operations prepares the state in Eq. (4.7) and measures only the first qubit:



(4.9)

Indeed, following Eq. (4.8) we should obtain 0 and 1 with equal probability.

Once you measured the first qubit and obtained some outcome $a \in \{0,1\}$, what is the quantum state of the second and the third qubit? Following the same procedure as for probabilistic bits in §3.1.3, we first collect all terms of $|\psi\rangle$ where the first qubit is in the state that corresponds to the outcome of interest:

$$\psi_{a00}|a00\rangle + \psi_{a01}|a01\rangle + \psi_{a10}|a10\rangle + \psi_{a11}|a11\rangle.$$

Next, we leave out the first qubit in all four terms, since it was measured:

$$\psi_{a00}|00\rangle + \psi_{a01}|01\rangle + \psi_{a10}|10\rangle + \psi_{a11}|11\rangle.$$

Finally, we normalize this so that we obtain a valid two-qubit state. For this, we want to find a number c such that $\frac{\psi_{a00}}{c}|00\rangle + \frac{\psi_{a01}}{c}|01\rangle + \frac{\psi_{a10}}{c}|10\rangle + \frac{\psi_{a11}}{c}|11\rangle$ is a qubit state, i.e.,

$$\left(\frac{\psi_{a00}}{c}\right)^2 + \left(\frac{\psi_{a01}}{c}\right)^2 + \left(\frac{\psi_{a10}}{c}\right)^2 + \left(\frac{\psi_{a11}}{c}\right)^2 = 1.$$

The overall sign is not important, so we can simply use

$$c = \sqrt{\psi_{a00}^2 + \psi_{a01}^2 + \psi_{a10}^2 + \psi_{a11}^2},$$

which is the square root of the probability in Eq. (4.8).

To summarize, if you measure the first qubit of a three-qubit state as in Eq. (4.1), you obtain outcome $a \in \{0,1\}$ with probability

$$p_a = \psi_{a00}^2 + \psi_{a01}^2 + \psi_{a10}^2 + \psi_{a11}^2 \tag{4.10}$$

and the resulting two-qubit state $|\psi_a\rangle$ on the remaining two qubits is

$$|\psi_a\rangle = \frac{\psi_{a00}\,|00\rangle + \psi_{a01}\,|01\rangle + \psi_{a10}\,|10\rangle + \psi_{a11}\,|11\rangle}{\sqrt{\psi_{a00}^2 + \psi_{a01}^2 + \psi_{a10}^2 + \psi_{a11}^2}}. \tag{4.11}$$

What does this mean in the situation of (4.9), where we prepare the state $\frac{1}{\sqrt{2}}\,|000\rangle + \frac{1}{\sqrt{2}}\,|111\rangle$ and then measure the first qubit? If the measurement outcome is 0 (which happens with probability $1/2$), the remaining two qubits are in state

$$\frac{\frac{1}{\sqrt{2}}\,|00\rangle}{\sqrt{\frac{1}{2}}} = |00\rangle\,.$$

If instead the outcome is 1 then the remaining qubits are similarly in state $|11\rangle$.

Since measuring one qubit out of many can be quite tricky, let us discuss another method for doing this. Consider again a general three-qubit state $|\psi\rangle$ like the one in Eq. (4.1) and assume that we want to measure the first qubit. We can rewrite the eight terms in the expression of $|\psi\rangle$ as follows:

$$\begin{aligned} |\psi\rangle = {} & \sqrt{p_0}\,|0\rangle \otimes \frac{\psi_{000}\,|00\rangle + \psi_{001}\,|01\rangle + \psi_{010}\,|10\rangle + \psi_{011}\,|11\rangle}{\sqrt{p_0}} \\ & + \sqrt{p_1}\,|1\rangle \otimes \frac{\psi_{100}\,|00\rangle + \psi_{101}\,|01\rangle + \psi_{110}\,|10\rangle + \psi_{111}\,|11\rangle}{\sqrt{p_1}}. \end{aligned} \tag{4.12}$$

We can now rewrite this as

$$|\psi\rangle = \sqrt{p_0}\,|0\rangle \otimes |\psi_0\rangle + \sqrt{p_1}\,|1\rangle \otimes |\psi_1\rangle\,, \tag{4.13}$$

where the p_a are probabilities (namely, the ones from Eq. (4.10)) and the $|\psi_a\rangle$ are quantum states (the two-qubit states from Eq. (4.11)).

In fact, whenever you manage to write your quantum state in the form of Eq. (4.13) then you can simply read off the probabilities of the measurement outcomes in this way, and also see what the state on the remaining two qubits is after the measurement. For example,

$$\frac{1}{\sqrt{2}}\,|000\rangle + \frac{1}{\sqrt{2}}\,|111\rangle = \frac{1}{\sqrt{2}}\,|0\rangle \otimes |00\rangle + \frac{1}{\sqrt{2}}\,|1\rangle \otimes |11\rangle\,,$$

This confirms that in our running example (4.9), both outcomes happen with probability $1/2$ and that the state of the remaining qubits is either $|00\rangle$ or $|11\rangle$, depending on the outcome. This method of first grouping terms and then normalizing them is rather intuitive and generally very useful. However, when using it you should be very careful not to forget to correctly normalize the states! That is, whatever constants $\sqrt{p_a}$ you pull out in front of the two basis states in Eqs. (4.12) and (4.13), they should satisfy $p_0 + p_1 = 1$ and the states $|\psi_0\rangle$ and $|\psi_1\rangle$ on the remaining qubits should be properly normalized, see Eq. (4.3).

We can proceed completely analogously if we have more than three qubits, or we want to measure another qubit than the first, or if we want to measure more than a single qubit! For example, suppose we were to measure the first *two* qubits of a general three-qubit state $|\psi\rangle$

from Eq. (4.1). Then the measurement outcome consists of two bits, a and b, occurring with probabilities

$$p_{a,b} = \psi_{ab0}^2 + \psi_{ab1}^2, \tag{4.14}$$

and the remaining qubit after the measurement is in state

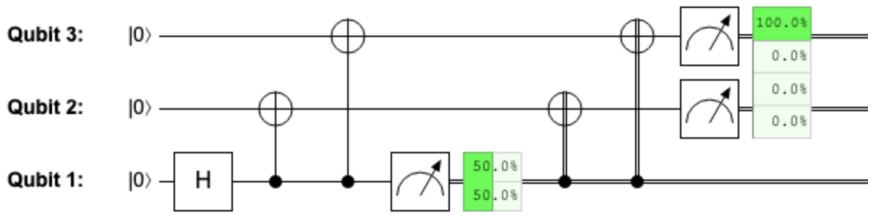
$$|\psi_{a,b}\rangle = \frac{\psi_{ab0}\,|0\rangle + \psi_{ab1}\,|1\rangle}{\sqrt{\psi_{ab0}^2 + \psi_{ab1}^2}}. \tag{4.15}$$

Exercise 4.7: Two out of three

What are the probabilities of outcomes if you measure the first two qubits of the three-qubit state in Eq. (4.7)? Use QUIRKY to confirm your result.

If we measure some of the qubits but not others, we will often want to use measurement outcomes to determine if an operation should be applied to the remaining qubits or not. For example, suppose that in the situation of (4.9) you want to reset the remaining two qubits to state $|00\rangle$. Now, if the measurement outcome is zero, nothing needs to be done. But if the measurement outcome is one, then we know that the two remaining qubits are in state $|11\rangle$ and we would like to reset them back to $|00\rangle$. This can be done by applying a NOT operation to each of them. However, how do we know whether we should really apply this operation or not because this depends on the earlier measurement outcome on the first qubit. Hence, we would like to apply it only when the measurement outcome is 1. In other words, we want to apply a controlled-NOT gate, where the control is now a classical bit (the outcome of the measurement) but the target is still quantum.

In QUIRKY we can realize this as you would expect, namely by using a classical bit as the control and a quantum bit as the target, as in the following:



Here, after the first qubit is measured, we apply two further controlled-NOT operations that are controlled by the measurement outcome and then we measure the remaining two qubits. The picture shows that we indeed successfully reset the two qubits, since measuring them yields $[00]$ with 100% probability.

If we liked, we could describe these operations in terms of 'hybrid' states that consist of one bit and two qubits, e.g.,

$$\mathrm{CNOT}_{1\to 2}[a] \otimes |b, c\rangle = [a] \otimes |a \oplus b, c\rangle\,,$$

but we will not need this level of formality.

4.2 Quantum surprises

We will now discuss some interesting phenomena that arise when we deal with quantum bits. The following sections can be read largely independently, so feel free to start with the one that interests you most.

4.2.1 No cloning

When we have a classical bit, it is easy to *copy* or clone it – simply look at it and copy what you see:

$$[0] \mapsto [00],$$
$$[1] \mapsto [11].$$

Can we also clone quantum bits?

Let us assume for a moment that this is possible. This would mean that there exists a quantum operation C that, given any qubit in state $|\psi\rangle$ and a fresh qubit in state $|0\rangle$, acts as

$$C(|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes |\psi\rangle \tag{4.16}$$

to produce two copies of $|\psi\rangle$ out of a single copy. (Why do we supply the fresh qubit? This is so that C has the same number of input qubits as output qubits.)

For example, the cloner would operate as follows on the basis states:

$$\begin{aligned} C\,|00\rangle &= |00\rangle\,, \\ C\,|10\rangle &= |11\rangle\,. \end{aligned} \tag{4.17}$$

Just like we can easily clone a classical bit, it is easy to find a quantum operation that clones the basis states. For example, the controlled-NOT operation $\mathrm{CNOT}_{1\to 2}$ from Eq. (3.46) does this job.

But is there a quantum operation that can clone an arbitrary unknown qubit state, not just a basis state? In the following homework problem, you will show that this is *not* possible.

Homework 4.1: No cloning

In this homework problem we want to prove that there exists no quantum operation C that satisfies Eq. (4.16). We will use a trick called *proof by contradiction*. This means that we will show that if such a cloning operation C existed then this would imply something that we know to be wrong (e.g., "$0 = 1$"). From this we can then conclude that no such C can exist.

Thus let us start by assuming that there is a quantum operation C that satisfies Eq. (4.16). Now you can calculate $C(|+\rangle \otimes |0\rangle)$ in two different ways:

1. First use Eq. (4.16) and then write the result in the form of Eq. (3.30).
2. First expand $|+\rangle \otimes |0\rangle$ in the form of Eq. (3.30), then use that C is linear, and finally apply Eq. (4.16).

Do you get the same answer in both cases? If not, what can you conclude?

This famous result is known as the no cloning theorem. The same conclusion (and likely also the argument that you gave in Homework 4.1) applies also to probabilistic bits! Here is an intuitive explanation for why we can copy neither probabilistic nor quantum information. If this were possible then, given a probabilistic bit in an unknown state p or a qubit in an unknown state $|\psi\rangle$, we could first produce as many copies of p and $|\psi\rangle$ as we like. Given these copies, we could then measure them in various ways and use the obtained data to estimate the probabilities of p or the amplitudes of $|\psi\rangle$ to arbitrary precision (just like we did in in §2.5.1 to figure out the inner workings of the yellow mystery box). Thus, we could from a single probabilistic or quantum bit learn an arbitrary amount of information. This should clearly not be possible!

Indeed, if this were possible then we would live in a very strange world (much stranger than the one described by quantum mechanics)! For example, imagine a coin whose probability of heads is $p = 0.1011010010\ldots$ where the binary digits encode the whole content of Wikipedia

as well as all YouTube videos and all pictures of cats you can find on the internet. If cloning probabilistic bits were possible, I could flip this coin once and write down which outcome I got. This is a probabilistic bit of information that is equal to 0 with probability p. If I send this probabilistic bit to you and you have the ability to clone it, you could produce as many copies of it as you want and then measure them all. By looking at the measurement outcomes and counting how many zeroes you got, you could estimate the probability p. In fact, by producing sufficiently many copies of the original bit, you could estimate this probability arbitrarily well! In particular, you would be able to extract out any binary digit of p and hence also all the information encoded in p, including the cat picture number 65535!

This should clearly be impossible, since otherwise we would not need USB drives, data centers, or to pay for our mobile phone data connection – we could just store all our information in a single probabilistic bit and transmit it all by sending this bit to somebody else! This is certainly too good to be true...

4.2.2 One-time pad

Before discussing teleportation of quantum states, it is useful to first understand a simpler procedure for probabilistic bits, which is called one-time pad. This procedure allows Alice to encrypt a message and send it to Bob in such a way that only Bob can understand what the message is. That is, in case anyone else intercepts the message (such as their classmate Eve), they would have no idea what the actual message is. The fact that this is even possible is somewhat surprising. Indeed, what advantage does Bob have over Eve so that he can decode Alice's message correctly while Eve has absolutely no clue about its contents?

The trick is for Alice and Bob to first meet in a coffee shop. Take two coins, glue them together with a chewing gum, flip the resulting 'double coin', and separate the two coins again. Alice and Bob each take one of the flipped coins. Now they share a pair of random bits that are described by the state from (3.5), namely

$$r = \frac{1}{2}([00] + [11]).$$

You can think of this as a shared secret! Moreover, only Alice and Bob know what this secret is – to find it out, they can simply look at their respective coins (thus measuring them). They will both see the same side, and each of the sides occurs with probability $1/2$. This is a very good secret, since Eve cannot predict it better than blindly guessing!

Now, let us see how Alice and Bob can make use of it. Suppoose that Alice has a secret message $m \in \{0, 1\}$ that she wants to send to Bob. The overall state of all their bits is described by the state

$$[m] \otimes r = \frac{1}{2}([m00] + [m11]), \tag{4.18}$$

where the first two bits (m and the first half of r) belong to Alice and the third bit (the second half of r) belongs to Bob.

To send her message, Alice needs to look at her half of the shared random bit r and

1. if she sees 0 then she sends m over to Bob as it is,
2. if she sees 1 then she sends $\mathrm{NOT}(m)$ over to Bob.

Let's imagine that Eve intercepts this message. What does she see? Irrespective of the value of m, she will see 0 with probability $1/2$ and 1 with probability $1/2$. This is because Alice inverts m with probability $1/2$, which effectively randomizes m in such a way that Eve sees it as a uniformly random bit.

But what about Bob? Doesn't Alice's message appear uniformly random to him as well? Luckily, Bob has the other half of the secret random bit they shared. While originally Alice's

message appears random to him, too, he can decode it by applying exactly the same procedure as Alice: look at his half of the shared random bit and

1. if he sees 0 then he takes Alice's message as it is,

2. if he sees 1 then he applies a NOT operation to Alice's message.

Overall, Alice's message is either transmitted as it is or inverted twice, meaning that Bob always understands it correctly. However, from Eve's perspective it has been inverted with probability $1/2$, meaning that what she sees is a uniformly random bit. Hence, this is a perfectly secure way for Alice and Bob to communicate!

Let us understand more formally what is going on here. When Alice inverts her first bit if her second bit is equal to 1, this is the same as applying $\mathrm{CNOT}_{2\to 1}$ on her two bits. Next, Alice sends the first bit over to Bob. Then, when Bob decodes Alice's message, what he does is to apply a $\mathrm{CNOT}_{3\to 1}$ operation (which he can do since, in addition to the third bit, he now also holds the first bit). The resulting state is

$$\mathrm{CNOT}_{3\to 1}\,\mathrm{CNOT}_{2\to 1}([m]\otimes r).$$

We can evaluate this as follows:

$$\mathrm{CNOT}_{3\to 1}\,\mathrm{CNOT}_{2\to 1}\,\frac{1}{2}([m,0,0]+[m,1,1]) = \mathrm{CNOT}_{3\to 1}\,\frac{1}{2}([m,0,0]+[\mathrm{NOT}(m),1,1])$$
$$= \frac{1}{2}([m,0,0]+[\mathrm{NOT}(\mathrm{NOT}(m)),1,1]) = \frac{1}{2}([m,0,0]+[m,1,1]) = [m]\otimes r.$$

Thus, the message bit is back in its original state but is now with Bob.

An interesting aspect of the above one-time pad protocol is not only that it lets Alice send a deterministic message $[m]$ to Bob but even a probabilistic one. It follows by linearity that if Alice's message is instead a probabilistic bit with distribution p, then the initial state is $p\otimes r$ and the final state is again $p\otimes r$, where this time p is with Bob. However, from Eve's perspective the transmitted message is still uniformly random. The surprising part about this is that by sending a uniformly random bit Alice manages to secretly transmit a probabilistic bit whose distribution she might not even be aware of.

This procedure is quite similar to quantum teleportation, where Alice can transmit a qubit state $|\psi\rangle$ to Bob by sending two (instead of one) uniformly random bit. For teleportation, they have to use a shared maximally entangled state $|\Phi^+\rangle$ instead of the shared random bit r. In both cases, the shared resource is measured and thus consumed during the procedure. We discuss this in the following section.

4.2.3 Quantum teleportation

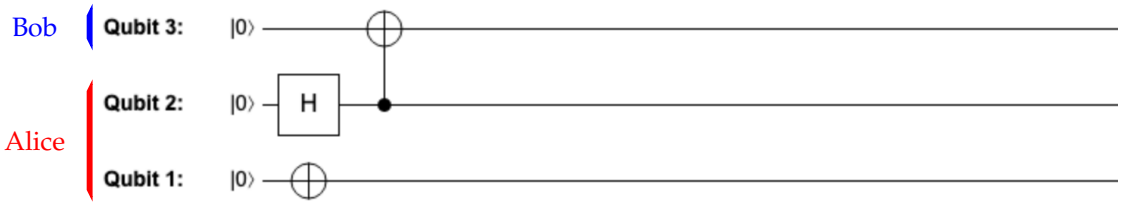
While it is not possible to clone quantum bits, we can certainly *move* quantum bits from one place to another. Indeed, any qubit is stored inside some physical object or particle that carries the qubit. For example, if Alice stores her qubit as the polarization of a photon, she can simply send this photon to Bob. What is more surprising, however, is that one can move a quantum bit from one place to another by sending a finite number of classical bits (in fact, two bits suffice). The reason this is surprising is because a general qubit state $|\psi(\theta)\rangle$ is specified by an arbitrary angle θ, see Eq. (2.5), which generally cannot be encoded in a finite number of bits. Because of this surprising property this procedure is known as teleportation. We will see shortly that we need entanglement to make it work!

The starting point for teleportation is the following scenario. We imagine that Alice has two qubits and Bob has one qubit. Alice's first qubit is the *message qubit* that she wants to send to Bob, and it starts out in some arbitrary state $|\psi\rangle$ – which may well be unknown to Alice herself!

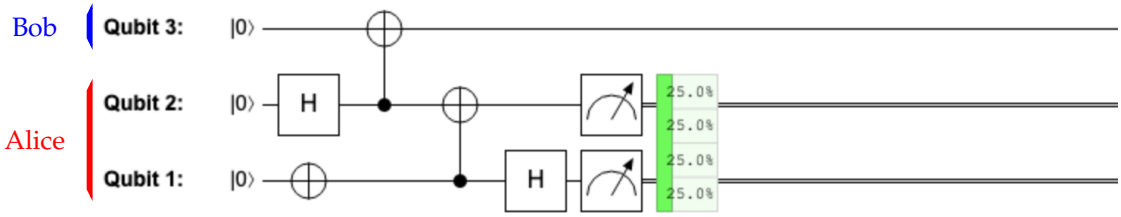
Her second qubit and Bob's qubit are in a maximally entangled state $|\Phi^+\rangle$. Thus, the three qubits are in the following state:

$$|\psi\rangle \otimes |\Phi^+\rangle ,$$

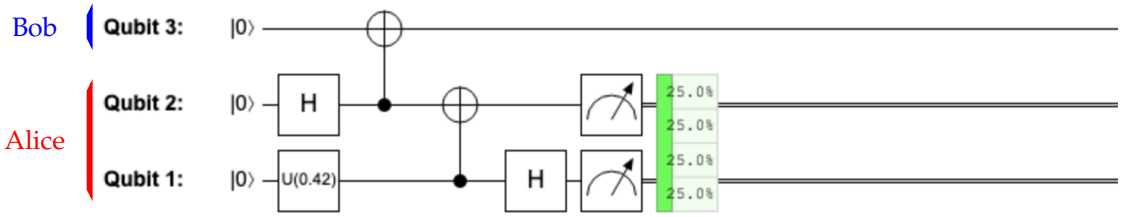
where the first two qubits belong to Alice and the last qubit belongs to Bob (recall from Eq. (3.52) that $|\Phi^+\rangle$ is a two-qubit state). The following QUIRKY circuit shows what this looks like in case that Alice wants to send a qubit in state $|\psi\rangle = |1\rangle = \text{NOT}\,|0\rangle$:



What should be the next step? Eventually, the goal is for Bob to obtain Alice's qubit – since we cannot clone the qubit, this means that Alice has to perform some action that 'destroys' her qubit. A good way of going about this is to perform a measurement. But simply measuring Alice's two qubits will do them no good, since we know that we cannot infer the state of $|\psi\rangle$ from a single measurement. This means that Alice should first apply some quantum operation on both of her qubits and then measure them. It turns out that the correct strategy for her is to perform the same operations that you used to discriminate the four Bell states in Exercise 3.13:



Note that in this example Alice's four measurement outcomes occur with 25% each. Here is another example, corresponding to the situation where Alice tries to teleport the state $|\psi(0.42)\rangle$ instead:



It seems that, whatever the state of Alice's message qubit, the four probabilities are always the same. This is already encouraging, since it means that the measurement gives Alice no information at all about her message qubit! Recall that we want her initial state $|\psi\rangle$ to end up completely with Bob, meaning that she should not extract or keep any information about this state around.

In general, the state right before Alice's measurements looks as follows:

$$(H \otimes I \otimes I)\,(\text{CNOT}_{1\to 2} \otimes I)\,\left(|\psi\rangle \otimes |\Phi^+\rangle\right)$$

Note that we have to be careful, since the last two tensor products are *not* aligned (cf. Exercise 4.3). Thus we compute:

$$\begin{aligned}
&(H \otimes I \otimes I)\,(\mathrm{CNOT}_{1\to 2} \otimes I)\,\left(|\psi\rangle \otimes \left|\Phi^+\right\rangle\right) \\
&= \frac{1}{\sqrt{2}}\,(H \otimes I \otimes I)\,(\mathrm{CNOT}_{1\to 2} \otimes I)\,(\psi_0\,|000\rangle + \psi_0\,|011\rangle + \psi_1\,|100\rangle + \psi_1\,|111\rangle) \\
&= \frac{1}{\sqrt{2}}\,(H \otimes I \otimes I)\,(\psi_0\,|000\rangle + \psi_0\,|011\rangle + \psi_1\,|110\rangle + \psi_1\,|101\rangle) \\
&= \frac{1}{2}\,(\psi_0\,|000\rangle + \psi_0\,|100\rangle + \psi_0\,|011\rangle + \psi_0\,|111\rangle + \psi_1\,|010\rangle - \psi_1\,|110\rangle + \psi_1\,|001\rangle - \psi_1\,|101\rangle) \\
&= |00\rangle \otimes \frac{\psi_0\,|0\rangle + \psi_1\,|1\rangle}{2} + |01\rangle \otimes \frac{\psi_1\,|0\rangle + \psi_0\,|1\rangle}{2} \\
&+ |10\rangle \otimes \frac{\psi_0\,|0\rangle - \psi_1\,|1\rangle}{2} + |11\rangle \otimes \frac{-\psi_1\,|0\rangle + \psi_0\,|1\rangle}{2}.
\end{aligned}$$

This is the overall state right before Alice's measurement. We can compute the probability of her measurement outcomes as discussed in Eq. (4.14). Namely, to compute the probability $p_{a,b}$ of outcome $[ab]$ we simply add the squared amplitudes of the relevant basis states $|ab0\rangle$ and $|ab1\rangle$. In each case, one of the amplitudes is $\psi_0/2$ and the other is $\pm\psi_1/2$, so their squares always sum to the same result:

$$\begin{aligned}
p_{00} &= \left(\frac{\psi_0}{2}\right)^2 + \left(\frac{\psi_1}{2}\right)^2 = \frac{\psi_0^2 + \psi_1^2}{4} = \frac{1}{4}, \\
p_{01} &= \left(\frac{\psi_1}{2}\right)^2 + \left(\frac{\psi_0}{2}\right)^2 = \frac{1}{4}, \\
p_{10} &= \left(\frac{\psi_0}{2}\right)^2 + \left(\frac{-\psi_1}{2}\right)^2 = \frac{1}{4}, \\
p_{11} &= \left(\frac{-\psi_1}{2}\right)^2 + \left(\frac{\psi_0}{2}\right)^2 = \frac{1}{4}.
\end{aligned}$$

Each outcome occurs with 25%. This confirms what we previously observed with QUIRKY.

After Alice's measurement, the only remaining quantum bit is Bob's qubit. Let us denote its state by $|\psi'_{a,b}\rangle$, since it depends on Alice's measurement outcome. We can determine it using Eq. (4.15) or, much easier, using the grouping and normalization method in Eq. (4.13). Either way, the result is that Bob's qubit is in one of the following four states:

$$\begin{aligned}
\left|\psi'_{00}\right\rangle &= \psi_0\,|0\rangle + \psi_1\,|1\rangle\,, \\
\left|\psi'_{01}\right\rangle &= \psi_1\,|0\rangle + \psi_0\,|1\rangle\,, \\
\left|\psi'_{10}\right\rangle &= \psi_0\,|0\rangle - \psi_1\,|1\rangle\,, \\
\left|\psi'_{11}\right\rangle &= -\psi_1\,|0\rangle + \psi_0\,|1\rangle\,.
\end{aligned}$$

When $[ab] = [00]$, Bob's state coincides exactly with Alice's original state $|\psi\rangle$ that she wanted to teleport:

$$\left|\psi'_{00}\right\rangle = |\psi\rangle\,.$$

In the other three cases Bob's quantum state is slightly garbled. However, if Alice sends her measurement outcomes (i.e., the two bits a and b) to Bob then he can apply an appropriate correction operation to 'fix up' his state:

$$\begin{aligned}
\mathrm{NOT}\left|\psi'_{01}\right\rangle &= |\psi\rangle\,, \\
Z\left|\psi'_{10}\right\rangle &= |\psi\rangle\,, \\
Z\,\mathrm{NOT}\left|\psi'_{11}\right\rangle &= |\psi\rangle\,.
\end{aligned}$$

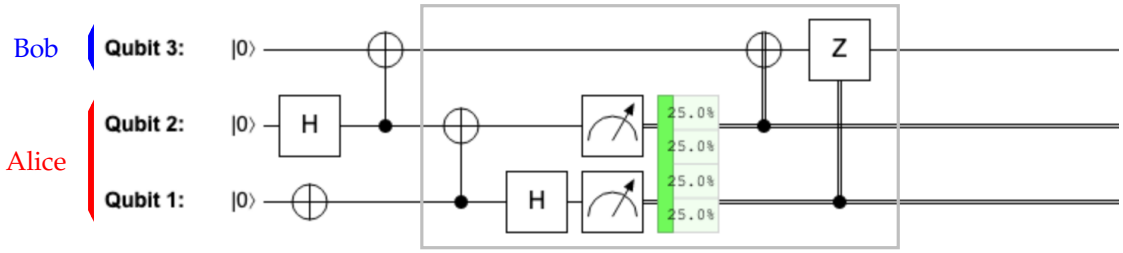
These four cases can be summarized in the following simple procedure for Bob:

1. look at bit b and if $b = 1$ then apply NOT,

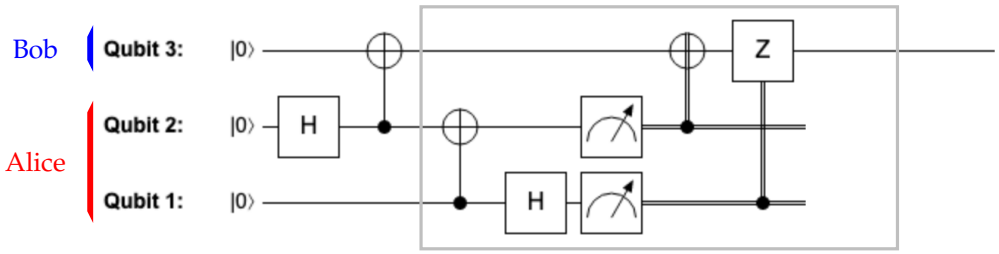
2. look at bit a and if $a = 1$ then apply Z.

We can implement this using a controlled-NOT and a controlled-Z operation, where the controls are classical bits. You can create the controlled-Z operation in the same way as the controlled NOT operation – we discussed this at the end of §3.2.4. Phew, this was quite a bit of work!

What does all this look like in QUIRKY? The end result is the following quantum circuit:

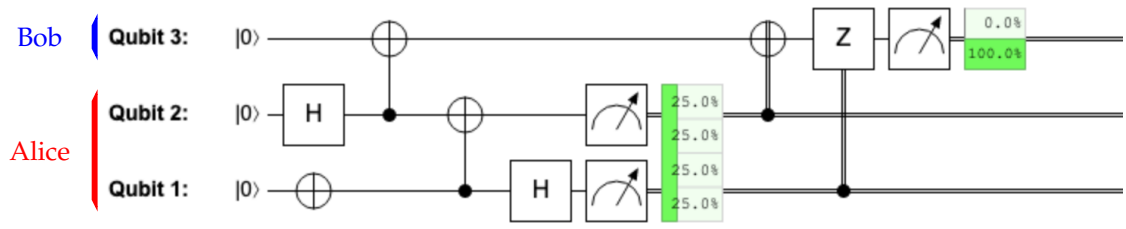


We added a gray box about the relevant part – the teleportation circuit – to separate it from the rest of the circuit that creates the input states. Here is a picture that shows only the teleportation circuit and the creation of the maximally entangled state, without specifying Alice's first qubit:



We also cut off the wires for Alice's two classical bits, since they are no longer of interest once Alice has sent the two measurement outcomes to Bob. We also removed the probability display, since it is not an actual quantum operation but just a way for us to inspect the circuit in QUIRKY. Thus, there is one input qubit for Alice's message, two qubits for the maximally entangled state, and one output qubit on Bob's side. The effect of the teleportation is simply to pass through an arbitrary state $|\psi\rangle$ from the input qubit on Alice's side to the output qubit on Bob's side. Crucially, inside the box, only classical bits are being sent from Alice to Bob!

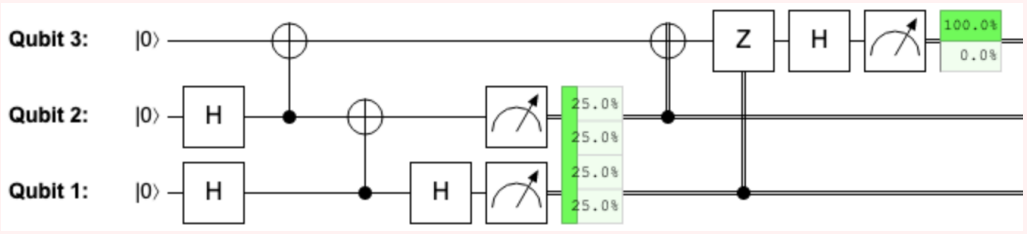
Let's make sure that we didn't make a mistake. Since we expect that Bob's qubit ends up in state $|1\rangle$ after the teleportation procedure, we can add a simple measurement to test if this is indeed what happened:



Indeed, we get outcome 1 with 100% probability, which confirms that we successfully teleported the $|1\rangle$ state! Now, $|1\rangle$ is not a particularly interesting state to teleport. How about the $|+\rangle$ state, which caused us trouble before when we discussed cloning? In the following homework you will test the teleportation circuit first for the $|+\rangle$ state and then for arbitrary single-qubit states.

Homework 4.2: Testing teleportation

1. Why does the following circuit confirm that the $|+\rangle$ state was teleported correctly?



2. How you can you similarly test if a quantum state $|\psi(\theta)\rangle$ was teleported correctly?

The teleportation circuit is quite remarkable. It allows to send a quantum bit from Alice to Bob by transmitting two classical bits only, provided that Alice and Bob share a maximally entangled state. However, it is not only useful for applications (we will discuss some of them below), but it also gives some interesting perspective on the difference between classical and quantum bits. Recall that at the end of last week we discussed superdense coding, which allowed us to send two bits by transmitting a single quantum bit, again using one maximally entangled state beween Alice and Bob. This shows that:

Given a free supply of quantum entanglement, sending two bits is completely equivalent to sending one quantum bit!

Note that in neither teleportation nor superdense coding can we reuse the maximally entangled state once the procedure is finished – it is the 'fuel' that is consumed by either procedure.

4.2.4 A glance at quantum networks

By repeatedly using teleportation, we can communicate quantum bits between distant nodes. For example, suppose Alice, her donkey robot, and Bob are in the following situation:

$$\text{Alice's donkey robot} \longleftrightarrow \text{Alice} \longleftrightarrow \text{Bob},$$

where each "$\leftrightarrow$" arrow indicates a maximally entangled state. That is, the joint state of our three protagonists is the following four-qubit state:

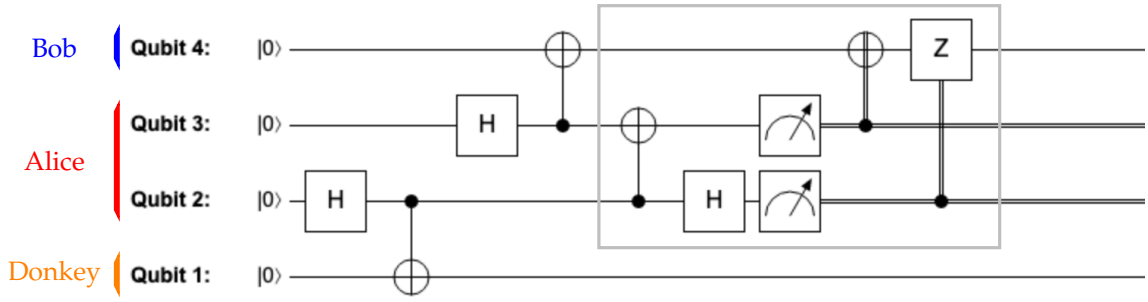
$$\left|\Phi^+\right\rangle \otimes \left|\Phi^+\right\rangle,$$

where the middle two qubits belong to Alice – the first one is entangled with the robot and the second one with Bob.

Note that the donkey robot is not entangled with Bob directly! Nevertheless, if it had a quantum message to send to Bob then this could be done by running the teleportation procedure twice: first, we teleport the message from the donkey robot to Alice (consuming the first maximally entangled state) and subsequently from Alice to Bob (consuming the remaining maximally entangled state). This is similar to how, say, your mobile phone connects to a nearby base station, which in turn 'repeats' or 'relays' the signal to another mobile phone tower (and so forth). While quantum mechanically we cannot copy a qubit, due to the no cloning theorem (c.f. Homework 4.1), we can still teleport it over long distances!

Entanglement is not only useful for teleportation but also for many other things. Is there also a way to use teleportation to create entanglement between Alice's donkey robot and Bob (which they could then use for other purposes)?

Intuitively, it seems that Alice simply needs to teleport her first qubit (the one that is entangled with the donkey robot) to Bob. In QUIRKY, this would look as follows:



Here, we first create the two maximally entangled states and we then apply the same teleportation circuit as above (gray box). Intuitively, we might hope that this results in a maximally entangled state between the donkey robot and Bob. In the following homework, you can confirm that this is indeed the case.

Homework 4.3: Teleporting an entangled qubit

Confirm using QUIRKY that at the end of the circuit the donkey's qubit and Bob's qubit are in the maximally entangled state $|\Phi^+\rangle$.

In the same way, we can use teleportation to create entanglement between more and more distant nodes. E.g., suppose we are in the following situation:

$$\text{Alice's donkey robot} \longleftrightarrow \text{Alice} \longleftrightarrow \text{Bob} \longleftrightarrow \text{Bob's squirrel robot.}$$

If Alice first teleports her first qubit to Bob and Bob subsequently teleports his first qubit to his squirrel robot, this results in a maximally entangled state between the two robots. Let's hope that the robots only use this entanglement for benevolent purposes!

Establishing entanglement over long distances will be an important functionality once we try to connect quantum computers in a small network or, dreaming boldly, in a future 'quantum internet'. Several of us are already thinking hard how to realize this in practice and how to best use long-distance entanglement for interesting applications.

4.2.5 The uncertainty principle

For the last phenomenon that we want to discuss, we will only need a single quantum bit. Recall the rules for measuring a single quantum bit from Eq. (2.6), as in the following picture:

Given a quantum state $|\psi\rangle = \psi_0 |0\rangle + \psi_1 |1\rangle$, we obtain the two possible outcomes with probabilities

$$p_0 = \psi_0^2, \qquad p_1 = \psi_1^2. \tag{4.19}$$

What are the deterministic states for which we get one of the outcomes with certainty? These are states for which one probability is 100% and the other is 0%. In other words, states where one amplitude is ± 1 and the other is zero. Up to possibly an overall minus sign, which we know from Exercise 2.7 is irrelevant, there are only two such states:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{4.20}$$

i.e., the basis states. These are the only states for which we can predict the measurement outcome perfectly well, so we will say that there is no uncertainty in the measurement outcome.

What happens if we first perform an operation on the qubit and then perform a measurement? For example, suppose that we first apply a Hadamard operation and then a measurement, as in the following picture:

Here, the only states for which we are completely certain about the measurement outcome are

$$|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}, \qquad |-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} \tag{4.21}$$

(up to overall sign). This is because the Hadamard operation maps $|+\rangle, |-\rangle$ back to the basis states $|0\rangle, |1\rangle$ (you showed this in Exercise 4.5); and the latter states are precisely the states with complete certainty about the final measurement, as we discussed above. We can also see this by computing the probabilities of the two measurement outcomes

$$q_0 = \frac{(\psi_0 + \psi_1)^2}{2}, \qquad q_1 = \frac{(\psi_0 - \psi_1)^2}{2}. \tag{4.22}$$

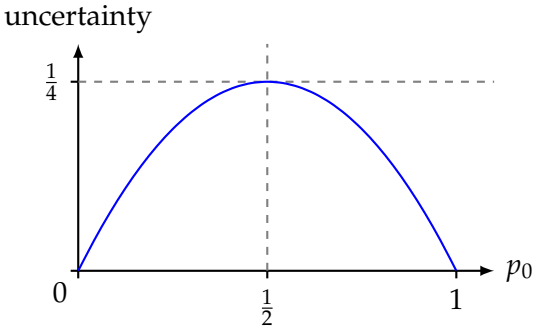
If $q_1 = 0$ then the state is $|+\rangle$, while if $q_0 = 0$ then the state is $|-\rangle$ (up to overall sign).

These are all calculations that we have seen several times – but there is an interesting observation that we have not made before. Since no state appears in *both* Eqs. (4.20) and (4.21), this means that, for every state, at least one of the two procedures will have some uncertainty in the measurement outcome. This result is nothing but the famous *Heisenberg uncertainty principle*!

Can we make this observation more quantitative? We first need a way to quantify the uncertainty or 'randomness' given by a probability distribution $p = \binom{p_0}{p_1}$. The following function is a good choice:

$$\text{uncertainty}(p) = p_0(1 - p_0) = p_0 p_1.$$

It maps values to $[0, 1/4]$, is minimal if one of the outcomes is zero (i.e., if $p_0 = 0$ or $p_0 = 1$) and it is maximal if both outcomes are equally likely (i.e., if $p_0 = p_1 = 1/2$). Here is a plot which confirms these properties:



Now suppose that we start with a state $|\psi\rangle$ and perform either of the two procedures described above. That is, we either measure the state directly or we first apply a Hadamard operation and then measure. The corresponding uncertainties are uncertainty(p) and uncertainty(q), where the distributions $p = \binom{p_0}{p_1}$ and $q = \binom{q_0}{q_1}$ are given in Eqs. (4.19) and (4.22) above. Then:

$$\text{uncertainty}(p) + \text{uncertainty}(q) > 0.$$

Indeed, this inequality means precisely that there exists no state for which both uncertainties are simultaneously zero. In the following homework problem, you will show a stronger result:

Homework 4.4: Uncertainty tradeoff

Show that, for every qubit state $|\psi\rangle$:

$$\text{uncertainty}(p) + \text{uncertainty}(q) = \frac{1}{4}. \tag{4.23}$$

Moreover, find a qubit state $|\psi\rangle$ with $\text{uncertainty}(p) = \text{uncertainty}(q)$.

Bonus question: Construct this state using QUIRKY, and confirm that $\text{uncertainty}(p) = \text{uncertainty}(q)$ by using QUIRKY.

The formula that you just proved in Homework 4.4 is quite remarkable: it shows that there is a simple tradeoff between the uncertainties of the two procedures. In particular, if one procedure has zero uncertainty then the other produces a uniformly random outcome! [12]

[12] We can also see this directly. E.g., if we directly measure $|+\rangle$ (a state that has zero uncertainty for the second procedure) without first doing a Hadamard, we get outcomes 0 and 1 with equal probability.

4.3 Exercise solutions

Solution to Exercise 4.1

$$\begin{aligned}
&\left|\Phi^-\right\rangle \otimes \left|\Psi^-\right\rangle \\
&= \left(\frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle\right) \\
&= \frac{1}{2}|0001\rangle - \frac{1}{2}|0010\rangle - \frac{1}{2}|1101\rangle + \frac{1}{2}|1110\rangle .
\end{aligned}$$

Solution to Exercise 4.2

Let us first expand the given product state in the form of Eq. (4.2):

$$\left|\Phi^+\right\rangle \otimes |1\rangle = \frac{1}{\sqrt{2}}|001\rangle + \frac{1}{\sqrt{2}}|111\rangle .$$

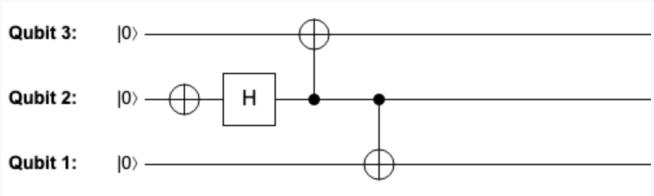
Thus:

$$\begin{aligned}
&H_2\left(\left|\Phi^+\right\rangle \otimes |1\rangle\right) = H_2\left(\frac{1}{\sqrt{2}}|001\rangle + \frac{1}{\sqrt{2}}|111\rangle\right) = \frac{1}{\sqrt{2}}H_2|001\rangle + \frac{1}{\sqrt{2}}H_2|111\rangle \\
&= \frac{1}{\sqrt{2}}|0\rangle \otimes H|0\rangle \otimes |1\rangle + \frac{1}{\sqrt{2}}|1\rangle \otimes H|1\rangle \otimes |1\rangle \\
&= \frac{1}{\sqrt{2}}|0\rangle \otimes |+\rangle \otimes |1\rangle + \frac{1}{\sqrt{2}}|1\rangle \otimes |-\rangle \otimes |1\rangle \\
&= \frac{1}{\sqrt{2}}|0\rangle \otimes \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes |1\rangle \; + \; \frac{1}{\sqrt{2}}|1\rangle \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) \otimes |1\rangle \\
&= \frac{1}{2}|001\rangle + \frac{1}{2}|011\rangle + \frac{1}{2}|101\rangle - \frac{1}{2}|111\rangle ,
\end{aligned}$$

where we used Eq. (2.20) to compute the action of H on the basis states.

Solution to Exercise 4.3

1. We saw how to prepare $\left|\Phi^-\right\rangle$ in Exercise 3.12. Thus, the following circuit does the job:



2. Here is the resulting state:

$$\begin{aligned}
(\mathrm{CNOT}_{2\to 1} \otimes I)(|0\rangle \otimes \left|\Phi^-\right\rangle) &= (\mathrm{CNOT}_{2\to 1} \otimes I)\left(\frac{1}{\sqrt{2}}|000\rangle - \frac{1}{\sqrt{2}}|011\rangle\right) \\
&= \frac{1}{\sqrt{2}}|000\rangle - \frac{1}{\sqrt{2}}|111\rangle .
\end{aligned}$$

Solution to Exercise 4.4

Let

$$\begin{aligned}|\psi\rangle = \psi_{000}|000\rangle + \psi_{001}|001\rangle + \psi_{010}|010\rangle + \psi_{011}|011\rangle \\ + \psi_{100}|100\rangle + \psi_{101}|101\rangle + \psi_{110}|110\rangle + \psi_{111}|111\rangle\end{aligned}$$

be an arbitrary three-qubit quantum state. The result of applying the Toffoli operation is

$$\begin{aligned}|\psi'\rangle = T|\psi\rangle = \psi_{000}|000\rangle + \psi_{001}|001\rangle + \psi_{010}|010\rangle + \psi_{011}|011\rangle \\ + \psi_{100}|100\rangle + \psi_{101}|101\rangle + \psi_{110}|\mathbf{111}\rangle + \psi_{111}|\mathbf{110}\rangle.\end{aligned}$$

We highlighted the two basis states that changed in bold. Note that the only change is that the amplitudes of $|110\rangle$ and $|111\rangle$ were swapped. Thus it is clear that if $\sum_{a,b,c=0}^{1}\psi_{a,b,c}^2 = 1$ then also $\sum_{a,b,c=0}^{1}(\psi'_{a,b,c})^2 = 1$. Thus, T maps quantum states to quantum states.

Solution to Exercise 4.5

1. By applying H to $|+\rangle$ and $|-\rangle$ we get

$$\begin{aligned}H|+\rangle = H\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{2}\big((|0\rangle + |1\rangle) + (|0\rangle - |1\rangle)\big) = |0\rangle\,, \\ H|-\rangle = H\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{2}\big((|0\rangle + |1\rangle) - (|0\rangle - |1\rangle)\big) = |1\rangle\,.\end{aligned}$$

2. To show this identity, we only need to check that HZH acts the same way as NOT on the basis vectors (by linearity, this would mean that they act the same way on all qubit states):

$$\begin{aligned}HZH|0\rangle = HZ\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = H\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = H|-\rangle = |1\rangle\,, \\ HZH|1\rangle = HZ\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = H\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = H|+\rangle = |0\rangle\,.\end{aligned}$$

   In both cases HZH inverts the bit, so it is implementing the same operation as NOT.

3. The first part of the exercise shows that applying the Hadamard gate twice does nothing: $HH = I$. Thus,

$$Z = (HH)Z(HH) = H(HZH)H = H\,\text{NOT}\,H$$

   where the last step is by part 2 of the exercise.

Solution to Exercise 4.6

1. Using Eq. (2.15),

$$U(\theta_2)U(\theta_1)\,|\psi(\alpha)\rangle = U(\theta_2)\,|\psi(\alpha+\theta_1)\rangle = |\psi(\alpha+\theta_1+\theta_2)\rangle = U(\theta_1+\theta_2)\,|\psi(\alpha)\rangle\,.$$

   This holds for any state $|\psi(\alpha)\rangle$, implying that $U(\theta_2)U(\theta_1) = U(\theta)$ where $\theta = \theta_1 + \theta_2$. This is also clear geometrically: if you first rotate by angle θ_1 and then by angle θ_2, together this amounts to a rotation by angle $\theta = \theta_1 + \theta_2$.

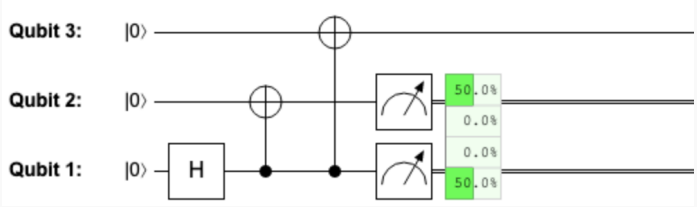
2. Here is an elegant solution. Recall from Eq. (2.19) that we can express a general reflection in two different forms: $V(\theta) = \mathrm{NOT}\,U(\theta) = U(-\theta)\,\mathrm{NOT}$. If use the first expression for $V(\theta_1)$ and the second expression for $V(\theta_2)$, we get

$$V(\theta_2)V(\theta_1) = U(-\theta_2)\,\mathrm{NOT}\,\mathrm{NOT}\,U(\theta_1) = U(-\theta_2)U(\theta_1) = U(\theta_1-\theta_2),$$

   where we used the fact that doing two consecutive NOTs amounts to doing nothing and that two consecutive rotations amount to a single rotation with the two angles being added, as shown above.

Solution to Exercise 4.7

Following Eq. (4.14), we should obtain [00] and [11], with probability 50% each. Indeed:



Quest 5: Algorithm virtuoso

One of the main motivations for studying quantum computing is the fact that quantum computers can solve some problems much faster than we know using the computers we currently have. To distinguish quantum computers from ordinary computers, we will use the general term classical computer to refer to any computational device that we currently have. This includes your laptop or desktop computer, but also very small computers such as in your smartphone or smartwatch, as well as very big and powerful *supercomputers* that may occupy a complete room. What distinguishes all these computers from quantum computers is not how big, small, slow, or fast they are, but the fact that their inner workings can be described by classical physics (more specifically, electromagnetism). In other words, their hardware operates in ways that can be described by old physical theories that predate quantum mechanics. This is a bit similar to how the music composed by Mozart and Bach is called *classical music*. Just like classical computers, classical music also does not take full advantage of more advanced musical instruments, such as the electric guitar or synthesizers.

As a consequence of the kind of hardware that classical computers have, all information they store – be it a picture, a sound file, a video or a web page – are represented by bitstrings, i.e., long sequences of zeroes and ones. This information is processed by following some rules that describe how these zeroes and ones should be modified to get a useful answer. We call this sequence of instructions an algorithm. You can think of an algorithm as a recipe – it is a sequence of instruction such that if you carefully follow them you will get the desired outcome, such as a chocolate brownie! For example, an algorithm might take two binary strings as input and produce another binary string that contains the sum of the two original strings (when interpreted as numbers) as output. Just like recipes, algorithms were originally executed by people. In fact, the word 'computer' used to refer to a person who is computing. Nowadays, however, algorithms are run on actual computers. Since computers generally are not very smart, they need the algorithm to be described to them in an extremely precise way. This description, a program, is a concrete implementation of the abstract algorithm in such a way that the computer can understand it. For this, we need to use some programming language which the computer is able to translate by itself into elementary operations on zeroes and ones, and then execute them in the actual hardware.

When you design an algorithm, program it in your computer, and run it, you want to get the answer in some reasonable amount of time. The actual time it takes to get the answer, however, depends on many factors:

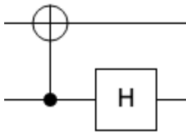
1. how fast your computer is at executing different elementary instructions,
2. whether the input of your program is read from a hard disk, a network connection, or directly from memory,
3. which programming language you use to specify your program (and the version of the compiler or interpreter that runs the program),

and so forth. This makes it very hard to compare different algorithms.

To make the comparison of different algorithms easier and more fair, computer scientists do not look at how long it takes to run the algorithm on a specific computer configured in some particular way. Instead they count the number of elementary operations the algorithm performs. This way they can make sure that they are comparing the algorithms themselves and not the computers the algorithms are running on (indeed, a good algorithm running on a very slow computer might appear worse than a bad algorithm running on a very fast computer). More specifically, what computer scientists want to know is how the number of operations grows with the *size* of the problem the algorithm must solve. Indeed, the larger amount of data you

need to process, the longer it will take no matter how good the algorithm is. So what you want to know is whether your algorithm will still be able to cope when the data gets extremely large. The area of computer science that studies this is called computational complexity.

In quantum computing, we are interested in designing quantum algorithms which solve computational problems by manipulating quantum states instead of bitstrings. The elementary instructions that we will use are gates, such as the Hadamard gate, the controlled-NOT gate, or a measurement. We will specify our quantum algorithms pictorially in terms of a quantum circuit, which you already have a lot of experience crafting in QUIRKY. But we could also use a textual representation like you would expect from an ordinary computer program. For example, the left-hand side circuit could be represented by the right-hand program text:




```
CNOT q1 -> q2;
H q1;
```

where `q1` and `q2` refer to the two qubits (recall that the bottom wire corresponds to the *first* qubit). Given a computational problem, we can then compare the number of elementary instructions that the best known quantum and classical algorithms use to solve it. In this way, we can get a precise understanding of the advantage offered by future quantum computers.

5.1 Talking to oracles

In this quest, we will look at several quantum algorithms and see how they compare to classical algorithms that solve the same problem. Since it is generally very hard to understand how many elementary operations are required to solve a certain computational problem, in this quest we will look at a simpler measure of algorithm complexity (this is also what computer scientists do when they want to make their life a bit simpler).

When a computer is running a program, it has to perform several different types of operations. The slowest type of operations are those that need to access data. For example, when reading it from the memory, hard drive or – in the worst case – another computer accessible over the internet. Once the relevant piece of data has been read, processing it can be relatively quick. Because of this, we can get a rough idea for how long a certain algorithm will run if we only count those instructions that access data.

You might be familiar with this when opening a complicated web page or a large document. It can take a while for it to load, but once it has loaded, interacting with the web page or scrolling around the document and inserting another line is usually quite quick.

Another way to think about this, which we will use in this quest, is that the information you are trying to access is actually produced by another algorithm or a subroutine within your algorithm. Moreover, this subroutine is very slow. For example, it might be reading the information from the hard disk or accessing it through the internet. Or it might not even have the answer readily available and instead have to produce it from scratch by performing some very complicated calculation. Either way, this subroutine always takes a very long time to come up with the answer, so you want to call it as few times as possible. We call such subroutine an oracle since it is very wise and knows all the answers, but also a bit slow and hence needs to think for a while to deliver the answer.

More formally, a classical oracle is just a function $f\colon \{0,1\}^n \to \{0,1\}$ where $\{0,1\}^n$ denotes the set of all n-bit binary strings. You can think of the input $x \in \{0,1\}^n$ to this function as a question and the output, the bit $f(x)$, as the answer. Every time you evaluate the function f on some input, you are asking a question that corresponds to that input, and you get a yes/no answer that corresponds to the value of the function.

For example, you can model the access to a hard drive or memory using an oracle. Say, if you have 4 bits of memory, you can model them as a function $f : \{0,1\}^2 \to \{0,1\}$ that, given a

binary address, returns the corresponding bit. For example, if you want to find out all four bits, you would need to evaluate f four times to get the four values $f(00), f(01), f(10), f(11)$.

What is important to note is that in our setup the kinds of computational problems we are interested to solve are *not* about finding the value of f on a specific input. Indeed, such problems are trivial since they can be solved by consulting the oracle just once, because this is precisely what the oracle does – it tells you the value of the function on any input of your choice. Hence, the kinds of problems we are interested are more subtle. What we want is to determine some property of the function f by evaluating it as few times as possible.

For example, let's say we wanted to know whether $f(x) = 0$ for all $x \in \{0,1\}^n$. In this case, what we could do is to ask the oracle about random values of x until we find one such that $f(x) = 1$. We will soon see other examples of such problems.

5.1.1 Reversible computation

Before we get carried away with trying to find exciting new quantum algorithms, let us first make sure that we can still compute on a quantum computer everything that we can compute on an ordinary computer. In other words, let us first make sure that quantum computers are actually not *less* powerful than ordinary (classical) computers! This is not entirely obvious, since the way that quantum computers work is very different. In particular, all operations on a quantum computer are reversible or invertible, as we already mentioned in §2.4.2 and §4.1.3, but this is normally not the case on an ordinary computer. Who has never erased a file by accident or forgotten to save the changes in their document and lost all their work? If all operations would be reversible, you would never have to worry about such trivial matters.

This raises the following question: how can we see that quantum computers can compute everything that ordinary computers can compute? One way to address this is to show that reversibility does not in fact restrict what an ordinary computer can compute, and hence it is also not a restriction for quantum computers. In other words, we will show that any computation can be made reversible and hence run on a quantum computer.

To get an idea of how this can be done, let us consider the simple example of computing the logical AND function of two bits. If both bits are 1, AND evaluates to 1, otherwise it evaluates to 0. Thus, we can represent the AND function by the following function table:

x_1	x_2	$\mathrm{AND}(x_1, x_2)$
0	0	0
0	1	0
1	0	0
1	1	1

(5.1)

Using the notation from §3.1, we can write this down mathematically as the following operation on two bits:

$$[x_1, x_2] \mapsto [\mathrm{AND}(x_1, x_2)].$$

Clearly this operation is not reversible because it does not have the same number of output bits as input bits. Indeed, if you only know that the AND of two bits is 0 then you cannot infer the precise state of the two bits – as the function table shows, there are three possible options.

How can we fix this problem? Let's try to keep the first bit around and introduce a second output bit in which we store the answer:

$$[x_1, x_2] \mapsto [x_1, \mathrm{AND}(x_1, x_2)].$$

This is better, since now we are mapping two bits to two bits. But is it reversible? That is, given the output, can we always reconstruct the input? Well, we can surely reconstruct the first bit of

the input, x_1, since we also have it in the output. How about x_2 then? If $x_1 = 0$ then according to Eq. (5.1) the output is $[00]$ – irrespective of the value of x_2. This means that, again, we cannot always reconstruct the input and hence this approach also sadly does not work.

This starts to look hopeless. Is it even possible at all to implement the AND function in a reversible fashion? When you get stuck, you have to think outside the box! In this case, who said that we should limit ourselves to two bits only? If we keep both input bits around and introduce a third bit to store the answer, this should surely make the operation reversible:

$$[x_1, x_2, 0] \mapsto [x_1, x_2, \mathrm{AND}(x_1, x_2)]. \tag{5.2}$$

Now there is no problem to invert the operation – given any output bitstring, we can simply forget what the last bit contains and replace it by $[0]$ to recover the input bitstring. For example, if the output is $[111]$ then the input must have been $[110]$.

So are we done? Not so fast! Note that Eq. (5.2) only specifies the operation *partially* – if the input is $[111]$, or any other bitstring that ends with 1, then Eq. (5.2) does *not* say how the operation should act on this input. Since the four input strings that end with 0 are mapped to four distinct output strings, it is clear that we can extend Eq. (5.2) in some arbitrary fashion to a reversible operation of three bits. But is there some systematic way of going about this?

For this, note that Eq. (5.2) flips the last bit from $[0]$ to $[1]$ if $\mathrm{AND}(x_1, x_2) = 1$. Similarly, if the last bit instead happens to be $[1]$, we could simply define our operation to flip it back to $[0]$ whenever $\mathrm{AND}(x_1, x_2) = 1$. In other words, we can extend Eq. (5.2) to *all* possible input strings as follows:

$$[x_1, x_2, y] \mapsto [x_1, x_2, y \oplus \mathrm{AND}(x_1, x_2)], \tag{5.3}$$

for all $x_1, x_2, y \in \{0, 1\}$, where '$\oplus$' denotes addition modulo 2.

The operation in Eq. (5.3) is now defined on all possible inputs. But is it finally reversible? Yes, it is! In fact, this operation is its own inverse! That is, if we do the operation twice, we get the original input back:

$$\begin{aligned}[x_1, x_2, y] &\mapsto [x_1, x_2, y \oplus \mathrm{AND}(x_1, x_2)] \\ &\mapsto [x_1, x_2, y \oplus \mathrm{AND}(x_1, x_2) \oplus \mathrm{AND}(x_1, x_2)] = [x_1, x_2, y].\end{aligned}$$

In the third step we used that $a \oplus a = 0$ for any $a \in \{0, 1\}$, see Eq. (3.20). Thus we have successfully found a way to compute the AND function in a reversible way.

5.1.2 Bit oracles

This idea works not only for the logical AND function but in fact for any function

$$f\colon \{0, 1\}^n \to \{0, 1\}$$

that takes n bits as an input and returns a single bit. For any such function f we can define a reversible operation on $(n + 1)$ bits:

$$[x_1, \ldots, x_n, y] \mapsto [x_1, \ldots, x_n, y \oplus f(x_1, \ldots, x_n)] \tag{5.4}$$

for all $x_1, \ldots, x_n, y \in \{0, 1\}$. Just like Eq. (5.3), this operation is its own inverse, i.e., applying it twice is equivalent to doing nothing.

It is reassuring that we can implement any function $f : \{0, 1\}^n \to \{0, 1\}$ reversibly as in Eq. (5.4). First of all, it means that any computation on a classical computer can be done in such a way that we can recover the original input. Second, once the computation is made reversible, we can run it also on a quantum computer. This implies that quantum computers can compute everything that classical computers can! Third, this means that we can implement any function f as an oracle on a quantum computer.

Let's see how this actually works. The quantum version of the operation in Eq. (5.4) is defined as follows:

$$U_f \left|x_1, \ldots, x_n, y\right\rangle = \left|x_1, \ldots, x_n, y \oplus f(x_1, \ldots, x_n)\right\rangle \tag{5.5}$$

for all $x_1, \ldots, x_n, y \in \{0, 1\}$. Eq. (5.5) defines how U_f acts on all basis states of $n+1$ qubits. As usual, we extend this definition to an arbitrary state on $n+1$ qubits by linearity. Since U_f simply permutes the basis states, you can check as in Exercise 4.4 that it defines a valid quantum operation.

We call the quantum operation U_f defined in Eq. (5.5) the bit oracle for f (we could also call the function in Eq. (5.4) a classical bit oracle, but we will not need this name). The term 'oracle' simply means that applying this operation is like asking an all-powerful oracle to tell us the value of the function on any given input. We do not know how the oracle is implemented precisely nor where it gets its answer from, we will just count how many questions we need to ask the oracle to learn some property of the function f. Many interesting algorithmic problems can be modeled in this fashion, as we will see in the remainder of this quest.

The concept of an oracle is a bit like playing the 'guess my number' game. Your friend (the oracle) comes up with some number and you ask them questions of the form "is your number x"? Your friend answers each of these questions with either "yes" or "no". In other words, your friend is hiding a function that evaluates to "no" on all inputs, except on one (the number they have in mind). With every question you ask, you gain more information about which number it could possibly be. You may wonder how many questions you need to ask to determine the number. Moreover, what if you could ask your questions to a quantum oracle such as U_f instead of your friend? Could you figure out the answer with less questions? In this week's quest, we will see several interesting examples where this indeed is the case!

Let us discuss some examples of bit oracles. Assume f is the AND function. According to Eq. (5.1), we can interpret AND as multiplication modulo 2 since $\text{AND}(x_1, x_2) = x_1 x_2$. The corresponding bit oracle

$$U_{\text{AND}} \left|a, b, c\right\rangle = \left|a, b, c \oplus ab\right\rangle$$

is nothing but the Toffoli gate from Exercise 4.4. Even for $n = 1$ and the function $f(x) = x$ that simply returns its input, we get an interesting result: for all $a, b \in \{0, 1\}$,

$$U_f \left|a, b\right\rangle = \left|a, b \oplus a\right\rangle .$$

This is just the familiar controlled-NOT gate $\text{CNOT}_{1\to 2}$ from Eq. (3.47) in §3.2.4! Thus, the bit oracle construction reproduces several interesting quantum operations that we had previously defined by hand. In the following exercise you can try to implement all other bit oracles for functions of a single bit.

Exercise 5.1: Bit oracle for a one-bit function

Let $f : \{0, 1\} \to \{0, 1\}$ be a function with a single input and output bit. Such a function is fully specified by the values $f(0), f(1) \in \{0, 1\}$. These are two bits, so there are precisely four such functions. We just discussed how to implement the bit oracle U_f for the function $f(x) = x$. Can you implement the bit oracle U_f for the other three functions in QUIRKY?

5.1.3 Sign oracles

Since the bit oracle U_f is a quantum operation, we can not only apply it to basis states $\left|x_1, \ldots, x_n, y\right\rangle$ but also to general quantum states. Why would we like to do such a thing? Well, if we only ever ask the oracle 'classical' questions then there is little hope of achieving a quantum speedup! Given this motivation, let us investigate how the bit oracle U_f for an

arbitrary function $f : \{0,1\}^n \to \{0,1\}$ acts when the last register is set to $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$. First, notice the following interesting fact:

$$\text{NOT}\,|-\rangle = \frac{1}{\sqrt{2}}(\text{NOT}\,|0\rangle - \text{NOT}\,|1\rangle) = \frac{1}{\sqrt{2}}(|1\rangle - |0\rangle) = -\,|-\rangle\,.$$

That is, if we invert a qubit in state $|-\rangle$ then we pick up a sign. We can similarly compute how the bit oracle acts on a state of the form $|x_1,\ldots,x_n\rangle \otimes |-\rangle$. By linearity, Eq. (5.5) gives us

$$\begin{aligned}
&U_f\,(|x_1,\ldots,x_n\rangle \otimes |-\rangle) \\
&= U_f\big(\frac{1}{\sqrt{2}}\,|x_1,\ldots,x_n,0\rangle - \frac{1}{\sqrt{2}}\,|x_1,\ldots,x_n,1\rangle\big) \\
&= \frac{1}{\sqrt{2}}\,|x_1,\ldots,x_n,f(x_1,\ldots,x_n)\rangle - \frac{1}{\sqrt{2}}\,|x_1,\ldots,x_n,f(x_1,\ldots,x_n)\oplus 1\rangle \\
&= |x_1,\ldots,x_n\rangle \otimes \frac{1}{\sqrt{2}}\big(|f(x_1,\ldots,x_n)\rangle - |f(x_1,\ldots,x_n)\oplus 1\rangle\big) \\
&= (-1)^{f(x_1,\ldots,x_n)}\,|x_1,\ldots,x_n\rangle \otimes |-\rangle\,.
\end{aligned}$$

In other words, we end up returning the last qubit back in the state $|-\rangle$, but we pick up an overall minus sign when $f(x_1,\ldots,x_n) = 1$. Effectively, we have implemented the following quantum operation on the first n qubits:

$$O_f\,|x_1,\ldots,x_n\rangle = (-1)^{f(x_1,\ldots,x_n)}\,|x_1,\ldots,x_n\rangle\,, \tag{5.6}$$

for all bitstrings $x_1,\ldots,x_n \in \{0,1\}$. We call O_f the sign oracle for f.

Interestingly, for a function $f\colon \{0,1\}^n \to \{0,1\}$, the sign oracle O_f operates on n qubits since it stores the output in the amplitude. This is different from the bit oracle U_f, which stores the output in an additional qubit and therefore operates on $n+1$ qubits.

At first glance, the sign oracle does not seem to do much since it only introduces an overall sign when acting on a basis state, which we know from Exercise 2.7 cannot be observed. However, when we apply it to a superposition then the sign oracle can introduce relative signs, so we can get an interesting result. For example, for $n = 2$ qubits, if we apply a sign oracle O_f to a general two-qubit state

$$|\psi\rangle = \psi_{00}\,|00\rangle + \psi_{01}\,|01\rangle + \psi_{10}\,|10\rangle + \psi_{11}\,|11\rangle$$

then we obtain

$$O_f\,|\psi\rangle = (-1)^{f(0,0)}\psi_{00}\,|00\rangle + (-1)^{f(0,1)}\psi_{01}\,|01\rangle + (-1)^{f(1,0)}\psi_{10}\,|10\rangle + (-1)^{f(1,1)}\psi_{11}\,|11\rangle\,.$$

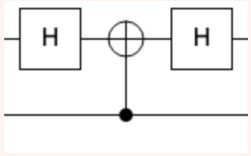
As it turns out, the sign oracle O_f is indeed useful and often much easier to apply in quantum algorithms than the bit oracle U_f, so from now on we will not use the bit oracle anymore.

Exercise 5.2: Sign oracle for a one-bit function

Recall from Exercise 5.1 that there are four functions $f : \{0,1\} \to \{0,1\}$ with a single input and output bit. Can you implement the sign oracle O_f for each of them in QUIRKY?

Homework 5.1: Determine the function from its sign oracle

Consider the following two-qubit circuit (as usual, the bottom wire is the first qubit):



For what function $f\colon \{0,1\}^2 \to \{0,1\}$ is this the corresponding sign oracle?

Hint: Use that H NOT $H = Z$, which follows from Exercise 4.5.

Let us briefly summarize what we have achieved so far: Using bit oracles, we can ask a quantum computer to evaluate a function $f\colon \{0,1\}^n \to \{0,1\}$ in exactly the same way as we would ask an ordinary computer that operates reversibly (compare Eqs. (5.4) and (5.5)). This is important, because it means that we are not comparing apples and oranges when we ask how many questions to the bit oracle U_f are required to learn something about f versus how many times one would have to evaluate f on an ordinary computer to learn the same thing. And since we just showed that the sign oracle O_f can always be implemented using the bit oracle U_f, it makes no difference if we ask questions to the sign oracle O_f instead.

5.2 Quantum algorithms

In this section we will see several quantum algorithms that can solve a computational problem much faster than any classical algorithm. Such speedups are very surprising because quantum mechanics at first glance does not appear to have anything to do with computation. Nevertheless, quantum-mechanical phenomena such as interference can enable very impressive computational speedups. As already explained earlier, we are working in a computational setting where we count only the number of questions. That is, assuming the ability to evaluate some function f on any input, on how many inputs do you need to evaluate it to determine some property of f. Equivalently, having access to an oracle that can evaluate f, how many times do you need to use this oracle to determine some property of f.

5.2.1 Deutsch's algorithm

It is late on Sunday evening. Alice and Bob have just finished their homework assignments for the quantum computing class and are about to watch a 3D movie. When they turn on their holographic television set, they discover that the movie screening has been postponed due to unexpected dramatic news coverage coming from the International Transgalactic Station. There has been a terrible accident and a module containing two crew members, Hila and Iman, has separated from the main mothership. The last message received from the module was that Iman has been injured and is heavily bleeding – he is in need of an urgent blood transfer. What makes the situation complicated is that Hila and Iman are somewhat in shock and forgot their own blood type – they can only remember that they each had blood type A or B. The news presenter is appealing to the general public for help with suggesting a way for Hila and Iman to determine if they have the same blood type, because if that was the case Hila could transfer her blood to Iman to save his life. This is because the medical kit in Hila and Iman's module includes a lympho-transcoder that is able to convert either of the two blood types to the opposite one. Hence, even if it turns out that their blood types are mismatching, Hila can convert her's to the opposite one using the lympho-transcoder.

Upon hearing these news, Alice and Bob immediately abandon their plan to watch a movie and start to ponder what could be done to help Hila and Iman. The news coverage continues with some additional information. Luckily, it turns out that the module involved in the accident

contains a database chip that stores Hila and Iman's blood type. We can model this by a function $f\colon \{0,1\} \to \{0,1\}$, where

$$f(0) = \begin{cases} 0 & \text{if Hila's blood type is A,} \\ 1 & \text{if Hila's blood type is B.} \end{cases}$$

and

$$f(1) = \begin{cases} 0 & \text{if Iman's blood type is A,} \\ 1 & \text{if Iman's blood type is B.} \end{cases}$$

What needs to be determined is whether $f(0) = f(1)$ or not!

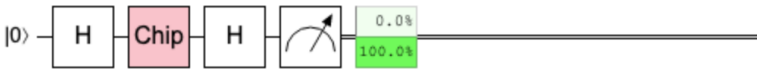
The solution seems at hand: Hila and Iman simply need to query the database twice to read out their respective blood types, $f(0)$ and $f(1)$, and then compare the two values to see if they are the same. Unfortunately, the accident has partly fried the control logic of the database chip and the news presenter reports that after the first query the database chip will likely burn out completely.

Our two protagonists are at an impasse. Clearly, any classical algorithm needs to evaluate f twice in order to determine whether $f(0) = f(1)$. Indeed, if you know only the value of $f(0)$ then whether $f(0) = f(1)$ still depends on the value of $f(1)$ and, unless you also compute $f(1)$, you would not be able to tell whether $f(0) = f(1)$. Similarly, if you know only $f(1)$ then you cannot compare it with $f(0)$ unless you also know what the value of $f(0)$ is. No matter what strategy you use, you need to know both $f(0)$ and $f(1)$ in order to determine whether $f(0) = f(1)$. Is there really no way around this?

After flipping through some manuals, Bob discovers that the database chip can be switched into *quantum mode*. When enabled, the database chip no longer evaluates the function classically but instead implements the sign oracle O_f. Could this somehow be used to solve the problem? Alice thinks about it for a second and suddenly realizes that this is precisely what Deutsch's algorithm is for! Alice and Bob quickly go over some calculations to confirm that it works and sit down to write an intergalactic e-mail with instructions to Hila and Iman on how they can solve the problem. Their instructions are as follows:

1. Prepare a qubit in state $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$.
2. Use the database chip in quantum mode to apply the operation O_f.
3. Apply the Hadamard gate H to the output qubit and measure it.
4. If the outcome is 0 then Hila and Iman have the same blood type and otherwise they have different blood types.

Note that in this procedure Hila and Iman only query the database chip *once* to determine if they have the same blood type. Here is an implementation of the algorithm in QUIRKY:



The picture shows that the outcome is 1, so Hila and Iman have different blood types.

But why does Deutsch's algorithm work? Let us analyze it step by step. The first Hadamard gate creates the state $|+\rangle = H|0\rangle$. Next, we apply the sign oracle O_f, which leads to the state

$$\begin{aligned} O_f|+\rangle &= \frac{1}{\sqrt{2}} O_f|0\rangle + \frac{1}{\sqrt{2}} O_f|1\rangle \\ &= \frac{1}{\sqrt{2}} (-1)^{f(0)}|0\rangle + \frac{1}{\sqrt{2}} (-1)^{f(1)}|1\rangle . \end{aligned}$$

After applying the second Hadamard, we obtain the following state:

$$\begin{aligned} HO_f |+\rangle &= \frac{1}{\sqrt{2}}(-1)^{f(0)} H|0\rangle + \frac{1}{\sqrt{2}}(-1)^{f(1)} H|1\rangle \\ &= \frac{1}{\sqrt{2}}(-1)^{f(0)} |+\rangle + \frac{1}{\sqrt{2}}(-1)^{f(1)} |-\rangle \\ &= \frac{1}{2}(-1)^{f(0)} (|0\rangle + |1\rangle) + \frac{1}{2}(-1)^{f(1)} (|0\rangle - |1\rangle) \\ &= \frac{(-1)^{f(0)} + (-1)^{f(1)}}{2} |0\rangle + \frac{(-1)^{f(0)} - (-1)^{f(1)}}{2} |1\rangle . \end{aligned} \tag{5.7}$$

Note that the two signs $(-1)^{f(0)}$ and $(-1)^{f(1)}$ are added in the first amplitude, but subtracted in the second amplitude. Depending on the values of $f(0)$ and $f(1)$, for each amplitude we will observe either constructive or destructive interference (see §2.6.1). In fact, which of the two amplitudes will remain is determined only by whether $f(0)$ and $f(1)$ are equal or not:

$$\begin{aligned} f(0) = f(1): &\qquad HO_f |+\rangle = \pm |0\rangle , \\ f(0) \neq f(1): &\qquad HO_f |+\rangle = \pm |1\rangle . \end{aligned} \tag{5.8}$$

It is a good exercise to verify this explicitly:

Exercise 5.3: Verifying Deutsch's algorithm

Recall from Exercise 5.1 that there are four functions $f : \{0,1\} \to \{0,1\}$. For each function, compute the state $HO_f |+\rangle$ using Eq. (5.7).

Eq. (5.8) shows that the final measurement yields outcome 0 if and only if $f(0) = f(1)$. Thus, Deutsch's algorithm correctly determines whether $f(0) = f(1)$. Importantly, it evaluates the function $f : \{0,1\} \to \{0,1\}$ only once by using the sign oracle. In contrast, we discussed above that any classical algorithm necessarily needs to evaluate both function values $f(0)$ and $f(1)$ separately.

Another way to interpret Deutsch's algorithm is that it computes the sum of the two bits $f(0)$ and $f(1)$ modulo two. This is because $f(0) \oplus f(1) = 0$ if and only if $f(0) = f(1)$. Indeed, recall from Eq. (3.20) that addition modulo two works as follows:

x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

(5.9)

This is also why the sum modulo two is known as the *XOR* (short for 'exclusive OR') of the two bits, since it is one if exactly one of the two bits is set.

5.2.2 Hadamard transform and interference

While Deutsch's algorithm is very surprising, it achieves only a very small improvement over the best classical algorithm, namely 1 evaluation of f instead of 2. This could be useful if evaluating f takes a very long time, say a year. But if it takes only a millisecond, then most people would not mind to wait for two milliseconds to get the answer (and pay much less for it because they would not need to use a quantum computer)! To demonstrate the usefulness of quantum computers, we would like to speed up computations by more than just a factor of 2.

There is no hope of getting a larger speedup if we only look at functions of a single bit, since those can be fully determined by two evaluations: $f(0)$ and $f(1)$. We will therefore look

more generally at functions $f\colon \{0,1\}^n \to \{0,1\}$ with n input bits, since there are many more such functions (indeed, there are 2^{2^n} of them). Recall that our goal is *not* to evaluate a function (indeed, we can do that with a single query to the oracle) but rather to learn some interesting property of the function that relates the values it takes on different inputs. Are there properties of n-bit functions that we can learn very efficiently by using clever quantum algorithms?

To generalize Deutsch's algorithm, observe that its key ingredient was to introduce a Hadamard gate H before and after the sign oracle. We can do something very similar if we have an n-bit function. In this case, the sign oracle O_f is a quantum operation of n qubits, so we could simply apply Hadamard gates on all qubits before and after the oracle. The quantum operation that applies Hadamards on each of the n qubits in parallel is called the Hadamard transform. Recall from §3.2.3 that we can write it as follows:

$$H \otimes \cdots \otimes H.$$

Let us first see what happens if we apply the Hadamard transform to a basis state. For $|0\ldots0\rangle$, the result is simply the uniform superposition over all basis states:

$$\begin{aligned}(H \otimes \cdots \otimes H)\,|0\ldots0\rangle &= H\,|0\rangle \otimes \cdots \otimes H\,|0\rangle \\ &= \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes \cdots \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \\ &= \frac{1}{\sqrt{2^n}}\left(|0\ldots00\rangle + |0\ldots01\rangle + \ldots + |1\ldots11\rangle\right).\end{aligned}$$

This state is a superposition of 2^n basis states. There is a more compact way of writing this:

$$(H \otimes \cdots \otimes H)\,|0\ldots0\rangle = \frac{1}{\sqrt{2^n}} \sum_{y_1,\ldots,y_n \in \{0,1\}} |y_1,\ldots,y_n\rangle \tag{5.10}$$

The notation $\sum_{y_1,\ldots,y_n\in\{0,1\}}$ means that you compute the sum of $|y_1,\ldots,y_n\rangle$ for all possible choices of the bits $y_1,\ldots,y_n$.

More generally, the output on an arbitrary basis state is always a superposition of all 2^n basis states with all amplitudes equal except for signs. Figuring out what exactly those signs are is a bit tricky. As a warm-up, try it first for the $n = 1$ and $n = 2$ cases.

Exercise 5.4: Two Hadamards

Recall from Eq. (2.20) that $H\,|x_1\rangle = \left(|0\rangle + (-1)^{x_1}\,|1\rangle\right)/\sqrt{2}$, for any $x_1 \in \{0,1\}$.

1. Write the state $H\,|x_1\rangle$, for arbitrary $x_1 \in \{0,1\}$, as follows:

$$H\,|x_1\rangle = \frac{1}{\sqrt{2}} \sum_{y_1\in\{0,1\}} (-1)^{\boxed{???}}\,|y_1\rangle\,,$$

   where $\boxed{???}$ is some expression involving $x_1, y_1 \in \{0,1\}$. Determine this expression.

2. Write the state $(H \otimes H)\,|x_1, x_2\rangle$, for arbitrary $x_1, x_2 \in \{0,1\}$, in the form

$$(H \otimes H)\,|x_1, x_2\rangle = \frac{1}{2} \sum_{y_1,y_2\in\{0,1\}} (-1)^{\boxed{???}}\,|y_1, y_2\rangle\,,$$

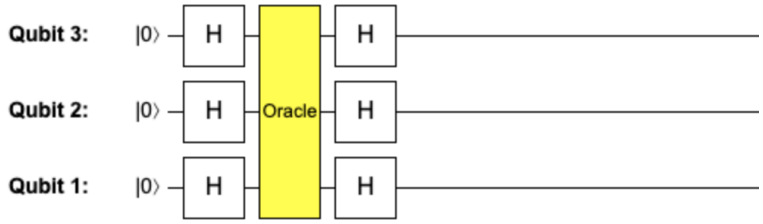
   where $\boxed{???}$ stands for some expression involving $x_1, x_2, y_1, y_2 \in \{0,1\}$. Can you determine what this expression is?

Did you figure out the solution? Good, then you are allowed to read on!

In general, you can derive the following formula that describes the sign pattern of the amplitudes you get when applying the Hadamard transform to any n-qubit basis state: For any $x_1, \ldots, x_n \in \{0,1\}$,

$$(H \otimes \cdots \otimes H) \left|x_1, \ldots, x_n\right\rangle = \frac{1}{\sqrt{2^n}} \sum_{y_1,\ldots,y_n \in \{0,1\}} (-1)^{x_1 y_1 + \ldots + x_n y_n} \left|y_1, \ldots, y_n\right\rangle . \tag{5.11}$$

We can now generalize the Deutsch algorithm in a straightforward way. Starting with the n-qubit basis state $|0 \ldots 0\rangle$, we first apply the Hadamard transform, next the sign oracle O_f for the function $f\colon \{0,1\}^n \to \{0,1\}$ of interest, and finally another Hadamard transform. For example, for $n = 3$, this corresponds to the following QUIRKY circuit:



For general n, the mathematical formula for the final n-qubit state is:

$$(H \otimes \cdots \otimes H) O_f (H \otimes \cdots \otimes H) \left|0 \ldots 0\right\rangle . \tag{5.12}$$

Can we write down this state more explicitly? As before, we can compute this step by step. First, we apply the Hadamard transform to the all-zeros basis state. By Eq. (5.10), we obtain the uniform superposition over all basis states:

$$(H \otimes \cdots \otimes H) \left|0 \ldots 0\right\rangle = \frac{1}{\sqrt{2^n}} \sum_{x_1,\ldots,x_n \in \{0,1\}} \left|x_1, \ldots, x_n\right\rangle .$$

Next we apply the sign oracle O_f:

$$O_f (H \otimes \cdots \otimes H) \left|0 \ldots 0\right\rangle = \frac{1}{\sqrt{2^n}} \sum_{x_1,\ldots,x_n \in \{0,1\}} (-1)^{f(x_1,\ldots,x_n)} \left|x_1, \ldots, x_n\right\rangle .$$

What does the final Hadamard transform achieve? By linearity, we can apply Eq. (5.11) to each basis vector, so we obtain the following expression for the state in Eq. (5.12):

$$\begin{aligned}
&(H \otimes \cdots \otimes H) O_f (H \otimes \cdots \otimes H) \left|0 \ldots 0\right\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{x_1,\ldots,x_n \in \{0,1\}} (-1)^{f(x_1,\ldots,x_n)} \frac{1}{\sqrt{2^n}} \sum_{y_1,\ldots,y_n \in \{0,1\}} (-1)^{x_1 y_1 + \ldots + x_n y_n} \left|y_1, \ldots, y_n\right\rangle \\
&= \sum_{y_1,\ldots,y_n \in \{0,1\}} \frac{1}{2^n} \left(\sum_{x_1,\ldots,x_n \in \{0,1\}} (-1)^{x_1 y_1 + \ldots + x_n y_n} (-1)^{f(x_1,\ldots,x_n)} \right) \left|y_1, \ldots, y_n\right\rangle ,
\end{aligned} \tag{5.13}$$

where we exchanged the two sums to obtain the last equality. For $n = 1$ this is exactly the same as Eq. (5.7). In general, however, this expression looks quite unwieldy and hard to interpret – it seems like the circuit computes some superposition of basis states, where the amplitude is some strange sum of plus and minus signs!

Let us try to get some intuition by looking at the amplitude of $|0 \ldots 0\rangle$ in Eq. (5.13). The square of this number is the probability that, when we measure the n qubits, all outcomes are zero. Since this amplitude corresponds to $y_1 = \ldots = y_n = 0$, it is given simply by

$$\frac{1}{2^n} \sum_{x_1,\ldots,x_n \in \{0,1\}} (-1)^{f(x_1,\ldots,x_n)} .$$

What does this mean? Suppose that there are N_f input bitstrings for which f evaluates to zero and $2^n - N_f$ bitstrings for which f evaluates to one. Then,

$$\frac{1}{2^n} \sum_{x_1,\ldots,x_n \in \{0,1\}} (-1)^{f(x_1,\ldots,x_n)} = \frac{N_f - (2^n - N_f)}{2^n} = \frac{2N_f - 2^n}{2^n}.$$

There are two interesting extreme cases:[13]

- If f is a *constant function* then either $N_f = 0$ (for the all-zeros function) or $N_f = 2^n$ (for the all-ones function). Either way, the amplitude is $\pm 2^n/2^n = \pm 1$. Since the squares of all amplitudes should sum to 1, we conclude that all other amplitudes in Eq. (5.13) must be 0. In other words, whenever f is constant, the state in Eq. (5.13) is simply $\pm |0\ldots0\rangle$. If we measure all n qubits of this state then all outcomes will be zero.

- If f is a *balanced function*, which means that there are as many zeros as ones, then $N_f = 2^n/2$ and so the amplitude is 0. This means that if we measure the state in Eq. (5.13) then we can never get all outcomes equal to zero. Hence, for a balanced function at least one of the measurement outcomes will always be one.

Note how these two cases correspond to very different patterns of interference (see §2.6.1). In the first case, the amplitude of $|0\ldots0\rangle$ gets amplified to ± 1 due to a highly focused *constructive* interference, while at the same time all other amplitudes simultaneously vanish due to a massively widespread destructive interference. In the second case, $|0\ldots0\rangle$ experiences *destructive* interference, causing non-zero amplitudes to pop up elsewhere. Hadamard transform illustrates the central importance of interference in quantum computing. In the next two sections, we will make crucial use of it to design quantum algorithms on a large number of qubits.

5.2.3 Deutsch-Jozsa algorithm

Are the above observations useful for anything? Yes, they are! Suppose $f\colon \{0,1\}^n \to \{0,1\}$ is an unknown function which is either constant or balanced. Then there is a simple quantum algorithm which can determine which of these two is the case, by using only a single evaluation of f (i.e., one application of O_f).

This algorithm is called the Deutsch-Jozsa algorithm, and it works in five simple steps:

1. Start with $|0\ldots0\rangle$.
2. Apply the Hadamard transform $H \otimes \cdots \otimes H$.
3. Apply the sign oracle O_f corresponding to the function f.
4. Apply the Hadamard transform $H \otimes \cdots \otimes H$ again.
5. Measure all qubits. If all outcomes are zero, the function f must be constant. Otherwise it must be balanced.

To see how this compares with classical algorithms, notice that any classical algorithm needs to evaluate the function f at least $\frac{2^n}{2} + 1$ times in the worst case. Indeed, suppose that we learn the function on half of the input bitstrings (i.e., on $2^n/2$ inputs) and we get the same answer every time. Then we still cannot conclude that the function is constant, since it could be the case that the function gives the opposite answer on the remaining half of the input bitstrings while still being balanced. Hence this is the worst case scenario. In this scenario, we have wasted

[13] For $n = 1$, every function is either constant or balanced. For $n > 1$ this is not true (e.g., the AND function is neither constant nor balanced).

$2^n/2$ questions and still have not learned anything useful. To be certain whether f is constant or balanced, we need to evaluate it on one more input. In total, this amounts to $\frac{2^n}{2}+1$ evaluations of f, compared to 1 evaluation in the quantum case. Note that even for moderate values such as $n = 100$ this difference is so dramatic that you would not be able to evaluate f so many times in any reasonable amount of time (indeed, by that time the sun would run out of fuel and you would have to relocate to another solar system).

To summarize: If $f\colon \{0,1\}^n \to \{0,1\}$ is a function that is either constant or balanced then the Deutsch-Jozsa algorithm can determine *using only one evaluation* of f which of the two is the case. This is exponentially better than any classical algorithm, which needs to evaluate the function on $\frac{2^n}{2}+1$ many inputs in the worst case.

Homework 5.2: Run Deutsch-Jozsa

The yellow Oracle box in QUIRKY implements the sign oracle for a function that is either constant or balanced. Implement the Deutsch-Jozsa algorithm in QUIRKY and use it to determine which of the two is the case.

5.2.4 Bernstein-Vazirani algorithm

Above, we discussed how to use the Hadamard transform to solve an interesting problem. Given a function $f\colon \{0,1\}^n \to \{0,1\}$ and the promise that f is either constant or balanced, the Deutsch-Jozsa algorithm was able to determine which of these two options was the case.

In this section we will discuss another interesting problem that one can solve by a slight variant of the same procedure. As before, we will start with a promise about the unknown function f that we are dealing with. This time, instead of assuming that it is constant or balanced, we will assume that it is of the following special form:

$$f(x_1,\ldots,x_n) = x_1a_1 \oplus \ldots \oplus x_na_n, \tag{5.14}$$

where $a_1,\ldots,a_n \in \{0,1\}$ is some fixed bitstring that defines the function.

If $n = 1$ then there are only two such functions:

- $f(x_1) = 0$, corresponding to $a_1 = 0$, and
- $f(x_1) = x_1$, corresponding to $a_1 = 1$.

If $n = 2$ then there are already four such functions:

- $f(x_1,x_2) = 0$, corresponding to $[a_1,a_2] = [0,0]$,
- $f(x_1,x_2) = x_2$, corresponding to $[a_1,a_2] = [0,1]$,
- $f(x_1,x_2) = x_1$, corresponding to $[a_1,a_2] = [1,0]$, and
- $f(x_1,x_2) = x_1 \oplus x_2$, corresponding to $[a_1,a_2] = [1,1]$.

Note that each of these functions computes the sum modulo two of some *subset* of the variables x_i. Which subset? Whenever $a_i = 1$, the corresponding variable x_i is included in the subset.

In general, there are 2^n choices of the bitstring $a_1,\ldots,a_n$ and hence 2^n functions f of the special form (5.14). In fact, we may think of Eq. (5.14) as a way of *hiding* the bitstring

$$[a_1,\ldots,a_n]$$

in the function f. How many evaluations of the function do we need to uncover it? Since

$$\begin{aligned} a_1 &= f(1,0,\ldots,0,0), \\ a_2 &= f(0,1,\ldots,0,0), \\ &\vdots \\ a_n &= f(0,0,\ldots,0,1), \end{aligned}$$

we conclude that n evaluations of the function f are certainly enough. For any classical algorithm, this is also optimal: each time you evaluate the function you only learn a single bit. Since the unknown bits $a_1, \ldots, a_n$ are completely arbitrary and there are n of them, you need to evaluate the function at least n times to learn them all.

We will now see that we can do much better using a quantum algorithm. To start, let us compute how the sign oracle for the function in Eq. (5.14) acts on the basis states:

$$\begin{aligned} O_f \left|x_1, \ldots, x_n\right\rangle &= (-1)^{x_1 a_1 \oplus \ldots \oplus x_n a_n} \left|x_1, \ldots, x_n\right\rangle \\ &= (-1)^{x_1 a_1 + \ldots + x_n a_n} \left|x_1, \ldots, x_n\right\rangle . \end{aligned} \tag{5.15}$$

In the second step we used that $(-1)^a$ only depends on a modulo two (i.e., on whether a is even or odd), so it does not matter whether we use addition modulo 2 ('$\oplus$') or the ordinary addition. How can this sign oracle be implemented? We do not really care, since our algorithm will treat the oracle as a black box. But since it is a nice exercise, you can try to figure this out in the following problem.

Exercise 5.5: Implementing the sign oracle (optional)

In this problem, you will implement the sign oracle for functions of the form (5.14).

1. When $n = 2$ there are four such functions, as we discussed above. Can you find for each of them a QUIRKY circuit that implements the sign oracle?

2. Explain in words or pictures how you can implement the sign oracle in the general case (i.e., when $n \geq 1$ and the bits $a_1, \ldots, a_n \in \{0, 1\}$ are arbitrary).

We now present the Bernstein-Vazirani algorithm, which uncovers the hidden bitstring $[a_1, \ldots, a_n]$ using a single evaluation of the sign oracle for f:

1. Start with $\left|0 \ldots 0\right\rangle$.

2. Apply the Hadamard transform $H \otimes \cdots \otimes H$.

3. Apply the sign oracle O_f corresponding to the function f.

4. Apply the Hadamard transform $H \otimes \cdots \otimes H$ again.

5. Measure all qubits. The measurement outcome is precisely the bitstring $[a_1, \ldots, a_n]$.

The algorithm is identical to the Deutsch-Jozsa algorithm in §5.2.3 except for the last step, which is even simpler.

Homework 5.3: Run Bernstein-Vazirani

The orange Oracle box in QUIRKY implements the sign oracle for a function of the form (5.14) with $n = 4$. Implement the Bernstein-Vazirani algorithm in QUIRKY and use it to determine the hidden bitstring $[a_1, a_2, a_3, a_4]$.

Why does this algorithm work? Now it is your turn to do the analysis!

Exercise 5.6: Verify Bernstein-Vazirani

The function $f(x_1, x_2) = x_2$ corresponds to the bitstring $[a_1, a_2] = [0, 1]$, as we saw earlier. Show that when you run the Bernstein-Vazirani algorithm for this function, the measurement outcome is indeed always $[a_1, a_2] = [0, 1]$. Don't just verify this using QUIRKY, but write down the state after each step yourself.

In the following homework problem you can analyze the general case:

Homework 5.4: Verify Bernstein-Vazirani (challenging)

Show that, when you run the Bernstein-Vazirani algorithm for a function of the form (5.14), the measurement outcome is $[a_1, \ldots, a_n]$ with 100% probability.

Hint: Since the first four steps of the algorithm are identical to the Deutsch-Jozsa algorithm, the state right before the measurement is given by Eq. (5.13).

5.3 Searching with Grover

After returning safely back to Earth, Hila and Iman have become good friends with Alice and Bob. The four of them decide to spend New Year's Eve together. This day also coincides with the drawing of the annual quantum lottery! They know that only one of the four can win the grand prize, but who will it be? If we label our four protagonists by bitstrings, say,

Name	x_1	x_2
Alice	0	0
Bob	0	1
Hila	1	0
Iman	1	1

then we can model the lottery by a function $f\colon \{0,1\}^2 \to \{0,1\}$ which evaluates to 1 for the bitstring corresponding to the winner. For example, if $f(1,0) = 1$ then Hila is the winner of this year's lottery.

How can our friends determine the winner? Using a classical algorithm they have to evaluate the function up to three times to determine the winner. Indeed, suppose that they learn that $f(0,1) = 0$ and $f(1,0) = 0$ – then they still do not know whether Alice or Iman is the winner! For archaic reasons that have long been forgotten, the rules of the lottery only allow evaluating the function once. But naturally, the lottery is happy to apply the sign oracle O_f to any two-qubit states of our protagonists' liking – after all, it is a quantum lottery…

Alice and her friends get together and start pondering. After a while, Bob grows impatient and suggests: *"Let us just follow the first few steps of Deutsch-Jozsa and Bernstein-Vazirani – surely the same trick will work once again…"* The others do not really know of a better alternative, so they go ahead and prepare the state

$$(H \otimes H)(|0\rangle \otimes |0\rangle) = |+\rangle \otimes |+\rangle = \frac{1}{2}\big(|00\rangle + |01\rangle + |10\rangle + |11\rangle\big).$$

Next, they hand the state to the quantum lottery, which applies the sign oracle O_f and returns the state. Let a and b denote the two bits that label the winner, i.e., $f(a,b) = 1$ and all other function values are zero. Then the two-qubit state returned by the lottery is the following:

$$\begin{aligned}
&\frac{1}{2}\big(|00\rangle + |01\rangle + |10\rangle + |11\rangle - 2\,|a,b\rangle\big) \\
&= \frac{1}{2}\big(|00\rangle + |01\rangle + |10\rangle + |11\rangle\big) - |a,b\rangle \\
&= |+\rangle \otimes |+\rangle - |a\rangle \otimes |b\rangle\,,
\end{aligned}$$

where the $-2\,|a,b\rangle$ in the first line replaces one of the plus signs by a minus sign. After applying a Hadamard transform, they arrive at the following state:

$$\begin{aligned}
&|0\rangle \otimes |0\rangle - H\,|a\rangle \otimes H\,|b\rangle \\
&= |00\rangle - \frac{1}{2}\Big(|0\rangle + (-1)^a\,|1\rangle\Big) \otimes \Big(|0\rangle + (-1)^b\,|1\rangle\Big) \\
&= -\frac{1}{2}\Big(-|00\rangle + (-1)^a\,|10\rangle + (-1)^b\,|01\rangle + (-1)^{a+b}\,|11\rangle\Big)
\end{aligned}$$

Now our friends are confused and not quite sure what to do. Hila has an idea: *"I don't really like the minus sign in front of $|00\rangle$. Why don't we apply a quantum operation that looks as follows?"*

$$\begin{aligned}
|00\rangle &\mapsto -|00\rangle \\
|01\rangle &\mapsto |01\rangle \\
|10\rangle &\mapsto |10\rangle \\
|11\rangle &\mapsto |11\rangle
\end{aligned} \tag{5.16}$$

Iman joins in: *"I think I know how to build this using a controlled-Z operation and a handful of NOTs..."* In no time, our friends arrive at the following state:

$$-\frac{1}{2}\Big(|00\rangle + (-1)^a\,|10\rangle + (-1)^b\,|01\rangle + (-1)^{a+b}\,|11\rangle\Big)$$

After only a brief moment, Alice realizes: *"This two-qubit state can be written as a tensor product!"*

$$-\frac{1}{2}\Big(|0\rangle + (-1)^a\,|1\rangle\Big) \otimes \Big(|0\rangle + (-1)^b\,|1\rangle\Big) = -(H \otimes H)\,|a,b\rangle\,.$$

Bob is elated: *"Aha! We only need to apply another Hadamard transform and measure both qubits..."* Can you also figure out who the winner of this year's quantum lottery will be?

Homework 5.5: Quantum lottery

1. Write the quantum operation in Eq. (5.16) using a controlled-Z operation and a few NOT operations.

   Hint: Recall that the controlled-Z operation $CZ_{1\to 2}$ acts on basis states $|x,y\rangle$ as follows: If $x = 0$ then it does nothing. If $x = 1$ then it acts as a Z on the second qubit. You learned last week how to build it in QUIRKY.

2. Build the quantum algorithm that Alice, Bob, Hila, and Iman came up with in QUIRKY, and determine the winner of this year's quantum Lottery.

The quantum algorithm that our friends just discovered is a special case of Grover's algorithm. Grovers algorithm solves the following problem: Given an oracle for a function $f\colon \{0,1\}^n \to \{0,1\}$, it finds $x_1,\ldots,x_n \in \{0,1\}$ such that $f(x_1,\ldots,x_n) = 1$. We can think of this problem as finding a lottery winner among 2^n participants or, less prosaically, finding an item that satisfies some property of interest in an unstructured database (by which we mean that the database entries are not sorted or similarly). By the same argument that we gave before, any classical algorithm will in the worst case need to look at all but one of the 2^n many entries before it is done (also in the average case one still needs to look at about half the entries). In contrast, Grover's algorithm only needs to use the oracle only a number of times that is proportional to $\sqrt{2^n}$, which grows much more slowly with n! Grover's algorithm is a very versatile tool which gives a square-root speedup for many computational problems.

5.3.1 Angle amplification

The final quantum algorithm that we will discuss is a very important subroutine that is used in many other quantum algorithms (for example, it is at the heart of Grover's algorithm for functions with more than $n = 2$ input bits).

The problem this subroutine solves might at first appear very strange. Indeed, it is a purely quantum problem that does not even have a meaningful classical formulation. Nevertheless, it appears naturally in various other algorithms, which makes the subroutine very useful. This highlights how different the ideas behind quantum algorithms are and that novel ways of thinking are required for inventing new quantum algorithms!

Moreover, this is an example of a problem where the oracle needs to be consulted more than once. This is different from all of the quantum algorithms we considered earlier in this quest, which used the oracle only once.

Homework 5.6: What is the angle? (challenging)

Alice and Bob are distraught. "We're so sorry for the trouble," they tell you. "We built this beautiful reflection $V(\theta)$ with angle

$$\theta = +\frac{\pi}{4k} \quad \text{or} \quad \theta = -\frac{\pi}{4k},$$

but we just cannot remember which one it was – we only remember the integer $k \geq 1$!"[a] What adds to their trouble is that the gate will self-destruct when used more than k times.

Can you help them out? Your task, if you choose to accept it, is to determine which of the two angles is the correct one by using the gate $V(\theta)$ at most k times.

Hint: Two consecutive reflections make a rotation. What do you get by combining NOT and $V(\theta)$? (You don't need the general formula in Exercise 4.6 to answer this question.)

[a] If you like, you can think of this reflection as an oracle that hides one of the two angles in a funny way.

5.4 Your quantum journey

This last problem concludes *The Quantum Quest*. Wow, it has already been five weeks – this class has been quite the journey! We really hope that you had a good time during the past weeks and that you learned a lot of interesting mathematics.

If you can't get enough of quantum computing, do not despair. By now, you are already a seasoned wizard of quantum bits and well-equipped to study a more advanced book on your own. Why don't you look and see if your local library has a copy of the book *'Quantum Computation and Quantum Information'* by Michael Nielsen and Isaac Chuang?

5.5 Exercise solutions

Solution to Exercise 5.1

The other three functions are $f = \text{NOT}$ and the two constant functions $f(0) = f(1) = 0$ and $f(0) = f(1) = 1$.

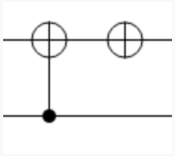
- For the NOT function:

  $$U_{\text{NOT}} |a, b\rangle = |a, b \oplus \text{NOT}(a)\rangle = |a, b \oplus a \oplus 1\rangle = |a, \text{NOT}(b \oplus a)\rangle ,$$

  which we recognize can be written as a composition of a controlled-NOT operation and a NOT operation on the second qubit, i.e.,

  $$U_{\text{NOT}} = (I \otimes \text{NOT})\, \text{CNOT}_{1\to 2}.$$

  In QUIRKY this looks as follows:



- For the all-zeros function $f(0) = f(1) = 0$:

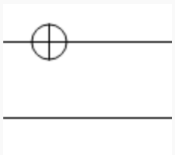
  $$U_f |a, b\rangle = |a, b\rangle ,$$

  so we do not have to do anything at all:

- For the all-ones function $f(0) = f(1) = 1$:

  $$U_f |a, b\rangle = |a, b \oplus 1\rangle = |a, \text{NOT}(b)\rangle ,$$

  so we only need to invert the second qubit:



Solution to Exercise 5.2

There are four functions: the 'identity' function $f(x) = x$, the NOT function, the all-zeros function, and the all-ones function.

- For the identity function $f(x) = x$, Eq. (5.6) reads

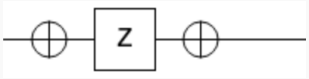
$$O_f |x\rangle = (-1)^x |x\rangle ,$$

so this is precisely the Z gate:

- For the NOT function $f(x) = \mathrm{NOT}(x)$, we want

$$O_f |x\rangle = (-1)^{\mathrm{NOT}(x)} |x\rangle = \mathrm{NOT}\, Z\, \mathrm{NOT} |x\rangle ,$$

which corresponds to the following sequence:



- For the all-zeros function with $f(0) = f(1) = 0$:

$$O_f |x\rangle = |x\rangle ,$$

so we do not have to do anything at all:

- For the all-ones function with $f(0) = f(1) = 1$:

$$O_f |x\rangle = - |x\rangle ,$$

which we can achieve by combining the first two oracles in sequence:

Indeed, the first oracle adds a minus sign if $x = 1$, while the second oracle adds a minus sign if $x = 0$, so we get a minus sign in either case:

$$\mathrm{NOT}\, Z\, \mathrm{NOT}\, Z |0\rangle = \mathrm{NOT}\, Z\, \mathrm{NOT} |0\rangle = - |0\rangle ,$$
$$\mathrm{NOT}\, Z\, \mathrm{NOT}\, Z |1\rangle = \mathrm{NOT}\, Z\, \mathrm{NOT}(- |1\rangle) = -\mathrm{NOT}\, Z\, \mathrm{NOT} |1\rangle = - |1\rangle .$$

In the second to last step, we used linearity to move the minus sign to the front.

Solution to Exercise 5.3

We evaluate Eq. (5.7) for all four functions.

- For $f(x) = x$:

$$HO_f |+\rangle = \frac{1+(-1)}{2} |0\rangle + \frac{1-(-1)}{2} |1\rangle = |1\rangle .$$

- For $f(x) = \mathrm{NOT}(x)$:

$$HO_f |+\rangle = \frac{-1+1}{2} |0\rangle + \frac{-1-1}{2} |1\rangle = -|1\rangle .$$

- For the all-zeros function:

$$HO_f |+\rangle = \frac{1+1}{2} |0\rangle + \frac{1-1}{2} |1\rangle = |0\rangle .$$

- For the all-ones function:

$$HO_f |+\rangle = \frac{(-1)+(-1)}{2} |0\rangle + \frac{(-1)-(-1)}{2} |1\rangle = -|0\rangle .$$

Solution to Exercise 5.4

1. Here $\boxed{???}$ stands for $x_1 y_1$.
2. For $n = 2$,

$$\begin{aligned}
(H \otimes H) |x_1, x_2\rangle &= (H |x_1\rangle) \otimes (H |x_2\rangle) \\
&= \left(\frac{1}{\sqrt{2}} \sum_{y_1 \in \{0,1\}} (-1)^{x_1 y_1} |y_1\rangle \right) \otimes \left(\frac{1}{\sqrt{2}} \sum_{y_2 \in \{0,1\}} (-1)^{x_2 y_2} |y_2\rangle \right) \\
&= \frac{1}{2} \sum_{y_1, y_2 \in \{0,1\}} (-1)^{x_1 y_1} (-1)^{x_2 y_2} |y_1\rangle \otimes |y_2\rangle \\
&= \frac{1}{2} \sum_{y_1, y_2 \in \{0,1\}} (-1)^{x_1 y_1 + x_2 y_2} |y_1, y_2\rangle ,
\end{aligned}$$

   so $\boxed{???}$ stands for $x_1 y_1 + x_2 y_2$.

Solution to Exercise 5.5

1. Here are the four functions and their sign oracles:

   - For $[a_1, a_2] = [0,0]$, $O_f |x_1, x_2\rangle = |x_1, x_2\rangle$, so the sign oracle does nothing at all.
   - For $[a_1, a_2] = [0,1]$, $O_f |x_1, x_2\rangle = (-1)^{x_2} |x_1, x_2\rangle$, which is the same as $I \otimes Z$.
   - For $[a_1, a_2] = [1,0]$, $O_f |x_1, x_2\rangle = (-1)^{x_1} |x_1, x_2\rangle$, which is the same as $Z \otimes I$.
   - For $[a_1, a_2] = [1,1]$, $O_f |x_1, x_2\rangle = (-1)^{x_1+x_2} |x_1, x_2\rangle = (-1)^{x_1}(-1)^{x_2} |x_1, x_2\rangle$, which is the same as $Z \otimes Z$.

   It is clear what these four operations look like in QUIRKY.

2. The general pattern is now clear. For an arbitrary function of the form (5.14):

$$O_f |x_1, \ldots, x_n\rangle = (-1)^{x_1 a_1 + \ldots + x_n a_n} |x_1, \ldots, x_n\rangle = (Z^{a_1} \otimes \cdots \otimes Z^{a_n}) |x_1, \ldots, x_n\rangle ,$$

   where we write $Z^1 = Z$ and $Z^0 = I$. In other words, we apply a Z gate on the j-th qubit if and only if $a_j = 1$.

Solution to Exercise 5.6

In step 1 we start with:

$$|00\rangle$$

In step 2 we apply $H \otimes H$ and obtain:

$$\begin{aligned} &\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \\ =&\frac{1}{2} (|00\rangle + |01\rangle + |10\rangle + |11\rangle) \end{aligned}$$

In step 3 we apply the sign oracle O_f for $f(x_1, x_2) = x_2$:

$$\frac{1}{2} (|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$

In step 4 we apply again $H \otimes H$, resulting in:

$$\begin{aligned} &\frac{1}{2}\Big((H \otimes H) |00\rangle - (H \otimes H) |01\rangle + (H \otimes H) |10\rangle - (H \otimes H) |11\rangle\Big) \\ =&\frac{1}{4}\Big((|00\rangle + |01\rangle + |10\rangle + |11\rangle) - (|00\rangle - |01\rangle + |10\rangle - |11\rangle) \\ &+ (|00\rangle + |01\rangle - |10\rangle - |11\rangle) - (|00\rangle - |01\rangle - |10\rangle + |11\rangle)\Big) \\ =&|01\rangle . \end{aligned}$$

In step 5 we measure both qubits and always obtain the result $[0,1]$.