I Fundamental concepts

1 Introduction and overview

Science offers the boldest metaphysics of the age. It is a thoroughly human construct, driven by the faith that if we dream, press to discover, explain, and dream again, thereby plunging repeatedly into new terrain, the world will somehow come clearer and we will grasp the true strangeness of the universe. And the strangeness will all prove to be connected, and make sense.
– Edward O. Wilson

Information is physical.
– Rolf Landauer

What are the fundamental concepts of quantum computation and quantum information? How did these concepts develop? To what uses may they be put? How will they be presented in this book? The purpose of this introductory chapter is to answer these questions by developing in broad brushstrokes a picture of the field of quantum computation and quantum information. The intent is to communicate a basic understanding of the central concepts of the field, perspective on how they have been developed, and to help you decide how to approach the rest of the book.

Our story begins in Section 1.1 with an account of the historical context in which quantum computation and quantum information has developed. Each remaining section in the chapter gives a brief introduction to one or more fundamental concepts from the field: quantum bits (Section 1.2), quantum computers, quantum gates and quantum circuits (Section 1.3), quantum algorithms (Section 1.4), experimental quantum information processing (Section 1.5), and quantum information and communication (Section 1.6).

Along the way, illustrative and easily accessible developments such as quantum teleportation and some simple quantum algorithms are given, using the basic mathematics taught in this chapter. The presentation is self-contained, and designed to be accessible even without a background in computer science or physics. As we move along, we give pointers to more in-depth discussions in later chapters, where references and suggestions for further reading may also be found.

If as you read you're finding the going rough, skip on to a spot where you feel more comfortable. At points we haven't been able to avoid using a little technical lingo which won't be completely explained until later in the book. Simply accept it for now, and come back later when you understand all the terminology in more detail. The emphasis in this first chapter is on the big picture, with the details to be filled in later.

1.1 Global perspectives

Quantum computation and quantum information is the study of the information processing tasks that can be accomplished using quantum mechanical systems. Sounds pretty

simple and obvious, doesn't it? Like many simple but profound ideas it was a long time before anybody thought of doing information processing using quantum mechanical systems. To see why this is the case, we must go back in time and look in turn at each of the fields which have contributed fundamental ideas to quantum computation and quantum information – quantum mechanics, computer science, information theory, and cryptography. As we take our short historical tour of these fields, think of yourself first as a physicist, then as a computer scientist, then as an information theorist, and finally as a cryptographer, in order to get some feel for the disparate perspectives which have come together in quantum computation and quantum information.

1.1.1 History of quantum computation and quantum information

Our story begins at the turn of the twentieth century when an unheralded revolution was underway in science. A series of crises had arisen in physics. The problem was that the theories of physics at that time (now dubbed *classical physics*) were predicting absurdities such as the existence of an 'ultraviolet catastrophe' involving infinite energies, or electrons spiraling inexorably into the atomic nucleus. At first such problems were resolved with the addition of *ad hoc* hypotheses to classical physics, but as a better understanding of atoms and radiation was gained these attempted explanations became more and more convoluted. The crisis came to a head in the early 1920s after a quarter century of turmoil, and resulted in the creation of the modern theory of *quantum mechanics*. Quantum mechanics has been an indispensable part of science ever since, and has been applied with enormous success to everything under and inside the Sun, including the structure of the atom, nuclear fusion in stars, superconductors, the structure of DNA, and the elementary particles of Nature.

What is quantum mechanics? Quantum mechanics is a mathematical framework or set of rules for the construction of physical theories. For example, there is a physical theory known as *quantum electrodynamics* which describes with fantastic accuracy the interaction of atoms and light. Quantum electrodynamics is built up within the framework of quantum mechanics, but it contains specific rules not determined by quantum mechanics. The relationship of quantum mechanics to specific physical theories like quantum electrodynamics is rather like the relationship of a computer's operating system to specific applications software – the operating system sets certain basic parameters and modes of operation, but leaves open how specific tasks are accomplished by the applications.

The rules of quantum mechanics are simple but even experts find them counter-intuitive, and the earliest antecedents of quantum computation and quantum information may be found in the long-standing desire of physicists to better understand quantum mechanics. The best known critic of quantum mechanics, Albert Einstein, went to his grave unreconciled with the theory he helped invent. Generations of physicists since have wrestled with quantum mechanics in an effort to make its predictions more palatable. One of the goals of quantum computation and quantum information is to develop tools which sharpen our intuition about quantum mechanics, and make its predictions more transparent to human minds.

For example, in the early 1980s, interest arose in whether it might be possible to use quantum effects to signal faster than light – a big no-no according to Einstein's theory of relativity. The resolution of this problem turns out to hinge on whether it is possible to *clone* an unknown quantum state, that is, construct a copy of a quantum state. If cloning were possible, then it would be possible to signal faster than light using quantum effects.

However, cloning – so easy to accomplish with classical information (consider the words in front of you, and where they came from!) – turns out not to be possible in general in quantum mechanics. This *no-cloning theorem*, discovered in the early 1980s, is one of the earliest results of quantum computation and quantum information. Many refinements of the no-cloning theorem have since been developed, and we now have conceptual tools which allow us to understand how well a (necessarily imperfect) quantum cloning device might work. These tools, in turn, have been applied to understand other aspects of quantum mechanics.

A related historical strand contributing to the development of quantum computation and quantum information is the interest, dating to the 1970s, of obtaining *complete control over single quantum systems*. Applications of quantum mechanics prior to the 1970s typically involved a gross level of control over a bulk sample containing an enormous number of quantum mechanical systems, none of them directly accessible. For example, superconductivity has a superb quantum mechanical explanation. However, because a superconductor involves a huge (compared to the atomic scale) sample of conducting metal, we can only probe a few aspects of its quantum mechanical nature, with the individual quantum systems constituting the superconductor remaining inaccessible. Systems such as particle accelerators do allow limited access to individual quantum systems, but again provide little control over the constituent systems.

Since the 1970s many techniques for controlling single quantum systems have been developed. For example, methods have been developed for trapping a single atom in an 'atom trap', isolating it from the rest of the world and allowing us to probe many different aspects of its behavior with incredible precision. The scanning tunneling microscope has been used to move single atoms around, creating designer arrays of atoms at will. Electronic devices whose operation involves the transfer of only single electrons have been demonstrated.

Why all this effort to attain complete control over single quantum systems? Setting aside the many technological reasons and concentrating on pure science, the principal answer is that researchers have done this on a hunch. Often the most profound insights in science come when we develop a method for probing a new regime of Nature. For example, the invention of radio astronomy in the 1930s and 1940s led to a spectacular sequence of discoveries, including the galactic core of the Milky Way galaxy, pulsars, and quasars. Low temperature physics has achieved its amazing successes by finding ways to lower the temperatures of different systems. In a similar way, by obtaining complete control over single quantum systems, we are exploring untouched regimes of Nature in the hope of discovering new and unexpected phenomena. We are just now taking our first steps along these lines, and already a few interesting surprises have been discovered in this regime. What else shall we discover as we obtain more complete control over single quantum systems, and extend it to more complex systems?

Quantum computation and quantum information fit naturally into this program. They provide a useful series of challenges at varied levels of difficulty for people devising methods to better manipulate single quantum systems, and stimulate the development of new experimental techniques and provide guidance as to the most interesting directions in which to take experiment. Conversely, the ability to control single quantum systems is essential if we are to harness the power of quantum mechanics for applications to quantum computation and quantum information.

Despite this intense interest, efforts to build quantum information processing systems

have resulted in modest success to date. Small quantum computers, capable of doing dozens of operations on a few quantum bits (or *qubits*) represent the state of the art in quantum computation. Experimental prototypes for doing *quantum cryptography* – a way of communicating in secret across long distances – have been demonstrated, and are even at the level where they may be useful for some real-world applications. However, it remains a great challenge to physicists and engineers of the future to develop techniques for making large-scale quantum information processing a reality.

Let us turn our attention from quantum mechanics to another of the great intellectual triumphs of the twentieth century, computer science. The origins of computer science are lost in the depths of history. For example, cuneiform tablets indicate that by the time of Hammurabi (circa 1750 B.C.) the Babylonians had developed some fairly sophisticated algorithmic ideas, and it is likely that many of those ideas date to even earlier times.

The modern incarnation of computer science was announced by the great mathematician Alan Turing in a remarkable 1936 paper. Turing developed in detail an abstract notion of what we would now call a programmable computer, a model for computation now known as the *Turing machine*, in his honor. Turing showed that there is a *Universal Turing Machine* that can be used to simulate any other Turing machine. Furthermore, he claimed that the Universal Turing Machine *completely captures* what it means to perform a task by algorithmic means. That is, if an algorithm can be performed on *any* piece of hardware (say, a modern personal computer), then there is an equivalent algorithm for a Universal Turing Machine which performs exactly the same task as the algorithm running on the personal computer. This assertion, known as the *Church–Turing thesis* in honor of Turing and another pioneer of computer science, Alonzo Church, asserts the equivalence between the physical concept of what class of algorithms can be performed on *some physical device* with the rigorous mathematical concept of a Universal Turing Machine. The broad acceptance of this thesis laid the foundation for the development of a rich theory of computer science.

Not long after Turing's paper, the first computers constructed from electronic components were developed. John von Neumann developed a simple theoretical model for how to put together in a practical fashion all the components necessary for a computer to be fully as capable as a Universal Turing Machine. Hardware development truly took off, though, in 1947, when John Bardeen, Walter Brattain, and Will Shockley developed the transistor. Computer hardware has grown in power at an amazing pace ever since, so much so that the growth was codified by Gordon Moore in 1965 in what has come to be known as *Moore's law*, which states that computer power will double for constant cost roughly once every two years.

Amazingly enough, Moore's law has approximately held true in the decades since the 1960s. Nevertheless, most observers expect that this dream run will end some time during the first two decades of the twenty-first century. Conventional approaches to the fabrication of computer technology are beginning to run up against fundamental difficulties of size. Quantum effects are beginning to interfere in the functioning of electronic devices as they are made smaller and smaller.

One possible solution to the problem posed by the eventual failure of Moore's law is to move to a different computing paradigm. One such paradigm is provided by the theory of quantum computation, which is based on the idea of using quantum mechanics to perform computations, instead of classical physics. It turns out that while an ordinary computer can be used to simulate a quantum computer, it appears to be impossible to

perform the simulation in an *efficient* fashion. Thus quantum computers offer an essential speed advantage over classical computers. This speed advantage is so significant that many researchers believe that *no* conceivable amount of progress in classical computation would be able to overcome the gap between the power of a classical computer and the power of a quantum computer.

What do we mean by 'efficient' versus 'inefficient' simulations of a quantum computer? Many of the key notions needed to answer this question were actually invented before the notion of a quantum computer had even arisen. In particular, the idea of *efficient* and *inefficient* algorithms was made mathematically precise by the field of *computational complexity*. Roughly speaking, an efficient algorithm is one which runs in time polynomial in the size of the problem solved. In contrast, an inefficient algorithm requires super-polynomial (typically exponential) time. What was noticed in the late 1960s and early 1970s was that it seemed as though the Turing machine model of computation was at least as powerful as any other model of computation, in the sense that a problem which could be solved efficiently in some model of computation could also be solved efficiently in the Turing machine model, by using the Turing machine to simulate the other model of computation. This observation was codified into a strengthened version of the Church–Turing thesis:

Any algorithmic process can be simulated efficiently using a Turing machine.

The key strengthening in the strong Church–Turing thesis is the word *efficiently*. If the strong Church–Turing thesis is correct, then it implies that no matter what type of machine we use to perform our algorithms, that machine can be simulated efficiently using a standard Turing machine. This is an important strengthening, as it implies that for the purposes of analyzing whether a given computational task can be accomplished efficiently, we may restrict ourselves to the analysis of the Turing machine model of computation.

One class of challenges to the strong Church–Turing thesis comes from the field of *analog computation*. In the years since Turing, many different teams of researchers have noticed that certain types of analog computers can efficiently solve problems believed to have no efficient solution on a Turing machine. At first glance these analog computers appear to violate the strong form of the Church–Turing thesis. Unfortunately for analog computation, it turns out that when realistic assumptions about the presence of noise in analog computers are made, their power disappears in all known instances; they cannot efficiently solve problems which are not efficiently solvable on a Turing machine. This lesson – that the effects of realistic noise must be taken into account in evaluating the efficiency of a computational model – was one of the great early challenges of quantum computation and quantum information, a challenge successfully met by the development of a theory of *quantum error-correcting codes* and *fault-tolerant quantum computation*. Thus, unlike analog computation, quantum computation can in principle tolerate a finite amount of noise and still retain its computational advantages.

The first major challenge to the strong Church–Turing thesis arose in the mid 1970s, when Robert Solovay and Volker Strassen showed that it is possible to test whether an integer is prime or composite using a *randomized algorithm*. That is, the Solovay–Strassen test for primality used randomness as an *essential* part of the algorithm. The algorithm did not determine whether a given integer was prime or composite with certainty. Instead, the algorithm could determine that a number was *probably* prime or else composite *with*

certainty. By repeating the Solovay–Strassen test a few times it is possible to determine with near certainty whether a number is prime or composite. The Solovay-Strassen test was of especial significance at the time it was proposed as no deterministic test for primality was then known, nor is one known at the time of this writing. Thus, it seemed as though computers with access to a random number generator would be able to efficiently perform computational tasks with no efficient solution on a conventional deterministic Turing machine. This discovery inspired a search for other randomized algorithms which has paid off handsomely, with the field blossoming into a thriving area of research.

Randomized algorithms pose a challenge to the strong Church–Turing thesis, suggesting that there are efficiently soluble problems which, nevertheless, cannot be efficiently solved on a deterministic Turing machine. This challenge appears to be easily resolved by a simple modification of the strong Church–Turing thesis:

Any algorithmic process can be simulated efficiently using a probabilistic Turing machine.

This *ad hoc* modification of the strong Church–Turing thesis should leave you feeling rather queasy. Might it not turn out at some later date that yet another model of computation allows one to efficiently solve problems that are not efficiently soluble within Turing's model of computation? Is there any way we can find a single model of computation which is guaranteed to be able to efficiently simulate any other model of computation?

Motivated by this question, in 1985 David Deutsch asked whether the laws of physics could be use to *derive* an even stronger version of the Church–Turing thesis. Instead of adopting *ad hoc* hypotheses, Deutsch looked to physical theory to provide a foundation for the Church–Turing thesis that would be as secure as the status of that physical theory. In particular, Deutsch attempted to define a computational device that would be capable of efficiently simulating an *arbitrary* physical system. Because the laws of physics are ultimately quantum mechanical, Deutsch was naturally led to consider computing devices based upon the principles of quantum mechanics. These devices, quantum analogues of the machines defined forty-nine years earlier by Turing, led ultimately to the modern conception of a quantum computer used in this book.

At the time of writing it is not clear whether Deutsch's notion of a Universal Quantum Computer is sufficient to efficiently simulate an arbitrary physical system. Proving or refuting this conjecture is one of the great open problems of the field of quantum computation and quantum information. It is possible, for example, that some effect of quantum field theory or an even more esoteric effect based in string theory, quantum gravity or some other physical theory may take us beyond Deutsch's Universal Quantum Computer, giving us a still more powerful model for computation. At this stage, we simply don't know.

What Deutsch's model of a quantum computer did enable was a challenge to the strong form of the Church–Turing thesis. Deutsch asked whether it is possible for a quantum computer to efficiently solve computational problems which have no efficient solution on a classical computer, even a probabilistic Turing machine. He then constructed a simple example suggesting that, indeed, quantum computers might have computational powers exceeding those of classical computers.

This remarkable first step taken by Deutsch was improved in the subsequent decade by many people, culminating in Peter Shor's 1994 demonstration that two enormously important problems – the problem of finding the prime factors of an integer, and the so-

called 'discrete logarithm' problem – could be solved efficiently on a quantum computer. This attracted widespread interest because these two problems were and still are widely believed to have no efficient solution on a classical computer. Shor's results are a powerful indication that quantum computers are more powerful than Turing machines, even probabilistic Turing machines. Further evidence for the power of quantum computers came in 1995 when Lov Grover showed that another important problem – the problem of conducting a search through some unstructured search space – could also be sped up on a quantum computer. While Grover's algorithm did not provide as spectacular a speed-up as Shor's algorithms, the widespread applicability of search-based methodologies has excited considerable interest in Grover's algorithm.

At about the same time as Shor's and Grover's algorithms were discovered, many people were developing an idea Richard Feynman had suggested in 1982. Feynman had pointed out that there seemed to be essential difficulties in simulating quantum mechanical systems on classical computers, and suggested that building computers based on the principles of quantum mechanics would allow us to avoid those difficulties. In the 1990s several teams of researchers began fleshing this idea out, showing that it is indeed possible to use quantum computers to efficiently simulate systems that have no known efficient simulation on a classical computer. It is likely that one of the major applications of quantum computers in the future will be performing simulations of quantum mechanical systems too difficult to simulate on a classical computer, a problem with profound scientific and technological implications.

What other problems can quantum computers solve more quickly than classical computers? The short answer is that we don't know. Coming up with good quantum algorithms seems to be *hard*. A pessimist might think that's because there's nothing quantum computers are good for other than the applications already discovered! We take a different view. Algorithm design for quantum computers is hard because designers face two difficult problems not faced in the construction of algorithms for classical computers. First, our human intuition is rooted in the classical world. If we use that intuition as an aid to the construction of algorithms, then the algorithmic ideas we come up with will be classical ideas. To design good quantum algorithms one must 'turn off' one's classical intuition for at least part of the design process, using truly quantum effects to achieve the desired algorithmic end. Second, to be truly interesting it is not enough to design an algorithm that is merely quantum mechanical. The algorithm must be *better* than any existing classical algorithm! Thus, it is possible that one may find an algorithm which makes use of truly quantum aspects of quantum mechanics, that is nevertheless not of widespread interest because classical algorithms with comparable performance characteristics exist. The combination of these two problems makes the construction of new quantum algorithms a challenging problem for the future.

Even more broadly, we can ask if there are any generalizations we can make about the power of quantum computers versus classical computers. What is it that makes quantum computers more powerful than classical computers – assuming that this is indeed the case? What class of problems can be solved efficiently on a quantum computer, and how does that class compare to the class of problems that can be solved efficiently on a classical computer? One of the most exciting things about quantum computation and quantum information is how *little* is known about the answers to these questions! It is a great challenge for the future to understand these questions better.

Having come up to the frontier of quantum computation, let's switch to the history

of another strand of thought contributing to quantum computation and quantum information: information theory. At the same time computer science was exploding in the 1940s, another revolution was taking place in our understanding of *communication*. In 1948 Claude Shannon published a remarkable pair of papers laying the foundations for the modern theory of information and communication.

Perhaps the key step taken by Shannon was *to mathematically define the concept of information*. In many mathematical sciences there is considerable flexibility in the choice of fundamental definitions. Try thinking naively for a few minutes about the following question: how would you go about mathematically defining the notion of an information source? Several *different* answers to this problem have found widespread use; however, the definition Shannon came up with seems to be far and away the most fruitful in terms of increased understanding, leading to a plethora of deep results and a theory with a rich structure which seems to accurately reflect many (though not all) real-world communications problems.

Shannon was interested in two key questions related to the communication of information over a communications channel. First, what resources are required to send information over a communications channel? For example, telephone companies need to know how much information they can reliably transmit over a given telephone cable. Second, can information be transmitted in such a way that it is protected against noise in the communications channel?

Shannon answered these two questions by proving the two fundamental theorems of information theory. The first, Shannon's *noiseless channel coding theorem*, quantifies the physical resources required to store the output from an information source. Shannon's second fundamental theorem, the *noisy channel coding theorem*, quantifies how much information it is possible to reliably transmit through a noisy communications channel. To achieve reliable transmission in the presence of noise, Shannon showed that *error-correcting codes* could be used to protect the information being sent. Shannon's noisy channel coding theorem gives an upper limit on the protection afforded by error-correcting codes. Unfortunately, Shannon's theorem does not explicitly give a practically useful set of error-correcting codes to achieve that limit. From the time of Shannon's papers until today, researchers have constructed more and better classes of error-correcting codes in their attempts to come closer to the limit set by Shannon's theorem. A sophisticated theory of error-correcting codes now exists offering the user a plethora of choices in their quest to design a good error-correcting code. Such codes are used in a multitude of places including, for example, compact disc players, computer modems, and satellite communications systems.

Quantum information theory has followed with similar developments. In 1995, Ben Schumacher provided an analogue to Shannon's noiseless coding theorem, and in the process defined the 'quantum bit' or 'qubit' as a tangible physical resource. However, no analogue to Shannon's noisy channel coding theorem is yet known for quantum information. Nevertheless, in analogy to their classical counterparts, a theory of quantum error-correction has been developed which, as already mentioned, allows quantum computers to compute effectively in the presence of noise, and also allows communication over noisy *quantum* channels to take place reliably.

Indeed, classical ideas of error-correction have proved to be enormously important in developing and understanding quantum error-correcting codes. In 1996, two groups working independently, Robert Calderbank and Peter Shor, and Andrew Steane, discov-

ered an important class of quantum codes now known as CSS codes after their initials. This work has since been subsumed by the stabilizer codes, independently discovered by Robert Calderbank, Eric Rains, Peter Shor and Neil Sloane, and by Daniel Gottesman. By building upon the basic ideas of classical linear coding theory, these discoveries greatly facilitated a rapid understanding of quantum error-correcting codes and their application to quantum computation and quantum information.

The theory of quantum error-correcting codes was developed to protect quantum states against noise. What about transmitting ordinary *classical* information using a quantum channel? How efficiently can this be done? A few surprises have been discovered in this arena. In 1992 Charles Bennett and Stephen Wiesner explained how to transmit *two* classical bits of information, while only transmitting *one* quantum bit from sender to receiver, a result dubbed *superdense coding*.

Even more interesting are the results in *distributed quantum computation*. Imagine you have two computers networked, trying to solve a particular problem. How much communication is required to solve the problem? Recently it has been shown that quantum computers can require *exponentially less* communication to solve certain problems than would be required if the networked computers were classical! Unfortunately, as yet these problems are not especially important in a practical setting, and suffer from some undesirable technical restrictions. A major challenge for the future of quantum computation and quantum information is to find problems of real-world importance for which distributed quantum computation offers a substantial advantage over distributed classical computation.

Let's return to information theory proper. The study of information theory begins with the properties of a single communications channel. In applications we often do not deal with a single communications channel, but rather with networks of many channels. The subject of *networked information theory* deals with the information carrying properties of such networks of communications channels, and has been developed into a rich and intricate subject.

By contrast, the study of networked quantum information theory is very much in its infancy. Even for very basic questions we know little about the information carrying abilities of networks of quantum channels. Several rather striking preliminary results have been found in the past few years; however, no unifying theory of networked information theory exists for quantum channels. One example of networked quantum information theory should suffice to convince you of the value such a general theory would have. Imagine that we are attempting to send quantum information from Alice to Bob through a noisy quantum channel. If that channel has zero capacity for quantum information, then it is impossible to reliably send *any* information from Alice to Bob. Imagine instead that we consider two copies of the channel, operating in synchrony. Intuitively it is clear (and can be rigorously justified) that such a channel also has zero capacity to send quantum information. However, if we instead *reverse* the direction of one of the channels, as illustrated in Figure 1.1, it turns out that sometimes we can obtain a non-zero capacity for the transmission of information from Alice to Bob! Counter-intuitive properties like this illustrate the strange nature of quantum information. Better understanding the information carrying properties of networks of quantum channels is a major open problem of quantum computation and quantum information.

Let's switch fields one last time, moving to the venerable old art and science of *cryptography*. Broadly speaking, cryptography is the problem of doing *communication* or

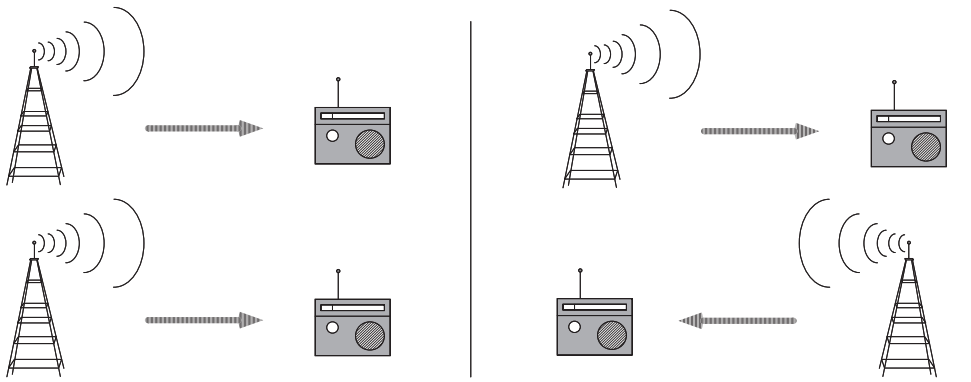


Figure 1.1. Classically, if we have two very noisy channels of zero capacity running side by side, then the combined channel has zero capacity to send information. Not surprisingly, if we reverse the direction of one of the channels, we still have zero capacity to send information. Quantum mechanically, reversing one of the zero capacity channels can actually allow us to send information!

computation involving two or more parties *who may not trust one another*. The best known cryptographic problem is the transmission of secret messages. Suppose two parties wish to communicate in secret. For example, you may wish to give your credit card number to a merchant in exchange for goods, hopefully without any malevolent third party intercepting your credit card number. The way this is done is to use a *cryptographic protocol*. We'll describe in detail how cryptographic protocols work later in the book, but for now it will suffice to make a few simple distinctions. The most important distinction is between *private key cryptosystems* and *public key cryptosystems*.

The way a private key cryptosystem works is that two parties, 'Alice' and 'Bob', wish to communicate by sharing a *private key*, which only they know. The exact form of the key doesn't matter at this point – think of a string of zeroes and ones. The point is that this key is used by Alice to *encrypt* the information she wishes to send to Bob. After Alice encrypts she sends the encrypted information to Bob, who must now recover the original information. Exactly how Alice encrypts the message *depends upon the private key*, so that to recover the original message Bob needs to know the private key, in order to undo the transformation Alice applied.

Unfortunately, private key cryptosystems have some severe problems in many contexts. The most basic problem is how to distribute the keys? In many ways, the key distribution problem is just as difficult as the original problem of communicating in private – a malevolent third party may be eavesdropping on the key distribution, and then use the intercepted key to decrypt some of the message transmission.

One of the earliest discoveries in quantum computation and quantum information was that quantum mechanics can be used to do key distribution in such a way that Alice and Bob's security can not be compromised. This procedure is known as *quantum cryptography* or *quantum key distribution*. The basic idea is to exploit the quantum mechanical principle that observation in general disturbs the system being observed. Thus, if there is an eavesdropper listening in as Alice and Bob attempt to transmit their key, the presence of the eavesdropper will be visible as a disturbance of the communications channel Alice and Bob are using to establish the key. Alice and Bob can then throw out the key bits established while the eavesdropper was listening in, and start over. The first quantum cryptographic ideas were proposed by Stephen Wiesner in the late 1960s, but unfortu-

nately were not accepted for publication! In 1984 Charles Bennett and Gilles Brassard, building on Wiesner's earlier work, proposed a protocol using quantum mechanics to distribute keys between Alice and Bob, without any possibility of a compromise. Since then numerous quantum cryptographic protocols have been proposed, and experimental prototypes developed. At the time of this writing, the experimental prototypes are nearing the stage where they may be useful in limited-scale real-world applications.

The second major type of cryptosystem is the *public key cryptosystem*. Public key cryptosystems don't rely on Alice and Bob sharing a secret key in advance. Instead, Bob simply publishes a 'public key', *which is made available to the general public*. Alice can make use of this public key to encrypt a message which she sends to Bob. What is interesting is that a third party *cannot* use Bob's public key to decrypt the message! Strictly speaking, we shouldn't say *cannot*. Rather, the encryption transformation is chosen in a very clever and non-trivial way so that it is *extremely difficult* (though not impossible) to invert, given only knowledge of the public key. To make inversion easy, Bob has a *secret key* matched to his public key, which together enable him to *easily* perform the decryption. This secret key is not known to anybody other than Bob, who can therefore be confident that only he can read the contents of Alice's transmission, to the extent that it is unlikely that anybody else has the computational power to invert the encryption, given only the public key. Public key cryptosystems solve the key distribution problem by making it unnecessary for Alice and Bob to share a private key before communicating.

Rather remarkably, public key cryptography did not achieve widespread use until the mid-1970s, when it was proposed independently by Whitfield Diffie and Martin Hellman, and by Ralph Merkle, revolutionizing the field of cryptography. A little later, Ronald Rivest, Adi Shamir, and Leonard Adleman developed the *RSA cryptosystem*, which at the time of writing is the most widely deployed public key cryptosystem, believed to offer a fine balance of security and practical usability. In 1997 it was disclosed that these ideas – public key cryptography, the Diffie–Hellman and RSA cryptosystems – were actually invented in the late 1960s and early 1970s by researchers working at the British intelligence agency GCHQ.

The key to the security of public key cryptosystems is that it should be difficult to invert the encryption stage if only the public key is available. For example, it turns out that inverting the encryption stage of RSA is a problem closely related to factoring. Much of the presumed security of RSA comes from the belief that factoring is a problem hard to solve on a classical computer. However, Shor's fast algorithm for factoring on a quantum computer could be used to break RSA! Similarly, there are other public key cryptosystems which can be broken if a fast algorithm for solving the discrete logarithm problem – like Shor's quantum algorithm for discrete logarithm – were known. This practical application of quantum computers to the breaking of cryptographic codes has excited much of the interest in quantum computation and quantum information.

We have been looking at the historical antecedents for quantum computation and quantum information. Of course, as the field has grown and matured, it has sprouted its own subfields of research, whose antecedents lie mainly within quantum computation and quantum information.

Perhaps the most striking of these is the study of *quantum entanglement*. Entanglement is a uniquely quantum mechanical *resource* that plays a key role in many of the most interesting applications of quantum computation and quantum information; entanglement is iron to the classical world's bronze age. In recent years there has been a

tremendous effort trying to better understand the properties of entanglement considered as a fundamental resource of Nature, of comparable importance to energy, information, entropy, or any other fundamental resource. Although there is as yet no complete theory of entanglement, some progress has been made in understanding this strange property of quantum mechanics. It is hoped by many researchers that further study of the properties of entanglement will yield insights that facilitate the development of new applications in quantum computation and quantum information.

1.1.2 Future directions

We've looked at some of the history and present status of quantum computation and quantum information. What of the future? What can quantum computation and quantum information offer to science, to technology, and to humanity? What benefits does quantum computation and quantum information confer upon its parent fields of computer science, information theory, and physics? What are the key open problems of quantum computation and quantum information? We will make a few very brief remarks about these overarching questions before moving onto more detailed investigations.

Quantum computation and quantum information has taught us to *think physically about computation*, and we have discovered that this approach yields many new and exciting capabilities for information processing and communication. Computer scientists and information theorists have been gifted with a new and rich paradigm for exploration. Indeed, in the broadest terms we have learned that *any physical theory*, not just quantum mechanics, may be used as the basis for a theory of information processing and communication. The fruits of these explorations may one day result in information processing devices with capabilities far beyond today's computing and communications systems, with concomitant benefits and drawbacks for society as a whole.

Quantum computation and quantum information certainly offer challenges aplenty to physicists, but it is perhaps a little subtle what quantum computation and quantum information offers to physics in the long term. We believe that just as we have learned to think physically about computation, we can also learn to *think computationally about physics*. Whereas physics has traditionally been a discipline focused on understanding 'elementary' objects and simple systems, many interesting aspects of Nature arise only when things become larger and more complicated. Chemistry and engineering deal with such complexity to some extent, but most often in a rather *ad hoc* fashion. One of the messages of quantum computation and information is that new tools are available for traversing the gulf between the small and the relatively complex: computation and algorithms provide systematic means for constructing and understanding such systems. Applying ideas from these fields is already beginning to yield new insights into physics. It is our hope that this perspective will blossom in years to come into a fruitful way of understanding all aspects of physics.

We've briefly examined some of the key motivations and ideas underlying quantum computation and quantum information. Over the remaining sections of this chapter we give a more technical but still accessible introduction to these motivations and ideas, with the hope of giving you a bird's-eye view of the field as it is presently poised.

1.2 Quantum bits

The *bit* is the fundamental concept of classical computation and classical information. Quantum computation and quantum information are built upon an analogous concept, the *quantum bit*, or *qubit* for short. In this section we introduce the properties of single and multiple qubits, comparing and contrasting their properties to those of classical bits.

What is a qubit? We're going to describe qubits as *mathematical objects* with certain specific properties. 'But hang on', you say, 'I thought qubits were physical objects.' It's true that qubits, like bits, are realized as actual physical systems, and in Section 1.5 and Chapter 7 we describe in detail how this connection between the abstract mathematical point of view and real systems is made. However, for the most part we treat qubits as abstract mathematical objects. The beauty of treating qubits as abstract entities is that it gives us the freedom to construct a general theory of quantum computation and quantum information which does not depend upon a specific system for its realization.

What then is a qubit? Just as a classical bit has a *state* – either 0 or 1 – a qubit also has a state. Two possible states for a qubit are the states $|0\rangle$ and $|1\rangle$, which as you might guess correspond to the states 0 and 1 for a classical bit. Notation like '$|\ \rangle$' is called the *Dirac notation*, and we'll be seeing it often, as it's the standard notation for states in quantum mechanics. The difference between bits and qubits is that a qubit can be in a state *other* than $|0\rangle$ or $|1\rangle$. It is also possible to form *linear combinations* of states, often called *superpositions*:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle. \tag{1.1}$$

The numbers α and β are complex numbers, although for many purposes not much is lost by thinking of them as real numbers. Put another way, the state of a qubit is a vector in a two-dimensional complex vector space. The special states $|0\rangle$ and $|1\rangle$ are known as *computational basis states*, and form an orthonormal basis for this vector space.

We can examine a bit to determine whether it is in the state 0 or 1. For example, computers do this all the time when they retrieve the contents of their memory. Rather remarkably, we cannot examine a qubit to determine its quantum state, that is, the values of α and β. Instead, quantum mechanics tells us that we can only acquire much more restricted information about the quantum state. When we measure a qubit we get either the result 0, with probability $|\alpha|^2$, or the result 1, with probability $|\beta|^2$. Naturally, $|\alpha|^2 + |\beta|^2 = 1$, since the probabilities must sum to one. Geometrically, we can interpret this as the condition that the qubit's state be normalized to length 1. Thus, in general a qubit's state is a unit vector in a two-dimensional complex vector space.

This dichotomy between the unobservable state of a qubit and the observations we can make lies at the heart of quantum computation and quantum information. In most of our abstract models of the world, there is a direct correspondence between elements of the abstraction and the real world, just as an architect's plans for a building are in correspondence with the final building. The lack of this direct correspondence in quantum mechanics makes it difficult to intuit the behavior of quantum systems; however, there is an indirect correspondence, for qubit states can be manipulated and transformed in ways which lead to measurement outcomes which depend distinctly on the different properties of the state. Thus, these quantum states have real, experimentally verifiable consequences, which we shall see are essential to the power of quantum computation and quantum information.

The ability of a qubit to be in a superposition state runs counter to our 'common sense' understanding of the physical world around us. A classical bit is like a coin: either heads or tails up. For imperfect coins, there may be intermediate states like having it balanced on an edge, but those can be disregarded in the ideal case. By contrast, a qubit can exist in a *continuum* of states between $|0\rangle$ and $|1\rangle$ – until it is observed. Let us emphasize again that when a qubit is measured, it only ever gives '0' or '1' as the measurement result – probabilistically. For example, a qubit can be in the state

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\,, \tag{1.2}$$

which, when measured, gives the result 0 fifty percent ($|1/\sqrt{2}|^2$) of the time, and the result 1 fifty percent of the time. We will return often to this state, which is sometimes denoted $|+\rangle$.

Despite this strangeness, qubits are decidedly real, their existence and behavior extensively validated by experiments (discussed in Section 1.5 and Chapter 7), and many different physical systems can be used to realize qubits. To get a concrete feel for how a qubit can be realized it may be helpful to list some of the ways this realization may occur: as the two different polarizations of a photon; as the alignment of a nuclear spin in a uniform magnetic field; as two states of an electron orbiting a single atom such as shown in Figure 1.2. In the atom model, the electron can exist in either the so-called 'ground' or 'excited' states, which we'll call $|0\rangle$ and $|1\rangle$, respectively. By shining light on the atom, with appropriate energy and for an appropriate length of time, it is possible to move the electron from the $|0\rangle$ state to the $|1\rangle$ state and vice versa. But more interestingly, by reducing the time we shine the light, an electron initially in the state $|0\rangle$ can be moved 'halfway' between $|0\rangle$ and $|1\rangle$, into the $|+\rangle$ state.

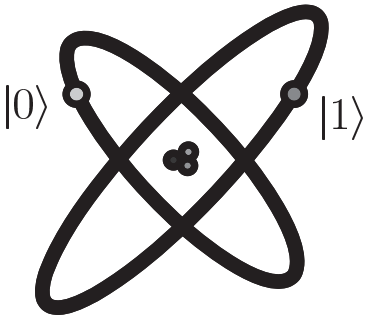


Figure 1.2. Qubit represented by two electronic levels in an atom.

Naturally, a great deal of attention has been given to the 'meaning' or 'interpretation' that might be attached to superposition states, and of the inherently probabilistic nature of observations on quantum systems. However, by and large, we shall not concern ourselves with such discussions in this book. Instead, our intent will be to develop mathematical and conceptual pictures which are predictive.

One picture useful in thinking about qubits is the following geometric representation.

Because $|\alpha|^2 + |\beta|^2 = 1$, we may rewrite Equation (1.1) as

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right), \tag{1.3}$$

where θ, φ and γ are real numbers. In Chapter 2 we will see that we can *ignore* the factor of $e^{i\gamma}$ out the front, because it has *no observable effects*, and for that reason we can effectively write

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle. \tag{1.4}$$

The numbers θ and φ define a point on the unit three-dimensional sphere, as shown in Figure 1.3. This sphere is often called the *Bloch sphere*; it provides a useful means of visualizing the state of a single qubit, and often serves as an excellent testbed for ideas about quantum computation and quantum information. Many of the operations on single qubits which we describe later in this chapter are neatly described within the Bloch sphere picture. However, it must be kept in mind that this intuition is limited because there is no simple generalization of the Bloch sphere known for multiple qubits.

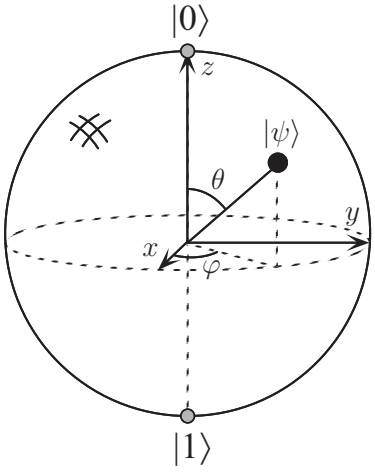


Figure 1.3. Bloch sphere representation of a qubit.

How much information is represented by a qubit? Paradoxically, there are an infinite number of points on the unit sphere, so that in principle one could store an entire text of Shakespeare in the infinite binary expansion of θ. However, this conclusion turns out to be misleading, because of the behavior of a qubit when observed. Recall that measurement of a qubit will give *only* either 0 or 1. Furthermore, measurement *changes* the state of a qubit, collapsing it from its superposition of $|0\rangle$ and $|1\rangle$ to the specific state consistent with the measurement result. For example, if measurement of $|+\rangle$ gives 0, then the post-measurement state of the qubit will be $|0\rangle$. Why does this type of collapse occur? Nobody knows. As discussed in Chapter 2, this behavior is simply one of the *fundamental postulates* of quantum mechanics. What is relevant for our purposes is that from a single measurement one obtains only a single bit of information about the state of the qubit, thus resolving the apparent paradox. It turns out that only if infinitely many

identically prepared qubits were measured would one be able to determine α and β for a qubit in the state given in Equation (1.1).

But an even more interesting question to ask might be: how much information is represented by a qubit *if we do not measure it*? This is a trick question, because how can one quantify information if it cannot be measured? Nevertheless, there is something conceptually important here, because when Nature evolves a closed quantum system of qubits, not performing any 'measurements', she apparently does keep track of all the continuous variables describing the state, like α and β. In a sense, in the state of a qubit, Nature conceals a great deal of 'hidden information'. And even more interestingly, we will see shortly that the potential amount of this extra 'information' grows exponentially with the number of qubits. Understanding this hidden *quantum information* is a question that we grapple with for much of this book, and which lies at the heart of what makes quantum mechanics a powerful tool for information processing.

1.2.1 Multiple qubits

Hilbert space is a big place.
– Carlton Caves

Suppose we have two qubits. If these were two classical bits, then there would be four possible states, 00, 01, 10, and 11. Correspondingly, a two qubit system has four *computational basis states* denoted $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. A pair of qubits can also exist in superpositions of these four states, so the quantum state of two qubits involves associating a complex coefficient – sometimes called an *amplitude* – with each computational basis state, such that the state vector describing the two qubits is

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle. \tag{1.5}$$

Similar to the case for a single qubit, the measurement result x (= 00, 01, 10 or 11) occurs with probability $|\alpha_x|^2$, with the state of the qubits after the measurement being $|x\rangle$. The condition that probabilities sum to one is therefore expressed by the *normalization* condition that $\sum_{x\in\{0,1\}^2} |\alpha_x|^2 = 1$, where the notation '$\{0,1\}^2$' means 'the set of strings of length two with each letter being either zero or one'. For a two qubit system, we could measure just a subset of the qubits, say the first qubit, and you can probably guess how this works: measuring the first qubit alone gives 0 with probability $|\alpha_{00}|^2 + |\alpha_{01}|^2$, leaving the post-measurement state

$$|\psi'\rangle = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}. \tag{1.6}$$

Note how the post-measurement state is *re-normalized* by the factor $\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}$ so that it still satisfies the normalization condition, just as we expect for a legitimate quantum state.

An important two qubit state is the *Bell state* or *EPR pair*,

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}. \tag{1.7}$$

This innocuous-looking state is responsible for many surprises in quantum computation

and quantum information. It is the key ingredient in quantum teleportation and superdense coding, which we'll come to in Section 1.3.7 and Section 2.3, respectively, and the prototype for many other interesting quantum states. The Bell state has the property that upon measuring the first qubit, one obtains two possible results: 0 with probability $1/2$, leaving the post-measurement state $|\varphi'\rangle = |00\rangle$, and 1 with probability $1/2$, leaving $|\varphi'\rangle = |11\rangle$. As a result, a measurement of the second qubit always gives the same result as the measurement of the first qubit. That is, the measurement outcomes are *correlated*. Indeed, it turns out that other types of measurements can be performed on the Bell state, by first applying some operations to the first or second qubit, and that interesting correlations still exist between the result of a measurement on the first and second qubit. These correlations have been the subject of intense interest ever since a famous paper by Einstein, Podolsky and Rosen, in which they first pointed out the strange properties of states like the Bell state. EPR's insights were taken up and greatly improved by John Bell, who proved an amazing result: the measurement correlations in the Bell state are *stronger than could ever exist between classical systems*. These results, described in detail in Section 2.6, were the first intimation that quantum mechanics allows information processing beyond what is possible in the classical world.

More generally, we may consider a system of n qubits. The computational basis states of this system are of the form $|x_1 x_2 \ldots x_n\rangle$, and so a quantum state of such a system is specified by 2^n amplitudes. For $n = 500$ this number is larger than the estimated number of atoms in the Universe! Trying to store all these complex numbers would not be possible on any conceivable classical computer. Hilbert space is indeed a big place. In principle, however, Nature manipulates such enormous quantities of data, even for systems containing only a few hundred atoms. It is as if Nature were keeping 2^{500} hidden pieces of scratch paper on the side, on which she performs her calculations as the system evolves. This enormous potential computational power is something we would very much like to take advantage of. But how can we think of quantum mechanics as computation?

1.3 Quantum computation

Changes occurring to a quantum state can be described using the language of *quantum computation*. Analogous to the way a classical computer is built from an electrical circuit containing wires and logic gates, a quantum computer is built from a *quantum circuit* containing wires and elementary *quantum gates* to carry around and manipulate the quantum information. In this section we describe some simple quantum gates, and present several example circuits illustrating their application, including a circuit which teleports qubits!

1.3.1 Single qubit gates

Classical computer circuits consist of *wires* and *logic gates*. The wires are used to carry information around the circuit, while the logic gates perform manipulations of the information, converting it from one form to another. Consider, for example, classical single bit logic gates. The only non-trivial member of this class is the NOT gate, whose operation is defined by its *truth table*, in which $0 \rightarrow 1$ and $1 \rightarrow 0$, that is, the 0 and 1 states are interchanged.

Can an analogous quantum NOT gate for qubits be defined? Imagine that we had some process which took the state $|0\rangle$ to the state $|1\rangle$, and vice versa. Such a process

would obviously be a good candidate for a quantum analogue to the NOT gate. However, specifying the action of the gate on the states $|0\rangle$ and $|1\rangle$ does not tell us what happens to superpositions of the states $|0\rangle$ and $|1\rangle$, without further knowledge about the properties of quantum gates. In fact, the quantum NOT gate acts *linearly*, that is, it takes the state

$$\alpha|0\rangle + \beta|1\rangle \tag{1.8}$$

to the corresponding state in which the role of $|0\rangle$ and $|1\rangle$ have been interchanged,

$$\alpha|1\rangle + \beta|0\rangle. \tag{1.9}$$

Why the quantum NOT gate acts linearly and not in some nonlinear fashion is a very interesting question, and the answer is not at all obvious. It turns out that this linear behavior is a general property of quantum mechanics, and very well motivated empirically; moreover, nonlinear behavior can lead to apparent paradoxes such as time travel, faster-than-light communication, and violations of the second laws of thermodynamics. We'll explore this point in more depth in later chapters, but for now we'll just take it as given.

There is a convenient way of representing the quantum NOT gate in matrix form, which follows directly from the linearity of quantum gates. Suppose we define a matrix X to represent the quantum NOT gate as follows:

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{1.10}$$

(The notation X for the quantum NOT is used for historical reasons.) If the quantum state $\alpha|0\rangle + \beta|1\rangle$ is written in a vector notation as

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \tag{1.11}$$

with the top entry corresponding to the amplitude for $|0\rangle$ and the bottom entry the amplitude for $|1\rangle$, then the corresponding output from the quantum NOT gate is

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}. \tag{1.12}$$

Notice that the action of the NOT gate is to take the state $|0\rangle$ and replace it by the state corresponding to the first column of the matrix X. Similarly, the state $|1\rangle$ is replaced by the state corresponding to the second column of the matrix X.

So quantum gates on a single qubit can be described by two by two matrices. Are there any constraints on what matrices may be used as quantum gates? It turns out that there are. Recall that the normalization condition requires $|\alpha|^2 + |\beta|^2 = 1$ for a quantum state $\alpha|0\rangle + \beta|1\rangle$. This must also be true of the quantum state $|\psi'\rangle = \alpha'|0\rangle + \beta'|1\rangle$ after the gate has acted. It turns out that the appropriate condition on the matrix representing the gate is that the matrix U describing the single qubit gate be *unitary*, that is $U^\dagger U = I$, where $U^\dagger$ is the *adjoint* of U (obtained by transposing and then complex conjugating U), and I is the two by two identity matrix. For example, for the NOT gate it is easy to verify that $X^\dagger X = I$.

Amazingly, this *unitarity* constraint is the *only* constraint on quantum gates. Any unitary matrix specifies a valid quantum gate! The interesting implication is that in contrast to the classical case, where only one non-trivial single bit gate exists – the NOT

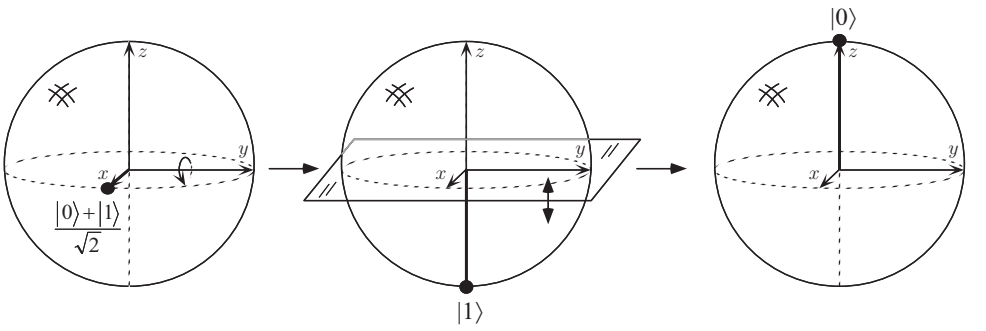


Figure 1.4. Visualization of the Hadamard gate on the Bloch sphere, acting on the input state $(|0\rangle + |1\rangle)/\sqrt{2}$.

gate – there are many non-trivial single qubit gates. Two important ones which we shall use later are the Z gate:

$$Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{1.13}$$

which leaves $|0\rangle$ unchanged, and flips the sign of $|1\rangle$ to give $-|1\rangle$, and the *Hadamard* gate,

$$H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{1.14}$$

This gate is sometimes described as being like a 'square-root of NOT' gate, in that it turns a $|0\rangle$ into $(|0\rangle + |1\rangle)/\sqrt{2}$ (first column of H), 'halfway' between $|0\rangle$ and $|1\rangle$, and turns $|1\rangle$ into $(|0\rangle - |1\rangle)/\sqrt{2}$ (second column of H), which is also 'halfway' between $|0\rangle$ and $|1\rangle$. Note, however, that H^2 is not a NOT gate, as simple algebra shows that $H^2 = I$, and thus applying H twice to a state does nothing to it.

The Hadamard gate is one of the most useful quantum gates, and it is worth trying to visualize its operation by considering the Bloch sphere picture. In this picture, it turns out that single qubit gates correspond to rotations and reflections of the sphere. The Hadamard operation is just a rotation of the sphere about the $\hat{y}$ axis by 90°, followed by a rotation about the $\hat{x}$ axis by 180°, as illustrated in Figure 1.4. Some important single qubit gates are shown in Figure 1.5, and contrasted with the classical case.

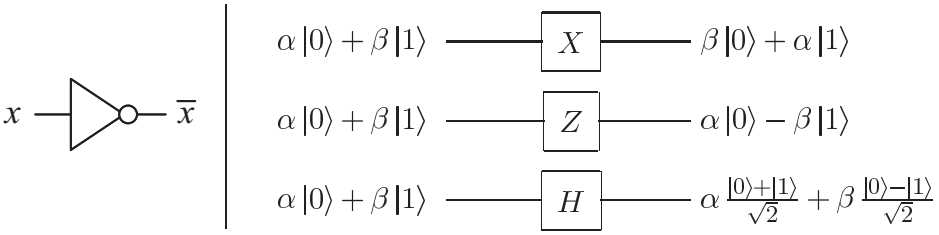


Figure 1.5. Single bit (left) and qubit (right) logic gates.

There are infinitely many two by two unitary matrices, and thus infinitely many single

qubit gates. However, it turns out that the properties of the complete set can be understood from the properties of a much smaller set. For example, as explained in Box 1.1, an arbitrary single qubit unitary gate can be decomposed as a product of rotations

$$\begin{bmatrix} \cos\frac{\gamma}{2} & -\sin\frac{\gamma}{2} \\ \sin\frac{\gamma}{2} & \cos\frac{\gamma}{2} \end{bmatrix}, \tag{1.15}$$

and a gate which we'll later understand as being a rotation about the $\hat{z}$ axis,

$$\begin{bmatrix} e^{-i\beta/2} & 0 \\ 0 & e^{i\beta/2} \end{bmatrix}, \tag{1.16}$$

together with a *(global) phase shift* – a constant multiplier of the form $e^{i\alpha}$. These gates can be broken down further – we don't need to be able to do these gates for arbitrary α, β and γ, but can build arbitrarily good approximations to such gates using only certain special *fixed* values of α, β and γ. In this way it is possible to build up an arbitrary single qubit gate using a *finite* set of quantum gates. More generally, an arbitrary quantum computation on any number of qubits can be generated by a finite set of gates that is said to be *universal* for quantum computation. To obtain such a universal set we first need to introduce some quantum gates involving multiple qubits.

Box 1.1: Decomposing single qubit operations

In Section 4.2 starting on page 174 we prove that an arbitrary 2×2 unitary matrix may be decomposed as

$$U = e^{i\alpha} \begin{bmatrix} e^{-i\beta/2} & 0 \\ 0 & e^{i\beta/2} \end{bmatrix} \begin{bmatrix} \cos\frac{\gamma}{2} & -\sin\frac{\gamma}{2} \\ \sin\frac{\gamma}{2} & \cos\frac{\gamma}{2} \end{bmatrix} \begin{bmatrix} e^{-i\delta/2} & 0 \\ 0 & e^{i\delta/2} \end{bmatrix}, \tag{1.17}$$

where α, β, γ, and δ are real-valued. Notice that the second matrix is just an ordinary rotation. It turns out that the first and last matrices can also be understood as rotations in a different plane. This decomposition can be used to give an exact prescription for performing an *arbitrary* single qubit quantum logic gate.

1.3.2 Multiple qubit gates

Now let us generalize from one to multiple qubits. Figure 1.6 shows five notable multiple bit classical gates, the AND, OR, XOR (exclusive-OR), NAND and NOR gates. An important theoretical result is that any function on bits can be computed from the composition of NAND gates alone, which is thus known as a *universal* gate. By contrast, the XOR alone or even together with NOT is not universal. One way of seeing this is to note that applying an XOR gate does not change the total parity of the bits. As a result, any circuit involving only NOT and XOR gates will, if two inputs x and y have the same parity, give outputs with the same parity, restricting the class of functions which may be computed, and thus precluding universality.

The prototypical multi-qubit quantum logic gate is the *controlled*-NOT or CNOT gate. This gate has two input qubits, known as the *control* qubit and the *target* qubit, respectively. The circuit representation for the CNOT is shown in the top right of Figure 1.6; the top line represents the control qubit, while the bottom line represents the target

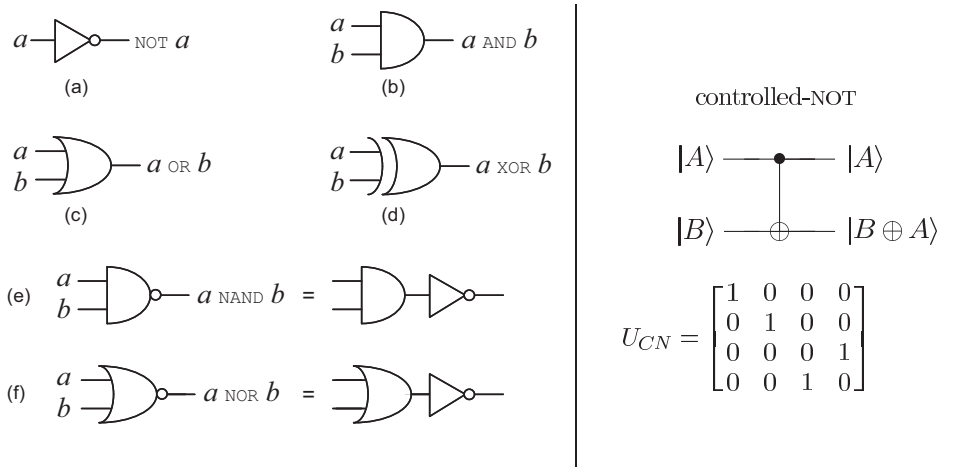


Figure 1.6. On the left are some standard single and multiple bit gates, while on the right is the prototypical multiple qubit gate, the controlled-NOT. The matrix representation of the controlled-NOT, U_{CN}, is written with respect to the amplitudes for $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$, in that order.

qubit. The action of the gate may be described as follows. If the control qubit is set to 0, then the target qubit is left alone. If the control qubit is set to 1, then the target qubit is flipped. In equations:

$$|00\rangle \rightarrow |00\rangle; \; |01\rangle \rightarrow |01\rangle; \; |10\rangle \rightarrow |11\rangle; \; |11\rangle \rightarrow |10\rangle. \tag{1.18}$$

Another way of describing the CNOT is as a generalization of the classical XOR gate, since the action of the gate may be summarized as $|A, B\rangle \rightarrow |A, B \oplus A\rangle$, where $\oplus$ is addition modulo two, which is exactly what the XOR gate does. That is, the control qubit and the target qubit are XORed and stored in the target qubit.

Yet another way of describing the action of the CNOT is to give a matrix representation, as shown in the bottom right of Figure 1.6. You can easily verify that the first column of U_{CN} describes the transformation that occurs to $|00\rangle$, and similarly for the other computational basis states, $|01\rangle$, $|10\rangle$, and $|11\rangle$. As for the single qubit case, the requirement that probability be conserved is expressed in the fact that U_{CN} is a *unitary matrix*, that is, $U_{CN}^{\dagger} U_{CN} = I$.

We noticed that the CNOT can be regarded as a type of generalized-XOR gate. Can other classical gates such as the NAND or the regular XOR gate be understood as unitary gates in a sense similar to the way the quantum NOT gate represents the classical NOT gate? It turns out that this is not possible. The reason is because the XOR and NAND gates are essentially *irreversible* or *non-invertible*. For example, given the output $A \oplus B$ from an XOR gate, it is not possible to determine what the inputs A and B were; there is an irretrievable *loss of information* associated with the irreversible action of the XOR gate. On the other hand, unitary quantum gates are *always* invertible, since the inverse of a unitary matrix is also a unitary matrix, and thus a quantum gate can always be inverted by another quantum gate. Understanding how to do classical logic in this *reversible* or *invertible* sense will be a crucial step in understanding how to harness the power of

quantum mechanics for computation. We'll explain the basic idea of how to do reversible computation in Section 1.4.1.

Of course, there are many interesting quantum gates other than the controlled-NOT. However, in a sense the controlled-NOT and single qubit gates are the prototypes for *all* other gates because of the following remarkable *universality* result: *Any multiple qubit logic gate may be composed from* CNOT *and single qubit gates*. The proof is given in Section 4.5, and is the quantum parallel of the universality of the NAND gate.

1.3.3 Measurements in bases other than the computational basis

We've described quantum measurements of a single qubit in the state $\alpha|0\rangle + \beta|1\rangle$ as yielding the result 0 or 1 and leaving the qubit in the corresponding state $|0\rangle$ or $|1\rangle$, with respective probabilities $|\alpha|^2$ and $|\beta|^2$. In fact, quantum mechanics allows somewhat more versatility in the class of measurements that may be performed, although certainly nowhere near enough to recover α and β from a single measurement!

Note that the states $|0\rangle$ and $|1\rangle$ represent just one of many possible choices of basis states for a qubit. Another possible choice is the set $|+\rangle \equiv (|0\rangle + |1\rangle)/\sqrt{2}$ and $|-\rangle \equiv (|0\rangle - |1\rangle)/\sqrt{2}$. An arbitrary state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ can be re-expressed in terms of the states $|+\rangle$ and $|-\rangle$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha\frac{|+\rangle + |-\rangle}{\sqrt{2}} + \beta\frac{|+\rangle - |-\rangle}{\sqrt{2}} = \frac{\alpha+\beta}{\sqrt{2}}|+\rangle + \frac{\alpha-\beta}{\sqrt{2}}|-\rangle. \quad (1.19)$$

It turns out that it is possible to treat the $|+\rangle$ and $|-\rangle$ states as though they were the computational basis states, and measure with respect to this new basis. Naturally, measuring with respect to the $|+\rangle$, $|-\rangle$ basis results in the result '+' with probability $|\alpha+\beta|^2/2$ and the result '$-$' with probability $|\alpha-\beta|^2/2$, with corresponding post-measurement states $|+\rangle$ and $|-\rangle$, respectively.

More generally, given any basis states $|a\rangle$ and $|b\rangle$ for a qubit, it is possible to express an arbitrary state as a linear combination $\alpha|a\rangle + \beta|b\rangle$ of those states. Furthermore, provided the states are *orthonormal*, it is possible to *perform a measurement with respect to the* $|a\rangle, |b\rangle$ *basis*, giving the result a with probability $|\alpha|^2$ and b with probability $|\beta|^2$. The orthonormality constraint is necessary in order that $|\alpha|^2 + |\beta|^2 = 1$ as we expect for probabilities. In an analogous way it is possible in principle to measure a quantum system of many qubits with respect to an arbitrary orthonormal basis. However, just because it is possible in principle does not mean that such a measurement can be done easily, and we return later to the question of how efficiently a measurement in an arbitrary basis can be performed.

There are many reasons for using this extended formalism for quantum measurements, but ultimately the best one is this: the formalism allows us to describe observed experimental results, as we will see in our discussion of the Stern–Gerlach experiment in Section 1.5.1. An even more sophisticated and convenient (but essentially equivalent) formalism for describing quantum measurements is described in the next chapter, in Section 2.2.3.

1.3.4 Quantum circuits

We've already met a few simple quantum circuits. Let's look in a little more detail at the elements of a quantum circuit. A simple quantum circuit containing three quantum gates is shown in Figure 1.7. The circuit is to be read from left-to-right. Each line

in the circuit represents a *wire* in the quantum circuit. This wire does not necessarily correspond to a physical wire; it may correspond instead to the passage of time, or perhaps to a physical particle such as a photon – a particle of light – moving from one location to another through space. It is conventional to assume that the state input to the circuit is a computational basis state, usually the state consisting of all $|0\rangle$s. This rule is broken frequently in the literature on quantum computation and quantum information, but it is considered polite to inform the reader when this is the case.

The circuit in Figure 1.7 accomplishes a simple but useful task – it swaps the states of the two qubits. To see that this circuit accomplishes the swap operation, note that the sequence of gates has the following sequence of effects on a computational basis state $|a, b\rangle$,

$$\begin{aligned} |a, b\rangle &\longrightarrow |a, a \oplus b\rangle \\ &\longrightarrow |a \oplus (a \oplus b), a \oplus b\rangle = |b, a \oplus b\rangle \\ &\longrightarrow |b, (a \oplus b) \oplus b\rangle = |b, a\rangle \, , \end{aligned} \tag{1.20}$$

where all additions are done modulo 2. The effect of the circuit, therefore, is to interchange the state of the two qubits.

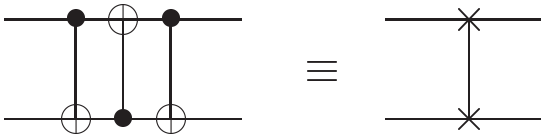


Figure 1.7. Circuit swapping two qubits, and an equivalent schematic symbol notation for this common and useful circuit.

There are a few features allowed in classical circuits that are not usually present in quantum circuits. First of all, we don't allow 'loops', that is, feedback from one part of the quantum circuit to another; we say the circuit is *acyclic*. Second, classical circuits allow wires to be 'joined' together, an operation known as FANIN, with the resulting single wire containing the bitwise OR of the inputs. Obviously this operation is not reversible and therefore not unitary, so we don't allow FANIN in our quantum circuits. Third, the inverse operation, FANOUT, whereby several copies of a bit are produced is also not allowed in quantum circuits. In fact, it turns out that quantum mechanics forbids the copying of a qubit, making the FANOUT operation impossible! We'll see an example of this in the next section when we attempt to design a circuit to copy a qubit.

As we proceed we'll introduce new quantum gates as needed. It's convenient to introduce another convention about quantum circuits at this point. This convention is illustrated in Figure 1.8. Suppose U is *any* unitary matrix acting on some number n of qubits, so U can be regarded as a quantum gate on those qubits. Then we can define a *controlled-U* gate which is a natural extension of the controlled-NOT gate. Such a gate has a single *control qubit*, indicated by the line with the black dot, and n *target qubits*, indicated by the boxed U. If the control qubit is set to 0 then nothing happens to the target qubits. If the control qubit is set to 1 then the gate U is applied to the target qubits. The prototypical example of the controlled-U gate is the controlled-NOT gate, which is a controlled-U gate with $U = X$, as illustrated in Figure 1.9.

Another important operation is measurement, which we represent by a 'meter' symbol,

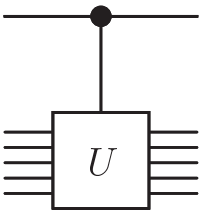


Figure 1.8. Controlled-U gate.

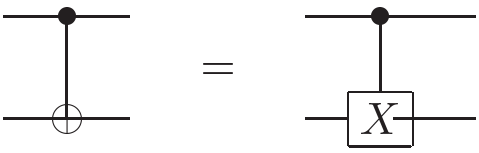


Figure 1.9. Two different representations for the controlled-NOT.

as shown in Figure 1.10. As previously described, this operation converts a single qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ into a probabilistic classical bit M (distinguished from a qubit by drawing it as a double-line wire), which is 0 with probability $|\alpha|^2$, or 1 with probability $|\beta|^2$.

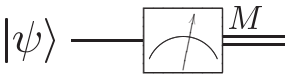


Figure 1.10. Quantum circuit symbol for measurement.

We shall find quantum circuits useful as models of all quantum processes, including but not limited to computation, communication, and even quantum noise. Several simple examples illustrate this below.

1.3.5 Qubit copying circuit?

The CNOT gate is useful for demonstrating one particularly fundamental property of quantum information. Consider the task of copying a classical bit. This may be done using a classical CNOT gate, which takes in the bit to copy (in some unknown state x) and a 'scratchpad' bit initialized to zero, as illustrated in Figure 1.11. The output is two bits, both of which are in the same state x.

Suppose we try to copy a qubit in the unknown state $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$ in the same manner by using a CNOT gate. The input state of the two qubits may be written as

$$\left[a\,|0\rangle + b\,|1\rangle\right]|0\rangle = a\,|00\rangle + b\,|10\rangle, \tag{1.21}$$

The function of CNOT is to negate the second qubit when the first qubit is 1, and thus the output is simply $a\,|00\rangle + b\,|11\rangle$. Have we successfully copied $|\psi\rangle$? That is, have we created the state $|\psi\rangle|\psi\rangle$? In the case where $|\psi\rangle = |0\rangle$ or $|\psi\rangle = |1\rangle$ that is indeed what this circuit does; it is possible to use quantum circuits to copy classical information encoded as a $|0\rangle$ or a $|1\rangle$. However, for a general state $|\psi\rangle$ we see that

$$|\psi\rangle|\psi\rangle = a^2|00\rangle + ab|01\rangle + ab|10\rangle + b^2|11\rangle. \tag{1.22}$$

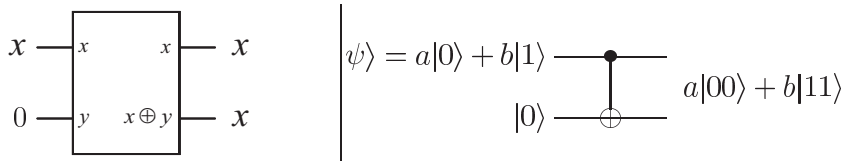


Figure 1.11. Classical and quantum circuits to 'copy' an unknown bit or qubit.

Comparing with $a|00\rangle + b|11\rangle$, we see that unless $ab = 0$ the 'copying circuit' above does *not* copy the quantum state input. In fact, it turns out to be *impossible* to make a copy of an unknown quantum state. This property, that qubits cannot be copied, is known as the *no-cloning* theorem, and it is one of the chief differences between quantum and classical information. The no-cloning theorem is discussed at more length in Box 12.1 on page 532; the proof is very simple, and we encourage you to skip ahead and read the proof now.

There is another way of looking at the failure of the circuit in Figure 1.11, based on the intuition that a qubit somehow contains 'hidden' information not directly accessible to measurement. Consider what happens when we measure one of the qubits of the state $a|00\rangle + b|11\rangle$. As previously described, we obtain either 0 or 1 with probabilities $|a|^2$ and $|b|^2$. However, once one qubit is measured, the state of the other one is completely determined, and no additional information can be gained about a and b. In this sense, the extra hidden information carried in the original qubit $|\psi\rangle$ was lost in the first measurement, and cannot be regained. If, however, the qubit had been copied, then the state of the other qubit should still contain some of that hidden information. Therefore, a copy cannot have been created.

1.3.6 Example: Bell states

Let's consider a slightly more complicated circuit, shown in Figure 1.12, which has a Hadamard gate followed by a CNOT, and transforms the four computational basis states according to the table given. As an explicit example, the Hadamard gate takes the input $|00\rangle$ to $(|0\rangle + |1\rangle)|0\rangle/\sqrt{2}$, and then the CNOT gives the output state $(|00\rangle + |11\rangle)/\sqrt{2}$. Note how this works: first, the Hadamard transform puts the top qubit in a superposition; this then acts as a control input to the CNOT, and the target gets inverted only when the control is 1. The output states

$$|\beta_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}; \tag{1.23}$$

$$|\beta_{01}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}; \tag{1.24}$$

$$|\beta_{10}\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}; \text{ and} \tag{1.25}$$

$$|\beta_{11}\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}, \tag{1.26}$$

are known as the *Bell states*, or sometimes the *EPR states* or *EPR pairs*, after some of the people – Bell, and Einstein, Podolsky, and Rosen – who first pointed out the strange properties of states like these. The mnemonic notation $|\beta_{00}\rangle, |\beta_{01}\rangle, |\beta_{10}\rangle, |\beta_{11}\rangle$ may be

understood via the equations

$$|\beta_{xy}\rangle \equiv \frac{|0,y\rangle + (-1)^x |1,\bar{y}\rangle}{\sqrt{2}}, \tag{1.27}$$

where $\bar{y}$ is the negation of y.

In	Out
$\|00\rangle$	$(\|00\rangle + \|11\rangle)/\sqrt{2} \equiv \|\beta_{00}\rangle$
$\|01\rangle$	$(\|01\rangle + \|10\rangle)/\sqrt{2} \equiv \|\beta_{01}\rangle$
$\|10\rangle$	$(\|00\rangle - \|11\rangle)/\sqrt{2} \equiv \|\beta_{10}\rangle$
$\|11\rangle$	$(\|01\rangle - \|10\rangle)/\sqrt{2} \equiv \|\beta_{11}\rangle$

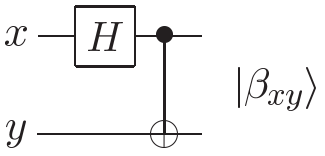


Figure 1.12. Quantum circuit to create Bell states, and its input–ouput quantum 'truth table'.

1.3.7 Example: quantum teleportation

We will now apply the techniques of the last few pages to understand something non-trivial, surprising, and a lot of fun – quantum teleportation! Quantum teleportation is a technique for moving quantum states around, even in the absence of a quantum communications channel linking the sender of the quantum state to the recipient.

Here's how quantum teleportation works. Alice and Bob met long ago but now live far apart. While together they generated an EPR pair, each taking one qubit of the EPR pair when they separated. Many years later, Bob is in hiding, and Alice's mission, should she choose to accept it, is to deliver a qubit $|\psi\rangle$ to Bob. She does not know the state of the qubit, and moreover can only send *classical* information to Bob. Should Alice accept the mission?

Intuitively, things look pretty bad for Alice. She doesn't know the state $|\psi\rangle$ of the qubit she has to send to Bob, and the laws of quantum mechanics prevent her from determining the state when she only has a single copy of $|\psi\rangle$ in her possession. What's worse, even if she did know the state $|\psi\rangle$, describing it precisely takes an infinite amount of classical information since $|\psi\rangle$ takes values in a *continuous* space. So even if she did know $|\psi\rangle$, it would take forever for Alice to describe the state to Bob. It's not looking good for Alice. Fortunately for Alice, quantum teleportation is a way of utilizing the entangled EPR pair in order to send $|\psi\rangle$ to Bob, with only a small overhead of classical communication.

In outline, the steps of the solution are as follows: Alice interacts the qubit $|\psi\rangle$ with her half of the EPR pair, and then measures the two qubits in her possession, obtaining one of four possible classical results, 00, 01, 10, and 11. She sends this information to Bob. Depending on Alice's classical message, Bob performs one of four operations on his half of the EPR pair. Amazingly, by doing this he can recover the original state $|\psi\rangle$!

The quantum circuit shown in Figure 1.13 gives a more precise description of quantum teleportation. The state to be teleported is $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are unknown amplitudes. The state input into the circuit $|\psi_0\rangle$ is

$$|\psi_0\rangle = |\psi\rangle|\beta_{00}\rangle \tag{1.28}$$

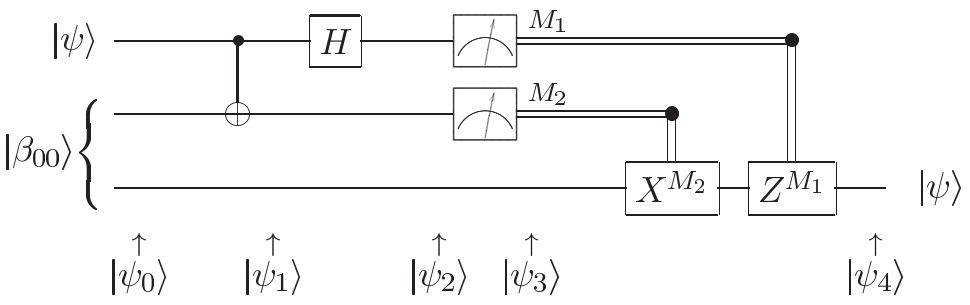


Figure 1.13. Quantum circuit for teleporting a qubit. The two top lines represent Alice's system, while the bottom line is Bob's system. The meters represent measurement, and the double lines coming out of them carry classical bits (recall that single lines denote qubits).

$$= \frac{1}{\sqrt{2}} \left[\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|00\rangle + |11\rangle) \right], \tag{1.29}$$

where we use the convention that the first two qubits (on the left) belong to Alice, and the third qubit to Bob. As we explained previously, Alice's second qubit and Bob's qubit start out in an EPR state. Alice sends her qubits through a CNOT gate, obtaining

$$|\psi_1\rangle = \frac{1}{\sqrt{2}} \left[\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|10\rangle + |01\rangle) \right]. \tag{1.30}$$

She then sends the first qubit through a Hadamard gate, obtaining

$$|\psi_2\rangle = \frac{1}{2} \left[\alpha(|0\rangle + |1\rangle)(|00\rangle + |11\rangle) + \beta(|0\rangle - |1\rangle)(|10\rangle + |01\rangle) \right]. \tag{1.31}$$

This state may be re-written in the following way, simply by regrouping terms:

$$\begin{aligned} |\psi_2\rangle = \frac{1}{2} \Big[& |00\rangle \left(\alpha|0\rangle + \beta|1\rangle \right) + |01\rangle \left(\alpha|1\rangle + \beta|0\rangle \right) \\ & + |10\rangle \left(\alpha|0\rangle - \beta|1\rangle \right) + |11\rangle \left(\alpha|1\rangle - \beta|0\rangle \right) \Big] . \end{aligned} \tag{1.32}$$

This expression naturally breaks down into four terms. The first term has Alice's qubits in the state $|00\rangle$, and Bob's qubit in the state $\alpha|0\rangle + \beta|1\rangle$ – which is the original state $|\psi\rangle$. If Alice performs a measurement and obtains the result 00 then Bob's system will be in the state $|\psi\rangle$. Similarly, from the previous expression we can read off Bob's post-measurement state, given the result of Alice's measurement:

$$00 \longmapsto |\psi_3(00)\rangle \equiv \left[\alpha|0\rangle + \beta|1\rangle \right] \tag{1.33}$$

$$01 \longmapsto |\psi_3(01)\rangle \equiv \left[\alpha|1\rangle + \beta|0\rangle \right] \tag{1.34}$$

$$10 \longmapsto |\psi_3(10)\rangle \equiv \left[\alpha|0\rangle - \beta|1\rangle \right] \tag{1.35}$$

$$11 \longmapsto |\psi_3(11)\rangle \equiv \left[\alpha|1\rangle - \beta|0\rangle \right]. \tag{1.36}$$

Depending on Alice's measurement outcome, Bob's qubit will end up in one of these four possible states. Of course, to know which state it is in, Bob must be told the result of Alice's measurement – we will show later that it is this fact which prevents teleportation

from being used to transmit information faster than light. Once Bob has learned the measurement outcome, Bob can 'fix up' his state, recovering $|\psi\rangle$, by applying the appropriate quantum gate. For example, in the case where the measurement yields 00, Bob doesn't need to do anything. If the measurement is 01 then Bob can fix up his state by applying the X gate. If the measurement is 10 then Bob can fix up his state by applying the Z gate. If the measurement is 11 then Bob can fix up his state by applying first an X and then a Z gate. Summing up, Bob needs to apply the transformation $Z^{M_1}X^{M_2}$ (note how time goes from left to right in circuit diagrams, but in matrix products terms on the *right* happen *first*) to his qubit, and he will recover the state $|\psi\rangle$.

There are many interesting features of teleportation, some of which we shall return to later in the book. For now we content ourselves with commenting on a couple of aspects. First, doesn't teleportation allow one to transmit quantum states faster than light? This would be rather peculiar, because the theory of relativity implies that faster than light information transfer could be used to send information backwards in time. Fortunately, quantum teleportation does not enable faster than light communication, because to complete the teleportation Alice must transmit her measurement result to Bob over a classical communications channel. We will show in Section 2.4.3 that without this classical communication, teleportation does not convey *any* information at all. The classical channel is limited by the speed of light, so it follows that quantum teleportation cannot be accomplished faster than the speed of light, resolving the apparent paradox.

A second puzzle about teleportation is that it appears to create a copy of the quantum state being teleported, in apparent violation of the no-cloning theorem discussed in Section 1.3.5. This violation is only illusory since after the teleportation process only the target qubit is left in the state $|\psi\rangle$, and the original data qubit ends up in one of the computational basis states $|0\rangle$ or $|1\rangle$, depending upon the measurement result on the first qubit.

What can we learn from quantum teleportation? Quite a lot! It's much more than just a neat trick one can do with quantum states. Quantum teleportation emphasizes the interchangeability of *different* resources in quantum mechanics, showing that one shared EPR pair together with two classical bits of communication is a resource at least the equal of one qubit of communication. Quantum computation and quantum information has revealed a plethora of methods for interchanging resources, many built upon quantum teleportation. In particular, in Chapter 10 we explain how teleportation can be used to build quantum gates which are resistant to the effects of noise, and in Chapter 12 we show that teleportation is intimately connected with the properties of quantum error-correcting codes. Despite these connections with other subjects, it is fair to say that we are only beginning to understand *why* it is that quantum teleportation is possible in quantum mechanics; in later chapters we endeavor to explain some of the insights that make such an understanding possible.

1.4 Quantum algorithms

What class of computations can be performed using quantum circuits? How does that class compare with the computations which can be performed using classical logical circuits? Can we find a task which a quantum computer may perform better than a classical computer? In this section we investigate these questions, explaining how to perform classical computations on quantum computers, giving some examples of problems for

which quantum computers offer an advantage over classical computers, and summarizing the known quantum algorithms.

1.4.1 Classical computations on a quantum computer

Can we simulate a classical logic circuit using a quantum circuit? Not surprisingly, the answer to this question turns out to be yes. It would be very surprising if this were not the case, as physicists believe that all aspects of the world around us, including classical logic circuits, can ultimately be explained using quantum mechanics. As pointed out earlier, the reason quantum circuits cannot be used to directly simulate classical circuits is because unitary quantum logic gates are inherently *reversible*, whereas many classical logic gates such as the NAND gate are inherently irreversible.

Any classical circuit can be replaced by an equivalent circuit containing only *reversible* elements, by making use of a reversible gate known as the *Toffoli gate*. The Toffoli gate has three input bits and three output bits, as illustrated in Figure 1.14. Two of the bits are *control bits* that are unaffected by the action of the Toffoli gate. The third bit is a *target bit* that is flipped if both control bits are set to 1, and otherwise is left alone. Note that applying the Toffoli gate twice to a set of bits has the effect $(a, b, c) \rightarrow (a, b, c \oplus ab) \rightarrow (a, b, c)$, and thus the Toffoli gate is a reversible gate, since it has an inverse – itself.

Inputs			Outputs		
a	b	c	a'	b'	c'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

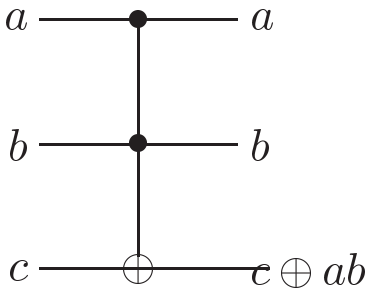


Figure 1.14. Truth table for the Toffoli gate, and its circuit representation.

The Toffoli gate can be used to simulate NAND gates, as shown in Figure 1.15, and can also be used to do FANOUT, as shown in Figure 1.16. With these two operations it becomes possible to simulate all other elements in a classical circuit, and thus an arbitrary classical circuit can be simulated by an equivalent reversible circuit.

The Toffoli gate has been described as a classical gate, but it can also be implemented as a quantum logic gate. By definition, the quantum logic implementation of the Toffoli gate simply permutes computational basis states in the same way as the classical Toffoli gate. For example, the quantum Toffoli gate acting on the state $|110\rangle$ flips the third qubit because the first two are set, resulting in the state $|111\rangle$. It is tedious but not difficult to write this transformation out as an 8 by 8 matrix, U, and verify explicitly that U is a unitary matrix, and thus the Toffoli gate is a legitimate quantum gate. The quantum Toffoli gate can be used to simulate irreversible classical logic gates, just as the classical

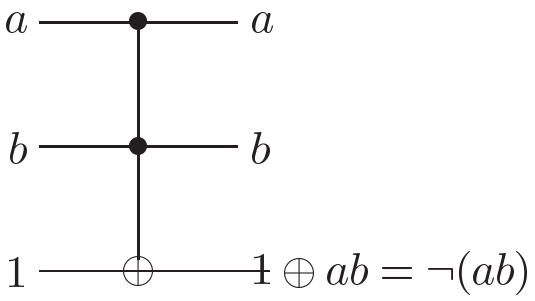


Figure 1.15. Classical circuit implementing a NAND gate using a Toffoli gate. The top two bits represent the input to the NAND, while the third bit is prepared in the standard state 1, sometimes known as an *ancilla* state. The output from the NAND is on the third bit.

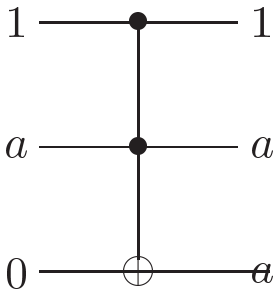


Figure 1.16. FANOUT with the Toffoli gate, with the second bit being the input to the FANOUT (and the other two bits standard ancilla states), and the output from FANOUT appearing on the second and third bits.

Toffoli gate was, and ensures that quantum computers are capable of performing any computation which a classical (deterministic) computer may do.

What if the classical computer is non-deterministic, that is, has the ability to generate random bits to be used in the computation? Not surprisingly, it is easy for a quantum computer to simulate this. To perform such a simulation it turns out to be sufficient to produce random fair coin tosses, which can be done by preparing a qubit in the state $|0\rangle$, sending it through a Hadamard gate to produce $(|0\rangle + |1\rangle)/\sqrt{2}$, and then measuring the state. The result will be $|0\rangle$ or $|1\rangle$ with 50/50 probability. This provides a quantum computer with the ability to efficiently simulate a non-deterministic classical computer.

Of course, if the ability to simulate classical computers were the only feature of quantum computers there would be little point in going to all the trouble of exploiting quantum effects! The advantage of quantum computing is that much more powerful functions may be computed using qubits and quantum gates. In the next few sections we explain how to do this, culminating in the Deutsch–Jozsa algorithm, our first example of a quantum algorithm able to solve a problem faster than any classical algorithm.

1.4.2 Quantum parallelism

Quantum parallelism is a fundamental feature of many quantum algorithms. Heuristically, and at the risk of over-simplifying, quantum parallelism allows quantum computers to evaluate a function $f(x)$ for many *different* values of x simultaneously. In this section we explain how quantum parallelism works, and some of its limitations.

Suppose $f(x) : \{0,1\} \rightarrow \{0,1\}$ is a function with a one-bit domain and range. A

convenient way of computing this function on a quantum computer is to consider a two qubit quantum computer which starts in the state $|x, y\rangle$. With an appropriate sequence of logic gates it is possible to transform this state into $|x, y \oplus f(x)\rangle$, where $\oplus$ indicates addition modulo 2; the first register is called the 'data' register, and the second register the 'target' register. We give the transformation defined by the map $|x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$ a name, U_f, and note that it is easily shown to be unitary. If $y = 0$, then the final state of the second qubit is just the value $f(x)$. (In Section 3.2.5 we show that given a classical circuit for computing f there is a quantum circuit of comparable efficiency which computes the transformation U_f on a quantum computer. For our purposes it can be considered to be a black box.)

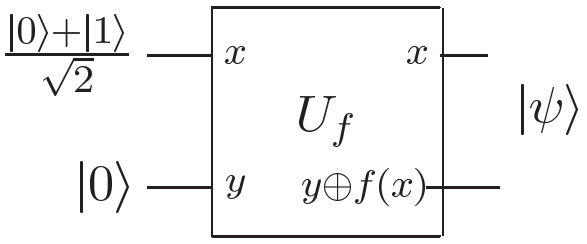


Figure 1.17. Quantum circuit for evaluating $f(0)$ and $f(1)$ simultaneously. U_f is the quantum circuit which takes inputs like $|x, y\rangle$ to $|x, y \oplus f(x)\rangle$.

Consider the circuit shown in Figure 1.17, which applies U_f to an input not in the computational basis. Instead, the data register is prepared in the superposition $(|0\rangle + |1\rangle)/\sqrt{2}$, which can be created with a Hadamard gate acting on $|0\rangle$. Then we apply U_f, resulting in the state:

$$\frac{|0, f(0)\rangle + |1, f(1)\rangle}{\sqrt{2}}. \tag{1.37}$$

This is a remarkable state! The different terms contain information about both $f(0)$ and $f(1)$; it is almost as if we have evaluated $f(x)$ for two values of x simultaneously, a feature known as 'quantum parallelism'. Unlike classical parallelism, where multiple circuits each built to compute $f(x)$ are executed simultaneously, here a *single* $f(x)$ circuit is employed to evaluate the function for multiple values of x simultaneously, by exploiting the ability of a quantum computer to be in superpositions of different states.

This procedure can easily be generalized to functions on an arbitrary number of bits, by using a general operation known as the *Hadamard transform*, or sometimes the *Walsh–Hadamard transform*. This operation is just n Hadamard gates acting in parallel on n qubits. For example, shown in Figure 1.18 is the case $n = 2$ with qubits initially prepared as $|0\rangle$, which gives

$$\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2} \tag{1.38}$$

as output. We write $H^{\otimes 2}$ to denote the parallel action of two Hadamard gates, and read '$\otimes$' as 'tensor'. More generally, the result of performing the Hadamard transform on n

qubits initially in the all $|0\rangle$ state is

$$\frac{1}{\sqrt{2^n}} \sum_x |x\rangle \,, \tag{1.39}$$

where the sum is over all possible values of x, and we write $H^{\otimes n}$ to denote this action. That is, the Hadamard transform produces an equal superposition of all computational basis states. Moreover, it does this extremely efficiently, producing a superposition of 2^n states using just n gates.

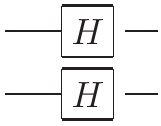


Figure 1.18. The Hadamard transform $H^{\otimes 2}$ on two qubits.

Quantum parallel evaluation of a function with an n bit input x and 1 bit output, $f(x)$, can thus be performed in the following manner. Prepare the $n + 1$ qubit state $|0\rangle^{\otimes n}|0\rangle$, then apply the Hadamard transform to the first n qubits, followed by the quantum circuit implementing U_f. This produces the state

$$\frac{1}{\sqrt{2^n}} \sum_x |x\rangle |f(x)\rangle \,. \tag{1.40}$$

In some sense, quantum parallelism enables all possible values of the function f to be evaluated simultaneously, even though we apparently only evaluated f once. However, this parallelism is *not* immediately useful. In our single qubit example, measurement of the state gives only *either* $|0, f(0)\rangle$ *or* $|1, f(1)\rangle$! Similarly, in the general case, measurement of the state $\sum_x |x, f(x)\rangle$ would give only $f(x)$ for a single value of x. Of course, a classical computer can do this easily! Quantum computation requires something more than just quantum parallelism to be useful; it requires the ability to *extract* information about more than one value of $f(x)$ from superposition states like $\sum_x |x, f(x)\rangle$. Over the next two sections we investigate examples of how this may be done.

1.4.3 Deutsch's algorithm

A simple modification of the circuit in Figure 1.17 demonstrates how quantum circuits can outperform classical ones by implementing *Deutsch's algorithm* (we actually present a simplified and improved version of the original algorithm; see 'History and further reading' at the end of the chapter). Deutsch's algorithm combines quantum parallelism with a property of quantum mechanics known as *interference*. As before, let us use the Hadamard gate to prepare the first qubit as the superposition $(|0\rangle + |1\rangle)/\sqrt{2}$, but now let us prepare the second qubit y as the superposition $(|0\rangle - |1\rangle)/\sqrt{2}$, using a Hadamard gate applied to the state $|1\rangle$. Let us follow the states along to see what happens in this circuit, shown in Figure 1.19.

The input state

$$|\psi_0\rangle = |01\rangle \tag{1.41}$$

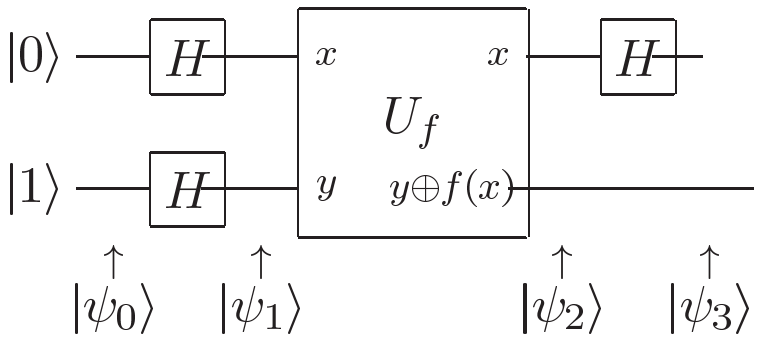


Figure 1.19. Quantum circuit implementing Deutsch's algorithm.

is sent through two Hadamard gates to give

$$|\psi_1\rangle = \left[\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]. \tag{1.42}$$

A little thought shows that if we apply U_f to the state $|x\rangle(|0\rangle - |1\rangle)/\sqrt{2}$ then we obtain the state $(-1)^{f(x)}|x\rangle(|0\rangle - |1\rangle)/\sqrt{2}$. Applying U_f to $|\psi_1\rangle$ therefore leaves us with one of two possibilities:

$$|\psi_2\rangle = \begin{cases} \pm\left[\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] & \text{if } f(0) = f(1) \\ \pm\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] & \text{if } f(0) \neq f(1). \end{cases} \tag{1.43}$$

The final Hadamard gate on the first qubit thus gives us

$$|\psi_3\rangle = \begin{cases} \pm|0\rangle\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] & \text{if } f(0) = f(1) \\ \pm|1\rangle\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] & \text{if } f(0) \neq f(1). \end{cases} \tag{1.44}$$

Realizing that $f(0) \oplus f(1)$ is 0 if $f(0) = f(1)$ and 1 otherwise, we can rewrite this result concisely as

$$|\psi_3\rangle = \pm|f(0) \oplus f(1)\rangle\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right], \tag{1.45}$$

so by measuring the first qubit we may determine $f(0) \oplus f(1)$. This is very interesting indeed: the quantum circuit has given us the ability to determine a *global property* of $f(x)$, namely $f(0) \oplus f(1)$, using only *one* evaluation of $f(x)$! This is faster than is possible with a classical apparatus, which would require at least two evaluations.

This example highlights the difference between quantum parallelism and classical randomized algorithms. Naively, one might think that the state $|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle$ corresponds rather closely to a probabilistic classical computer that evaluates $f(0)$ with probability one-half, or $f(1)$ with probability one-half. The difference is that in a classical computer these two alternatives forever exclude one another; in a quantum computer it is

possible for the two alternatives to *interfere* with one another to yield some global property of the function f, by using something like the Hadamard gate to recombine the different alternatives, as was done in Deutsch's algorithm. The essence of the design of many quantum algorithms is that a clever choice of function and final transformation allows efficient determination of useful global information about the function – information which cannot be attained quickly on a classical computer.

1.4.4 The Deutsch–Jozsa algorithm

Deutsch's algorithm is a simple case of a more general quantum algorithm, which we shall refer to as the Deutsch–Jozsa algorithm. The application, known as *Deutsch's problem*, may be described as the following game. Alice, in Amsterdam, selects a number x from 0 to $2^n - 1$, and mails it in a letter to Bob, in Boston. Bob calculates some function $f(x)$ and replies with the result, which is either 0 or 1. Now, Bob has promised to use a function f which is of one of two kinds; either $f(x)$ is *constant* for all values of x, or else $f(x)$ is *balanced*, that is, equal to 1 for exactly half of all the possible x, and 0 for the other half. Alice's goal is to determine with certainty whether Bob has chosen a constant or a balanced function, corresponding with him as little as possible. How fast can she succeed?

In the classical case, Alice may only send Bob one value of x in each letter. At worst, Alice will need to query Bob at least $2^n/2 + 1$ times, since she may receive $2^n/2$ 0s before finally getting a 1, telling her that Bob's function is balanced. The best deterministic classical algorithm she can use therefore requires $2^n/2 + 1$ queries. Note that in each letter, Alice sends Bob n bits of information. Furthermore, in this example, physical distance is being used to artificially elevate the cost of calculating $f(x)$, but this is not needed in the general problem, where $f(x)$ may be inherently difficult to calculate.

If Bob and Alice were able to exchange qubits, instead of just classical bits, and if Bob agreed to calculate $f(x)$ using a unitary transform U_f, then Alice could achieve her goal in just *one* correspondence with Bob, using the following algorithm.

Analogously to Deutsch's algorithm, Alice has an n qubit register to store her query in, and a single qubit register which she will give to Bob, to store the answer in. She begins by preparing both her query and answer registers in a superposition state. Bob will evaluate $f(x)$ using quantum parallelism and leave the result in the answer register. Alice then interferes states in the superposition using a Hadamard transform on the query register, and finishes by performing a suitable measurement to determine whether f was constant or balanced.

The specific steps of the algorithm are depicted in Figure 1.20. Let us follow the states through this circuit. The input state

$$|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle \tag{1.46}$$

is similar to that of Equation (1.41), but here the query register describes the state of n qubits all prepared in the $|0\rangle$ state. After the Hadamard transform on the query register and the Hadamard gate on the answer register we have

$$|\psi_1\rangle = \sum_{x\in\{0,1\}^n} \frac{|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]. \tag{1.47}$$

The query register is now a superposition of all values, and the answer register is in an

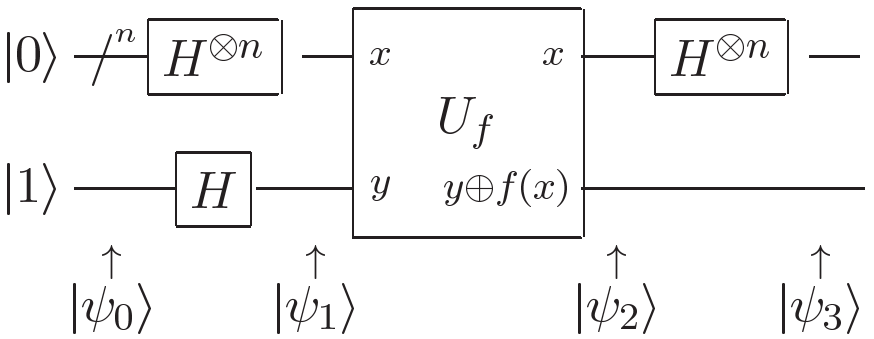


Figure 1.20. Quantum circuit implementing the general Deutsch–Jozsa algorithm. The wire with a '/' through it represents a set of n qubits, similar to the common engineering notation.

evenly weighted superposition of 0 and 1. Next, the function f is evaluated (by Bob) using $U_f : |x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$, giving

$$|\psi_2\rangle = \sum_x \frac{(-1)^{f(x)}|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]. \tag{1.48}$$

Alice now has a set of qubits in which the result of Bob's function evaluation is stored in the amplitude of the qubit superposition state. She now interferes terms in the superposition using a Hadamard transform on the query register. To determine the result of the Hadamard transform it helps to first calculate the effect of the Hadamard transform on a state $|x\rangle$. By checking the cases $x = 0$ and $x = 1$ separately we see that for a single qubit $H|x\rangle = \sum_z (-1)^{xz}|z\rangle/\sqrt{2}$. Thus

$$H^{\otimes n}|x_1, \ldots, x_n\rangle = \frac{\sum_{z_1,\ldots,z_n}(-1)^{x_1 z_1 + \cdots + x_n z_n}|z_1, \ldots, z_n\rangle}{\sqrt{2^n}}. \tag{1.49}$$

This can be summarized more succinctly in the very useful equation

$$H^{\otimes n}|x\rangle = \frac{\sum_z (-1)^{x \cdot z}|z\rangle}{\sqrt{2^n}}, \tag{1.50}$$

where $x \cdot z$ is the bitwise inner product of x and z, modulo 2. Using this equation and (1.48) we can now evaluate $|\psi_3\rangle$,

$$|\psi_3\rangle = \sum_z \sum_x \frac{(-1)^{x \cdot z + f(x)}|z\rangle}{2^n} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]. \tag{1.51}$$

Alice now observes the query register. Note that the amplitude for the state $|0\rangle^{\otimes n}$ is $\sum_x (-1)^{f(x)}/2^n$. Let's look at the two possible cases – f constant and f balanced – to discern what happens. In the case where f is constant the amplitude for $|0\rangle^{\otimes n}$ is +1 or -1, depending on the constant value $f(x)$ takes. Because $|\psi_3\rangle$ is of unit length it follows that all the other amplitudes must be zero, and an observation will yield 0s for all qubits in the query register. If f is balanced then the positive and negative contributions to the amplitude for $|0\rangle^{\otimes n}$ cancel, leaving an amplitude of zero, and a measurement must yield a result other than 0 on at least one qubit in the query register. Summarizing, if Alice

measures all 0s then the function is constant; otherwise the function is balanced. The Deutsch–Jozsa algorithm is summarized below.

Algorithm: Deutsch–Jozsa

Inputs: (1) A black box U_f which performs the transformation $|x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$, for $x \in \{0, \ldots, 2^n - 1\}$ and $f(x) \in \{0, 1\}$. It is promised that $f(x)$ is either *constant* for all values of x, or else $f(x)$ is *balanced*, that is, equal to 1 for exactly half of all the possible x, and 0 for the other half.

Outputs: 0 if and only if f is constant.

Runtime: One evaluation of U_f. Always succeeds.

Procedure:

1.	$\|0\rangle^{\otimes n}\|1\rangle$	initialize state
2.	$\rightarrow \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \|x\rangle \left[\frac{\|0\rangle - \|1\rangle}{\sqrt{2}}\right]$	create superposition using Hadamard gates
3.	$\rightarrow \sum_x (-1)^{f(x)} \|x\rangle \left[\frac{\|0\rangle - \|1\rangle}{\sqrt{2}}\right]$	calculate function f using U_f
4.	$\rightarrow \sum_z \sum_x \frac{(-1)^{x \cdot z + f(x)} \|z\rangle}{\sqrt{2^n}} \left[\frac{\|0\rangle - \|1\rangle}{\sqrt{2}}\right]$	perform Hadamard transform
5.	$\rightarrow z$	measure to obtain final output z

We've shown that a quantum computer can solve Deutsch's problem with one evaluation of the function f compared to the classical requirement for $2^n/2 + 1$ evaluations. This appears impressive, but there are several important caveats. First, Deutsch's problem is not an especially important problem; it has no known applications. Second, the comparison between classical and quantum algorithms is in some ways an apples and oranges comparison, as the method for evaluating the function is quite different in the two cases. Third, if Alice is allowed to use a probabilistic classical computer, then by asking Bob to evaluate $f(x)$ for a few randomly chosen x she can very quickly determine with high probability whether f is constant or balanced. This probabilistic scenario is perhaps more realistic than the deterministic scenario we have been considering. Despite these caveats, the Deutsch–Jozsa algorithm contains the seeds for more impressive quantum algorithms, and it is enlightening to attempt to understand the principles behind its operation.

Exercise 1.1: (Probabilistic classical algorithm) Suppose that the problem is not to distinguish between the constant and balanced functions *with certainty*, but rather, with some probability of error $\epsilon < 1/2$. What is the performance of the best classical algorithm for this problem?

1.4.5 Quantum algorithms summarized

The Deutsch–Jozsa algorithm suggests that quantum computers may be capable of solving some computational problems much more efficiently than classical computers. Unfortunately, the problem it solves is of little practical interest. Are there more interesting

problems whose solution may be obtained more efficiently using quantum algorithms? What are the principles underlying such algorithms? What are the ultimate limits of a quantum computer's computational power?

Broadly speaking, there are three classes of quantum algorithms which provide an advantage over known classical algorithms. First, there is the class of algorithms based upon quantum versions of the Fourier transform, a tool which is also widely used in classical algorithms. The Deutsch–Jozsa algorithm is an example of this type of algorithm, as are Shor's algorithms for factoring and discrete logarithm. The second class of algorithms is quantum search algorithms. The third class of algorithms is quantum simulation, whereby a quantum computer is used to simulate a quantum system. We now briefly describe each of these classes of algorithms, and then summarize what is known or suspected about the computational power of quantum computers.

Quantum algorithms based upon the Fourier transform

The discrete Fourier transform is usually described as transforming a set $x_0, \ldots, x_{N-1}$ of N complex numbers into a set of complex numbers $y_0, \ldots, y_{N-1}$ defined by

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i j k/N} x_j \, . \tag{1.52}$$

Of course, this transformation has an enormous number of applications in many branches of science; the Fourier transformed version of a problem is often easier than the original problem, enabling a solution.

The Fourier transform has proved so useful that a beautiful generalized theory of Fourier transforms has been developed which goes beyond the definition (1.52). This general theory involves some technical ideas from the character theory of finite groups, and we will not attempt to describe it here. What is important is that the Hadamard transform used in the Deutsch–Jozsa algorithm is an example of this generalized class of Fourier transforms. Moreover, many of the other important quantum algorithms also involve some type of Fourier transform.

The most important quantum algorithms known, Shor's fast algorithms for factoring and discrete logarithm, are two examples of algorithms based upon the Fourier transform defined in Equation (1.52). The Equation (1.52) does not appear terribly quantum mechanical in the form we have written it. Imagine, however, that we define a linear transformation U on n qubits by its action on computational basis states $|j\rangle$, where $0 \leq j \leq 2^n - 1$,

$$|j\rangle \longrightarrow \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i j k/2^n} |k\rangle \, . \tag{1.53}$$

It can be checked that this transformation is unitary, and in fact can be realized as a quantum circuit. Moreover, if we write out its action on superpositions,

$$\sum_{j=0}^{2^n-1} x_j |j\rangle \longrightarrow \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \left[\sum_{j=0}^{2^n-1} e^{2\pi i j k/2^n} x_j \right] |k\rangle = \sum_{k=0}^{2^n-1} y_k |k\rangle \, , \tag{1.54}$$

we see that it corresponds to a vector notation for the Fourier transform (1.52) for the case $N = 2^n$.

How quickly can we perform the Fourier transform? Classically, the fast Fourier transform takes roughly $N \log(N) = n2^n$ steps to Fourier transform $N = 2^n$ numbers. On a quantum computer, the Fourier transform can be accomplished using about $\log^2(N) = n^2$ steps, an exponential saving! The quantum circuit to do this is explained in Chapter 5.

This result seems to indicate that quantum computers can be used to very quickly compute the Fourier transform of a vector of 2^n complex numbers, which would be fantastically useful in a wide range of applications. However, that is *not* exactly the case; the Fourier transform is being performed on the information 'hidden' in the amplitudes of the quantum state. This information is not directly accessible to measurement. The catch, of course, is that if the output state is measured, it will collapse each qubit into the state $|0\rangle$ or $|1\rangle$, preventing us from learning the transform result y_k directly. This example speaks to the heart of the conundrum of devising a quantum algorithm. On the one hand, we can perform certain calculations on the 2^n amplitudes associated with n qubits far more efficiently than would be possible on a classical computer. But on the other hand, the results of such a calculation are not available to us if we go about it in a straightforward manner. More cleverness is required in order to harness the power of quantum computation.

Fortunately, it does turn out to be possible to utilize the quantum Fourier transform to efficiently solve several problems that are believed to have no efficient solution on a classical computer. These problems include Deutsch's problem, and Shor's algorithms for discrete logarithm and factoring. This line of thought culminated in Kitaev's discovery of a method to solve the *Abelian stabilizer problem*, and the generalization to the *hidden subgroup problem*,

> Let f be a function from a finitely generated group G to a finite set X such that f is constant on the cosets of a subgroup K, and distinct on each coset. Given a quantum black box for performing the unitary transform $U|g\rangle|h\rangle = |g\rangle|h \oplus f(g)\rangle$, for $g \in G$, $h \in X$, and $\oplus$ an appropriately chosen binary operation on X, find a generating set for K.

The Deutsch–Jozsa algorithm, Shor's algorithms, and related 'exponentially fast' quantum algorithms can all be viewed as special cases of this algorithm. The quantum Fourier transform and its applications are described in Chapter 5.

Quantum search algorithms

A completely different class of algorithms is represented by the quantum search algorithm, whose basic principles were discovered by Grover. The quantum search algorithm solves the following problem: Given a search space of size N, and no prior knowledge about the structure of the information in it, we want to find an element of that search space satisfying a known property. How long does it take to find an element satisfying that property? Classically, this problem requires approximately N operations, but the quantum search algorithm allows it to be solved using approximately $\sqrt{N}$ operations.

The quantum search algorithm offers only a quadratic speedup, as opposed to the more impressive exponential speedup offered by algorithms based on the quantum Fourier transform. However, the quantum search algorithm is still of great interest, since searching heuristics have a wider range of application than the problems solved using the quantum Fourier transform, and adaptations of the quantum search algorithm may have utility

for a very wide range of problems. The quantum search algorithm and its applications are described in Chapter 6.

Quantum simulation

Simulating naturally occurring quantum mechanical systems is an obvious candidate for a task at which quantum computers may excel, yet which is believed to be difficult on a classical computer. Classical computers have difficulty simulating general quantum systems for much the same reasons they have difficulty simulating quantum computers – the number of complex numbers needed to describe a quantum system generally grows *exponentially* with the size of the system, rather than linearly, as occurs in classical systems. In general, storing the quantum state of a system with n distinct components takes something like c^n bits of memory on a classical computer, where c is a constant which depends upon details of the system being simulated, and the desired accuracy of the simulation.

By contrast, a quantum computer can perform the simulation using kn qubits, where k is again a constant which depends upon the details of the system being simulated. This allows quantum computers to efficiently perform simulations of quantum mechanical systems that are believed not to be efficiently simulatable on a classical computer. A significant caveat is that even though a quantum computer can simulate many quantum systems far more efficiently than a classical computer, this does not mean that the fast simulation will allow the desired information about the quantum system to be obtained. When measured, a kn qubit simulation will collapse into a definite state, giving only kn bits of information; the c^n bits of 'hidden information' in the wavefunction is not entirely accessible. Thus, a crucial step in making quantum simulations useful is development of systematic means by which desired answers can be efficiently extracted; how to do this is only partially understood.

Despite this caveat, quantum simulation is likely to be an important application of quantum computers. The simulation of quantum systems is an important problem in many fields, notably quantum chemistry, where the computational constraints imposed by classical computers make it difficult to accurately simulate the behavior of even moderately sized molecules, much less the very large molecules that occur in many important biological systems. Obtaining faster and more accurate simulations of such systems may therefore have the welcome effect of enabling advances in other fields in which quantum phenomena are important.

In the future we may discover a physical phenomenon in Nature which cannot be efficiently simulated on a quantum computer. Far from being bad news, this would be wonderful! At the least, it will stimulate us to extend our models of computation to encompass the new phenomenon, and increase the power of our computational models beyond the existing quantum computing model. It also seems likely that very interesting new physical effects will be associated with any such phenomenon!

Another application for quantum simulation is as a general method to obtain insight into other quantum algorithms; for example, in Section 6.2 we explain how the quantum search algorithm can be viewed as the solution to a problem of quantum simulation. By approaching the problem in this fashion it becomes much easier to understand the origin of the quantum search algorithm.

Finally, quantum simulation also gives rise to an interesting and optimistic 'quantum corollary' to Moore's law. Recall that Moore's law states that the power of classical

computers will double once every two years or so, for constant cost. However, suppose we are simulating a quantum system on a classical computer, and want to add a single qubit (or a larger system) to the system being simulated. This doubles or more the memory requirements needed for a classical computer to store a description of the state of the quantum system, with a similar or greater cost in the time needed to simulate the dynamics. The quantum corollary to Moore's law follows from this observation, stating that quantum computers are keeping pace with classical computers provided a *single qubit* is added to the quantum computer every two years. This corollary should not be taken too seriously, as the exact nature of the gain, if any, of quantum computation over classical is not yet clear. Nevertheless, this heuristic statement helps convey why we should be interested in quantum computers, and hopeful that they will one day be able to outperform the most powerful classical computers, at least for some applications.

The power of quantum computation

How powerful are quantum computers? What gives them their power? Nobody yet knows the answers to these questions, despite the suspicions fostered by examples such as factoring, which strongly suggest that quantum computers are more powerful than classical computers. It is still possible that quantum computers are no more powerful than classical computers, in the sense that any problem which can be efficiently solved on a quantum computer can also be efficiently solved on a classical computer. On the other hand, it may eventually be proved that quantum computers are much more powerful than classical computers. We now take a brief look at what is known about the power of quantum computation.

Computational complexity theory is the subject of classifying the difficulty of various computational problems, both classical and quantum, and to understand the power of quantum computers we will first examine some general ideas from computational complexity. The most basic idea is that of a *complexity class*. A complexity class can be thought of as a collection of computational problems, all of which share some common feature with respect to the computational resources needed to solve those problems.

Two of the most important complexity classes go by the names **P** and **NP**. Roughly speaking, **P** is the class of computational problems that can be solved quickly on a classical computer. **NP** is the class of problems which have *solutions* which can be quickly checked on a classical computer. To understand the distinction between **P** and **NP**, consider the problem of finding the prime factors of an integer, n. So far as is known there is no fast way of solving this problem on a classical computer, which suggests that the problem is not in **P**. On the other hand, if somebody tells you that some number p is a factor of n, then we can quickly check that this is correct by dividing p into n, so factoring is a problem in **NP**.

It is clear that **P** is a subset of **NP**, since the ability to solve a problem implies the ability to check potential solutions. What is not so clear is whether or not there are problems in **NP** that are not in **P**. Perhaps the most important unsolved problem in theoretical computer science is to determine whether these two classes are different:

$$\mathbf{P} \stackrel{?}{\neq} \mathbf{NP}. \tag{1.55}$$

Most researchers believe that **NP** contains problems that are not in **P**. In particular, there is an important subclass of the **NP** problems, the **NP**-complete problems, that are

of especial importance for two reasons. First, there are thousands of problems, many highly important, that are known to be **NP**-complete. Second, any given **NP**-complete problem is in some sense 'at least as hard' as all other problems in **NP**. More precisely, an algorithm to solve a specific **NP**-complete problem can be adapted to solve any other problem in **NP**, with a small overhead. In particular, if $\mathbf{P} \neq \mathbf{NP}$, then it will follow that no **NP**-complete problem can be efficiently solved on a classical computer.

It is not known whether quantum computers can be used to quickly solve all the problems in **NP**, despite the fact that they can be used to solve some problems – like factoring – which are believed by many people to be in **NP** but not in **P**. (Note that factoring is not known to be **NP**-complete, otherwise we would already know how to efficiently solve all problems in **NP** using quantum computers.) It would certainly be very exciting if it were possible to solve all the problems in **NP** efficiently on a quantum computer. There is a very interesting negative result known in this direction which rules out using a simple variant of quantum parallelism to solve all the problems in **NP**. Specifically, one approach to the problem of solving problems in **NP** on a quantum computer is to try to use some form of quantum parallelism to search in parallel through all the possible solutions to the problem. In Section 6.6 we will show that no approach based upon such a search-based methodology can yield an efficient solution to all the problems in **NP**. While it is disappointing that this approach fails, it does not rule out that some deeper structure exists in the problems in **NP** that will allow them all to be solved quickly using a quantum computer.

P and **NP** are just two of a plethora of complexity classes that have been defined. Another important complexity class is **PSPACE**. Roughly speaking, **PSPACE** consists of those problems which can be solved using resources which are few in spatial size (that is, the computer is 'small'), but not necessarily in time ('long' computations are fine). **PSPACE** is believed to be strictly larger than both **P** and **NP** although, again, this has never been proved. Finally, the complexity class **BPP** is the class of problems that can be solved using randomized algorithms in polynomial time, if a bounded probability of error (say $1/4$) is allowed in the solution to the problem. **BPP** is widely regarded as being, even more so than **P**, the class of problems which should be considered efficiently soluble on a classical computer. We have elected to concentrate here on **P** rather than **BPP** because **P** has been studied in more depth, however many similar ideas and conclusions arise in connection with **BPP**.

What of quantum complexity classes? We can define **BQP** to be the class of all computational problems which can be solved efficiently on a quantum computer, where a bounded probability of error is allowed. (Strictly speaking this makes **BQP** more analogous to the classical complexity class **BPP** than to **P**, however we will ignore this subtlety for the purposes of the present discussion, and treat it as the analogue of **P**.) Exactly where **BQP** fits with respect to **P**, **NP** and **PSPACE** is as yet unknown. What is known is that quantum computers can solve all the problems in **P** efficiently, but that there are no problems outside of **PSPACE** which they can solve efficiently. Therefore, **BQP** lies somewhere between **P** and **PSPACE**, as illustrated in Figure 1.21. An important implication is that if it is proved that quantum computers are strictly more powerful than classical computers, then it will follow that **P** is not equal to **PSPACE**. Proving this latter result has been attempted without success by many computer scientists, suggesting that it may be non-trivial to prove that quantum computers are more powerful than classical computers, despite much evidence in favor of this proposition.

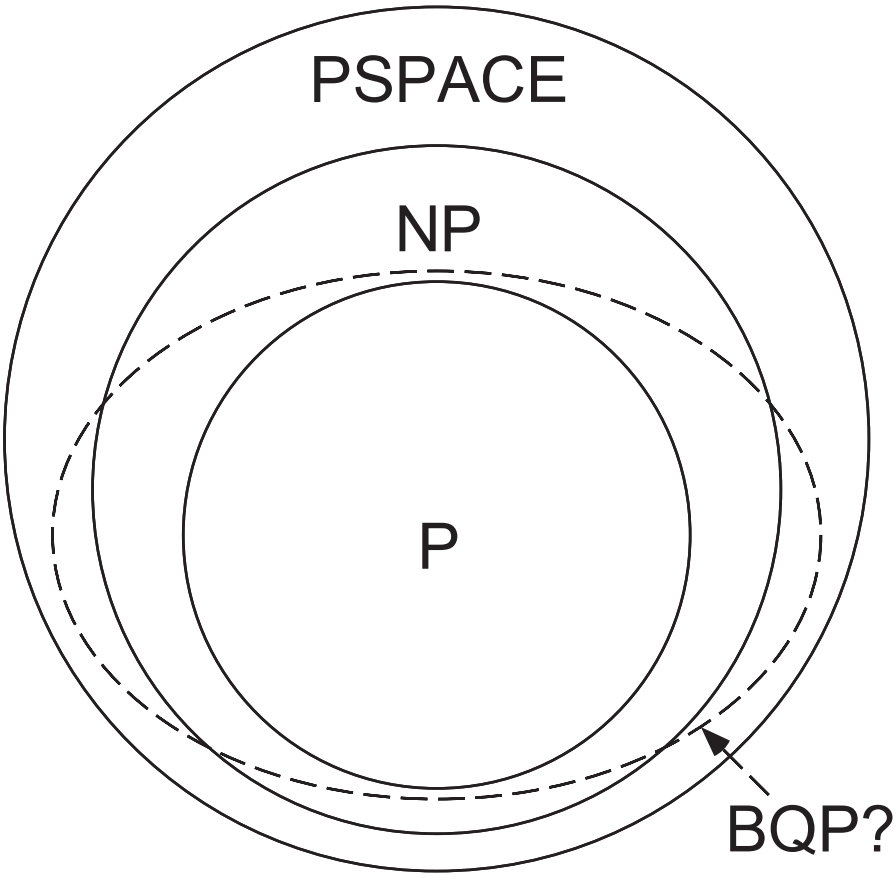


Figure 1.21. The relationship between classical and quantum complexity classes. Quantum computers can quickly solve any problem in **P**, and it is known that they can't solve problems outside of **PSPACE** quickly. Where quantum computers fit between **P** and **PSPACE** is not known, in part because we don't even know whether **PSPACE** is bigger than **P**!

We won't speculate further on the ultimate power of quantum computation now, preferring to wait until after we have better understood the principles on which fast quantum algorithms are based, a topic which occupies us for most of Part II of this book. What is already clear is that the *theory* of quantum computation poses interesting and significant challenges to the traditional notions of computation. What makes this an important challenge is that the theoretical model of quantum computation is believed to be *experimentally* realizable, because – to the best of our knowledge – this theory is consistent with the way Nature works. If this were not so then quantum computation would be just another mathematical curiosity.

1.5 Experimental quantum information processing

Quantum computation and quantum information is a wonderful theoretical discovery, but its central concepts, such as superpositions and entanglement, run counter to the intuition we garner from the everyday world around us. What evidence do we have that these ideas truly describe how Nature operates? Will the realization of large-scale quantum

computers be experimentally feasible? Or might there be some principle of physics which fundamentally prohibits their eventual scaling? In the next two sections we address these questions. We begin with a review of the famous 'Stern–Gerlach' experiment, which provides evidence for the existence of qubits in Nature. We then widen our scope, addressing the broader problem of how to build practical quantum information processing systems.

1.5.1 The Stern–Gerlach experiment

The qubit is a fundamental element for quantum computation and quantum information. How do we know that systems with the properties of qubits exist in Nature? At the time of writing there is an enormous amount of evidence that this is so, but in the early days of quantum mechanics the qubit structure was not at all obvious, and people struggled with phenomena that we may now understand in terms of qubits, that is, in terms of two level quantum systems.

A decisive (and very famous) early experiment indicating the qubit structure was conceived by Stern in 1921 and performed with Gerlach in 1922 in Frankfurt. In the original Stern–Gerlach experiment, hot atoms were 'beamed' from an oven through a magnetic field which caused the atoms to be deflected, and then the position of each atom was recorded, as illustrated in Figure 1.22. The original experiment was done with silver atoms, which have a complicated structure that obscures the effects we are discussing. What we describe below actually follows a 1927 experiment done using hydrogen atoms. The same basic effect is observed, but with hydrogen atoms the discussion is easier to follow. Keep in mind, though, that this privilege wasn't available to people in the early 1920s, and they had to be very ingenious to think up explanations for the more complicated effects they observed.

Hydrogen atoms contain a proton and an orbiting electron. You can think of this electron as a little 'electric current' around the proton. This electric current causes the atom to have a magnetic field; each atom has what physicists call a 'magnetic dipole moment'. As a result each atom behaves like a little bar magnet with an axis corresponding to the axis the electron is spinning around. Throwing little bar magnets through a magnetic field causes the magnets to be deflected by the field, and we expect to see a similar deflection of atoms in the Stern–Gerlach experiment.

How the atom is deflected depends upon both the atom's magnetic dipole moment – the axis the electron is spinning around – and the magnetic field generated by the Stern–Gerlach device. We won't go through the details, but suffice to say that by constructing the Stern–Gerlach device appropriately, we can cause the atom to be deflected by an amount that depends upon the $\hat{z}$ component of the atom's magnetic dipole moment, where $\hat{z}$ is some fixed external axis.

Two major surprises emerge when this experiment is performed. First, since the hot atoms exiting the oven would naturally be expected to have their dipoles oriented randomly in every direction, it would follow that there would be a continuous distribution of atoms seen at all angles exiting from the Stern–Gerlach device. Instead, what is seen is atoms emerging from a *discrete* set of angles. Physicists were able to explain this by assuming that the magnetic dipole moment of the atoms is *quantized*, that is, comes in discrete multiples of some fundamental amount.

This observation of quantization in the Stern–Gerlach experiment was surprising to physicists of the 1920s, but not completely astonishing because evidence for quantization

effects in other systems was becoming widespread at that time. What was truly surprising was the *number* of peaks seen in the experiment. The hydrogen atoms being used were such that they should have had *zero* magnetic dipole moment. Classically, this is surprising in itself, since it corresponds to no orbital motion of the electron, but based on what was known of quantum mechanics at that time this was an acceptable notion. Since the hydrogen atoms would therefore have zero magnetic moment, it was expected that only one beam of atoms would be seen, and this beam would not be deflected by the magnetic field. Instead, two beams were seen, one deflected up by the magnetic field, and the other deflected down!

This puzzling doubling was explained after considerable effort by positing that the electron in the hydrogen atom has associated with it a quantity called *spin*. This spin is not in any way associated to the usual rotational motion of the electron around the proton; it is an entirely new quantity to be associated with an electron. The great physicist Heisenberg labeled the idea 'brave' at the time it was suggested, and it is a brave idea, since it introduces an essentially new physical quantity into Nature. The spin of the electron is posited to make an *extra* contribution to the magnetic dipole moment of a hydrogen atom, in addition to the contribution due to the rotational motion of the electron.

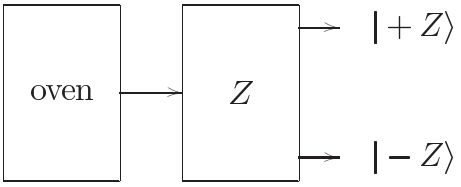


Figure 1.22. Abstract schematic of the Stern–Gerlach experiment. Hot hydrogen atoms are beamed from an oven through a magnetic field, causing a deflection either up ($|+Z\rangle$) or down ($|-Z\rangle$).

What is the proper description of the spin of the electron? As a first guess, we might hypothesize that the spin is specified by a single bit, telling the hydrogen atom to go up or down. Additional experimental results provide further useful information to determine if this guess needs refinement or replacement. Let's represent the original Stern–Gerlach apparatus as shown in Figure 1.22. Its outputs are two beams of atoms, which we shall call $|+Z\rangle$ and $|-Z\rangle$. (We're using suggestive notation which looks quantum mechanical, but of course you're free to use whatever notation you prefer.) Now suppose we cascade two Stern–Gerlach apparatus together, as shown in Figure 1.23. We arrange it so that the second apparatus is *tipped sideways*, so the magnetic field deflects atoms along the $\hat{x}$ axis. In our thought-experiment we'll block off the $|-Z\rangle$ output from the first Stern–Gerlach apparatus, while the $|+Z\rangle$ output is sent through a second apparatus oriented along the $\hat{x}$ axis. A detector is placed at the final output to measure the distribution of atoms along the $\hat{x}$ axis.

A classical magnetic dipole pointed in the $+\hat{z}$ direction has no net magnetic moment in the $\hat{x}$ direction, so we might expect that the final output would have one central peak. However, experimentally it is observed that there are two peaks of equal intensity! So perhaps these atoms are peculiar, and have definite magnetic moments along each axis, independently. That is, maybe each atom passing through the second apparatus can be

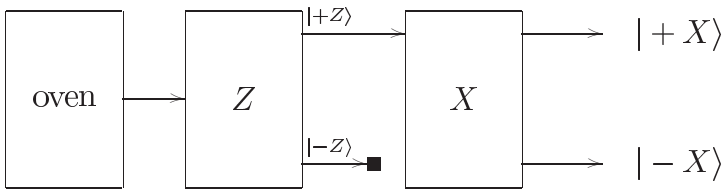


Figure 1.23. Cascaded Stern–Gerlach measurements.

described as being in a state we might write as $|+Z\rangle|+X\rangle$ or $|+Z\rangle|-X\rangle$, to indicate the two values for spin that might be observed.

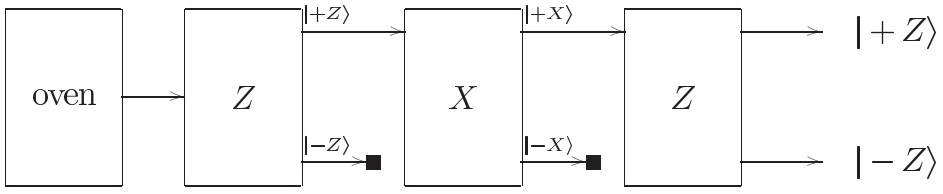


Figure 1.24. Three stage cascaded Stern–Gerlach measurements.

Another experiment, shown in Figure 1.24, can test this hypothesis by sending one beam of the previous output through a second $\hat{z}$ oriented Stern–Gerlach apparatus. If the atoms had retained their $|+Z\rangle$ orientation, then the output would be expected to have only one peak, at the $|+Z\rangle$ output. However, again *two* beams are observed at the final output, of equal intensity. Thus, the conclusion would seem to be that contrary to classical expectations, a $|+Z\rangle$ state consists of equal portions of $|+X\rangle$ and $|-X\rangle$ states, and a $|+X\rangle$ state consists of equal portions of $|+Z\rangle$ and $|-Z\rangle$ states. Similar conclusions can be reached if the Stern–Gerlach apparatus is aligned along some other axis, like the $\hat{y}$ axis.

The qubit model provides a simple explanation of this experimentally observed behavior. Let $|0\rangle$ and $|1\rangle$ be the states of a qubit, and make the assignments

$$|+Z\rangle \leftarrow |0\rangle \tag{1.56}$$

$$|-Z\rangle \leftarrow |1\rangle \tag{1.57}$$

$$|+X\rangle \leftarrow (|0\rangle + |1\rangle)/\sqrt{2}. \tag{1.58}$$

$$|-X\rangle \leftarrow (|0\rangle - |1\rangle)/\sqrt{2} \tag{1.59}$$

Then the results of the cascaded Stern–Gerlach experiment can be explained by assuming that the $\hat{z}$ Stern–Gerlach apparatus measures the spin (that is, the qubit) in the computational basis $|0\rangle, |1\rangle$, and the $\hat{x}$ Stern–Gerlach apparatus measures the spin with respect to the basis $(|0\rangle + |1\rangle)/\sqrt{2}, (|0\rangle - |1\rangle)/\sqrt{2}$. For example, in the cascaded $\hat{z}$-$\hat{x}$-$\hat{z}$ experiment, if we assume that the spins are in the state $|+Z\rangle = |0\rangle = (|+X\rangle + |-X\rangle)/\sqrt{2}$ after exiting the first Stern–Gerlach experiment, then the probability for obtaining $|+X\rangle$ out of the second apparatus is $1/2$, and the probability for $|-X\rangle$ is $1/2$. Similarly, the probability for obtaining $|+Z\rangle$ out of the third apparatus is $1/2$. A qubit model thus properly predicts results from this type of cascaded Stern–Gerlach experiment.

This example demonstrates how qubits could be a believable way of modeling systems in Nature. Of course it doesn't establish beyond all doubt that the qubit model is the correct way of understanding electron spin – far more experimental corroboration is required. Nevertheless, because of many experiments like these, we now believe that electron spin is best described by the qubit model. What is more, we believe that the qubit model (and generalizations of it to higher dimensions; quantum mechanics, in other words) is capable of describing *every* physical system. We now turn to the question of what systems are especially well adapted to quantum information processing.

1.5.2 Prospects for practical quantum information processing

Building quantum information processing devices is a great challenge for scientists and engineers of the third millennium. Will we rise to meet this challenge? Is it possible at all? Is it worth attempting? If so, how might the feat be accomplished? These are difficult and important questions, to which we essay brief answers in this section, to be expanded upon throughout the book.

The most fundamental question is whether there is any point of principle that prohibits us from doing one or more forms of quantum information processing? Two possible obstructions suggest themselves: that noise may place a fundamental barrier to useful quantum information processing; or that quantum mechanics may fail to be correct.

Noise is without a doubt a significant obstruction to the development of practical quantum information processing devices. Is it a *fundamentally irremovable* obstruction that will forever prevent the development of large-scale quantum information processing devices? The theory of quantum error-correcting codes strongly suggests that while quantum noise is a practical problem that needs to be addressed, it does not present a fundamental problem of *principle*. In particular, there is a *threshold theorem* for quantum computation, which states, roughly speaking, that provided the level of noise in a quantum computer can be reduced below a certain constant 'threshold' value, quantum error-correcting codes can be used to push it down even further, essentially *ad infinitum*, for a small overhead in the complexity of the computation. The threshold theorem makes some broad assumptions about the nature and magnitude of the noise occurring in a quantum computer, and the architecture available for performing quantum computation; however, provided those assumptions are satisfied, the effects of noise can be made essentially negligible for quantum information processing. Chapters 8, 10 and 12 discuss quantum noise, quantum error-correction and the threshold theorem in detail.

A second possibility that may preclude quantum information processing is if quantum mechanics is incorrect. Indeed, probing the validity of quantum mechanics (both relativistic and non-relativistic) is one reason for being interested in building quantum information processing devices. Never before have we explored a regime of Nature in which complete control has been obtained over large-scale quantum systems, and perhaps Nature may reveal some new surprises in this regime which are not adequately explained by quantum mechanics. If this occurs, it will be a momentous discovery in the history of science, and can be expected to have considerable consequences in other areas of science and technology, as did the discovery of quantum mechanics. Such a discovery might also impact quantum computation and quantum information; however, whether the impact would enhance, detract or not affect the power of quantum information processing cannot be predicted in advance. Until and unless such effects are found we have no way of knowing how they might affect information processing, so for the remainder of this book

we go with all the evidence to date and assume that quantum mechanics is a complete and correct description of the world.

Given that there is no fundamental obstacle to building quantum information processing devices, why should we invest enormous amounts of time and money in the attempt to do so? We have already discussed several reasons for wanting to do so: practical applications such as quantum cryptography and the factoring of large composite numbers; and the desire to obtain fundamental insights into Nature and into information processing.

These are good reasons, and justify a considerable investment of time and money in the effort to build quantum information processing devices. However, it is fair to say that a clearer picture of the relative power of quantum and classical information processing is needed in order to assess their relative merits. To obtain such a picture requires further theoretical work on the foundations of quantum computation and quantum information. Of particular interest is a decisive answer to the question 'Are quantum computers more powerful than classical computers?' Even if the answer to such a question eludes us for the time being, it would be useful to have a clear path of interesting applications at varying levels of complexity to aid researchers aiming to experimentally realize quantum information processing. Historically, the advance of technology is often hastened by the use of short- to medium-term incentives as a stepping-stone to long-term goals. Consider that microprocessors were initially used as controllers for elevators and other simple devices, before graduating to be the fundamental component in personal computers (and then on to who-knows-what). Below we sketch out a path of short- to medium-term goals for people interested in achieving the long-term goal of large-scale quantum information processing.

Surprisingly many small-scale applications of quantum computation and quantum information are known. Not all are as flashy as cousins like the quantum factoring algorithm, but the relative ease of implementing small-scale applications makes them extremely important as medium-term goals in themselves.

Quantum state tomography and quantum process tomography are two elementary processes whose perfection is of great importance to quantum computation and quantum information, as well as being of independent interest in their own right. Quantum state tomography is a method for determining the quantum state of a system. To do this, it has to overcome the 'hidden' nature of the quantum state – remember, the state can't be directly determined by a measurement – by performing repeated preparations of the same quantum state, which is then measured in different ways in order to build up a complete description of the quantum state. Quantum process tomography is a more ambitious (but closely related) procedure to completely characterize the *dynamics* of a quantum system. Quantum process tomography can, for example, be used to characterize the performance of an alleged quantum gate or quantum communications channel, or to determine the types and magnitudes of different noise processes in a system. Beside obvious applications to quantum computation and quantum information, quantum process tomography can be expected to have significant applications as a diagnostic tool to aid in the evaluation and improvement of primitive operations in any field of science and technology where quantum effects are important. Quantum state tomography and quantum process tomography are described in more detail in Chapter 8.

Various small-scale communications primitives are also of great interest. We have already mentioned quantum cryptography and quantum teleportation. The former is likely to be useful in practical applications involving the distribution of a small amount of key

material that needs to be highly secure. The uses of quantum teleportation are perhaps more open to question. We will see in Chapter 12 that teleportation may be an extremely useful primitive for transmitting quantum states between distant nodes in a network, in the presence of noise. The idea is to focus one's efforts on distributing EPR pairs between the nodes that wish to communicate. The EPR pairs may be corrupted during communication, but special 'entanglement distillation' protocols can then be used to 'clean up' the EPR pairs, enabling them to be used to teleport quantum states from one location to another. In fact, procotols based upon entanglement distillation and teleportation offer performance superior to more conventional quantum error-correction techniques in enabling noise free communication of qubits.

What of the medium-scale? A promising medium-scale application of quantum information processing is to the simulation of quantum systems. To simulate a quantum system containing even a few dozen 'qubits' (or the equivalent in terms of some other basic system) strains the resources of even the largest supercomputers. A simple calculation is instructive. Suppose we have a system containing 50 qubits. To describe the state of such a system requires $2^{50} \approx 10^{15}$ complex amplitudes. If the amplitudes are stored to 128 bits of precision, then it requires 256 bits or 32 bytes in order to store each amplitude, for a total of 32×10^{15} bytes of information, or about 32 thousand terabytes of information, well beyond the capacity of existing computers, and corresponding to about the storage capacity that might be expected to appear in supercomputers during the second decade of the twenty-first century, presuming that Moore's law continues on schedule. 90 qubits at the same level of precision requires 32×10^{27} bytes, which, even if implemented using single atoms to represent bits, would require kilograms (or more) of matter.

How useful will quantum simulations be? It seems likely that conventional methods will still be used to determine elementary properties of materials, such as bond strengths and basic spectroscopic properties. However, once the basic properties are well understood, it seems likely that quantum simulation will be of great utility as a laboratory for the design and testing of properties of novel molecules. In a conventional laboratory setup, many different types of 'hardware' – chemicals, detectors, and so on – may be required to test a wide variety of possible designs for a molecule. On a quantum computer, these different types of hardware can all be simulated in software, which is likely to be much less expensive and much faster. Of course, final design and testing must be performed with real physical systems; however, quantum computers may enable a much larger range of potential designs to be explored and evaluated *en route* to a better final design. It is interesting to note that such *ab initio* calculations to aid in the design of new molecules have been attempted on classical computers; however, they have met with limited success due to the enormous computational resources needed to simulate quantum mechanics on a classical computer. Quantum computers should be able to do much better in the relatively near future.

What of large-scale applications? Aside from scaling up applications like quantum simulation and quantum cryptography, relatively few large-scale applications are known: the factoring of large numbers, taking discrete logarithms, and quantum searching. Interest in the first two of these derives mainly from the *negative* effect they would have of limiting the viability of existing public key cryptographic systems. (They might also be of substantial practical interest to mathematicians interested in these problems simply for their own sake.) So it does not seem likely that factoring and discrete logarithm

will be all that important as applications for the long run. Quantum searching may be of tremendous use because of the wide utility of the search heuristic, and we discuss some possible applications in Chapter 6. What would really be superb are many more large-scale applications of quantum information processing. This is a great goal for the future!

Given a path of potential applications for quantum information processing, how can it be achieved in real physical systems? At the small scale of a few qubits there are already several working proposals for quantum information processing devices. Perhaps the easiest to realize are based upon *optical* techniques, that is, electromagnetic radiation. Simple devices like mirrors and beamsplitters can be used to do elementary manipulations of photons. Interestingly, a major difficulty has been producing single photons on demand; experimentalists have instead opted to use schemes which produce single photons 'every now and then', at random, and wait for such an event to occur. Quantum cryptography, superdense coding, and quantum teleportation have all been realized using such optical techniques. A major advantage of the optical techniques is that photons tend to be highly stable carriers of quantum mechanical information. A major disadvantage is that photons don't directly interact with one another. Instead, the interaction has to be mediated by something else, like an atom, which introduces additional noise and complications into the experiment. An *effective* interaction between two photons is set up, which essentially works in two steps: photon number one interacts with the atom, which in turn interacts with the second photon, causing an overall interaction between the two photons.

An alternative scheme is based upon methods for trapping different types of atom: there is the *ion trap*, in which a small number of charged atoms are trapped in a confined space; and *neutral atom traps*, for trapping uncharged atoms in a confined space. Quantum information processing schemes based upon atom traps use the atoms to store qubits. Electromagnetic radiation also shows up in these schemes, but in a rather different way than in what we referred to as the 'optical' approach to quantum information processing. In these schemes, photons are used to manipulate the information stored in the atoms themselves, rather than as the place the information is stored. Single qubit quantum gates can be performed by applying appropriate pulses of electromagnetic radiation to individual atoms. Neighboring atoms can interact with one another via (for example) dipole forces that enable quantum gates to be accomplished. Moreover, the exact nature of the interaction between neighboring atoms can be modified by applying appropriate pulses of electromagnetic radiation to the atoms, giving the experimentalist control over what gates are performed in the system. Finally, quantum measurement can be accomplished in these systems using the long established *quantum jumps* technique, which implements with superb accuracy the measurements in the computational basis used for quantum computation.

Another class of quantum information processing schemes is based upon *Nuclear Magnetic Resonance*, often known by its initials, NMR. These schemes store quantum information in the *nuclear spin* of atoms in a molecule, and manipulate that information using electromagnetic radiation. Such schemes pose special difficulties, because in NMR it is not possible to directly access individual nuclei. Instead, a huge number (typically around 10^{15}) of essentially identical molecules are stored in solution. Electromagnetic pulses are applied to the sample, causing each molecule to respond in roughly the same way. You should think of each molecule as being an independent computer, and the sample as containing a huge number of computers all running in parallel (classically).

NMR quantum information processing faces three special difficulties that make it rather different from other quantum information processing schemes. First, the molecules are typically prepared by letting them equilibrate at room temperature, which is so much higher than typical spin flip energies that the spins become nearly completely randomly oriented. This fact makes the initial state rather more 'noisy' than is desirable for quantum information processing. How this noise may be overcome is an interesting story that we tell in Chapter 7. A second problem is that the class of measurements that may be performed in NMR falls well short of the most general measurements we would like to perform in quantum information processing. Nevertheless, for many instances of quantum information processing the class of measurements allowed in NMR is sufficient. Third, because molecules cannot be individually addressed in NMR you might ask how it is that individual qubits can be manipulated in an appropriate way. Fortunately, different nuclei in the molecule can have different properties that allow them to be individually addressed – or at least addressed at a sufficiently fine-grained scale to allow the operations essential for quantum computation.

Many of the elements required to perform large-scale quantum information processing can be found in existing proposals: superb state preparation and quantum measurements can be performed on a small number of qubits in the ion trap; superb dynamics can be performed in small molecules using NMR; fabrication technology in solid state systems allows designs to be scaled up tremendously. A single system having all these elements would be a long way down the road to a dream quantum computer. Unfortunately, all these systems are very different, and we are many, many years from having large-scale quantum computers. However, we believe that the existence of all these properties in existing (albeit different) systems does bode well for the long-term existence of large-scale quantum information processors. Furthermore, it suggests that there is a great deal of merit to pursuing *hybrid* designs which attempt to marry the best features of two or more existing technologies. For example, there is much work being done on trapping atoms inside *electromagnetic cavities*. This enables flexible manipulation of the atom inside the cavity via optical techniques, and makes possible real-time feedback control of single atoms in ways unavailable in conventional atom traps.

To conclude, note that it is important not to assess quantum information processing as though it were just another technology for information processing. For example, it is tempting to dismiss quantum computation as yet another technological fad in the evolution of the computer that will pass in time, much as other fads have passed – for example, the 'bubble memories' widely touted as the next big thing in memory during the early 1980s. This is a mistake, since quantum computation is an *abstract paradigm* for information processing that may have many *different* implementations in technology. One can compare two different proposals for quantum computing as regards their technological merits – it makes sense to compare a 'good' proposal to a 'bad' proposal – however even a very poor proposal for a quantum computer is of a different qualitative nature from a superb design for a classical computer.

1.6 Quantum information

The term 'quantum information' is used in two distinct ways in the field of quantum computation and quantum information. The first usage is as a broad catch-all for all manner of operations that might be interpreted as related to information processing

using quantum mechanics. This use encompasses subjects such as quantum computation, quantum teleportation, the no-cloning theorem, and virtually all other topics in this book.

The second use of 'quantum information' is much more specialized: it refers to the study of *elementary* quantum information processing tasks. It does not typically include, for example, quantum algorithm design, since the details of specific quantum algorithms are beyond the scope of 'elementary'. To avoid confusion we will use the term 'quantum information theory' to refer to this more specialized field, in parallel with the widely used term '(classical) information theory' to describe the corresponding classical field. Of course, the term 'quantum information theory' has a drawback of its own – it might be seen as implying that theoretical considerations are all that matter! Of course, this is not the case, and experimental demonstration of the elementary processes studied by quantum information theory is of great interest.

The purpose of this section is to introduce the basic ideas of quantum information theory. Even with the restriction to elementary quantum information processing tasks, quantum information theory may look like a disordered zoo to the beginner, with many apparently unrelated subjects falling under the 'quantum information theory' rubric. In part, that's because the subject is still under development, and it's not yet clear how all the pieces fit together. However, we can identify a few fundamental goals uniting work on quantum information theory:

(1) **Identify elementary classes of static resources in quantum mechanics**. An example is the qubit. Another example is the *bit*; classical physics arises as a special case of quantum physics, so it should not be surprising that elementary static resources appearing in classical information theory should also be of great relevance in quantum information theory. Yet another example of an elementary class of static resources is a Bell state shared between two distant parties.
(2) **Identify elementary classes of dynamical processes in quantum mechanics**. A simple example is *memory*, the ability to store a quantum state over some period of time. Less trivial processes are quantum information transmission between two parties, Alice and Bob; copying (or trying to copy) a quantum state, and the process of protecting quantum information processing against the effects of noise.
(3) **Quantify resource tradeoffs incurred performing elementary dynamical processes.** For example, what are the minimal resources required to reliably transfer quantum information between two parties using a noisy communications channel?

Similar goals define classical information theory; however, quantum information theory is broader in scope than classical information theory, for quantum information theory includes all the static and dynamic elements of classical information theory, as well as *additional* static and dynamic elements.

The remainder of this section describes some examples of questions studied by quantum information theory, in each case emphasizing the fundamental static and dynamic elements under consideration, and the resource tradeoffs being considered. We begin with an example that will appear quite familiar to classical information theorists: the problem of sending classical information through a quantum channel. We then begin to branch out and explore some of the new static and dynamic processes present in quantum mechanics, such as quantum error-correction, the problem of distinguishing quantum states, and entanglement transformation. The chapter concludes with some reflections on how the

tools of quantum information theory can be applied elsewhere in quantum computation and quantum information.

1.6.1 Quantum information theory: example problems

Classical information through quantum channels

The fundamental results of classical information theory are Shannon's *noiseless channel coding theorem* and Shannon's *noisy channel coding theorem*. The noiseless channel coding theorem quantifies how many bits are required to store information being emitted by a source of information, while the noisy channel coding theorem quantifies how much information can be reliably transmitted through a noisy communications channel.

What do we mean by an *information source*? Defining this notion is a fundamental problem of classical and quantum information theory, one we'll re-examine several times. For now, let's go with a provisional definition: a classical information source is described by a set of probabilities p_j, $j = 1, 2, \ldots, d$. Each use of the source results in the 'letter' j being emitted, chosen at random with probability p_j, independently for each use of the source. For instance, if the source were of English text, then the numbers j might correspond to letters of the alphabet and punctuation, with the probabilities p_j giving the relative frequencies with which the different letters appear in regular English text. Although it is not true that the letters in English appear in an independent fashion, for our purposes it will be a good enough approximation.

Regular English text includes a considerable amount of redundancy, and it is possible to exploit that redundancy to *compress* the text. For example, the letter 'e' occurs much more frequently in regular English text than does the letter 'z'. A good scheme for compressing English text will therefore represent the letter 'e' using fewer bits of information than it uses to represent 'z'. Shannon's noiseless channel coding theorem quantifies exactly how well such a compression scheme can be made to work. More precisely, the noiseless channel coding theorem tells us that a classical source described by probabilities p_j can be compressed so that on average each use of the source can be represented using $H(p_j)$ bits of information, where $H(p_j) \equiv -\sum_j p_j \log(p_j)$ is a function of the source probability distribution known as the *Shannon entropy*. Moreover, the noiseless channel coding theorem tells us that to attempt to represent the source using fewer bits than this will result in a high probability of error when the information is decompressed. (Shannon's noiseless channel coding theorem is discussed in much greater detail in Chapter 12.)

Shannon's noiseless coding theorem provides a good example where the goals of information theory listed earlier are all met. Two static resources are identified (goal number 1): the bit and the information source. A two-stage dynamic process is identified (goal 2), compressing an information source, and then decompressing to recover the information source. Finally a quantitative criterion for determining the resources consumed (goal 3) by an optimal data compression scheme is found.

Shannon's second major result, the noisy channel coding theorem, quantifies the amount of information that can be reliably transmitted through a noisy channel. In particular, suppose we wish to transfer the information being produced by some information source to another location through a noisy channel. That location may be at another point in space, or at another point in time – the latter is the problem of storing information in the presence of noise. The idea in both instances is to encode the information being produced using error-correcting codes, so that any noise introduced by the channel can be corrected at the other end of the channel. The way error-correcting codes achieve this

is by introducing enough redundancy into the information sent through the channel so that even after some of the information has been corrupted it is still possible to recover the original message. For example, suppose the noisy channel is for the transmission of single bits, and the noise in the channel is such that to achieve reliable transmission each bit produced by the source must be encoded using two bits before being sent through the channel. We say that such a channel has a *capacity* of half a bit, since each use of the channel can be used to reliably convey roughly half a bit of information. Shannon's noisy channel coding theorem provides a general procedure for calculating the capacity of an arbitrary noisy channel.

Shannon's noisy channel coding theorem also achieves the three goals of information theory we stated earlier. Two types of static resources are involved (goal 1), the information source, and the bits being sent through the channel. Three dynamical processes are involved (goal 2). The primary process is the noise in the channel. To combat this noise we perform the dual processes of encoding and decoding the state in an error-correcting code. For a fixed noise model, Shannon's theorem tells us how much redundancy must be introduced by an optimal error-correction scheme if reliable information transmission is to be achieved (goal 3).

For both the noiseless and noisy channel coding theorems Shannon restricted himself to storing the output from an information source in classical systems – bits and the like. A natural question for quantum information theory is what happens if the storage medium is changed so that classical information is transmitted using quantum states as the medium. For example, it may be that Alice wishes to compress some classical information produced by an information source, transmitting the compressed information to Bob, who then decompresses it. If the medium used to store the compressed information is a quantum state, then Shannon's noiseless channel coding theorem cannot be used to determine the optimal compression and decompression scheme. One might wonder, for example, if using qubits allows a better compression rate than is possible classically. We'll study this question in Chapter 12, and prove that, in fact, qubits do not allow any significant saving in the amount of communication required to transmit information over a noiseless channel.

Naturally, the next step is to investigate the problem of transmitting classical information through a *noisy* quantum channel. Ideally, what we'd like is a result that quantifies the *capacity* of such a channel for the transmission of information. Evaluating the capacity is a very tricky job for several reasons. Quantum mechanics gives us a huge variety of noise models, since it takes place in a continuous space, and it is not at all obvious how to adapt classical error-correction techniques to combat the noise. Might it be advantageous, for example, to encode the classical information using *entangled* states, which are then transmitted one piece at a time through the noisy channel? Or perhaps it will be advantageous to decode using entangled measurements? In Chapter 12 we'll prove the *HSW (Holevo–Schumacher–Westmoreland) theorem*, which provides a lower bound on the capacity of such a channel. Indeed, it is widely believed that the HSW theorem provides an exact evaluation of the capacity, although a complete proof of this is not yet known! What remains at issue is whether or not encoding using entangled states can be used to raise the capacity beyond the lower bound provided by the HSW theorem. All evidence to date suggests that this doesn't help raise the capacity, but it is still a fascinating open problem of quantum information theory to determine the truth or falsity of this conjecture.

Quantum information through quantum channels

Classical information is, of course, not the only static resource available in quantum mechanics. Quantum states themselves are a natural static resource, even more natural than classical information. Let's look at a *different* quantum analogue of Shannon's coding theorems, this time involving the compression and decompression of quantum states.

To begin, we need to define some quantum notion of an information source, analogous to the classical definition of an information source. As in the classical case, there are several different ways of doing this, but for the sake of definiteness let's make the provisional definition that a quantum source is described by a set of probabilities p_j and corresponding quantum states $|\psi_j\rangle$. Each use of the source produces a state $|\psi_j\rangle$ with probability p_j, with different uses of the source being independent of one another.

Is it possible to compress the output from such a quantum mechanical source? Consider the case of a qubit source which outputs the state $|0\rangle$ with probability p and the state $|1\rangle$ with probability $1-p$. This is essentially the same as a classical source emitting single bits, either 0 with probability p, or 1 with probability $1-p$, so it is not surprising that similar techniques can be used to compress the source so that only $H(p, 1-p)$ qubits are required to store the compressed source, where $H(\cdot)$ is again the Shannon entropy function.

What if the source had instead been producing the state $|0\rangle$ with probability p, and the state $(|0\rangle + |1\rangle)/\sqrt{2}$ with probability $1-p$? The standard techniques of classical data compression no longer apply, since in general it is not possible for us to distinguish the states $|0\rangle$ and $(|0\rangle + |1\rangle)/\sqrt{2}$. Might it still be possible to perform some type of compression operation?

It turns out that a type of compression is still possible, even in this instance. What is interesting is that the compression may no longer be *error-free*, in the sense that the quantum states being produced by the source may be slightly distorted by the compression–decompression procedure. Nevertheless, we require that this distortion ought to become very small and ultimately negligible in the limit of large blocks of source output being compressed. To quantify the distortion we introduce a *fidelity* measure for the compression scheme, which measures the average distortion introduced by the compression scheme. The idea of quantum data compression is that the compressed data should be recovered with very good fidelity. Think of the fidelity as being analogous to the probability of doing the decompression correctly – in the limit of large block lengths, it should tend towards the no error limit of 1.

Schumacher's noiseless channel coding theorem quantifies the resources required to do quantum data compression, with the restriction that it be possible to recover the source with fidelity close to 1. In the case of a source producing orthogonal quantum states $|\psi_j\rangle$ with probabilities p_j Schumacher's theorem reduces to telling us that the source may be compressed down to but not beyond the classical limit $H(p_j)$. However, in the more general case of non-orthogonal states being produced by the source, Schumacher's theorem tells us how much a quantum source may be compressed, and the answer is *not* the Shannon entropy $H(p_j)$! Instead, a new entropic quantity, the *von Neumann* entropy, turns out to be the correct answer. In general, the von Neumann entropy agrees with the Shannon entropy if and only if the states $|\psi_j\rangle$ are orthogonal. Otherwise, the von Neumann entropy for the source $p_j, |\psi_j\rangle$ is in general strictly *smaller* than the Shannon entropy $H(p_j)$. Thus, for example, a source producing the state $|0\rangle$ with probability p

and $(|0\rangle + |1\rangle)/\sqrt{2}$ with probability $1 - p$ can be reliably compressed using fewer than $H(p, 1-p)$ qubits per use of the source!

The basic intuition for this decrease in resources required can be understood quite easily. Suppose the source emitting states $|0\rangle$ with probability p and $(|0\rangle + |1\rangle)/\sqrt{2}$ with probability $1 - p$ is used a large number n times. Then by the law of large numbers, with high probability the source emits about np copies of $|0\rangle$ and $n(1-p)$ copies of $(|0\rangle + |1\rangle)/\sqrt{2}$. That is, it has the form

$$|0\rangle^{\otimes np} \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right)^{\otimes n(1-p)}, \tag{1.60}$$

up to re-ordering of the systems involved. Suppose we expand the product of $|0\rangle + |1\rangle$ terms on the right hand side. Since $n(1-p)$ is large, we can again use the law of large numbers to deduce that the terms in the product will be roughly one-half $|0\rangle$s and one-half $|1\rangle$s. That is, the $|0\rangle + |1\rangle$ product can be well approximated by a superposition of states of the form

$$|0\rangle^{\otimes n(1-p)/2} |1\rangle^{\otimes n(1-p)/2}. \tag{1.61}$$

Thus the state emitted by the source can be approximated as a superposition of terms of the form

$$|0\rangle^{\otimes n(1+p)/2} |1\rangle^{\otimes n(1-p)/2}. \tag{1.62}$$

How many states of this form are there? Roughly n choose $n(1+p)/2$, which by Stirling's approximation is equal to $N \equiv 2^{nH[(1+p)/2,(1-p)/2]}$. A simple compression method then is to label all states of the form (1.62) $|c_1\rangle$ through $|c_N\rangle$. It is possible to perform a unitary transform on the n qubits emitted from the source that takes $|c_j\rangle$ to $|j\rangle|0\rangle^{\otimes n - nH[(1+p)/2,(1-p)/2]}$, since j is an $nH[(1+p)/2, (1-p)/2]$ bit number. The compression operation is to perform this unitary transformation, and then drop the final $n - nH[(1+p)/2, (1-p)/2]$ qubits, leaving a compressed state of $nH[(1+p)/2, (1-p)/2]$ qubits. To decompress we append the state $|0\rangle^{\otimes n - nH[(1+p)/2,(1-p)/2]}$ to the compressed state, and perform the inverse unitary transformation.

This procedure for quantum data compression and decompression results in a storage requirement of $H[(1+p)/2, (1-p)/2]$ qubits per use of the source, which whenever $p \geq 1/3$ is an improvement over the $H(p, 1-p)$ qubits we might naively have expected from Shannon's noiseless channel coding theorem. In fact, Schumacher's noiseless channel coding theorem allows us to do somewhat better even than this, as we will see in Chapter 12; however, the essential reason in that construction is the same as the reason we were able to compress here: we exploited the fact that $|0\rangle$ and $(|0\rangle + |1\rangle)/\sqrt{2}$ are not orthogonal. Intuitively, the states contain some redundancy since both have a component in the $|0\rangle$ direction, which results in more physical similarity than would be obtained from orthogonal states. It is this redundancy that we have exploited in the coding scheme just described, and which is used in the full proof of Schumacher's noiseless channel coding theorem. Note that the restriction $p \geq 1/3$ arises because when $p < 1/3$ this particular scheme doesn't exploit the redundancy in the states: we end up effectively *increasing* the redundancy present in the problem! Of course, this is an artifact of the particular scheme we have chosen, and the general solution exploits the redundancy in a much more sensible way to achieve data compression.

Schumacher's noiseless channel coding theorem is an analogue of Shannon's noiseless

channel coding theorem for the compression and decompression of quantum states. Can we find an analogue of Shannon's noisy channel coding theorem? Considerable progress on this important question has been made, using the theory of quantum error-correcting codes; however, a fully satisfactory analogue has not yet been found. We review some of what is known about the quantum channel capacity in Chapter 12.

Quantum distinguishability

Thus far all the dynamical processes we have considered – compression, decompression, noise, encoding and decoding error-correcting codes – arise in both classical and quantum information theory. However, the introduction of new types of information, such as quantum states, enlarges the class of dynamical processes beyond those considered in classical information theory. A good example is the problem of distinguishing quantum states. Classically, we are used to being able to distinguish different items of information, at least in principle. In practice, of course, a smudged letter 'a' written on a page may be very difficult to distinguish from a letter 'o', but in principle it is possible to distinguish between the two possibilities with perfect certainty.

On the other hand, quantum mechanically it is *not* always possible to distinguish between arbitrary states. For example, there is no process allowed by quantum mechanics that will reliably distinguish between the states $|0\rangle$ and $(|0\rangle + |1\rangle)/\sqrt{2}$. Proving this rigorously requires tools we don't presently have available (it is done in Chapter 2), but by considering examples it's pretty easy to convince oneself that it is not possible. Suppose, for example, that we try to distinguish the two states by measuring in the computational basis. Then, if we have been given the state $|0\rangle$, the measurement will yield 0 with probability 1. However, when we measure $(|0\rangle + |1\rangle)/\sqrt{2}$ the measurement yields 0 with probability $1/2$ and 1 with probability $1/2$. Thus, while a measurement result of 1 implies that state must have been $(|0\rangle + |1\rangle)/\sqrt{2}$, since it couldn't have been $|0\rangle$, we can't infer anything about the identity of the quantum state from a measurement result of 0.

This indistinguishability of non-orthogonal quantum states is at the heart of quantum computation and quantum information. It is the essence of our assertion that a quantum state contains hidden information that is not accessible to measurement, and thus plays a key role in quantum algorithms and quantum cryptography. One of the central problems of quantum information theory is to develop measures quantifying how well non-orthogonal quantum states may be distinguished, and much of Chapters 9 and 12 is concerned with this goal. In this introduction we'll limit ourselves to pointing out two interesting aspects of indistinguishability – a connection with the possibility of faster-than-light communication, and an application to 'quantum money.'

Imagine for a moment that we could distinguish between arbitrary quantum states. We'll show that this implies the ability to communicate faster than light, using entanglement. Suppose Alice and Bob share an entangled pair of qubits in the state $(|00\rangle + |11\rangle)/\sqrt{2}$. Then, if Alice measures in the computational basis, the post-measurement states will be $|00\rangle$ with probability $1/2$, and $|11\rangle$ with probability $1/2$. Thus Bob's system is either in the state $|0\rangle$, with probability $1/2$, or in the state $|1\rangle$, with probability $1/2$. Suppose, however, that Alice had instead measured in the $|+\rangle, |-\rangle$ basis. Recall that $|0\rangle = (|+\rangle + |-\rangle)/\sqrt{2}$ and $|1\rangle = (|+\rangle - |-\rangle)/\sqrt{2}$. A little algebra shows that the initial state of Alice and Bob's system may be rewritten as $(|++\rangle + |--\rangle)/\sqrt{2}$. Therefore, if Alice measures in the $|+\rangle, |-\rangle$ basis, the state of Bob's system after the measurement

will be $|+\rangle$ or $|-\rangle$ with probability $1/2$ each. So far, this is all basic quantum mechanics. But if Bob had access to a device that could distinguish the four states $|0\rangle, |1\rangle, |+\rangle, |-\rangle$ from one another, then he could tell whether Alice had measured in the computational basis, or in the $|+\rangle, |-\rangle$ basis. Moreover, he could get that information *instantaneously*, as soon as Alice had made the measurement, providing a means by which Alice and Bob could achieve faster-than-light communication! Of course, we know that it is not possible to distinguish non-orthogonal quantum states; this example shows that this restriction is also intimately tied to other physical properties which we expect the world to obey.

The indistinguishability of non-orthogonal quantum states need not always be a handicap. Sometimes it can be a boon. Imagine that a bank produces banknotes imprinted with a (classical) serial number, and a sequence of qubits each in either the state $|0\rangle$ or $(|0\rangle + |1\rangle)/\sqrt{2}$. Nobody but the bank knows what sequence of these two states is embedded in the note, and the bank maintains a list matching serial numbers to embedded states. The note is impossible to counterfeit exactly, because it is impossible for a would-be counterfeiter to determine with certainty the state of the qubits in the original note, without destroying them. When presented with the banknote a merchant (of certifiable repute) can verify that it is not a counterfeit by calling the bank, telling them the serial number, and then asking what sequence of states were embedded in the note. They can then check that the note is genuine by measuring the qubits in the $|0\rangle, |1\rangle$ or $(|0\rangle + |1\rangle)/\sqrt{2}, (|0\rangle - |1\rangle)/\sqrt{2}$ basis, as directed by the bank. With probability which increases exponentially to one with the number of qubits checked, any would-be counterfeiter will be detected at this stage! This idea is the basis for numerous other quantum cryptographic protocols, and demonstrates the utility of the indistinguishability of non-orthogonal quantum states.

Exercise 1.2: Explain how a device which, upon input of one of two non-orthogonal quantum states $|\psi\rangle$ or $|\varphi\rangle$ correctly identified the state, could be used to build a device which cloned the states $|\psi\rangle$ and $|\varphi\rangle$, in violation of the no-cloning theorem. Conversely, explain how a device for cloning could be used to distinguish non-orthogonal quantum states.

Creation and transformation of entanglement

Entanglement is another elementary static resource of quantum mechanics. Its properties are amazingly different from those of the resources most familiar from classical information theory, and they are not yet well understood; we have at best an incomplete collage of results related to entanglement. We don't yet have all the language needed to understand the solutions, but let's at least look at two information-theoretic problems related to entanglement.

Creating entanglement is a simple dynamical process of interest in quantum information theory. How many qubits must two parties exchange if they are to create a particular entangled state shared between them, given that they share no prior entanglement? A second dynamical process of interest is *transforming entanglement* from one form into another. Suppose, for example, that Alice and Bob share between them a Bell state, and wish to transform it into some other type of entangled state. What resources do they need to accomplish this task? Can they do it without communicating? With classical communication only? If quantum communication is required then how much quantum communication is required?

Answering these and more complex questions about the creation and transformation of entanglement forms a fascinating area of study in its own right, and also promises to give insight into tasks such as quantum computation. For example, a distributed quantum computation may be viewed as simply a method for generating entanglement between two or more parties; lower bounds on the amount of communication that must be done to perform such a distributed quantum computation then follow from lower bounds on the amount of communication that must be performed to create appropriate entangled states.

1.6.2 Quantum information in a wider context

We have given but the barest glimpse of quantum information theory. Part III of this book discusses quantum information theory in much greater detail, especially Chapter 11, which deals with fundamental properties of entropy in quantum and classical information theory, and Chapter 12, which focuses on pure quantum information theory.

Quantum information theory is the most abstract part of quantum computation and quantum information, yet in some sense it is also the most fundamental. The question driving quantum information theory, and ultimately all of quantum computation and quantum information, is *what makes quantum information processing tick?* What is it that separates the quantum and the classical world? What resources, unavailable in a classical world, are being utilized in a quantum computation? Existing answers to these questions are foggy and incomplete; it is our hope that the fog may yet lift in the years to come, and we will obtain a clear appreciation for the possibilities and limitations of quantum information processing.

Problem 1.1: (Feynman-Gates conversation) Construct a friendly imaginary discussion of about 2000 words between Bill Gates and Richard Feynman, set in the present, on the future of computation. (*Comment*: You might like to try waiting until you've read the rest of the book before attempting this question. See the 'History and further reading' below for pointers to one possible answer for this question.)

Problem 1.2: What is the most significant discovery yet made in quantum computation and quantum information? Write an essay of about 2000 words for an educated lay audience about the discovery. (*Comment*: As for the previous problem, you might like to try waiting until you've read the rest of the book before attempting this question.)

History and further reading

Most of the material in this chapter is revisited in more depth in later chapters. Therefore the historical references and further reading below are limited to material which does not recur in later chapters.

Piecing together the historical context in which quantum computation and quantum information have developed requires a broad overview of the history of many fields. We have tried to tie this history together in this chapter, but inevitably much background material was omitted due to limited space and expertise. The following recommendations attempt to redress this omission.

The history of quantum mechanics has been told in many places. We recommend especially the outstanding works of Pais[Pai82, Pai86, Pai91]. Of these three, [Pai86] is most directly concerned with the development of quantum mechanics; however, Pais' biographies of Einstein[Pai82] and of Bohr[Pai91] also contain much material of interest, at a less intense level. The rise of technologies based upon quantum mechanics has been described by Milburn[Mil97, Mil98]. Turing's marvelous paper on the foundations of computer science[Tur36] is well worth reading. It can be found in the valuable historical collection of Davis[Dav65]. Hofstadter[Hof79] and Penrose[Pen89] contain entertaining and informative discussions of the foundations of computer science. Shasha and Lazere's biography of fifteen leading computer scientists[SL98] gives considerable insight into many different facets of the history of computer science. Finally, Knuth's awesome series of books[Knu97, Knu98a, Knu98b] contain an amazing amount of historical information. Shannon's brilliant papers founding information theory make excellent reading[Sha48] (also reprinted in [SW49]). MacWilliams and Sloane[MS77] is not only an excellent text on error-correcting codes, but also contains an enormous amount of useful historical information. Similarly, Cover and Thomas[CT91] is an excellent text on information theory, with extensive historical information. Shannon's collected works, together with many useful historical items have been collected in a large volume[SW93] edited by Sloane and Wyner. Slepian has also collected a useful set of reprints on information theory[Sle74]. Cryptography is an ancient art with an intricate and often interesting history. Kahn[Kah96] is a huge history of cryptography containing a wealth of information. For more recent developments we recommend the books by Menezes, van Oorschot, and Vanstone[MvOV96], Schneier[Sch96a], and by Diffie and Landau[DL98].

Quantum teleportation was discovered by Bennett, Brassard, Crépeau, Jozsa, Peres, and Wootters[BBC+93], and later experimentally realized in various different forms by Boschi, Branca, De Martini, Hardy and Popescu[BBM+98] using optical techniques, by Bouwmeester, Pan, Mattle, Eibl, Weinfurter, and Zeilinger[BPM+97] using photon polarization, by Furusawa, Sørensen, Braunstein, Fuchs, Kimble, and Polzik using 'squeezed' states of light[FSB+98], and by Nielsen, Knill, and Laflamme using NMR[NKL98].

Deutsch's problem was posed by Deutsch[Deu85], and a one-bit solution was given in the same paper. The extension to the general n-bit case was given by Deutsch and Jozsa[DJ92]. The algorithms in these early papers have been substantially improved subsequently by Cleve, Ekert, Macchiavello, and Mosca[CEMM98], and independently in unpublished work by Tapp. In this chapter we have given the improved version of the algorithm, which fits very nicely into the hidden subgroup problem framework that will later be discussed in Chapter 5. The original algorithm of Deutsch only worked probabilistically; Deutsch and Jozsa improved this to obtain a deterministic algorithm, but their method required two function evaluations, in contrast to the improved algorithms presented in this chapter. Nevertheless, it is still conventional to refer to these algorithms as Deutsch's algorithm and the Deutsch–Jozsa algorithm in honor of two huge leaps forward: the concrete demonstration by Deutsch that a quantum computer could do something faster than a classical computer; and the extension by Deutsch and Jozsa which demonstrated for the first time a similar gap for the scaling of the time required to solve a problem.

Excellent discussions of the Stern–Gerlach experiment can be found in standard quantum mechanics textbooks such as the texts by Sakurai[Sak95], Volume III of Feynman, Leighton and Sands[FLS65a], and Cohen-Tannoudji, Diu and Laloë[CTDL77a, CTDL77b].

Problem 1.1 was suggested by the lovely article of Rahim[Rah99].

2 Introduction to quantum mechanics

I ain't no physicist but I know what matters.
– Popeye the Sailor

Quantum mechanics: Real Black Magic Calculus
– Albert Einstein

Quantum mechanics is the most accurate and complete description of the world known. It is also the basis for an understanding of quantum computation and quantum information. This chapter provides all the necessary background knowledge of quantum mechanics needed for a thorough grasp of quantum computation and quantum information. No prior knowledge of quantum mechanics is assumed.

Quantum mechanics is easy to learn, despite its reputation as a difficult subject. The reputation comes from the difficulty of some *applications*, like understanding the structure of complicated molecules, which aren't fundamental to a grasp of the subject; we won't be discussing such applications. The only prerequisite for understanding is some familiarity with elementary linear algebra. Provided you have this background you can begin working out simple problems in a few hours, even with no prior knowledge of the subject.

Readers already familiar with quantum mechanics can quickly skim through this chapter, to become familiar with our (mostly standard) notational conventions, and to assure themselves of familiarity with all the material. Readers with little or no prior knowledge should work through the chapter in detail, pausing to attempt the exercises. If you have difficulty with an exercise, move on, and return later to make another attempt.

The chapter begins with a review of some material from linear algebra in Section 2.1. This section assumes familiarity with elementary linear algebra, but introduces the notation used by physicists to describe quantum mechanics, which is different to that used in most introductions to linear algebra. Section 2.2 describes the basic postulates of quantum mechanics. Upon completion of the section, you will have understood *all* of the fundamental principles of quantum mechanics. This section contains numerous simple exercises designed to help consolidate your grasp of this material. The remaining sections of the chapter, and of this book, elucidate upon this material, without introducing fundamentally new physical principles. Section 2.3 explains *superdense coding*, a surprising and illuminating example of quantum information processing which combines many of the postulates of quantum mechanics in a simple setting. Sections 2.4 and 2.5 develop powerful mathematical tools – the *density operator*, *purifications*, and the *Schmidt decomposition* – which are especially useful in the study of quantum computation and quantum information. Understanding these tools will also help you consolidate your understanding of elementary quantum mechanics. Finally, Section 2.6 examines the question of how quantum mechanics goes beyond the usual 'classical' understanding of the way the world works.

2.1 Linear algebra

This book is written as much to disturb and annoy as to instruct.
– The first line of *About Vectors*, by Banesh Hoffmann.

Life is complex – it has both real and imaginary parts.
– Anonymous

Linear algebra is the study of vector spaces and of linear operations on those vector spaces. A good understanding of quantum mechanics is based upon a solid grasp of elementary linear algebra. In this section we review some basic concepts from linear algebra, and describe the standard notations which are used for these concepts in the study of quantum mechanics. These notations are summarized in Figure 2.1 on page 62, with the quantum notation in the left column, and the linear-algebraic description in the right column. You may like to glance at the table, and see how many of the concepts in the right column you recognize.

In our opinion the chief obstacle to assimilation of the postulates of quantum mechanics is not the postulates themselves, but rather the large body of linear algebraic notions required to understand them. Coupled with the unusual Dirac notation adopted by physicists for quantum mechanics, it can appear (falsely) quite fearsome. For these reasons, we advise the reader not familiar with quantum mechanics to quickly read through the material which follows, pausing mainly to concentrate on understanding the absolute basics of the notation being used. Then proceed to a careful study of the main topic of the chapter – the postulates of quantum mechanics – returning to study the necessary linear algebraic notions and notations in more depth, as required.

The basic objects of linear algebra are *vector spaces*. The vector space of most interest to us is $\mathbf{C}^n$, the space of all n-tuples of complex numbers, $(z_1, \ldots, z_n)$. The elements of a vector space are called *vectors*, and we will sometimes use the column matrix notation

$$\begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} \tag{2.1}$$

to indicate a vector. There is an *addition* operation defined which takes pairs of vectors to other vectors. In $\mathbf{C}^n$ the addition operation for vectors is defined by

$$\begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} + \begin{bmatrix} z'_1 \\ \vdots \\ z'_n \end{bmatrix} \equiv \begin{bmatrix} z_1 + z'_1 \\ \vdots \\ z_n + z'_n \end{bmatrix}, \tag{2.2}$$

where the addition operations on the right are just ordinary additions of complex numbers. Furthermore, in a vector space there is a *multiplication by a scalar* operation. In $\mathbf{C}^n$ this operation is defined by

$$z \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} \equiv \begin{bmatrix} z z_1 \\ \vdots \\ z z_n \end{bmatrix}, \tag{2.3}$$

where z is a *scalar*, that is, a complex number, and the multiplications on the right are ordinary multiplication of complex numbers. Physicists sometimes refer to complex numbers as *c-numbers*.

Quantum mechanics is our main motivation for studying linear algebra, so we will use the standard notation of quantum mechanics for linear algebraic concepts. The standard quantum mechanical notation for a vector in a vector space is the following:

$$|\psi\rangle. \tag{2.4}$$

ψ is a label for the vector (any label is valid, although we prefer to use simple labels like ψ and φ). The $|\cdot\rangle$ notation is used to indicate that the object is a vector. The entire object $|\psi\rangle$ is sometimes called a *ket*, although we won't use that terminology often.

A vector space also contains a special *zero vector*, which we denote by 0. It satisfies the property that for any other vector $|v\rangle$, $|v\rangle + 0 = |v\rangle$. Note that we do not use the ket notation for the zero vector – it is the only exception we shall make. The reason for making the exception is because it is conventional to use the 'obvious' notation for the zero vector, $|0\rangle$, to mean something else entirely. The scalar multiplication operation is such that $z0 = 0$ for any complex number z. For convenience, we use the notation $(z_1, \ldots, z_n)$ to denote a column matrix with entries $z_1, \ldots, z_n$. In $\mathbf{C}^n$ the zero element is $(0, 0, \ldots, 0)$. A *vector subspace* of a vector space V is a subset W of V such that W is also a vector space, that is, W must be closed under scalar multiplication and addition.

Notation	Description
z^*	Complex conjugate of the complex number z. $(1 + i)^* = 1 - i$
$\|\psi\rangle$	Vector. Also known as a *ket*.
$\langle\psi\|$	Vector dual to $\|\psi\rangle$. Also known as a *bra*.
$\langle\varphi\|\psi\rangle$	Inner product between the vectors $\|\varphi\rangle$ and $\|\psi\rangle$.
$\|\varphi\rangle \otimes \|\psi\rangle$	Tensor product of $\|\varphi\rangle$ and $\|\psi\rangle$.
$\|\varphi\rangle\|\psi\rangle$	Abbreviated notation for tensor product of $\|\varphi\rangle$ and $\|\psi\rangle$.
A^*	Complex conjugate of the A matrix.
A^T	Transpose of the A matrix.
$A^\dagger$	Hermitian conjugate or adjoint of the A matrix, $A^\dagger = (A^T)^*$. $\left[\begin{array}{cc} a & b \\ c & d \end{array}\right]^\dagger = \left[\begin{array}{cc} a^* & c^* \\ b^* & d^* \end{array}\right]$.
$\langle\varphi\|A\|\psi\rangle$	Inner product between $\|\varphi\rangle$ and $A\|\psi\rangle$. Equivalently, inner product between $A^\dagger\|\varphi\rangle$ and $\|\psi\rangle$.

Figure 2.1. Summary of some standard quantum mechanical notation for notions from linear algebra. This style of notation is known as the *Dirac* notation.

2.1.1 Bases and linear independence

A *spanning set* for a vector space is a set of vectors $|v_1\rangle, \ldots, |v_n\rangle$ such that any vector $|v\rangle$ in the vector space can be written as a linear combination $|v\rangle = \sum_i a_i |v_i\rangle$ of vectors

in that set. For example, a spanning set for the vector space $\mathbf{C}^2$ is the set

$$|v_1\rangle \equiv \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad |v_2\rangle \equiv \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{2.5}$$

since any vector

$$|v\rangle = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \tag{2.6}$$

in $\mathbf{C}^2$ can be written as a linear combination $|v\rangle = a_1|v_1\rangle + a_2|v_2\rangle$ of the vectors $|v_1\rangle$ and $|v_2\rangle$. We say that the vectors $|v_1\rangle$ and $|v_2\rangle$ *span* the vector space $\mathbf{C}^2$.

Generally, a vector space may have many different spanning sets. A second spanning set for the vector space $\mathbf{C}^2$ is the set

$$|v_1\rangle \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \quad |v_2\rangle \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \tag{2.7}$$

since an arbitrary vector $|v\rangle = (a_1, a_2)$ can be written as a linear combination of $|v_1\rangle$ and $|v_2\rangle$,

$$|v\rangle = \frac{a_1 + a_2}{\sqrt{2}}|v_1\rangle + \frac{a_1 - a_2}{\sqrt{2}}|v_2\rangle. \tag{2.8}$$

A set of non-zero vectors $|v_1\rangle, \ldots, |v_n\rangle$ are *linearly dependent* if there exists a set of complex numbers $a_1, \ldots, a_n$ with $a_i \neq 0$ for at least one value of i, such that

$$a_1|v_1\rangle + a_2|v_2\rangle + \cdots + a_n|v_n\rangle = 0. \tag{2.9}$$

A set of vectors is *linearly independent* if it is not linearly dependent. It can be shown that any two sets of linearly independent vectors which span a vector space V contain the same number of elements. We call such a set a *basis* for V. Furthermore, such a basis set always exists. The number of elements in the basis is defined to be the *dimension* of V. In this book we will only be interested in *finite dimensional* vector spaces. There are many interesting and often difficult questions associated with infinite dimensional vector spaces. We won't need to worry about these questions.

Exercise 2.1: (Linear dependence: example) Show that $(1, -1), (1, 2)$ and $(2, 1)$ are linearly dependent.

2.1.2 Linear operators and matrices

A *linear operator* between vector spaces V and W is defined to be any function $A : V \rightarrow W$ which is linear in its inputs,

$$A\left(\sum_i a_i|v_i\rangle\right) = \sum_i a_i A\left(|v_i\rangle\right). \tag{2.10}$$

Usually we just write $A|v\rangle$ to denote $A(|v\rangle)$. When we say that a linear operator A is defined *on* a vector space, V, we mean that A is a linear operator from V to V. An important linear operator on any vector space V is the *identity operator*, I_V, defined by the equation $I_V|v\rangle \equiv |v\rangle$ for all vectors $|v\rangle$. Where no chance of confusion arises we drop the subscript V and just write I to denote the identity operator. Another important linear operator is the *zero operator*, which we denote 0. The zero operator maps all vectors to

the zero vector, $0|v\rangle \equiv 0$. It is clear from (2.10) that once the action of a linear operator A on a basis is specified, the action of A is completely determined on all inputs.

Suppose V, W, and X are vector spaces, and $A : V \rightarrow W$ and $B : W \rightarrow X$ are linear operators. Then we use the notation BA to denote the *composition* of B with A, defined by $(BA)(|v\rangle) \equiv B(A(|v\rangle))$. Once again, we write $BA|v\rangle$ as an abbreviation for $(BA)(|v\rangle)$.

The most convenient way to understand linear operators is in terms of their *matrix representations*. In fact, the linear operator and matrix viewpoints turn out to be completely equivalent. The matrix viewpoint may be more familiar to you, however. To see the connection, it helps to first understand that an m by n complex matrix A with entries A_{ij} is in fact a linear operator sending vectors in the vector space $\mathbf{C}^n$ to the vector space $\mathbf{C}^m$, under matrix multiplication of the matrix A by a vector in $\mathbf{C}^n$. More precisely, the claim that the matrix A is a linear operator just means that

$$A\left(\sum_i a_i |v_i\rangle\right) = \sum_i a_i A|v_i\rangle \tag{2.11}$$

is true as an equation where the operation is matrix multiplication of A by column vectors. Clearly, this is true!

We've seen that matrices can be regarded as linear operators. Can linear operators be given a matrix representation? In fact they can, as we now explain. This equivalence between the two viewpoints justifies our interchanging terms from matrix theory and operator theory throughout the book. Suppose $A : V \rightarrow W$ is a linear operator between vector spaces V and W. Suppose $|v_1\rangle, \ldots, |v_m\rangle$ is a basis for V and $|w_1\rangle, \ldots, |w_n\rangle$ is a basis for W. Then for each j in the range $1, \ldots, m$, there exist complex numbers A_{1j} through A_{nj} such that

$$A|v_j\rangle = \sum_i A_{ij} |w_i\rangle. \tag{2.12}$$

The matrix whose entries are the values A_{ij} is said to form a *matrix representation* of the operator A. This matrix representation of A is completely equivalent to the operator A, and we will use the matrix representation and abstract operator viewpoints interchangeably. Note that to make the connection between matrices and linear operators we must specify a set of input and output basis states for the input and output vector spaces of the linear operator.

Exercise 2.2: (Matrix representations: example) Suppose V is a vector space with basis vectors $|0\rangle$ and $|1\rangle$, and A is a linear operator from V to V such that $A|0\rangle = |1\rangle$ and $A|1\rangle = |0\rangle$. Give a matrix representation for A, with respect to the input basis $|0\rangle, |1\rangle$, and the output basis $|0\rangle, |1\rangle$. Find input and output bases which give rise to a different matrix representation of A.

Exercise 2.3: (Matrix representation for operator products) Suppose A is a linear operator from vector space V to vector space W, and B is a linear operator from vector space W to vector space X. Let $|v_i\rangle, |w_j\rangle$, and $|x_k\rangle$ be bases for the vector spaces V, W, and X, respectively. Show that the matrix representation for the linear transformation BA is the matrix product of the matrix representations for B and A, with respect to the appropriate bases.

Exercise 2.4: (Matrix representation for identity) Show that the identity operator on a vector space V has a matrix representation which is one along the diagonal and zero everywhere else, if the matrix representation is taken with respect to the same input and output bases. This matrix is known as the *identity matrix*.

2.1.3 The Pauli matrices

Four extremely useful matrices which we shall often have occasion to use are the *Pauli matrices*. These are 2 by 2 matrices, which go by a variety of notations. The matrices, and their corresponding notations, are depicted in Figure 2.2. The Pauli matrices are so useful in the study of quantum computation and quantum information that we encourage you to memorize them by working through in detail the many examples and exercises based upon them in subsequent sections.

$$\sigma_0 \equiv I \equiv \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \sigma_1 \equiv \sigma_x \equiv X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\sigma_2 \equiv \sigma_y \equiv Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad \sigma_3 \equiv \sigma_z \equiv Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Figure 2.2. The Pauli matrices. Sometimes I is omitted from the list with just X, Y and Z known as the Pauli matrices.

2.1.4 Inner products

An *inner product* is a function which takes as input two vectors $|v\rangle$ and $|w\rangle$ from a vector space and produces a complex number as output. For the time being, it will be convenient to write the inner product of $|v\rangle$ and $|w\rangle$ as $(|v\rangle, |w\rangle)$. This is not the standard quantum mechanical notation; for pedagogical clarity the $(\cdot, \cdot)$ notation will be useful occasionally in this chapter. The standard quantum mechanical notation for the inner product $(|v\rangle, |w\rangle)$ is $\langle v|w\rangle$, where $|v\rangle$ and $|w\rangle$ are vectors in the inner product space, and the notation $\langle v|$ is used for the *dual vector* to the vector $|v\rangle$; the dual is a linear operator from the inner product space V to the complex numbers $\mathbf{C}$, defined by $\langle v|(|w\rangle) \equiv \langle v|w\rangle \equiv (|v\rangle, |w\rangle)$. We will see shortly that the matrix representation of dual vectors is just a row vector.

A function $(\cdot, \cdot)$ from $V \times V$ to $\mathbf{C}$ is an inner product if it satisfies the requirements that:

(1) $(\cdot, \cdot)$ is linear in the second argument,

$$\left(|v\rangle, \sum_i \lambda_i |w_i\rangle \right) = \sum_i \lambda_i \left(|v\rangle, |w_i\rangle \right). \tag{2.13}$$

(2) $(|v\rangle, |w\rangle) = (|w\rangle, |v\rangle)^*$.
(3) $(|v\rangle, |v\rangle) \geq 0$ with equality if and only if $|v\rangle = 0$.

For example, $\mathbf{C}^n$ has an inner product defined by

$$((y_1, \ldots, y_n), (z_1, \ldots, z_n)) \equiv \sum_i y_i^* z_i = \begin{bmatrix} y_1^* \ldots y_n^* \end{bmatrix} \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}. \tag{2.14}$$

We call a vector space equipped with an inner product an *inner product space*.

Exercise 2.5: Verify that $(\cdot, \cdot)$ just defined is an inner product on $\mathbf{C}^n$.

Exercise 2.6: Show that any inner product $(\cdot, \cdot)$ is conjugate-linear in the first argument,

$$\left(\sum_i \lambda_i |w_i\rangle, |v\rangle \right) = \sum_i \lambda_i^* (|w_i\rangle, |v\rangle). \tag{2.15}$$

Discussions of quantum mechanics often refer to *Hilbert space*. In the finite dimensional complex vector spaces that come up in quantum computation and quantum information, a Hilbert space is *exactly the same thing* as an inner product space. From now on we use the two terms interchangeably, preferring the term Hilbert space. In infinite dimensions Hilbert spaces satisfy additional technical restrictions above and beyond inner product spaces, which we will not need to worry about.

Vectors $|w\rangle$ and $|v\rangle$ are *orthogonal* if their inner product is zero. For example, $|w\rangle \equiv (1, 0)$ and $|v\rangle \equiv (0, 1)$ are orthogonal with respect to the inner product defined by (2.14). We define the *norm* of a vector $|v\rangle$ by

$$\| |v\rangle \| \equiv \sqrt{\langle v|v\rangle}. \tag{2.16}$$

A *unit vector* is a vector $|v\rangle$ such that $\| |v\rangle \| = 1$. We also say that $|v\rangle$ is *normalized* if $\| |v\rangle \| = 1$. It is convenient to talk of *normalizing* a vector by dividing by its norm; thus $|v\rangle / \| |v\rangle \|$ is the *normalized* form of $|v\rangle$, for any non-zero vector $|v\rangle$. A set $|i\rangle$ of vectors with index i is *orthonormal* if each vector is a unit vector, and distinct vectors in the set are orthogonal, that is, $\langle i|j\rangle = \delta_{ij}$, where i and j are both chosen from the index set.

Exercise 2.7: Verify that $|w\rangle \equiv (1, 1)$ and $|v\rangle \equiv (1, -1)$ are orthogonal. What are the normalized forms of these vectors?

Suppose $|w_1\rangle, \ldots, |w_d\rangle$ is a basis set for some vector space V with an inner product. There is a useful method, the *Gram–Schmidt* procedure, which can be used to produce an orthonormal basis set $|v_1\rangle, \ldots, |v_d\rangle$ for the vector space V. Define $|v_1\rangle \equiv |w_1\rangle / \| |w_1\rangle \|$, and for $1 \leq k \leq d-1$ define $|v_{k+1}\rangle$ inductively by

$$|v_{k+1}\rangle \equiv \frac{|w_{k+1}\rangle - \sum_{i=1}^{k} \langle v_i|w_{k+1}\rangle |v_i\rangle}{\| |w_{k+1}\rangle - \sum_{i=1}^{k} \langle v_i|w_{k+1}\rangle |v_i\rangle \|}. \tag{2.17}$$

It is not difficult to verify that the vectors $|v_1\rangle, \ldots, |v_d\rangle$ form an orthonormal set which is also a basis for V. Thus, any finite dimensional vector space of dimension d has an orthonormal basis, $|v_1\rangle, \ldots, |v_d\rangle$.

Exercise 2.8: Prove that the Gram–Schmidt procedure produces an orthonormal basis for V.

From now on, when we speak of a matrix representation for a linear operator, we mean a matrix representation with respect to orthonormal input and output bases. We also use the convention that if the input and output spaces for a linear operator are the same, then the input and output bases are the same, unless noted otherwise.

With these conventions, the inner product on a Hilbert space can be given a convenient matrix representation. Let $|w\rangle = \sum_i w_i|i\rangle$ and $|v\rangle = \sum_j v_j|j\rangle$ be representations of vectors $|w\rangle$ and $|v\rangle$ with respect to some orthonormal basis $|i\rangle$. Then, since $\langle i|j\rangle = \delta_{ij}$,

$$\langle v|w\rangle = \left(\sum_i v_i|i\rangle, \sum_j w_j|j\rangle\right) = \sum_{ij} v_i^* w_j \delta_{ij} = \sum_i v_i^* w_i \tag{2.18}$$

$$= \begin{bmatrix} v_1^* \ldots v_n^* \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}. \tag{2.19}$$

That is, the inner product of two vectors is equal to the vector inner product between two matrix representations of those vectors, provided the representations are written with respect to the same orthonormal basis. We also see that the dual vector $\langle v|$ has a nice interpretation as the row vector whose components are complex conjugates of the corresponding components of the column vector representation of $|v\rangle$.

There is a useful way of representing linear operators which makes use of the inner product, known as the *outer product* representation. Suppose $|v\rangle$ is a vector in an inner product space V, and $|w\rangle$ is a vector in an inner product space W. Define $|w\rangle\langle v|$ to be the linear operator from V to W whose action is defined by

$$\left(|w\rangle\langle v|\right)\left(|v'\rangle\right) \equiv |w\rangle\langle v|v'\rangle = \langle v|v'\rangle|w\rangle. \tag{2.20}$$

This equation fits beautifully into our notational conventions, according to which the expression $|w\rangle\langle v|v'\rangle$ could potentially have one of two meanings: we will use it to denote the result when the *operator* $|w\rangle\langle v|$ acts on $|v'\rangle$, and it has an existing interpretation as the result of multiplying $|w\rangle$ by the complex number $\langle v|v'\rangle$. Our definitions are chosen so that these two potential meanings coincide. Indeed, we *define* the former in terms of the latter!

We can take linear combinations of outer product operators $|w\rangle\langle v|$ in the obvious way. By definition $\sum_i a_i|w_i\rangle\langle v_i|$ is the linear operator which, when acting on $|v'\rangle$, produces $\sum_i a_i|w_i\rangle\langle v_i|v'\rangle$ as output.

The usefulness of the outer product notation can be discerned from an important result known as the *completeness relation* for orthonormal vectors. Let $|i\rangle$ be any orthonormal basis for the vector space V, so an arbitrary vector $|v\rangle$ can be written $|v\rangle = \sum_i v_i|i\rangle$ for some set of complex numbers v_i. Note that $\langle i|v\rangle = v_i$ and therefore

$$\left(\sum_i |i\rangle\langle i|\right)|v\rangle = \sum_i |i\rangle\langle i|v\rangle = \sum_i v_i|i\rangle = |v\rangle. \tag{2.21}$$

Since the last equation is true for all $|v\rangle$ it follows that

$$\sum_i |i\rangle\langle i| = I. \tag{2.22}$$

This equation is known as the *completeness relation*. One application of the completeness relation is to give a means for representing any operator in the outer product notation. Suppose $A : V \to W$ is a linear operator, $|v_i\rangle$ is an orthonormal basis for V, and $|w_j\rangle$ an orthonormal basis for W. Using the completeness relation twice we obtain

$$A = I_W A I_V \tag{2.23}$$

$$= \sum_{ij} |w_j\rangle\langle w_j|A|v_i\rangle\langle v_i| \tag{2.24}$$

$$= \sum_{ij} \langle w_j|A|v_i\rangle |w_j\rangle\langle v_i|, \tag{2.25}$$

which is the outer product representation for A. We also see from this equation that A has matrix element $\langle w_j|A|v_i\rangle$ in the ith column and jth row, with respect to the input basis $|v_i\rangle$ and output basis $|w_j\rangle$.

A second application illustrating the usefulness of the completeness relation is the *Cauchy–Schwarz inequality*. This important result is discussed in Box 2.1, on this page.

Exercise 2.9: (Pauli operators and the outer product) The Pauli matrices (Figure 2.2 on page 65) can be considered as operators with respect to an orthonormal basis $|0\rangle, |1\rangle$ for a two-dimensional Hilbert space. Express each of the Pauli operators in the outer product notation.

Exercise 2.10: Suppose $|v_i\rangle$ is an orthonormal basis for an inner product space V. What is the matrix representation for the operator $|v_j\rangle\langle v_k|$, with respect to the $|v_i\rangle$ basis?

Box 2.1: The Cauchy-Schwarz inequality

The *Cauchy–Schwarz inequality* is an important geometric fact about Hilbert spaces. It states that for any two vectors $|v\rangle$ and $|w\rangle$, $|\langle v|w\rangle|^2 \leq \langle v|v\rangle\langle w|w\rangle$. To see this, use the Gram–Schmidt procedure to construct an orthonormal basis $|i\rangle$ for the vector space such that the first member of the basis $|i\rangle$ is $|w\rangle/\sqrt{\langle w|w\rangle}$. Using the completeness relation $\sum_i |i\rangle\langle i| = I$, and dropping some non-negative terms gives

$$\langle v|v\rangle\langle w|w\rangle = \sum_i \langle v|i\rangle\langle i|v\rangle\langle w|w\rangle \tag{2.26}$$

$$\geq \frac{\langle v|w\rangle\langle w|v\rangle}{\langle w|w\rangle}\langle w|w\rangle \tag{2.27}$$

$$= \langle v|w\rangle\langle w|v\rangle = |\langle v|w\rangle|^2, \tag{2.28}$$

as required. A little thought shows that equality occurs if and only if $|v\rangle$ and $|w\rangle$ are linearly related, $|v\rangle = z|w\rangle$ or $|w\rangle = z|v\rangle$, for some scalar z.

2.1.5 Eigenvectors and eigenvalues

An *eigenvector* of a linear operator A on a vector space is a non-zero vector $|v\rangle$ such that $A|v\rangle = v|v\rangle$, where v is a complex number known as the *eigenvalue* of A corresponding to $|v\rangle$. It will often be convenient to use the notation v both as a label for the eigenvector, and to represent the eigenvalue. We assume that you are familiar with the elementary properties of eigenvalues and eigenvectors – in particular, how to find them, via the characteristic equation. The *characteristic function* is defined to be $c(\lambda) \equiv \det|A - \lambda I|$,

where det is the *determinant* function for matrices; it can be shown that the characteristic function depends only upon the operator A, and not on the specific matrix representation used for A. The solutions of the *characteristic equation* $c(\lambda) = 0$ are the eigenvalues of the operator A. By the fundamental theorem of algebra, every polynomial has at least one complex root, so every operator A has at least one eigenvalue, and a corresponding eigenvector. The *eigenspace* corresponding to an eigenvalue v is the set of vectors which have eigenvalue v. It is a vector subspace of the vector space on which A acts.

A *diagonal representation* for an operator A on a vector space V is a representation $A = \sum_i \lambda_i |i\rangle\langle i|$, where the vectors $|i\rangle$ form an orthonormal set of eigenvectors for A, with corresponding eigenvalues λ_i. An operator is said to be *diagonalizable* if it has a diagonal representation. In the next section we will find a simple set of necessary and sufficient conditions for an operator on a Hilbert space to be diagonalizable. As an example of a diagonal representation, note that the Pauli Z matrix may be written

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|, \tag{2.29}$$

where the matrix representation is with respect to orthonormal vectors $|0\rangle$ and $|1\rangle$, respectively. Diagonal representations are sometimes also known as *orthonormal decompositions*.

When an eigenspace is more than one dimensional we say that it is *degenerate*. For example, the matrix A defined by

$$A \equiv \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.30}$$

has a two-dimensional eigenspace corresponding to the eigenvalue 2. The eigenvectors $(1, 0, 0)$ and $(0, 1, 0)$ are said to be *degenerate* because they are linearly independent eigenvectors of A with the same eigenvalue.

Exercise 2.11: (Eigendecomposition of the Pauli matrices) Find the eigenvectors, eigenvalues, and diagonal representations of the Pauli matrices X, Y, and Z.

Exercise 2.12: Prove that the matrix

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{2.31}$$

is not diagonalizable.

2.1.6 Adjoints and Hermitian operators

Suppose A is any linear operator on a Hilbert space, V. It turns out that there exists a unique linear operator $A^\dagger$ on V such that for all vectors $|v\rangle, |w\rangle \in V$,

$$(|v\rangle, A|w\rangle) = (A^\dagger|v\rangle, |w\rangle). \tag{2.32}$$

This linear operator is known as the *adjoint* or *Hermitian conjugate* of the operator A. From the definition it is easy to see that $(AB)^\dagger = B^\dagger A^\dagger$. By convention, if $|v\rangle$ is a vector, then we define $|v\rangle^\dagger \equiv \langle v|$. With this definition it is not difficult to see that $(A|v\rangle)^\dagger = \langle v|A^\dagger$.

Exercise 2.13: If $|w\rangle$ and $|v\rangle$ are any two vectors, show that $(|w\rangle\langle v|)^\dagger = |v\rangle\langle w|$.

Exercise 2.14: (Anti-linearity of the adjoint) Show that the adjoint operation is anti-linear,

$$\left(\sum_i a_i A_i\right)^\dagger = \sum_i a_i^* A_i^\dagger. \tag{2.33}$$

Exercise 2.15: Show that $(A^\dagger)^\dagger = A$.

In a matrix representation of an operator A, the action of the Hermitian conjugation operation is to take the matrix of A to the conjugate-transpose matrix, $A^\dagger \equiv (A^*)^T$, where the $*$ indicates complex conjugation, and T indicates the transpose operation. For example, we have

$$\begin{bmatrix} 1+3i & 2i \\ 1+i & 1-4i \end{bmatrix}^\dagger = \begin{bmatrix} 1-3i & 1-i \\ -2i & 1+4i \end{bmatrix}. \tag{2.34}$$

An operator A whose adjoint is A is known as a *Hermitian* or *self-adjoint* operator. An important class of Hermitian operators is the *projectors*. Suppose W is a k-dimensional vector subspace of the d-dimensional vector space V. Using the Gram–Schmidt procedure it is possible to construct an orthonormal basis $|1\rangle, \ldots, |d\rangle$ for V such that $|1\rangle, \ldots, |k\rangle$ is an orthonormal basis for W. By definition,

$$P \equiv \sum_{i=1}^{k} |i\rangle\langle i| \tag{2.35}$$

is the *projector* onto the subspace W. It is easy to check that this definition is independent of the orthonormal basis $|1\rangle, \ldots, |k\rangle$ used for W. From the definition it can be shown that $|v\rangle\langle v|$ is Hermitian for any vector $|v\rangle$, so P is Hermitian, $P^\dagger = P$. We will often refer to the 'vector space' P, as shorthand for the vector space onto which P is a projector. The *orthogonal complement* of P is the operator $Q \equiv I - P$. It is easy to see that Q is a projector onto the vector space spanned by $|k+1\rangle, \ldots, |d\rangle$, which we also refer to as the *orthogonal complement* of P, and may denote by Q.

Exercise 2.16: Show that any projector P satisfies the equation $P^2 = P$.

An operator A is said to be *normal* if $AA^\dagger = A^\dagger A$. Clearly, an operator which is Hermitian is also normal. There is a remarkable representation theorem for normal operators known as the *spectral decomposition*, which states that an operator is a normal operator if and only if it is diagonalizable. This result is proved in Box 2.2 on page 72, which you should read closely.

Exercise 2.17: Show that a normal matrix is Hermitian if and only if it has real eigenvalues.

A matrix U is said to be *unitary* if $U^\dagger U = I$. Similarly an operator U is unitary if $U^\dagger U = I$. It is easily checked that an operator is unitary if and only if each of its matrix representations is unitary. A unitary operator also satisfies $UU^\dagger = I$, and therefore U is normal and has a spectral decomposition. Geometrically, unitary operators are important because they preserve inner products between vectors. To see this, let $|v\rangle$ and $|w\rangle$ be any

two vectors. Then the inner product of $U|v\rangle$ and $U|w\rangle$ is the same as the inner product of $|v\rangle$ and $|w\rangle$,

$$(U|v\rangle, U|w\rangle) = \langle v|U^\dagger U|w\rangle = \langle v|I|w\rangle = \langle v|w\rangle. \tag{2.36}$$

This result suggests the following elegant outer product representation of any unitary U. Let $|v_i\rangle$ be any orthonormal basis set. Define $|w_i\rangle \equiv U|v_i\rangle$, so $|w_i\rangle$ is also an orthonormal basis set, since unitary operators preserve inner products. Note that $U = \sum_i |w_i\rangle\langle v_i|$. Conversely, if $|v_i\rangle$ and $|w_i\rangle$ are any two orthonormal bases, then it is easily checked that the operator U defined by $U \equiv \sum_i |w_i\rangle\langle v_i|$ is a unitary operator.

Exercise 2.18: Show that all eigenvalues of a unitary matrix have modulus 1, that is, can be written in the form $e^{i\theta}$ for some real θ.

Exercise 2.19: (Pauli matrices: Hermitian and unitary) Show that the Pauli matrices are Hermitian and unitary.

Exercise 2.20: (Basis changes) Suppose A' and A'' are matrix representations of an operator A on a vector space V with respect to two different orthonormal bases, $|v_i\rangle$ and $|w_i\rangle$. Then the elements of A' and A'' are $A'_{ij} = \langle v_i|A|v_j\rangle$ and $A''_{ij} = \langle w_i|A|w_j\rangle$. Characterize the relationship between A' and A''.

A special subclass of Hermitian operators is extremely important. This is the *positive operators*. A positive operator A is defined to be an operator such that for any vector $|v\rangle$, $(|v\rangle, A|v\rangle)$ is a real, non-negative number. If $(|v\rangle, A|v\rangle)$ is *strictly* greater than zero for all $|v\rangle \neq 0$ then we say that A is *positive definite*. In Exercise 2.24 on this page you will show that any positive operator is automatically Hermitian, and therefore by the spectral decomposition has diagonal representation $\sum_i \lambda_i |i\rangle\langle i|$, with non-negative eigenvalues λ_i.

Exercise 2.21: Repeat the proof of the spectral decomposition in Box 2.2 for the case when M is Hermitian, simplifying the proof wherever possible.

Exercise 2.22: Prove that two eigenvectors of a Hermitian operator with different eigenvalues are necessarily orthogonal.

Exercise 2.23: Show that the eigenvalues of a projector P are all either 0 or 1.

Exercise 2.24: (Hermiticity of positive operators) Show that a positive operator is necessarily Hermitian. (*Hint*: Show that an arbitrary operator A can be written $A = B + iC$ where B and C are Hermitian.)

Exercise 2.25: Show that for any operator A, $A^\dagger A$ is positive.

2.1.7 Tensor products

The *tensor product* is a way of putting vector spaces together to form larger vector spaces. This construction is crucial to understanding the quantum mechanics of multiparticle systems. The following discussion is a little abstract, and may be difficult to follow if you're not already familiar with the tensor product, so feel free to skip ahead now and revisit later when you come to the discussion of tensor products in quantum mechanics.

Suppose V and W are vector spaces of dimension m and n respectively. For convenience we also suppose that V and W are Hilbert spaces. Then $V \otimes W$ (read 'V tensor

Box 2.2: The spectral decomposition – important!

The *spectral decomposition* is an extremely useful representation theorem for normal operators.

Theorem 2.1: (**Spectral decomposition**) Any normal operator M on a vector space V is diagonal with respect to some orthonormal basis for V. Conversely, any diagonalizable operator is normal.

Proof

The converse is a simple exercise, so we prove merely the forward implication, by induction on the dimension d of V. The case $d = 1$ is trivial. Let λ be an eigenvalue of M, P the projector onto the λ eigenspace, and Q the projector onto the orthogonal complement. Then $M = (P + Q)M(P + Q) = PMP + QMP + PMQ + QMQ$. Obviously $PMP = \lambda P$. Furthermore, $QMP = 0$, as M takes the subspace P into itself. We claim that $PMQ = 0$ also. To see this, let $|v\rangle$ be an element of the subspace P. Then $MM^\dagger|v\rangle = M^\dagger M|v\rangle = \lambda M^\dagger|v\rangle$. Thus, $M^\dagger|v\rangle$ has eigenvalue λ and therefore is an element of the subspace P. It follows that $QM^\dagger P = 0$. Taking the adjoint of this equation gives $PMQ = 0$. Thus $M = PMP + QMQ$. Next, we prove that QMQ is normal. To see this, note that $QM = QM(P + Q) = QMQ$, and $QM^\dagger = QM^\dagger(P + Q) = QM^\dagger Q$. Therefore, by the normality of M, and the observation that $Q^2 = Q$,

$$\begin{aligned} QMQ\,QM^\dagger Q &= QMQM^\dagger Q && (2.37)\\ &= QMM^\dagger Q && (2.38)\\ &= QM^\dagger MQ && (2.39)\\ &= QM^\dagger QMQ && (2.40)\\ &= QM^\dagger Q\,QMQ\,, && (2.41) \end{aligned}$$

so QMQ is normal. By induction, QMQ is diagonal with respect to some orthonormal basis for the subspace Q, and PMP is already diagonal with respect to some orthonormal basis for P. It follows that $M = PMP + QMQ$ is diagonal with respect to some orthonormal basis for the total vector space. □

In terms of the outer product representation, this means that M can be written as $M = \sum_i \lambda_i |i\rangle\langle i|$, where λ_i are the eigenvalues of M, $|i\rangle$ is an orthonormal basis for V, and each $|i\rangle$ an eigenvector of M with eigenvalue λ_i. In terms of projectors, $M = \sum_i \lambda_i P_i$, where λ_i are again the eigenvalues of M, and P_i is the projector onto the λ_i eigenspace of M. These projectors satisfy the completeness relation $\sum_i P_i = I$, and the orthonormality relation $P_i P_j = \delta_{ij} P_i$.

W') is an mn dimensional vector space. The elements of $V \otimes W$ are linear combinations of 'tensor products' $|v\rangle \otimes |w\rangle$ of elements $|v\rangle$ of V and $|w\rangle$ of W. In particular, if $|i\rangle$ and $|j\rangle$ are orthonormal bases for the spaces V and W then $|i\rangle \otimes |j\rangle$ is a basis for $V \otimes W$. We often use the abbreviated notations $|v\rangle|w\rangle$, $|v, w\rangle$ or even $|vw\rangle$ for the tensor product

$|v\rangle \otimes |w\rangle$. For example, if V is a two-dimensional vector space with basis vectors $|0\rangle$ and $|1\rangle$ then $|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle$ is an element of $V \otimes V$.

By definition the tensor product satisfies the following basic properties:

(1) For an arbitrary scalar z and elements $|v\rangle$ of V and $|w\rangle$ of W,

$$z\left(|v\rangle \otimes |w\rangle\right) = \left(z|v\rangle\right) \otimes |w\rangle = |v\rangle \otimes \left(z|w\rangle\right). \tag{2.42}$$

(2) For arbitrary $|v_1\rangle$ and $|v_2\rangle$ in V and $|w\rangle$ in W,

$$\left(|v_1\rangle + |v_2\rangle\right) \otimes |w\rangle = |v_1\rangle \otimes |w\rangle + |v_2\rangle \otimes |w\rangle. \tag{2.43}$$

(3) For arbitrary $|v\rangle$ in V and $|w_1\rangle$ and $|w_2\rangle$ in W,

$$|v\rangle \otimes \left(|w_1\rangle + |w_2\rangle\right) = |v\rangle \otimes |w_1\rangle + |v\rangle \otimes |w_2\rangle. \tag{2.44}$$

What sorts of linear operators act on the space $V \otimes W$? Suppose $|v\rangle$ and $|w\rangle$ are vectors in V and W, and A and B are linear operators on V and W, respectively. Then we can define a linear operator $A \otimes B$ on $V \otimes W$ by the equation

$$(A \otimes B)(|v\rangle \otimes |w\rangle) \equiv A|v\rangle \otimes B|w\rangle. \tag{2.45}$$

The definition of $A \otimes B$ is then extended to all elements of $V \otimes W$ in the natural way to ensure linearity of $A \otimes B$, that is,

$$(A \otimes B)\left(\sum_i a_i |v_i\rangle \otimes |w_i\rangle\right) \equiv \sum_i a_i A|v_i\rangle \otimes B|w_i\rangle. \tag{2.46}$$

It can be shown that $A \otimes B$ defined in this way is a well-defined linear operator on $V \otimes W$. This notion of the tensor product of two operators extends in the obvious way to the case where $A : V \rightarrow V'$ and $B : W \rightarrow W'$ map between different vector spaces. Indeed, an arbitrary linear operator C mapping $V \otimes W$ to $V' \otimes W'$ can be represented as a linear combination of tensor products of operators mapping V to V' and W to W',

$$C = \sum_i c_i A_i \otimes B_i, \tag{2.47}$$

where by definition

$$\left(\sum_i c_i A_i \otimes B_i\right) |v\rangle \otimes |w\rangle \equiv \sum_i c_i A_i |v\rangle \otimes B_i |w\rangle. \tag{2.48}$$

The inner products on the spaces V and W can be used to define a natural inner product on $V \otimes W$. Define

$$\left(\sum_i a_i |v_i\rangle \otimes |w_i\rangle, \sum_j b_j |v'_j\rangle \otimes |w'_j\rangle\right) \equiv \sum_{ij} a_i^* b_j \langle v_i|v'_j\rangle \langle w_i|w'_j\rangle. \tag{2.49}$$

It can be shown that the function so defined is a well-defined inner product. From this inner product, the inner product space $V \otimes W$ inherits the other structure we are familiar with, such as notions of an adjoint, unitarity, normality, and Hermiticity.

All this discussion is rather abstract. It can be made much more concrete by moving

to a convenient matrix representation known as the *Kronecker product*. Suppose A is an m by n matrix, and B is a p by q matrix. Then we have the matrix representation:

$$A \otimes B \equiv \overbrace{\left[\begin{array}{cccc} A_{11}B & A_{12}B & \ldots & A_{1n}B \\ A_{21}B & A_{22}B & \ldots & A_{2n}B \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1}B & A_{m2}B & \ldots & A_{mn}B \end{array}\right]}^{nq} \left.\begin{array}{c} \\ \\ \\ \\ \end{array}\right\} mp \,. \tag{2.50}$$

In this representation terms like $A_{11}B$ denote p by q submatrices whose entries are proportional to B, with overall proportionality constant A_{11}. For example, the tensor product of the vectors $(1, 2)$ and $(2, 3)$ is the vector

$$\left[\begin{array}{c} 1 \\ 2 \end{array}\right] \otimes \left[\begin{array}{c} 2 \\ 3 \end{array}\right] = \left[\begin{array}{c} 1 \times 2 \\ 1 \times 3 \\ 2 \times 2 \\ 2 \times 3 \end{array}\right] = \left[\begin{array}{c} 2 \\ 3 \\ 4 \\ 6 \end{array}\right] . \tag{2.51}$$

The tensor product of the Pauli matrices X and Y is

$$X \otimes Y = \left[\begin{array}{cc} 0 \cdot Y & 1 \cdot Y \\ 1 \cdot Y & 0 \cdot Y \end{array}\right] = \left[\begin{array}{cccc} 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \\ 0 & -i & 0 & 0 \\ i & 0 & 0 & 0 \end{array}\right] . \tag{2.52}$$

Finally, we mention the useful notation $|\psi\rangle^{\otimes k}$, which means $|\psi\rangle$ tensored with itself k times. For example $|\psi\rangle^{\otimes 2} = |\psi\rangle \otimes |\psi\rangle$. An analogous notation is also used for operators on tensor product spaces.

Exercise 2.26: Let $|\psi\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. Write out $|\psi\rangle^{\otimes 2}$ and $|\psi\rangle^{\otimes 3}$ explicitly, both in terms of tensor products like $|0\rangle|1\rangle$, and using the Kronecker product.

Exercise 2.27: Calculate the matrix representation of the tensor products of the Pauli operators (a) X and Z; (b) I and X; (c) X and I. Is the tensor product commutative?

Exercise 2.28: Show that the transpose, complex conjugation, and adjoint operations distribute over the tensor product,

$$(A \otimes B)^* = A^* \otimes B^*; \; (A \otimes B)^T = A^T \otimes B^T; \; (A \otimes B)^\dagger = A^\dagger \otimes B^\dagger . \tag{2.53}$$

Exercise 2.29: Show that the tensor product of two unitary operators is unitary.

Exercise 2.30: Show that the tensor product of two Hermitian operators is Hermitian.

Exercise 2.31: Show that the tensor product of two positive operators is positive.

Exercise 2.32: Show that the tensor product of two projectors is a projector.

Exercise 2.33: The Hadamard operator on one qubit may be written as

$$H = \frac{1}{\sqrt{2}} \left[(|0\rangle + |1\rangle)\langle 0| + (|0\rangle - |1\rangle)\langle 1| \right] . \tag{2.54}$$

Show explicitly that the Hadamard transform on n qubits, $H^{\otimes n}$, may be written as

$$H^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x,y} (-1)^{x \cdot y} |x\rangle\langle y|. \tag{2.55}$$

Write out an explicit matrix representation for $H^{\otimes 2}$.

2.1.8 Operator functions

There are many important functions which can be defined for operators and matrices. Generally speaking, given a function f from the complex numbers to the complex numbers, it is possible to define a corresponding matrix function on normal matrices (or some subclass, such as the Hermitian matrices) by the following construction. Let $A = \sum_a a|a\rangle\langle a|$ be a spectral decomposition for a normal operator A. Define $f(A) \equiv \sum_a f(a)|a\rangle\langle a|$. A little thought shows that $f(A)$ is uniquely defined. This procedure can be used, for example, to define the square root of a positive operator, the logarithm of a positive-definite operator, or the exponential of a normal operator. As an example,

$$\exp(\theta Z) = \begin{bmatrix} e^{\theta} & 0 \\ 0 & e^{-\theta} \end{bmatrix}, \tag{2.56}$$

since Z has eigenvectors $|0\rangle$ and $|1\rangle$.

Exercise 2.34: Find the square root and logarithm of the matrix

$$\begin{bmatrix} 4 & 3 \\ 3 & 4 \end{bmatrix}. \tag{2.57}$$

Exercise 2.35: (Exponential of the Pauli matrices) Let $\vec{v}$ be any real, three-dimensional unit vector and θ a real number. Prove that

$$\exp(i\theta \vec{v} \cdot \vec{\sigma}) = \cos(\theta) I + i \sin(\theta) \vec{v} \cdot \vec{\sigma}, \tag{2.58}$$

where $\vec{v} \cdot \vec{\sigma} \equiv \sum_{i=1}^{3} v_i \sigma_i$. This exercise is generalized in Problem 2.1 on page 117.

Another important matrix function is the *trace* of a matrix. The trace of A is defined to be the sum of its diagonal elements,

$$\mathrm{tr}(A) \equiv \sum_i A_{ii}. \tag{2.59}$$

The trace is easily seen to be *cyclic*, $\mathrm{tr}(AB) = \mathrm{tr}(BA)$, and *linear*, $\mathrm{tr}(A + B) = \mathrm{tr}(A) + \mathrm{tr}(B)$, $\mathrm{tr}(zA) = z\,\mathrm{tr}(A)$, where A and B are arbitrary matrices, and z is a complex number. Furthermore, from the cyclic property it follows that the trace of a matrix is invariant under the unitary *similarity transformation* $A \rightarrow UAU^{\dagger}$, as $\mathrm{tr}(UAU^{\dagger}) = \mathrm{tr}(U^{\dagger}UA) = \mathrm{tr}(A)$. In light of this result, it makes sense to define the trace of an *operator* A to be the trace of any matrix representation of A. The invariance of the trace under unitary similarity transformations ensures that the trace of an operator is well defined.

As an example of the trace, suppose $|\psi\rangle$ is a unit vector and A is an arbitrary operator. To evaluate $\mathrm{tr}(A|\psi\rangle\langle\psi|)$ use the Gram–Schmidt procedure to extend $|\psi\rangle$ to an

orthonormal basis $|i\rangle$ which includes $|\psi\rangle$ as the first element. Then we have

$$\mathrm{tr}(A|\psi\rangle\langle\psi|) = \sum_i \langle i|A|\psi\rangle\langle\psi|i\rangle \tag{2.60}$$

$$= \langle\psi|A|\psi\rangle. \tag{2.61}$$

This result, that $\mathrm{tr}(A|\psi\rangle\langle\psi|) = \langle\psi|A|\psi\rangle$ is extremely useful in evaluating the trace of an operator.

Exercise 2.36: Show that the Pauli matrices except for I have trace zero.

Exercise 2.37: (Cyclic property of the trace) If A and B are two linear operators show that

$$\mathrm{tr}(AB) = \mathrm{tr}(BA). \tag{2.62}$$

Exercise 2.38: (Linearity of the trace) If A and B are two linear operators, show that

$$\mathrm{tr}(A + B) = \mathrm{tr}(A) + \mathrm{tr}(B) \tag{2.63}$$

and if z is an arbitrary complex number show that

$$\mathrm{tr}(zA) = z\mathrm{tr}(A). \tag{2.64}$$

Exercise 2.39: (The Hilbert–Schmidt inner product on operators) The set L_V of linear operators on a Hilbert space V is obviously a vector space – the sum of two linear operators is a linear operator, zA is a linear operator if A is a linear operator and z is a complex number, and there is a zero element 0. An important additional result is that the vector space L_V can be given a natural inner product structure, turning it into a Hilbert space.

(1) Show that the function $(\cdot,\cdot)$ on $L_V \times L_V$ defined by

$$(A, B) \equiv \mathrm{tr}(A^\dagger B) \tag{2.65}$$

is an inner product function. This inner product is known as the *Hilbert–Schmidt* or *trace* inner product.

(2) If V has d dimensions show that L_V has dimension d^2.

(3) Find an orthonormal basis of Hermitian matrices for the Hilbert space L_V.

2.1.9 The commutator and anti-commutator

The *commutator* between two operators A and B is defined to be

$$[A, B] \equiv AB - BA. \tag{2.66}$$

If $[A, B] = 0$, that is, $AB = BA$, then we say A *commutes* with B. Similarly, the *anti-commutator* of two operators A and B is defined by

$$\{A, B\} \equiv AB + BA; \tag{2.67}$$

we say A *anti-commutes* with B if $\{A, B\} = 0$. It turns out that many important properties of pairs of operators can be deduced from their commutator and anti-commutator. Perhaps the most useful relation is the following connection between the commutator and the property of being able to *simultaneously diagonalize* Hermitian operators A and B,

that is, write $A = \sum_i a_i |i\rangle\langle i|$, $B = \sum_i b_i |i\rangle\langle i|$, where $|i\rangle$ is some common orthonormal set of eigenvectors for A and B.

Theorem 2.2: **(Simultaneous diagonalization theorem)** Suppose A and B are Hermitian operators. Then $[A, B] = 0$ if and only if there exists an orthonormal basis such that both A and B are diagonal with respect to that basis. We say that A and B are *simultaneously diagonalizable* in this case.

This result connects the commutator of two operators, which is often easy to compute, to the property of being simultaneously diagonalizable, which is *a priori* rather difficult to determine. As an example, consider that

$$[X, Y] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} - \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.68}$$

$$= 2i \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.69}$$

$$= 2iZ\,, \tag{2.70}$$

so X and Y do not commute. You have already shown, in Exercise 2.11, that X and Y do not have common eigenvectors, as we expect from the simultaneous diagonalization theorem.

Proof
You can (and should!) easily verify that if A and B are diagonal in the same orthonormal basis then $[A, B] = 0$. To show the converse, let $|a, j\rangle$ be an orthonormal basis for the eigenspace V_a of A with eigenvalue a; the index j is used to label possible degeneracies. Note that

$$AB|a, j\rangle = BA|a, j\rangle = aB|a, j\rangle, \tag{2.71}$$

and therefore $B|a, j\rangle$ is an element of the eigenspace V_a. Let P_a denote the projector onto the space V_a and define $B_a \equiv P_a B P_a$. It is easy to see that the restriction of B_a to the space V_a is Hermitian on V_a, and therefore has a spectral decomposition in terms of an orthonormal set of eigenvectors which span the space V_a. Let's call these eigenvectors $|a, b, k\rangle$, where the indices a and b label the eigenvalues of A and B_a, and k is an extra index to allow for the possibility of a degenerate B_a. Note that $B|a, b, k\rangle$ is an element of V_a, so $B|a, b, k\rangle = P_a B|a, b, k\rangle$. Moreover we have $P_a|a, b, k\rangle = |a, b, k\rangle$, so

$$B|a, b, k\rangle = P_a B P_a |a, b, k\rangle = b|a, b, k\rangle. \tag{2.72}$$

It follows that $|a, b, k\rangle$ is an eigenvector of B with eigenvalue b, and therefore $|a, b, k\rangle$ is an orthonormal set of eigenvectors of both A and B, spanning the entire vector space on which A and B are defined. That is, A and B are simultaneously diagonalizable. □

Exercise 2.40: (Commutation relations for the Pauli matrices) Verify the commutation relations

$$[X, Y] = 2iZ; \quad [Y, Z] = 2iX; \quad [Z, X] = 2iY. \tag{2.73}$$

There is an elegant way of writing this using ϵ_{jkl}, the antisymmetric tensor on

three indices, for which $\epsilon_{jkl} = 0$ except for $\epsilon_{123} = \epsilon_{231} = \epsilon_{312} = 1$, and $\epsilon_{321} = \epsilon_{213} = \epsilon_{132} = -1$:

$$[\sigma_j, \sigma_k] = 2i \sum_{l=1}^{3} \epsilon_{jkl} \sigma_l. \tag{2.74}$$

Exercise 2.41: (**Anti-commutation relations for the Pauli matrices**) Verify the anti-commutation relations

$$\{\sigma_i, \sigma_j\} = 0 \tag{2.75}$$

where $i \neq j$ are both chosen from the set $1, 2, 3$. Also verify that ($i = 0, 1, 2, 3$)

$$\sigma_i^2 = I. \tag{2.76}$$

Exercise 2.42: Verify that

$$AB = \frac{[A, B] + \{A, B\}}{2}. \tag{2.77}$$

Exercise 2.43: Show that for $j, k = 1, 2, 3$,

$$\sigma_j \sigma_k = \delta_{jk} I + i \sum_{l=1}^{3} \epsilon_{jkl} \sigma_l. \tag{2.78}$$

Exercise 2.44: Suppose $[A, B] = 0$, $\{A, B\} = 0$, and A is invertible. Show that B must be 0.

Exercise 2.45: Show that $[A, B]^\dagger = [B^\dagger, A^\dagger]$.

Exercise 2.46: Show that $[A, B] = -[B, A]$.

Exercise 2.47: Suppose A and B are Hermitian. Show that $i[A, B]$ is Hermitian.

2.1.10 The polar and singular value decompositions

The *polar* and *singular value* decompositions are useful ways of breaking linear operators up into simpler parts. In particular, these decompositions allow us to break general linear operators up into products of unitary operators and positive operators. While we don't understand the structure of general linear operators terribly well, we do understand unitary operators and positive operators in quite some detail. The polar and singular value decompositions allow us to apply this understanding to better understand general linear operators.

Theorem 2.3: (**Polar decomposition**) Let A be a linear operator on a vector space V. Then there exists unitary U and positive operators J and K such that

$$A = UJ = KU, \tag{2.79}$$

where the unique positive operators J and K satisfying these equations are defined by $J \equiv \sqrt{A^\dagger A}$ and $K \equiv \sqrt{AA^\dagger}$. Moreover, if A is invertible then U is unique.

We call the expression $A = UJ$ the *left polar decomposition* of A, and $A = KU$ the *right polar decomposition* of A. Most often, we'll omit the 'right' or 'left' nomenclature, and use the term 'polar decomposition' for both expressions, with context indicating which is meant.

Proof
$J \equiv \sqrt{A^\dagger A}$ is a positive operator, so it can be given a spectral decomposition, $J = \sum_i \lambda_i |i\rangle\langle i|$ ($\lambda_i \geq 0$). Define $|\psi_i\rangle \equiv A|i\rangle$. From the definition, we see that $\langle\psi_i|\psi_i\rangle = \lambda_i^2$. Consider for now only those i for which $\lambda_i \neq 0$. For those i define $|e_i\rangle \equiv |\psi_i\rangle/\lambda_i$, so the $|e_i\rangle$ are normalized. Moreover, they are orthogonal, since if $i \neq j$ then $\langle e_i|e_j\rangle = \langle i|A^\dagger A|j\rangle/\lambda_i\lambda_j = \langle i|J^2|j\rangle/\lambda_i\lambda_j = 0$.

We have been considering i such that $\lambda_i \neq 0$. Now use the Gram–Schmidt procedure to extend the orthonormal set $|e_i\rangle$ so it forms an orthonormal basis, which we also label $|e_i\rangle$. Define a unitary operator $U \equiv \sum_i |e_i\rangle\langle i|$. When $\lambda_i \neq 0$ we have $UJ|i\rangle = \lambda_i|e_i\rangle = |\psi_i\rangle = A|i\rangle$. When $\lambda_i = 0$ we have $UJ|i\rangle = 0 = |\psi_i\rangle$. We have proved that the action of A and UJ agree on the basis $|i\rangle$, and thus that $A = UJ$.

J is unique, since multiplying $A = UJ$ on the left by the adjoint equation $A^\dagger = JU^\dagger$ gives $J^2 = A^\dagger A$, from which we see that $J = \sqrt{A^\dagger A}$, uniquely. A little thought shows that if A is invertible, then so is J, so U is uniquely determined by the equation $U = AJ^{-1}$. The proof of the right polar decomposition follows, since $A = UJ = UJU^\dagger U = KU$, where $K \equiv UJU^\dagger$ is a positive operator. Since $AA^\dagger = KUU^\dagger K = K^2$ we must have $K = \sqrt{AA^\dagger}$, as claimed. □

The singular value decomposition combines the polar decomposition and the spectral theorem.

Corollary 2.4: (**Singular value decomposition**) Let A be a square matrix. Then there exist unitary matrices U and V, and a diagonal matrix D with non-negative entries such that

$$A = UDV\,. \tag{2.80}$$

The diagonal elements of D are called the *singular values* of A.

Proof
By the polar decomposition, $A = SJ$, for unitary S, and positive J. By the spectral theorem, $J = TDT^\dagger$, for unitary T and diagonal D with non-negative entries. Setting $U \equiv ST$ and $V \equiv T^\dagger$ completes the proof. □

Exercise 2.48: What is the polar decomposition of a positive matrix P? Of a unitary matrix U? Of a Hermitian matrix, H?

Exercise 2.49: Express the polar decomposition of a normal matrix in the outer product representation.

Exercise 2.50: Find the left and right polar decompositions of the matrix

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{2.81}$$

2.2 The postulates of quantum mechanics

All understanding begins with our not accepting the world as it appears.
– Alan Kay

The most incomprehensible thing about the world is that it is comprehensible.
– Albert Einstein

Quantum mechanics is a mathematical framework for the development of physical theories. On its own quantum mechanics doesn't tell you what laws a physical system must obey, but it does provide a mathematical and conceptual framework for the development of such laws. In the next few sections we give a complete description of the basic postulates of quantum mechanics. These postulates provide a connection between the physical world and the mathematical formalism of quantum mechanics.

The postulates of quantum mechanics were derived after a long process of trial and (mostly) error, which involved a considerable amount of guessing and fumbling by the originators of the theory. Don't be surprised if the motivation for the postulates is not always clear; even to experts the basic postulates of quantum mechanics appear surprising. What you should expect to gain in the next few sections is a good working grasp of the postulates – how to apply them, and when.

2.2.1 State space

The first postulate of quantum mechanics sets up the arena in which quantum mechanics takes place. The arena is our familiar friend from linear algebra, Hilbert space.

> **Postulate 1**: Associated to any isolated physical system is a complex vector space with inner product (that is, a Hilbert space) known as the *state space* of the system. The system is completely described by its *state vector*, which is a unit vector in the system's state space.

Quantum mechanics does *not* tell us, for a given physical system, what the state space of that system is, nor does it tell us what the state vector of the system is. Figuring that out for a *specific* system is a difficult problem for which physicists have developed many intricate and beautiful rules. For example, there is the wonderful theory of quantum electrodynamics (often known as QED), which describes how atoms and light interact. One aspect of QED is that it tells us what state spaces to use to give quantum descriptions of atoms and light. We won't be much concerned with the intricacies of theories like QED (except in so far as they apply to physical realizations, in Chapter 7), as we are mostly interested in the general framework provided by quantum mechanics. For our purposes it will be sufficient to make some very simple (and reasonable) assumptions about the state spaces of the systems we are interested in, and stick with those assumptions.

The simplest quantum mechanical system, and the system which we will be most concerned with, is the *qubit*. A qubit has a two-dimensional state space. Suppose $|0\rangle$ and $|1\rangle$ form an orthonormal basis for that state space. Then an arbitrary state vector in the state space can be written

$$|\psi\rangle = a|0\rangle + b|1\rangle, \tag{2.82}$$

where a and b are complex numbers. The condition that $|\psi\rangle$ be a unit vector, $\langle\psi|\psi\rangle = 1$, is therefore equivalent to $|a|^2 + |b|^2 = 1$. The condition $\langle\psi|\psi\rangle = 1$ is often known as the *normalization condition* for state vectors.

We will take the qubit as our fundamental quantum mechanical system. Later, in Chapter 7, we will see that there are real physical systems which may be described in terms of qubits. For now, though, it is sufficient to think of qubits in abstract terms, without reference to a specific realization. Our discussions of qubits will always be referred to some orthonormal set of basis vectors, $|0\rangle$ and $|1\rangle$, which should be thought of as being fixed in advance. Intuitively, the states $|0\rangle$ and $|1\rangle$ are analogous to the two values 0 and 1 which a bit may take. The way a qubit differs from a bit is that *superpositions* of these two states, of the form $a|0\rangle + b|1\rangle$, can also exist, in which it is not possible to say that the qubit is definitely in the state $|0\rangle$, or definitely in the state $|1\rangle$.

We conclude with some useful terminology which is often used in connection with the description of quantum states. We say that any linear combination $\sum_i \alpha_i|\psi_i\rangle$ is a superposition of the states $|\psi_i\rangle$ with *amplitude* α_i for the state $|\psi_i\rangle$. So, for example, the state

$$\frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{2.83}$$

is a superposition of the states $|0\rangle$ and $|1\rangle$ with amplitude $1/\sqrt{2}$ for the state $|0\rangle$, and amplitude $-1/\sqrt{2}$ for the state $|1\rangle$.

2.2.2 Evolution

How does the state, $|\psi\rangle$, of a quantum mechanical system change with time? The following postulate gives a prescription for the description of such state changes.

> **Postulate 2**: The evolution of a *closed* quantum system is described by a *unitary transformation*. That is, the state $|\psi\rangle$ of the system at time t_1 is related to the state $|\psi'\rangle$ of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2,
>
> $$|\psi'\rangle = U|\psi\rangle\,. \tag{2.84}$$

Just as quantum mechanics does not tell us the state space or quantum state of a *particular* quantum system, it does not tell us which unitary operators U describe real-world quantum dynamics. Quantum mechanics merely assures us that the evolution of any closed quantum system may be described in such a way. An obvious question to ask is: what unitary operators are natural to consider? In the case of single qubits, it turns out that *any* unitary operator at all can be realized in realistic systems.

Let's look at a few examples of unitary operators on a single qubit which are important in quantum computation and quantum information. We have already seen several examples of such unitary operators – the Pauli matrices, defined in Section 2.1.3, and the quantum gates described in Chapter 1. As remarked in Section 1.3.1, the X matrix is often known as the quantum NOT gate, by analogy to the classical NOT gate. The X and Z Pauli matrices are also sometimes referred to as the *bit flip* and *phase flip* matrices: the X matrix takes $|0\rangle$ to $|1\rangle$, and $|1\rangle$ to $|0\rangle$, thus earning the name bit flip; and the Z matrix leaves $|0\rangle$ invariant, and takes $|1\rangle$ to $-|1\rangle$, with the extra factor of -1 added known as a *phase factor*, thus justifying the term phase flip. We will not use the term phase flip for

Z very often, since it is easily confused with the phase gate to be defined in Chapter 4. (Section 2.2.7 contains more discussion of the many uses of the term 'phase'.)

Another interesting unitary operator is the *Hadamard gate*, which we denote H. This has the action $H|0\rangle \equiv (|0\rangle + |1\rangle)/\sqrt{2}$, $H|1\rangle \equiv (|0\rangle - |1\rangle)/\sqrt{2}$, and corresponding matrix representation

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{2.85}$$

Exercise 2.51: Verify that the Hadamard gate H is unitary.

Exercise 2.52: Verify that $H^2 = I$.

Exercise 2.53: What are the eigenvalues and eigenvectors of H?

Postulate 2 requires that the system being described be closed. That is, it is not interacting in any way with other systems. In reality, of course, all systems (except the Universe as a whole) interact at least somewhat with other systems. Nevertheless, there are interesting systems which can be described to a good approximation as being closed, and which are described by unitary evolution to some good approximation. Furthermore, at least in principle every open system can be described as part of a larger closed system (the Universe) which is undergoing unitary evolution. Later, we'll introduce more tools which allow us to describe systems which are not closed, but for now we'll continue with the description of the evolution of closed systems.

Postulate 2 describes how the quantum states of a closed quantum system at two different times are related. A more refined version of this postulate can be given which describes the evolution of a quantum system in *continuous time*. From this more refined postulate we will recover Postulate 2. Before we state the revised postulate, it is worth pointing out two things. First, a notational remark. The operator H appearing in the following discussion is not the same as the Hadamard operator, which we just introduced. Second, the following postulate makes use of the apparatus of differential equations. Readers with little background in the study of differential equations should be reassured that they will not be necessary for much of the book, with the exception of parts of Chapter 7, on real physical implementations of quantum information processing.

> **Postulate 2′**: The time evolution of the state of a closed quantum system is described by the *Schrödinger equation*,
>
> $$i\hbar \frac{d|\psi\rangle}{dt} = H|\psi\rangle. \tag{2.86}$$
>
> In this equation, $\hbar$ is a physical constant known as *Planck's constant* whose value must be experimentally determined. The exact value is not important to us. In practice, it is common to absorb the factor $\hbar$ into H, effectively setting $\hbar = 1$. H is a fixed Hermitian operator known as the *Hamiltonian* of the closed system.

If we know the Hamiltonian of a system, then (together with a knowledge of $\hbar$) we understand its dynamics completely, at least in principle. In general figuring out the Hamiltonian needed to describe a particular physical system is a very difficult problem – much of twentieth century physics has been concerned with this problem – which requires substantial input from experiment in order to be answered. From our point of

view this is a problem of *detail* to be addressed by physical theories built within the framework of quantum mechanics – what Hamiltonian do we need to describe atoms in such-and-such a configuration – and is not a question that needs to be addressed by the theory of quantum mechanics itself. Most of the time in our discussion of quantum computation and quantum information we won't need to discuss Hamiltonians, and when we do, we will usually just posit that some matrix is the Hamiltonian as a starting point, and proceed from there, without attempting to justify the use of that Hamiltonian.

Because the Hamiltonian is a Hermitian operator it has a spectral decomposition

$$H = \sum_E E|E\rangle\langle E|, \tag{2.87}$$

with eigenvalues E and corresponding normalized eigenvectors $|E\rangle$. The states $|E\rangle$ are conventionally referred to as *energy eigenstates*, or sometimes as *stationary states*, and E is the *energy* of the state $|E\rangle$. The lowest energy is known as the *ground state energy* for the system, and the corresponding energy eigenstate (or eigenspace) is known as the *ground state*. The reason the states $|E\rangle$ are sometimes known as stationary states is because their only change in time is to acquire an overall numerical factor,

$$|E\rangle \rightarrow \exp(-iEt/\hbar)|E\rangle. \tag{2.88}$$

As an example, suppose a single qubit has Hamiltonian

$$H = \hbar\omega X. \tag{2.89}$$

In this equation ω is a parameter that, in practice, needs to be experimentally determined. We won't worry about the parameter overly much here – the point is to give you a feel for the sort of Hamiltonians that are sometimes written down in the study of quantum computation and quantum information. The energy eigenstates of this Hamiltonian are obviously the same as the eigenstates of X, namely $(|0\rangle + |1\rangle)/\sqrt{2}$ and $(|0\rangle - |1\rangle)/\sqrt{2}$, with corresponding energies $\hbar\omega$ and $-\hbar\omega$. The ground state is therefore $(|0\rangle - |1\rangle)/\sqrt{2}$, and the ground state energy is $-\hbar\omega$.

What is the connection between the Hamiltonian picture of dynamics, Postulate 2′, and the unitary operator picture, Postulate 2? The answer is provided by writing down the solution to Schrödinger's equation, which is easily verified to be:

$$|\psi(t_2)\rangle = \exp\left[\frac{-iH(t_2 - t_1)}{\hbar}\right]|\psi(t_1)\rangle = U(t_1, t_2)|\psi(t_1)\rangle\,, \tag{2.90}$$

where we define

$$U(t_1, t_2) \equiv \exp\left[\frac{-iH(t_2 - t_1)}{\hbar}\right]. \tag{2.91}$$

You will show in the exercises that this operator is unitary, and furthermore, that any unitary operator U can be realized in the form $U = \exp(iK)$ for some Hermitian operator K. There is therefore a one-to-one correspondence between the discrete-time description of dynamics using unitary operators, and the continuous time description using Hamiltonians. For most of the book we use the unitary formulation of quantum dynamics.

Exercise 2.54: Suppose A and B are commuting Hermitian operators. Prove that $\exp(A)\exp(B) = \exp(A + B)$. (*Hint*: Use the results of Section 2.1.9.)

Exercise 2.55: Prove that $U(t_1, t_2)$ defined in Equation (2.91) is unitary.

Exercise 2.56: Use the spectral decomposition to show that $K \equiv -i \log(U)$ is Hermitian for any unitary U, and thus $U = \exp(iK)$ for some Hermitian K.

In quantum computation and quantum information we often speak of *applying* a unitary operator to a particular quantum system. For example, in the context of quantum circuits we may speak of applying the unitary gate X to a single qubit. Doesn't this contradict what we said earlier, about unitary operators describing the evolution of a *closed* quantum system? After all, if we are 'applying' a unitary operator, then that implies that there is an external 'we' who is interacting with the quantum system, and the system is not closed.

An example of this occurs when a laser is focused on an atom. After a lot of thought and hard work it is possible to write down a Hamiltonian describing the total atom–laser system. The interesting thing is that when we write down the Hamiltonian for the atom–laser system and consider the effects on the atom alone, the behavior of the state vector of the atom turns out to be almost but not quite perfectly described by another Hamiltonian, the *atomic Hamiltonian*. The atomic Hamiltonian contains terms related to laser intensity, and other parameters of the laser, which we can vary at will. It is *as if* the evolution of the atom were being described by a Hamiltonian which we can vary at will, despite the atom not being a closed system.

More generally, for many systems like this it turns out to be possible to write down a *time-varying* Hamiltonian for a quantum system, in which the Hamiltonian for the system is not a constant, but varies according to some parameters which are under an experimentalist's control, and which may be changed during the course of an experiment. The system is not, therefore, closed, but it does evolve according to Schrödinger's equation with a time-varying Hamiltonian, to some good approximation.

The upshot is that to begin we will often describe the evolution of quantum systems – even systems which aren't closed – using unitary operators. The main exception to this, quantum measurement, will be described in the next section. Later on we will investigate in more detail possible deviations from unitary evolution due to the interaction with other systems, and understand more precisely the dynamics of realistic quantum systems.

2.2.3 Quantum measurement

We postulated that closed quantum systems evolve according to unitary evolution. The evolution of systems which don't interact with the rest of the world is all very well, but there must also be times when the experimentalist and their experimental equipment – an external physical system in other words – observes the system to find out what is going on inside the system, an interaction which makes the system no longer closed, and thus not necessarily subject to unitary evolution. To explain what happens when this is done, we introduce Postulate 3, which provides a means for describing the effects of measurements on quantum systems.

> **Postulate 3**: Quantum measurements are described by a collection $\{M_m\}$ of *measurement operators*. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|\psi\rangle$ immediately before the measurement then the probability that result m occurs is

given by

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle\,, \tag{2.92}$$

and the state of the system after the measurement is

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}\,. \tag{2.93}$$

The measurement operators satisfy the *completeness equation*,

$$\sum_m M_m^\dagger M_m = I\,. \tag{2.94}$$

The completeness equation expresses the fact that probabilities sum to one:

$$1 = \sum_m p(m) = \sum_m \langle\psi|M_m^\dagger M_m|\psi\rangle\,. \tag{2.95}$$

This equation being satisfied for all $|\psi\rangle$ is equivalent to the completeness equation. However, the completeness equation is much easier to check directly, so that's why it appears in the statement of the postulate.

A simple but important example of a measurement is the *measurement of a qubit in the computational basis*. This is a measurement on a single qubit with two outcomes defined by the two measurement operators $M_0 = |0\rangle\langle 0|$, $M_1 = |1\rangle\langle 1|$. Observe that each measurement operator is Hermitian, and that $M_0^2 = M_0, M_1^2 = M_1$. Thus the completeness relation is obeyed, $I = M_0^\dagger M_0 + M_1^\dagger M_1 = M_0 + M_1$. Suppose the state being measured is $|\psi\rangle = a|0\rangle + b|1\rangle$. Then the probability of obtaining measurement outcome 0 is

$$p(0) = \langle\psi|M_0^\dagger M_0|\psi\rangle = \langle\psi|M_0|\psi\rangle = |a|^2. \tag{2.96}$$

Similarly, the probability of obtaining the measurement outcome 1 is $p(1) = |b|^2$. The state after measurement in the two cases is therefore

$$\frac{M_0|\psi\rangle}{|a|} = \frac{a}{|a|}|0\rangle \tag{2.97}$$

$$\frac{M_1|\psi\rangle}{|b|} = \frac{b}{|b|}|1\rangle. \tag{2.98}$$

We will see in Section 2.2.7 that multipliers like $a/|a|$, which have modulus one, can effectively be ignored, so the two post-measurement states are effectively $|0\rangle$ and $|1\rangle$, just as described in Chapter 1.

The status of Postulate 3 as a fundamental postulate intrigues many people. Measuring devices are quantum mechanical systems, so the quantum system being measured and the measuring device together are part of a larger, isolated, quantum mechanical system. (It may be necessary to include quantum systems other than the system being measured and the measuring device to obtain a completely isolated system, but the point is that this can be done.) According to Postulate 2, the evolution of this larger isolated system can be described by a unitary evolution. Might it be possible to *derive* Postulate 3 as a consequence of this picture? Despite considerable investigation along these lines there is still disagreement between physicists about whether or not this is possible. We, however, are going to take the very pragmatic approach that in practice it is clear when to apply

Postulate 2 and when to apply Postulate 3, and not worry about deriving one postulate from the other.

Over the next few sections we apply Postulate 3 to several elementary but important measurement scenarios. Section 2.2.4 examines the problem of *distinguishing* a set of quantum states. Section 2.2.5 explains a special case of Postulate 3, the *projective* or *von Neumann* measurements. Section 2.2.6 explains another special case of Postulate 3, known as *POVM* measurements. Many introductions to quantum mechanics only discuss projective measurements, omitting a full discussion of Postulate 3 or of POVM elements. For this reason we have included Box 2.5 on page 91 which comments on the relationship between the different classes of measurement we describe.

Exercise 2.57: (Cascaded measurements are single measurements) Suppose $\{L_l\}$ and $\{M_m\}$ are two sets of measurement operators. Show that a measurement defined by the measurement operators $\{L_l\}$ followed by a measurement defined by the measurement operators $\{M_m\}$ is physically equivalent to a single measurement defined by measurement operators $\{N_{lm}\}$ with the representation $N_{lm} \equiv M_m L_l$.

2.2.4 Distinguishing quantum states

An important application of Postulate 3 is to the problem of *distinguishing quantum states*. In the classical world, distinct states of an object are usually distinguishable, at least in principle. For example, we can always identify whether a coin has landed heads or tails, at least in the ideal limit. Quantum mechanically, the situation is more complicated. In Section 1.6 we gave a plausible argument that non-orthogonal quantum states cannot be distinguished. With Postulate 3 as a firm foundation we can now give a much more convincing demonstration of this fact.

Distinguishability, like many ideas in quantum computation and quantum information, is most easily understood using the metaphor of a game involving two parties, Alice and Bob. Alice chooses a state $|\psi_i\rangle$ ($1 \leq i \leq n$) from some fixed set of states known to both parties. She gives the state $|\psi_i\rangle$ to Bob, whose task it is to identify the index i of the state Alice has given him.

Suppose the states $|\psi_i\rangle$ are orthonormal. Then Bob can do a quantum measurement to *distinguish* these states, using the following procedure. Define measurement operators $M_i \equiv |\psi_i\rangle\langle\psi_i|$, one for each possible index i, and an additional measurement operator M_0 defined as the positive square root of the positive operator $I - \sum_{i \neq 0} |\psi_i\rangle\langle\psi_i|$. These operators satisfy the completeness relation, and if the state $|\psi_i\rangle$ is prepared then $p(i) = \langle\psi_i|M_i|\psi_i\rangle = 1$, so the result i occurs with certainty. Thus, it is possible to reliably distinguish the orthonormal states $|\psi_i\rangle$.

By contrast, if the states $|\psi_i\rangle$ are not orthonormal then we can prove that there is *no quantum measurement capable of distinguishing the states*. The idea is that Bob will do a measurement described by measurement operators M_j, with outcome j. Depending on the outcome of the measurement Bob tries to guess what the index i was using some rule, $i = f(j)$, where $f(\cdot)$ represents the rule he uses to make the guess. The key to why Bob can't distinguish non-orthogonal states $|\psi_1\rangle$ and $|\psi_2\rangle$ is the observation that $|\psi_2\rangle$ can be decomposed into a (non-zero) component parallel to $|\psi_1\rangle$, and a component orthogonal to $|\psi_1\rangle$. Suppose j is a measurement outcome such that $f(j) = 1$, that is, Bob guesses that the state was $|\psi_1\rangle$ when he observes j. But because of the component of $|\psi_2\rangle$ parallel

to $|\psi_1\rangle$, there is a non-zero probability of getting outcome j when $|\psi_2\rangle$ is prepared, so sometimes Bob will make an error identifying which state was prepared. A more rigorous argument that non-orthogonal states can't be distinguished is given in Box 2.3, but this captures the essential idea.

Box 2.3: Proof that non-orthogonal states can't be reliably distinguished

A proof by contradiction shows that no measurement distinguishing the non-orthogonal states $|\psi_1\rangle$ and $|\psi_2\rangle$ is possible. Suppose such a measurement is possible. If the state $|\psi_1\rangle$ ($|\psi_2\rangle$) is prepared then the probability of measuring j such that $f(j) = 1$ ($f(j) = 2$) must be 1. Defining $E_i \equiv \sum_{j:f(j)=i} M_j^\dagger M_j$, these observations may be written as:

$$\langle\psi_1|E_1|\psi_1\rangle = 1; \quad \langle\psi_2|E_2|\psi_2\rangle = 1. \tag{2.99}$$

Since $\sum_i E_i = I$ it follows that $\sum_i \langle\psi_1|E_i|\psi_1\rangle = 1$, and since $\langle\psi_1|E_1|\psi_1\rangle = 1$ we must have $\langle\psi_1|E_2|\psi_1\rangle = 0$, and thus $\sqrt{E_2}|\psi_1\rangle = 0$. Suppose we decompose $|\psi_2\rangle = \alpha|\psi_1\rangle + \beta|\varphi\rangle$, where $|\varphi\rangle$ is orthonormal to $|\psi_1\rangle$, $|\alpha|^2 + |\beta|^2 = 1$, and $|\beta| < 1$ since $|\psi_1\rangle$ and $|\psi_2\rangle$ are not orthogonal. Then $\sqrt{E_2}|\psi_2\rangle = \beta\sqrt{E_2}|\varphi\rangle$, which implies a contradiction with (2.99), as

$$\langle\psi_2|E_2|\psi_2\rangle = |\beta|^2\langle\varphi|E_2|\varphi\rangle \leq |\beta|^2 < 1, \tag{2.100}$$

where the second last inequality follows from the observation that

$$\langle\varphi|E_2|\varphi\rangle \leq \sum_i \langle\varphi|E_i|\varphi\rangle = \langle\varphi|\varphi\rangle = 1. \tag{2.101}$$

2.2.5 Projective measurements

In this section we explain an important special case of the general measurement postulate, Postulate 3. This special class of measurements is known as *projective measurements*. For many applications of quantum computation and quantum information we will be concerned primarily with projective measurements. Indeed, projective measurements actually turn out to be *equivalent* to the general measurement postulate, when they are augmented with the ability to perform unitary transformations, as described in Postulate 2. We will explain this equivalence in detail in Section 2.2.8, as the statement of the measurement postulate for projective measurements is superficially rather different from the general postulate, Postulate 3.

Projective measurements: A projective measurement is described by an *observable*, M, a Hermitian operator on the state space of the system being observed. The observable has a spectral decomposition,

$$M = \sum_m m P_m, \tag{2.102}$$

where P_m is the projector onto the eigenspace of M with eigenvalue m. The possible outcomes of the measurement correspond to the eigenvalues, m, of the observable. Upon measuring the state $|\psi\rangle$, the probability of getting result m is

given by

$$p(m) = \langle \psi | P_m | \psi \rangle \, . \tag{2.103}$$

Given that outcome m occurred, the state of the quantum system immediately after the measurement is

$$\frac{P_m |\psi\rangle}{\sqrt{p(m)}} \, . \tag{2.104}$$

Projective measurements can be understood as a special case of Postulate 3. Suppose the measurement operators in Postulate 3, in addition to satisfying the completeness relation $\sum_m M_m^\dagger M_m = I$, also satisfy the conditions that M_m are orthogonal projectors, that is, the M_m are Hermitian, and $M_m M_{m'} = \delta_{m,m'} M_m$. With these additional restrictions, Postulate 3 reduces to a projective measurement as just defined.

Projective measurements have many nice properties. In particular, it is very easy to calculate average values for projective measurements. By definition, the average (see Appendix 1 for elementary definitions and results in probability theory) value of the measurement is

$$\begin{aligned} \mathbf{E}(M) &= \sum_m m \, p(m) && (2.110) \\ &= \sum_m m \langle \psi | P_m | \psi \rangle && (2.111) \\ &= \langle \psi | \left(\sum_m m P_m \right) | \psi \rangle && (2.112) \\ &= \langle \psi | M | \psi \rangle . && (2.113) \end{aligned}$$

This is a useful formula, which simplifies many calculations. The average value of the observable M is often written $\langle M \rangle \equiv \langle \psi | M | \psi \rangle$. From this formula for the average follows a formula for the standard deviation associated to observations of M,

$$\begin{aligned} [\Delta(M)]^2 &= \langle (M - \langle M \rangle)^2 \rangle && (2.114) \\ &= \langle M^2 \rangle - \langle M \rangle^2 . && (2.115) \end{aligned}$$

The standard deviation is a measure of the typical spread of the observed values upon measurement of M. In particular, if we perform a large number of experiments in which the state $|\psi\rangle$ is prepared and the observable M is measured, then the standard deviation $\Delta(M)$ of the observed values is determined by the formula $\Delta(M) = \sqrt{\langle M^2 \rangle - \langle M \rangle^2}$. This formulation of measurement and standard deviations in terms of observables gives rise in an elegant way to results such as the *Heisenberg uncertainty principle* (see Box 2.4).

Exercise 2.58: Suppose we prepare a quantum system in an eigenstate $|\psi\rangle$ of some observable M, with corresponding eigenvalue m. What is the average observed value of M, and the standard deviation?

Two widely used nomenclatures for measurements deserve emphasis. Rather than giving an observable to describe a projective measurement, often people simply list a complete set of orthogonal projectors P_m satisfying the relations $\sum_m P_m = I$ and $P_m P_{m'} =$

Box 2.4: The Heisenberg uncertainty principle

Perhaps the best known result of quantum mechanics is the *Heisenberg uncertainty principle*. Suppose A and B are two Hermitian operators, and $|\psi\rangle$ is a quantum state. Suppose $\langle\psi|AB|\psi\rangle = x + iy$, where x and y are real. Note that $\langle\psi|[A,B]|\psi\rangle = 2iy$ and $\langle\psi|\{A,B\}|\psi\rangle = 2x$. This implies that

$$|\langle\psi|[A,B]|\psi\rangle|^2 + |\langle\psi|\{A,B\}|\psi\rangle|^2 = 4\,|\langle\psi|AB|\psi\rangle|^2\,. \tag{2.105}$$

By the Cauchy–Schwarz inequality

$$|\langle\psi|AB|\psi\rangle|^2 \leq \langle\psi|A^2|\psi\rangle\langle\psi|B^2|\psi\rangle, \tag{2.106}$$

which combined with Equation (2.105) and dropping a non-negative term gives

$$|\langle\psi|[A,B]|\psi\rangle|^2 \leq 4\langle\psi|A^2|\psi\rangle\langle\psi|B^2|\psi\rangle. \tag{2.107}$$

Suppose C and D are two observables. Substituting $A = C - \langle C\rangle$ and $B = D - \langle D\rangle$ into the last equation, we obtain Heisenberg's uncertainty principle as it is usually stated:

$$\Delta(C)\Delta(D) \geq \frac{|\langle\psi|[C,D]|\psi\rangle|}{2}\,. \tag{2.108}$$

You should be wary of a common misconception about the uncertainty principle, that measuring an observable C to some 'accuracy' $\Delta(C)$ causes the value of D to be 'disturbed' by an amount $\Delta(D)$ in such a way that some sort of inequality similar to (2.108) is satisfied. While it is true that measurements in quantum mechanics cause disturbance to the system being measured, this is most emphatically *not* the content of the uncertainty principle.

The correct interpretation of the uncertainty principle is that if we prepare a large number of quantum systems in identical states, $|\psi\rangle$, and then perform measurements of C on some of those systems, and of D in others, then the standard deviation $\Delta(C)$ of the C results times the standard deviation $\Delta(D)$ of the results for D will satisfy the inequality (2.108).

As an example of the uncertainty principle, consider the observables X and Y when measured for the quantum state $|0\rangle$. In Equation (2.70) we showed that $[X,Y] = 2iZ$, so the uncertainty principle tells us that

$$\Delta(X)\Delta(Y) \geq \langle 0|Z|0\rangle = 1\,. \tag{2.109}$$

One elementary consequence of this is that $\Delta(X)$ and $\Delta(Y)$ must both be strictly greater than 0, as can be verified by direct calculation.

$\delta_{mm'}P_m$. The corresponding observable implicit in this usage is $M = \sum_m mP_m$. Another widely used phrase, to 'measure in a basis $|m\rangle$', where $|m\rangle$ form an orthonormal basis, simply means to perform the projective measurement with projectors $P_m = |m\rangle\langle m|$.

Let's look at an example of projective measurements on single qubits. First is the measurement of the observable Z. This has eigenvalues +1 and -1 with corresponding eigenvectors $|0\rangle$ and $|1\rangle$. Thus, for example, measurement of Z on the state $|\psi\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ gives the result +1 with probability $\langle\psi|0\rangle\langle 0|\psi\rangle = 1/2$, and similarly the

result -1 with probability $1/2$. More generally, suppose $\vec{v}$ is any real three-dimensional unit vector. Then we can define an observable:

$$\vec{v} \cdot \vec{\sigma} \equiv v_1 \sigma_1 + v_2 \sigma_2 + v_3 \sigma_3. \tag{2.116}$$

Measurement of this observable is sometimes referred to as a 'measurement of spin along the $\vec{v}$ axis', for historical reasons. The following two exercises encourage you to work out some elementary but important properties of such a measurement.

Exercise 2.59: Suppose we have qubit in the state $|0\rangle$, and we measure the observable X. What is the average value of X? What is the standard deviation of X?

Exercise 2.60: Show that $\vec{v} \cdot \vec{\sigma}$ has eigenvalues ± 1, and that the projectors onto the corresponding eigenspaces are given by $P_{\pm} = (I \pm \vec{v} \cdot \vec{\sigma})/2$.

Exercise 2.61: Calculate the probability of obtaining the result $+1$ for a measurement of $\vec{v} \cdot \vec{\sigma}$, given that the state prior to measurement is $|0\rangle$. What is the state of the system after the measurement if $+1$ is obtained?

2.2.6 POVM measurements

The quantum measurement postulate, Postulate 3, involves two elements. First, it gives a rule describing the measurement statistics, that is, the respective probabilities of the different possible measurement outcomes. Second, it gives a rule describing the post-measurement state of the system. However, for some applications the post-measurement state of the system is of little interest, with the main item of interest being the probabilities of the respective measurement outcomes. This is the case, for example, in an experiment where the system is measured only once, upon conclusion of the experiment. In such instances there is a mathematical tool known as the *POVM formalism* which is especially well adapted to the analysis of the measurements. (The acronym POVM stands for 'Positive Operator-Valued Measure', a technical term whose historical origins we won't worry about.) This formalism is a simple consequence of the general description of measurements introduced in Postulate 3, but the theory of POVMs is so elegant and widely used that it merits a separate discussion here.

Suppose a measurement described by measurement operators M_m is performed upon a quantum system in the state $|\psi\rangle$. Then the probability of outcome m is given by $p(m) = \langle \psi | M_m^\dagger M_m | \psi \rangle$. Suppose we define

$$E_m \equiv M_m^\dagger M_m. \tag{2.117}$$

Then from Postulate 3 and elementary linear algebra, E_m is a positive operator such that $\sum_m E_m = I$ and $p(m) = \langle \psi | E_m | \psi \rangle$. Thus the set of operators E_m are sufficient to determine the probabilities of the different measurement outcomes. The operators E_m are known as the *POVM elements* associated with the measurement. The complete set $\{E_m\}$ is known as a *POVM*.

As an example of a POVM, consider a projective measurement described by measurement operators P_m, where the P_m are projectors such that $P_m P_{m'} = \delta_{mm'} P_m$ and $\sum_m P_m = I$. In this instance (and only this instance) all the POVM elements are the same as the measurement operators themselves, since $E_m \equiv P_m^\dagger P_m = P_m$.

Box 2.5: General measurements, projective measurements, and POVMs

Most introductions to quantum mechanics describe only projective measurements, and consequently the general description of measurements given in Postulate 3 may be unfamiliar to many physicists, as may the POVM formalism described in Section 2.2.6. The reason most physicists don't learn the general measurement formalism is because most physical systems can only be measured in a very coarse manner. In quantum computation and quantum information we aim for an exquisite level of control over the measurements that may be done, and consequently it helps to use a more comprehensive formalism for the description of measurements.

Of course, when the other axioms of quantum mechanics are taken into account, projective measurements augmented by unitary operations turn out to be completely *equivalent* to general measurements, as shown in Section 2.2.8. So a physicist trained in the use of projective measurements might ask to what end we start with the general formalism, Postulate 3? There are several reasons for doing so. First, mathematically general measurements are in some sense simpler than projective measurements, since they involve fewer restrictions on the measurement operators; there is, for example, no requirement for general measurements analogous to the condition $P_i P_j = \delta_{ij} P_i$ for projective measurements. This simpler structure also gives rise to many useful properties for general measurements that are not possessed by projective measurements. Second, it turns out that there are important problems in quantum computation and quantum information – such as the optimal way to distinguish a set of quantum states – the answer to which involves a general measurement, rather than a projective measurement.

A third reason for preferring Postulate 3 as a starting point is related to a property of projective measurements known as *repeatability*. Projective measurements are repeatable in the sense that if we perform a projective measurement once, and obtain the outcome m, repeating the measurement gives the outcome m again and does not change the state. To see this, suppose $|\psi\rangle$ was the initial state. After the first measurement the state is $|\psi_m\rangle = \left(P_m|\psi\rangle\right) / \sqrt{\langle\psi|P_m|\psi\rangle}$. Applying P_m to $|\psi_m\rangle$ does not change it, so we have $\langle\psi_m|P_m|\psi_m\rangle = 1$, and therefore repeated measurement gives the result m each time, without changing the state.

This repeatability of projective measurements tips us off to the fact that many important measurements in quantum mechanics are not projective measurements. For instance, if we use a silvered screen to measure the position of a photon we destroy the photon in the process. This certainly makes it impossible to repeat the measurement of the photon's position! Many other quantum measurements are also not repeatable in the same sense as a projective measurement. For such measurements, the general measurement postulate, Postulate 3, must be employed.

Where do POVMs fit in this picture? POVMs are best viewed as a special case of the general measurement formalism, providing the simplest means by which one can study general measurement statistics, without the necessity for knowing the post-measurement state. They are a mathematical convenience that sometimes gives extra insight into quantum measurements.

Exercise 2.62: Show that any measurement where the measurement operators and the POVM elements coincide is a projective measurement.

Above we noticed that the POVM operators are positive and satisfy $\sum_m E_m = I$. Suppose now that $\{E_m\}$ is some arbitrary set of positive operators such that $\sum_m E_m = I$. We will show that there exists a set of measurement operators M_m defining a measurement described by the POVM $\{E_m\}$. Defining $M_m \equiv \sqrt{E_m}$ we see that $\sum_m M_m^\dagger M_m = \sum_m E_m = I$, and therefore the set $\{M_m\}$ describes a measurement with POVM $\{E_m\}$. For this reason it is convenient to *define* a POVM to be any set of operators $\{E_m\}$ such that: (a) each operator E_m is positive; and (b) the *completeness relation* $\sum_m E_m = I$ is obeyed, expressing the fact that probabilities sum to one. To complete the description of POVMs, we note again that given a POVM $\{E_m\}$, the probability of outcome m is given by $p(m) = \langle\psi|E_m|\psi\rangle$.

We've looked at projective measurements as an example of the use of POVMs, but it wasn't very exciting since we didn't learn much that was new. The following more sophisticated example illustrates the use of the POVM formalism as a guide for our intuition in quantum computation and quantum information. Suppose Alice gives Bob a qubit prepared in one of two states, $|\psi_1\rangle = |0\rangle$ or $|\psi_2\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. As explained in Section 2.2.4 it is impossible for Bob to determine whether he has been given $|\psi_1\rangle$ or $|\psi_2\rangle$ with perfect reliability. However, it is possible for him to perform a measurement which distinguishes the states some of the time, but *never* makes an error of mis-identification. Consider a POVM containing three elements,

$$E_1 \equiv \frac{\sqrt{2}}{1+\sqrt{2}}|1\rangle\langle 1|, \tag{2.118}$$

$$E_2 \equiv \frac{\sqrt{2}}{1+\sqrt{2}}\frac{\left(|0\rangle - |1\rangle\right)\left(\langle 0| - \langle 1|\right)}{2}, \tag{2.119}$$

$$E_3 \equiv I - E_1 - E_2. \tag{2.120}$$

It is straightforward to verify that these are positive operators which satisfy the completeness relation $\sum_m E_m = I$, and therefore form a legitimate POVM.

Suppose Bob is given the state $|\psi_1\rangle = |0\rangle$. He performs the measurement described by the POVM $\{E_1, E_2, E_3\}$. There is zero probability that he will observe the result E_1, since E_1 has been cleverly chosen to ensure that $\langle\psi_1|E_1|\psi_1\rangle = 0$. Therefore, if the result of his measurement is E_1 then Bob can safely conclude that the state he received must have been $|\psi_2\rangle$. A similar line of reasoning shows that if the measurement outcome E_2 occurs then it must have been the state $|\psi_1\rangle$ that Bob received. Some of the time, however, Bob will obtain the measurement outcome E_3, and he can infer nothing about the identity of the state he was given. The key point, however, is that Bob *never* makes a mistake identifying the state he has been given. This infallibility comes at the price that sometimes Bob obtains no information about the identity of the state.

This simple example demonstrates the utility of the POVM formalism as a simple and intuitive way of gaining insight into quantum measurements in instances where only the measurement statistics matter. In many instances later in the book we will only be concerned with measurement statistics, and will therefore use the POVM formalism rather than the more general formalism for measurements described in Postulate 3.

Exercise 2.63: Suppose a measurement is described by measurement operators M_m.

Show that there exist unitary operators U_m such that $M_m = U_m\sqrt{E_m}$, where E_m is the POVM associated to the measurement.

Exercise 2.64: Suppose Bob is given a quantum state chosen from a set $|\psi_1\rangle, \ldots, |\psi_m\rangle$ of linearly independent states. Construct a POVM $\{E_1, E_2, \ldots, E_{m+1}\}$ such that if outcome E_i occurs, $1 \leq i \leq m$, then Bob knows with certainty that he was given the state $|\psi_i\rangle$. (The POVM must be such that $\langle\psi_i|E_i|\psi_i\rangle > 0$ for each i.)

2.2.7 Phase

'Phase' is a commonly used term in quantum mechanics, with several different meanings dependent upon context. At this point it is convenient to review a couple of these meanings. Consider, for example, the state $e^{i\theta}|\psi\rangle$, where $|\psi\rangle$ is a state vector, and θ is a real number. We say that the state $e^{i\theta}|\psi\rangle$ is equal to $|\psi\rangle$, up to the *global phase factor* $e^{i\theta}$. It is interesting to note that the *statistics of measurement* predicted for these two states are the same. To see this, suppose M_m is a measurement operator associated to some quantum measurement, and note that the respective probabilities for outcome m occurring are $\langle\psi|M_m^\dagger M_m|\psi\rangle$ and $\langle\psi|e^{-i\theta}M_m^\dagger M_m e^{i\theta}|\psi\rangle = \langle\psi|M_m^\dagger M_m|\psi\rangle$. Therefore, from an observational point of view these two states are identical. For this reason we may ignore global phase factors as being irrelevant to the observed properties of the physical system.

There is another kind of phase known as the *relative phase*, which has quite a different meaning. Consider the states

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} \qquad \text{and} \qquad \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{2.121}$$

In the first state the amplitude of $|1\rangle$ is $1/\sqrt{2}$. For the second state the amplitude is $-1/\sqrt{2}$. In each case the *magnitude* of the amplitudes is the same, but they differ in sign. More generally, we say that two amplitudes, a and b, *differ by a relative phase* if there is a real θ such that $a = \exp(i\theta)b$. More generally still, two states are said to *differ by a relative phase* in some basis if each of the amplitudes in that basis is related by such a phase factor. For example, the two states displayed above are the same up to a relative phase shift because the $|0\rangle$ amplitudes are identical (a relative phase factor of 1), and the $|1\rangle$ amplitudes differ only by a relative phase factor of -1. The difference between relative phase factors and global phase factors is that for relative phase the phase factors may vary from amplitude to amplitude. This makes the relative phase a basis-dependent concept unlike global phase. As a result, states which differ only by relative phases in some basis give rise to physically observable differences in measurement statistics, and it is not possible to regard these states as physically equivalent, as we do with states differing by a global phase factor

Exercise 2.65: Express the states $(|0\rangle + |1\rangle)/\sqrt{2}$ and $(|0\rangle - |1\rangle)/\sqrt{2}$ in a basis in which they are *not* the same up to a relative phase shift.

2.2.8 Composite systems

Suppose we are interested in a composite quantum system made up of two (or more) distinct physical systems. How should we describe states of the composite system? The following postulate describes how the state space of a composite system is built up from the state spaces of the component systems.

Postulate 4: The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state $|\psi_i\rangle$, then the joint state of the total system is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$.

Why is the tensor product the mathematical structure used to describe the state space of a composite physical system? At one level, we can simply accept it as a basic postulate, not reducible to something more elementary, and move on. After all, we certainly expect that there be *some canonical way* of describing composite systems in quantum mechanics. Is there some other way we can arrive at this postulate? Here is one heuristic that is sometimes used. Physicists sometimes like to speak of the *superposition principle of quantum mechanics*, which states that if $|x\rangle$ and $|y\rangle$ are two states of a quantum system, then any superposition $\alpha|x\rangle + \beta|y\rangle$ should also be an allowed state of a quantum system, where $|\alpha|^2 + |\beta|^2 = 1$. For composite systems, it seems natural that if $|A\rangle$ is a state of system A, and $|B\rangle$ is a state of system B, then there should be some corresponding state, which we might denote $|A\rangle|B\rangle$, of the joint system AB. Applying the superposition principle to product states of this form, we arrive at the tensor product postulate given above. This is not a derivation, since we are not taking the superposition principle as a fundamental part of our description of quantum mechanics, but it gives you the flavor of the various ways in which these ideas are sometimes reformulated.

A variety of different notations for composite systems appear in the literature. Part of the reason for this proliferation is that different notations are better adapted for different applications, and we will also find it convenient to introduce some specialized notations on occasion. At this point it suffices to mention a useful subscript notation to denote states and operators on different systems, when it is not clear from context. For example, in a system containing three qubits, X_2 is the Pauli σ_x operator acting on the second qubit.

Exercise 2.66: Show that the average value of the observable $X_1 Z_2$ for a two qubit system measured in the state $(|00\rangle + |11\rangle)/\sqrt{2}$ is zero.

In Section 2.2.5 we claimed that projective measurements together with unitary dynamics are sufficient to implement a general measurement. The proof of this statement makes use of composite quantum systems, and is a nice illustration of Postulate 4 in action. Suppose we have a quantum system with state space Q, and we want to perform a measurement described by measurement operators M_m on the system Q. To do this, we introduce an *ancilla system*, with state space M, having an orthonormal basis $|m\rangle$ in one-to-one correspondence with the possible outcomes of the measurement we wish to implement. This ancilla system can be regarded as merely a mathematical device appearing in the construction, or it can be interpreted physically as an extra quantum system introduced into the problem, which we assume has a state space with the required properties.

Letting $|0\rangle$ be any fixed state of M, define an operator U on products $|\psi\rangle|0\rangle$ of states $|\psi\rangle$ from Q with the state $|0\rangle$ by

$$U|\psi\rangle|0\rangle \equiv \sum_m M_m|\psi\rangle|m\rangle. \tag{2.122}$$

Using the orthonormality of the states $|m\rangle$ and the completeness relation $\sum_m M_m^\dagger M_m =$

I, we can see that U preserves inner products between states of the form $|\psi\rangle|0\rangle$,

$$\langle\varphi|\langle 0|U^\dagger U|\psi\rangle|0\rangle = \sum_{m,m'} \langle\varphi|M_m^\dagger M_{m'}|\psi\rangle \langle m|m'\rangle \tag{2.123}$$

$$= \sum_m \langle\varphi|M_m^\dagger M_m|\psi\rangle \tag{2.124}$$

$$= \langle\varphi|\psi\rangle. \tag{2.125}$$

By the results of Exercise 2.67 it follows that U can be extended to a unitary operator on the space $Q \otimes M$, which we also denote by U.

Exercise 2.67: Suppose V is a Hilbert space with a subspace W. Suppose $U : W \rightarrow V$ is a linear operator which preserves inner products, that is, for any $|w_1\rangle$ and $|w_2\rangle$ in W,

$$\langle w_1|U^\dagger U|w_2\rangle = \langle w_1|w_2\rangle. \tag{2.126}$$

Prove that there exists a unitary operator $U' : V \rightarrow V$ which *extends* U. That is, $U'|w\rangle = U|w\rangle$ for all $|w\rangle$ in W, but U' is defined on the entire space V. Usually we omit the prime symbol $'$ and just write U to denote the extension.

Next, suppose we perform a projective measurement on the two systems described by projectors $P_m \equiv I_Q \otimes |m\rangle\langle m|$. Outcome m occurs with probability

$$p(m) = \langle\psi|\langle 0|U^\dagger P_m U|\psi\rangle|0\rangle \tag{2.127}$$

$$= \sum_{m',m''} \langle\psi|M_{m'}^\dagger\langle m'|(I_Q \otimes |m\rangle\langle m|)M_{m''}|\psi\rangle|m''\rangle \tag{2.128}$$

$$= \langle\psi|M_m^\dagger M_m|\psi\rangle, \tag{2.129}$$

just as given in Postulate 3. The joint state of the system QM after measurement, conditional on result m occurring, is given by

$$\frac{P_m U|\psi\rangle|0\rangle}{\sqrt{\langle\psi|U^\dagger P_m U|\psi\rangle}} = \frac{M_m|\psi\rangle|m\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}. \tag{2.130}$$

It follows that the state of system M after the measurement is $|m\rangle$, and the state of system Q is

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}, \tag{2.131}$$

just as prescribed by Postulate 3. Thus unitary dynamics, projective measurements, and the ability to introduce ancillary systems, together allow any measurement of the form described in Postulate 3 to be realized.

Postulate 4 also enables us to define one of the most interesting and puzzling ideas associated with composite quantum systems – *entanglement*. Consider the two qubit state

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}. \tag{2.132}$$

This state has the remarkable property that there are no single qubit states $|a\rangle$ and $|b\rangle$ such that $|\psi\rangle = |a\rangle|b\rangle$, a fact which you should now convince yourself of:

Exercise 2.68: Prove that $|\psi\rangle \neq |a\rangle|b\rangle$ for all single qubit states $|a\rangle$ and $|b\rangle$.

We say that a state of a composite system having this property (that it can't be written as a product of states of its component systems) is an *entangled* state. For reasons which nobody fully understands, entangled states play a crucial role in quantum computation and quantum information, and arise repeatedly through the remainder of this book. We have already seen entanglement play a crucial role in quantum teleportation, as described in Section 1.3.7. In this chapter we give two examples of the strange effects enabled by entangled quantum states, superdense coding (Section 2.3), and the violation of Bell's inequality (Section 2.6).

2.2.9 Quantum mechanics: a global view

We have now explained *all* the fundamental postulates of quantum mechanics. Most of the rest of the book is taken up with deriving consequences of these postulates. Let's quickly review the postulates and try to place them in some kind of global perspective.

Postulate 1 sets the arena for quantum mechanics, by specifying how the state of an isolated quantum system is to be described. Postulate 2 tells us that the dynamics of *closed* quantum systems are described by the Schrödinger equation, and thus by unitary evolution. Postulate 3 tells us how to extract information from our quantum systems by giving a prescription for the description of measurement. Postulate 4 tells us how the state spaces of different quantum systems may be combined to give a description of the composite system.

What's odd about quantum mechanics, at least by our classical lights, is that we can't directly observe the state vector. It's a little bit like a game of chess where you can never find out exactly where each piece is, but only know the rank of the board they are on. Classical physics – and our intuition – tells us that the fundamental properties of an object, like energy, position, and velocity, are directly accessible to observation. In quantum mechanics these quantities no longer appear as fundamental, being replaced by the state vector, which can't be directly observed. It is as though there is a *hidden world* in quantum mechanics, which we can only indirectly and imperfectly access. Moreover, merely observing a classical system does not necessarily change the state of the system. Imagine how difficult it would be to play tennis if each time you looked at the ball its position changed! But according to Postulate 3, observation in quantum mechanics is an invasive procedure that typically changes the state of the system.

What conclusions should we draw from these strange features of quantum mechanics? Might it be possible to reformulate quantum mechanics in a mathematically equivalent way so that it had a structure more like classical physics? In Section 2.6 we'll prove *Bell's inequality*, a surprising result that shows any attempt at such a reformulation is doomed to failure. We're stuck with the counter-intuitive nature of quantum mechanics. Of course, the proper reaction to this is glee, not sorrow! It gives us an opportunity to develop tools of thought that make quantum mechanics intuitive. Moreover, we can exploit the hidden nature of the state vector to do information processing tasks beyond what is possible in the classical world. Without this counter-intuitive behavior, quantum computation and quantum information would be a lot less interesting.

We can also turn this discussion about, and ask ourselves: 'If quantum mechanics is so different from classical physics, then how come the everyday world looks so classical?' Why do we see no evidence of a hidden state vector in our everyday lives? It turns out

that the classical world we see can be *derived* from quantum mechanics as an approximate description of the world that will be valid on the sort of time, length and mass scales we commonly encounter in our everyday lives. Explaining the details of how quantum mechanics gives rise to classical physics is beyond the scope of this book, but the interested reader should check out the discussion of this topic in 'History and further reading'at the end of Chapter 8.

2.3 Application: superdense coding

Superdense coding is a simple yet surprising application of elementary quantum mechanics. It combines in a concrete, non-trivial way all the basic ideas of elementary quantum mechanics, as covered in the previous sections, and is therefore an ideal example of the information processing tasks that can be accomplished using quantum mechanics.

Superdense coding involves two parties, conventionally known as 'Alice' and 'Bob', who are a long way away from one another. Their goal is to transmit some classical information from Alice to Bob. Suppose Alice is in possession of two classical bits of information which she wishes to send Bob, but is only allowed to send a single qubit to Bob. Can she achieve her goal?

Superdense coding tells us that the answer to this question is yes. Suppose Alice and Bob initially share a pair of qubits in the entangled state

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}. \tag{2.133}$$

Alice is initially in possession of the first qubit, while Bob has possession of the second qubit, as illustrated in Figure 2.3. Note that $|\psi\rangle$ is a fixed state; there is no need for Alice to have sent Bob any qubits in order to prepare this state. Instead, some third party may prepare the entangled state ahead of time, sending one of the qubits to Alice, and the other to Bob.

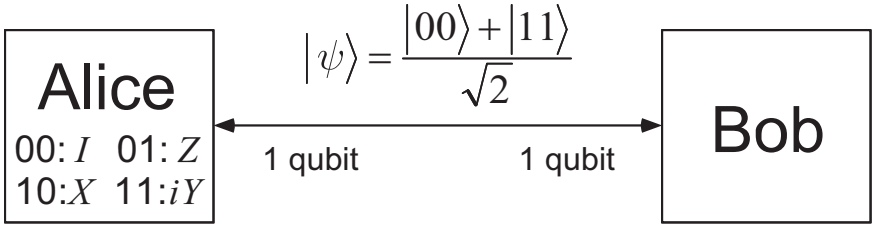


Figure 2.3. The initial setup for superdense coding, with Alice and Bob each in possession of one half of an entangled pair of qubits. Alice can use superdense coding to transmit two classical bits of information to Bob, using only a single qubit of communication and this preshared entanglement.

By sending the single qubit in her possession to Bob, it turns out that Alice can communicate two bits of classical information to Bob. Here is the procedure she uses. If she wishes to send the bit string '00' to Bob then she does nothing at all to her qubit. If she wishes to send '01' then she applies the phase flip Z to her qubit. If she wishes to send '10' then she applies the quantum NOT gate, X, to her qubit. If she wishes to send '11' then she applies the iY gate to her qubit. The four resulting states are easily seen

to be:

$$00 : |\psi\rangle \rightarrow \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2.134}$$

$$01 : |\psi\rangle \rightarrow \frac{|00\rangle - |11\rangle}{\sqrt{2}} \tag{2.135}$$

$$10 : |\psi\rangle \rightarrow \frac{|10\rangle + |01\rangle}{\sqrt{2}} \tag{2.136}$$

$$11 : |\psi\rangle \rightarrow \frac{|01\rangle - |10\rangle}{\sqrt{2}}. \tag{2.137}$$

As we noted in Section 1.3.6, these four states are known as the *Bell basis*, *Bell states*, or *EPR pairs*, in honor of several of the pioneers who first appreciated the novelty of entanglement. Notice that the Bell states form an orthonormal basis, and can therefore be distinguished by an appropriate quantum measurement. If Alice sends her qubit to Bob, giving Bob possession of both qubits, then by doing a measurement in the Bell basis Bob can determine which of the four possible bit strings Alice sent.

Summarizing, Alice, interacting with only a single qubit, is able to transmit two bits of information to Bob. Of course, two qubits are involved in the protocol, but Alice never need interact with the second qubit. Classically, the task Alice accomplishes would have been impossible had she only transmitted a single classical bit, as we will show in Chapter 12. Furthermore, this remarkable superdense coding protocol has received partial verification in the laboratory. (See 'History and further reading' for references to the experimental verification.) In later chapters we will see many other examples, some of them much more spectacular than superdense coding, of quantum mechanics being harnessed to perform information processing tasks. However, a key point can already be seen in this beautiful example: information is physical, and surprising physical theories such as quantum mechanics may predict surprising information processing abilities.

Exercise 2.69: Verify that the Bell basis forms an orthonormal basis for the two qubit state space.

Exercise 2.70: Suppose E is any positive operator acting on Alice's qubit. Show that $\langle\psi|E \otimes I|\psi\rangle$ *takes the same value* when $|\psi\rangle$ is any of the four Bell states. Suppose some malevolent third party ('Eve') intercepts Alice's qubit on the way to Bob in the superdense coding protocol. Can Eve infer anything about which of the four possible bit strings $00, 01, 10, 11$ Alice is trying to send? If so, how, or if not, why not?

2.4 The density operator

We have formulated quantum mechanics using the language of state vectors. An alternate formulation is possible using a tool known as the *density operator* or *density matrix*. This alternate formulation is mathematically equivalent to the state vector approach, but it provides a much more convenient language for thinking about some commonly encountered scenarios in quantum mechanics. The next three sections describe the density operator formulation of quantum mechanics. Section 2.4.1 introduces the density operator using the concept of an ensemble of quantum states. Section 2.4.2 develops some general

properties of the density operator. Finally, Section 2.4.3 describes an application where the density operator really shines – as a tool for the description of *individual subsystems* of a composite quantum system.

2.4.1 Ensembles of quantum states

The density operator language provides a convenient means for describing quantum systems whose state is not completely known. More precisely, suppose a quantum system is in one of a number of states $|\psi_i\rangle$, where i is an index, with respective probabilities p_i. We shall call $\{p_i, |\psi_i\rangle\}$ an *ensemble of pure states*. The density operator for the system is defined by the equation

$$\rho \equiv \sum_i p_i |\psi_i\rangle\langle\psi_i|. \tag{2.138}$$

The density operator is often known as the *density matrix*; we will use the two terms interchangeably. It turns out that all the postulates of quantum mechanics can be reformulated in terms of the density operator language. The purpose of this section and the next is to explain how to perform this reformulation, and explain when it is useful. Whether one uses the density operator language or the state vector language is a matter of taste, since both give the same results; however it is sometimes much easier to approach problems from one point of view rather than the other.

Suppose, for example, that the evolution of a closed quantum system is described by the unitary operator U. If the system was initially in the state $|\psi_i\rangle$ with probability p_i then after the evolution has occurred the system will be in the state $U|\psi_i\rangle$ with probability p_i. Thus, the evolution of the density operator is described by the equation

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \stackrel{U}{\longrightarrow} \sum_i p_i U|\psi_i\rangle\langle\psi_i|U^\dagger = U\rho U^\dagger. \tag{2.139}$$

Measurements are also easily described in the density operator language. Suppose we perform a measurement described by measurement operators M_m. If the initial state was $|\psi_i\rangle$, then the probability of getting result m is

$$p(m|i) = \langle\psi_i|M_m^\dagger M_m|\psi_i\rangle = \mathrm{tr}(M_m^\dagger M_m|\psi_i\rangle\langle\psi_i|), \tag{2.140}$$

where we have used Equation (2.61) to obtain the last equality. By the law of total probability (see Appendix 1 for an explanation of this and other elementary notions of probability theory) the probability of obtaining result m is

$$p(m) = \sum_i p(m|i)p_i \tag{2.141}$$

$$= \sum_i p_i \mathrm{tr}(M_m^\dagger M_m|\psi_i\rangle\langle\psi_i|) \tag{2.142}$$

$$= \mathrm{tr}(M_m^\dagger M_m \rho). \tag{2.143}$$

What is the density operator of the system after obtaining the measurement result m? If the initial state was $|\psi_i\rangle$ then the state after obtaining the result m is

$$|\psi_i^m\rangle = \frac{M_m|\psi_i\rangle}{\sqrt{\langle\psi_i|M_m^\dagger M_m|\psi_i\rangle}}. \tag{2.144}$$

Thus, after a measurement which yields the result m we have an ensemble of states $|\psi_i^m\rangle$ with respective probabilities $p(i|m)$. The corresponding density operator ρ_m is therefore

$$\rho_m = \sum_i p(i|m)|\psi_i^m\rangle\langle\psi_i^m| = \sum_i p(i|m)\frac{M_m|\psi_i\rangle\langle\psi_i|M_m^\dagger}{\langle\psi_i|M_m^\dagger M_m|\psi_i\rangle}. \tag{2.145}$$

But by elementary probability theory, $p(i|m) = p(m,i)/p(m) = p(m|i)p_i/p(m)$. Substituting from (2.143) and (2.140) we obtain

$$\rho_m = \sum_i p_i \frac{M_m|\psi_i\rangle\langle\psi_i|M_m^\dagger}{\mathrm{tr}(M_m^\dagger M_m \rho)} \tag{2.146}$$

$$= \frac{M_m \rho M_m^\dagger}{\mathrm{tr}(M_m^\dagger M_m \rho)}. \tag{2.147}$$

What we have shown is that the basic postulates of quantum mechanics related to unitary evolution and measurement can be rephrased in the language of density operators. In the next section we complete this rephrasing by giving an intrinsic characterization of the density operator that does not rely on the idea of a state vector.

Before doing so, however, it is useful to introduce some more language, and one more fact about the density operator. First, the language. A quantum system whose state $|\psi\rangle$ is known exactly is said to be in a *pure state*. In this case the density operator is simply $\rho = |\psi\rangle\langle\psi|$. Otherwise, ρ is in a *mixed state*; it is said to be a *mixture* of the different pure states in the ensemble for ρ. In the exercises you will be asked to demonstrate a simple criterion for determining whether a state is pure or mixed: a pure state satisfies $\mathrm{tr}(\rho^2) = 1$, while a mixed state satisfies $\mathrm{tr}(\rho^2) < 1$. A few words of warning about the nomenclature: sometimes people use the term 'mixed state' as a catch-all to include both pure and mixed quantum states. The origin for this usage seems to be that it implies that the writer is not necessarily *assuming* that a state is pure. Second, the term 'pure state' is often used in reference to a state vector $|\psi\rangle$, to distinguish it from a density operator ρ.

Finally, imagine a quantum system is prepared in the state ρ_i with probability p_i. It is not difficult to convince yourself that the system may be described by the density matrix $\sum_i p_i\rho_i$. A proof of this is to suppose that ρ_i arises from some ensemble $\{p_{ij}, |\psi_{ij}\rangle\}$ (note that i is fixed) of pure states, so the probability for being in the state $|\psi_{ij}\rangle$ is $p_i p_{ij}$. The density matrix for the system is thus

$$\rho = \sum_{ij} p_i p_{ij} |\psi_{ij}\rangle\langle\psi_{ij}| \tag{2.148}$$

$$= \sum_i p_i \rho_i, \tag{2.149}$$

where we have used the definition $\rho_i = \sum_j p_{ij}|\psi_{ij}\rangle\langle\psi_{ij}|$. We say that ρ is a *mixture* of the states ρ_i with probabilities p_i. This concept of a mixture comes up repeatedly in the analysis of problems like quantum noise, where the effect of the noise is to introduce ignorance into our knowledge of the quantum state. A simple example is provided by the measurement scenario described above. Imagine that, for some reason, our record of the result m of the measurement was lost. We would have a quantum system in the state ρ_m with probability $p(m)$, but would no longer know the actual value of m. The state of

such a quantum system would therefore be described by the density operator

$$\rho = \sum_m p(m)\rho_m \tag{2.150}$$

$$= \sum_m \mathrm{tr}(M_m^\dagger M_m \rho)\frac{M_m \rho M_m^\dagger}{\mathrm{tr}(M_m^\dagger M_m \rho)} \tag{2.151}$$

$$= \sum_m M_m \rho M_m^\dagger, \tag{2.152}$$

a nice compact formula which may be used as the starting point for analysis of further operations on the system.

2.4.2 General properties of the density operator

The density operator was introduced as a means of describing ensembles of quantum states. In this section we move away from this description to develop an intrinsic characterization of density operators that does not rely on an ensemble interpretation. This allows us to complete the program of giving a description of quantum mechanics that does not take as its foundation the state vector. We also take the opportunity to develop numerous other elementary properties of the density operator.

The class of operators that are density operators are characterized by the following useful theorem:

Theorem 2.5: (**Characterization of density operators**) An operator ρ is the density operator associated to some ensemble $\{p_i, |\psi_i\rangle\}$ if and only if it satisfies the conditions:

(1) **(Trace condition)** ρ has trace equal to one.
(2) **(Positivity condition)** ρ is a positive operator.

Proof

Suppose $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ is a density operator. Then

$$\mathrm{tr}(\rho) = \sum_i p_i \mathrm{tr}(|\psi_i\rangle\langle\psi_i|) = \sum_i p_i = 1, \tag{2.153}$$

so the trace condition $\mathrm{tr}(\rho) = 1$ is satisfied. Suppose $|\varphi\rangle$ is an arbitrary vector in state space. Then

$$\langle\varphi|\rho|\varphi\rangle = \sum_i p_i \langle\varphi|\psi_i\rangle\langle\psi_i|\varphi\rangle \tag{2.154}$$

$$= \sum_i p_i |\langle\varphi|\psi_i\rangle|^2 \tag{2.155}$$

$$\geq 0, \tag{2.156}$$

so the positivity condition is satisfied.

Conversely, suppose ρ is any operator satisfying the trace and positivity conditions. Since ρ is positive, it must have a spectral decomposition

$$\rho = \sum_j \lambda_j |j\rangle\langle j|, \tag{2.157}$$

where the vectors $|j\rangle$ are orthogonal, and λ_j are real, non-negative eigenvalues of ρ.

From the trace condition we see that $\sum_j \lambda_j = 1$. Therefore, a system in state $|j\rangle$ with probability λ_j will have density operator ρ. That is, the ensemble $\{\lambda_j, |j\rangle\}$ is an ensemble of states giving rise to the density operator ρ. □

This theorem provides a characterization of density operators that is intrinsic to the operator itself: we can *define* a density operator to be a positive operator ρ which has trace equal to one. Making this definition allows us to reformulate the postulates of quantum mechanics in the density operator picture. For ease of reference we state all the reformulated postulates here:

Postulate 1: Associated to any isolated physical system is a complex vector space with inner product (that is, a Hilbert space) known as the *state space* of the system. The system is completely described by its *density operator*, which is a positive operator ρ with trace one, acting on the state space of the system. If a quantum system is in the state ρ_i with probability p_i, then the density operator for the system is $\sum_i p_i \rho_i$.

Postulate 2: The evolution of a *closed* quantum system is described by a *unitary transformation*. That is, the state ρ of the system at time t_1 is related to the state ρ' of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2,

$$\rho' = U \rho U^\dagger. \tag{2.158}$$

Postulate 3: Quantum measurements are described by a collection $\{M_m\}$ of *measurement operators*. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is ρ immediately before the measurement then the probability that result m occurs is given by

$$p(m) = \mathrm{tr}(M_m^\dagger M_m \rho), \tag{2.159}$$

and the state of the system after the measurement is

$$\frac{M_m \rho M_m^\dagger}{\mathrm{tr}(M_m^\dagger M_m \rho)}. \tag{2.160}$$

The measurement operators satisfy the *completeness equation*,

$$\sum_m M_m^\dagger M_m = I. \tag{2.161}$$

Postulate 4: The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state ρ_i, then the joint state of the total system is $\rho_1 \otimes \rho_2 \otimes \ldots \rho_n$.

These reformulations of the fundamental postulates of quantum mechanics in terms of the density operator are, of course, mathematically equivalent to the description in terms of the state vector. Nevertheless, as a way of thinking about quantum mechanics, the density operator approach really shines for two applications: the description of quantum systems whose state is not known, and the description of subsystems of a composite

quantum system, as will be described in the next section. For the remainder of this section we flesh out the properties of the density matrix in more detail.

Exercise 2.71: (Criterion to decide if a state is mixed or pure) Let ρ be a density operator. Show that $\mathrm{tr}(\rho^2) \leq 1$, with equality if and only if ρ is a pure state.

It is a tempting (and surprisingly common) fallacy to suppose that the eigenvalues and eigenvectors of a density matrix have some special significance with regard to the ensemble of quantum states represented by that density matrix. For example, one might suppose that a quantum system with density matrix

$$\rho = \frac{3}{4}|0\rangle\langle 0| + \frac{1}{4}|1\rangle\langle 1|\,. \tag{2.162}$$

must be in the state $|0\rangle$ with probability $3/4$ and in the state $|1\rangle$ with probability $1/4$. However, this is not necessarily the case. Suppose we define

$$|a\rangle \equiv \sqrt{\frac{3}{4}}|0\rangle + \sqrt{\frac{1}{4}}|1\rangle \tag{2.163}$$

$$|b\rangle \equiv \sqrt{\frac{3}{4}}|0\rangle - \sqrt{\frac{1}{4}}|1\rangle, \tag{2.164}$$

and the quantum system is prepared in the state $|a\rangle$ with probability $1/2$ and in the state $|b\rangle$ with probability $1/2$. Then it is easily checked that the corresponding density matrix is

$$\rho = \frac{1}{2}|a\rangle\langle a| + \frac{1}{2}|b\rangle\langle b| = \frac{3}{4}|0\rangle\langle 0| + \frac{1}{4}|1\rangle\langle 1|. \tag{2.165}$$

That is, these two *different* ensembles of quantum states give rise to the *same* density matrix. In general, the eigenvectors and eigenvalues of a density matrix just indicate *one* of many possible ensembles that may give rise to a specific density matrix, and there is no reason to suppose it is an especially privileged ensemble.

A natural question to ask in the light of this discussion is what class of ensembles does give rise to a particular density matrix? The solution to this problem, which we now give, has surprisingly many applications in quantum computation and quantum information, notably in the understanding of quantum noise and quantum error-correction (Chapters 8 and 10). For the solution it is convenient to make use of vectors $|\tilde{\psi}_i\rangle$ which may not be normalized to unit length. We say the set $|\tilde{\psi}_i\rangle$ *generates* the operator $\rho \equiv \sum_i |\tilde{\psi}_i\rangle\langle\tilde{\psi}_i|$, and thus the connection to the usual ensemble picture of density operators is expressed by the equation $|\tilde{\psi}_i\rangle = \sqrt{p_i}|\psi_i\rangle$. When do two sets of vectors, $|\tilde{\psi}_i\rangle$ and $|\tilde{\varphi}_j\rangle$ generate the same operator ρ? The solution to this problem will enable us to answer the question of what ensembles give rise to a given density matrix.

Theorem 2.6: **(Unitary freedom in the ensemble for density matrices)** The sets $|\tilde{\psi}_i\rangle$ and $|\tilde{\varphi}_j\rangle$ generate the same density matrix if and only if

$$|\tilde{\psi}_i\rangle = \sum_j u_{ij}|\tilde{\varphi}_j\rangle\,, \tag{2.166}$$

where u_{ij} is a unitary matrix of complex numbers, with indices i and j, and we

'pad' whichever set of vectors $|\tilde{\psi}_i\rangle$ or $|\tilde{\varphi}_j\rangle$ is smaller with additional vectors 0 so that the two sets have the same number of elements.

As a consequence of the theorem, note that $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| = \sum_j q_j |\varphi_j\rangle\langle\varphi_j|$ for *normalized* states $|\psi_i\rangle, |\varphi_j\rangle$ and probability distributions p_i and q_j if and only if

$$\sqrt{p_i}|\psi_i\rangle = \sum_j u_{ij}\sqrt{q_j}|\varphi_j\rangle, \tag{2.167}$$

for some unitary matrix u_{ij}, and we may pad the smaller ensemble with entries having probability zero in order to make the two ensembles the same size. Thus, Theorem 2.6 characterizes the freedom in ensembles $\{p_i, |\psi_i\rangle\}$ giving rise to a given density matrix ρ. Indeed, it is easily checked that our earlier example of a density matrix with two different decompositions, (2.162), arises as a special case of this general result. Let's turn now to the proof of the theorem.

Proof
Suppose $|\tilde{\psi}_i\rangle = \sum_j u_{ij}|\tilde{\varphi}_j\rangle$ for some unitary u_{ij}. Then

$$\sum_i |\tilde{\psi}_i\rangle\langle\tilde{\psi}_i| = \sum_{ijk} u_{ij}u_{ik}^*|\tilde{\varphi}_j\rangle\langle\tilde{\varphi}_k| \tag{2.168}$$

$$= \sum_{jk}\left(\sum_i u_{ki}^\dagger u_{ij}\right)|\tilde{\varphi}_j\rangle\langle\tilde{\varphi}_k| \tag{2.169}$$

$$= \sum_{jk}\delta_{kj}|\tilde{\varphi}_j\rangle\langle\tilde{\varphi}_k| \tag{2.170}$$

$$= \sum_j |\tilde{\varphi}_j\rangle\langle\tilde{\varphi}_j|, \tag{2.171}$$

which shows that $|\tilde{\psi}_i\rangle$ and $|\tilde{\varphi}_j\rangle$ generate the same operator.

Conversely, suppose

$$A = \sum_i |\tilde{\psi}_i\rangle\langle\tilde{\psi}_i| = \sum_j |\tilde{\varphi}_j\rangle\langle\tilde{\varphi}_j|\,. \tag{2.172}$$

Let $A = \sum_k \lambda_k |k\rangle\langle k|$ be a decomposition for A such that the states $|k\rangle$ are orthonormal, and the λ_k are strictly positive. Our strategy is to relate the states $|\tilde{\psi}_i\rangle$ to the states $|\tilde{k}\rangle \equiv \sqrt{\lambda_k}|k\rangle$, and similarly relate the states $|\tilde{\varphi}_j\rangle$ to the states $|\tilde{k}\rangle$. Combining the two relations will give the result. Let $|\psi\rangle$ be any vector orthonormal to the space spanned by the $|\tilde{k}\rangle$, so $\langle\psi|\tilde{k}\rangle\langle\tilde{k}|\psi\rangle = 0$ for all k, and thus we see that

$$0 = \langle\psi|A|\psi\rangle = \sum_i \langle\psi|\tilde{\psi}_i\rangle\langle\tilde{\psi}_i|\psi\rangle = \sum_i |\langle\psi|\tilde{\psi}_i\rangle|^2. \tag{2.173}$$

Thus $\langle\psi|\tilde{\psi}_i\rangle = 0$ for all i and all $|\psi\rangle$ orthonormal to the space spanned by the $|\tilde{k}\rangle$. It follows that each $|\tilde{\psi}_i\rangle$ can be expressed as a linear combination of the $|\tilde{k}\rangle$, $|\tilde{\psi}_i\rangle = \sum_k c_{ik}|\tilde{k}\rangle$. Since $A = \sum_k |\tilde{k}\rangle\langle\tilde{k}| = \sum_i |\tilde{\psi}_i\rangle\langle\tilde{\psi}_i|$ we see that

$$\sum_k |\tilde{k}\rangle\langle\tilde{k}| = \sum_{kl}\left(\sum_i c_{ik}c_{il}^*\right)|\tilde{k}\rangle\langle\tilde{l}|. \tag{2.174}$$

The operators $|\tilde{k}\rangle\langle\tilde{l}|$ are easily seen to be linearly independent, and thus it must be that

$\sum_i c_{ik} c_{il}^* = \delta_{kl}$. This ensures that we may append extra columns to c to obtain a unitary matrix v such that $|\tilde{\psi}_i\rangle = \sum_k v_{ik} |\tilde{k}\rangle$, where we have appended zero vectors to the list of $|\tilde{k}\rangle$. Similarly, we can find a unitary matrix w such that $|\tilde{\varphi}_j\rangle = \sum_k w_{jk} |\tilde{k}\rangle$. Thus $|\tilde{\psi}_i\rangle = \sum_j u_{ij} |\tilde{\varphi}_j\rangle$, where $u = vw^\dagger$ is unitary. □

Exercise 2.72: (Bloch sphere for mixed states) The Bloch sphere picture for pure states of a single qubit was introduced in Section 1.2. This description has an important generalization to mixed states as follows.

(1) Show that an arbitrary density matrix for a mixed state qubit may be written as

$$\rho = \frac{I + \vec{r} \cdot \vec{\sigma}}{2}, \tag{2.175}$$

where $\vec{r}$ is a real three-dimensional vector such that $\|\vec{r}\| \leq 1$. This vector is known as the *Bloch vector* for the state ρ.

(2) What is the Bloch vector representation for the state $\rho = I/2$?

(3) Show that a state ρ is pure if and only if $\|\vec{r}\| = 1$.

(4) Show that for pure states the description of the Bloch vector we have given coincides with that in Section 1.2.

Exercise 2.73: Let ρ be a density operator. A *minimal ensemble* for ρ is an ensemble $\{p_i, |\psi_i\rangle\}$ containing a number of elements equal to the rank of ρ. Let $|\psi\rangle$ be any state in the support of ρ. (The *support* of a Hermitian operator A is the vector space spanned by the eigenvectors of A with non-zero eigenvalues.) Show that there is a minimal ensemble for ρ that contains $|\psi\rangle$, and moreover that in any such ensemble $|\psi\rangle$ must appear with probability

$$p_i = \frac{1}{\langle \psi_i | \rho^{-1} | \psi_i \rangle}, \tag{2.176}$$

where ρ^{-1} is defined to be the inverse of ρ, when ρ is considered as an operator acting only on the support of ρ. (This definition removes the problem that ρ may not have an inverse.)

2.4.3 The reduced density operator

Perhaps the deepest application of the density operator is as a descriptive tool for *subsystems* of a composite quantum system. Such a description is provided by the *reduced density operator*, which is the subject of this section. The reduced density operator is so useful as to be virtually indispensable in the analysis of composite quantum systems.

Suppose we have physical systems A and B, whose state is described by a density operator ρ^{AB}. The reduced density operator for system A is defined by

$$\rho^A \equiv \mathrm{tr}_B(\rho^{AB}), \tag{2.177}$$

where tr_B is a map of operators known as the *partial trace* over system B. The partial trace is defined by

$$\mathrm{tr}_B \left(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2| \right) \equiv |a_1\rangle\langle a_2| \, \mathrm{tr}(|b_1\rangle\langle b_2|), \tag{2.178}$$

where $|a_1\rangle$ and $|a_2\rangle$ are any two vectors in the state space of A, and $|b_1\rangle$ and $|b_2\rangle$ are any two vectors in the state space of B. The trace operation appearing on the right hand side

is the usual trace operation for system B, so $\mathrm{tr}(|b_1\rangle\langle b_2|) = \langle b_2|b_1\rangle$. We have defined the partial trace operation only on a special subclass of operators on AB; the specification is completed by requiring in addition to Equation (2.178) that the partial trace be linear in its input.

It is not obvious that the reduced density operator for system A is in any sense a description for the state of system A. The physical justification for making this identification is that the reduced density operator provides the correct measurement statistics for measurements made on system A. This is explained in more detail in Box 2.6 on page 107. The following simple example calculations may also help understand the reduced density operator. First, suppose a quantum system is in the product state $\rho^{AB} = \rho \otimes \sigma$, where ρ is a density operator for system A, and σ is a density operator for system B. Then

$$\rho^A = \mathrm{tr}_B(\rho \otimes \sigma) = \rho\,\mathrm{tr}(\sigma) = \rho, \tag{2.184}$$

which is the result we intuitively expect. Similarly, $\rho^B = \sigma$ for this state. A less trivial example is the Bell state $(|00\rangle + |11\rangle)/\sqrt{2}$. This has density operator

$$\rho = \left(\frac{|00\rangle + |11\rangle}{\sqrt{2}}\right)\left(\frac{\langle 00| + \langle 11|}{\sqrt{2}}\right) \tag{2.185}$$

$$= \frac{|00\rangle\langle 00| + |11\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 11|}{2}. \tag{2.186}$$

Tracing out the second qubit, we find the reduced density operator of the first qubit,

$$\rho^1 = \mathrm{tr}_2(\rho) \tag{2.187}$$

$$= \frac{\mathrm{tr}_2(|00\rangle\langle 00|) + \mathrm{tr}_2(|11\rangle\langle 00|) + \mathrm{tr}_2(|00\rangle\langle 11|) + \mathrm{tr}_2(|11\rangle\langle 11|)}{2} \tag{2.188}$$

$$= \frac{|0\rangle\langle 0|\langle 0|0\rangle + |1\rangle\langle 0|\langle 0|1\rangle + |0\rangle\langle 1|\langle 1|0\rangle + |1\rangle\langle 1|\langle 1|1\rangle}{2} \tag{2.189}$$

$$= \frac{|0\rangle\langle 0| + |1\rangle\langle 1|}{2} \tag{2.190}$$

$$= \frac{I}{2}. \tag{2.191}$$

Notice that this state is a *mixed state*, since $\mathrm{tr}((I/2)^2) = 1/2 < 1$. This is quite a remarkable result. The state of the joint system of two qubits is a pure state, that is, it is known *exactly*; however, the first qubit is in a mixed state, that is, a state about which we apparently do not have maximal knowledge. This strange property, that the joint state of a system can be completely known, yet a subsystem be in mixed states, is another hallmark of quantum entanglement.

Exercise 2.74: Suppose a composite of systems A and B is in the state $|a\rangle|b\rangle$, where $|a\rangle$ is a pure state of system A, and $|b\rangle$ is a pure state of system B. Show that the reduced density operator of system A alone is a pure state.

Exercise 2.75: For each of the four Bell states, find the reduced density operator for each qubit.

Quantum teleportation and the reduced density operator

A useful application of the reduced density operator is to the analysis of quantum teleportation. Recall from Section 1.3.7 that quantum teleportation is a procedure for sending

Box 2.6: Why the partial trace?

Why is the partial trace used to describe part of a larger quantum system? The reason for doing this is because the partial trace operation is the *unique* operation which gives rise to the correct description of *observable* quantities for subsystems of a composite system, in the following sense.

Suppose M is any observable on system A, and we have some measuring device which is capable of realizing measurements of M. Let $\tilde{M}$ denote the corresponding observable for the same measurement, performed on the composite system AB. Our immediate goal is to argue that $\tilde{M}$ is necessarily equal to $M \otimes I_B$. Note that if the system AB is prepared in the state $|m\rangle|\psi\rangle$, where $|m\rangle$ is an eigenstate of M with eigenvalue m, and $|\psi\rangle$ is any state of B, then the measuring device must yield the result m for the measurement, with probability one. Thus, if P_m is the projector onto the m eigenspace of the observable M, then the corresponding projector for $\tilde{M}$ is $P_m \otimes I_B$. We therefore have

$$\tilde{M} = \sum_m m P_m \otimes I_B = M \otimes I_B. \tag{2.179}$$

The next step is to show that the partial trace procedure gives the correct measurement statistics for observations on part of a system. Suppose we perform a measurement on system A described by the observable M. Physical consistency requires that any prescription for associating a 'state', ρ^A, to system A, must have the property that measurement averages be the same whether computed via ρ^A or ρ^{AB},

$$\mathrm{tr}(M\rho^A) = \mathrm{tr}(\tilde{M}\rho^{AB}) = \mathrm{tr}((M \otimes I_B)\rho^{AB}). \tag{2.180}$$

This equation is certainly satisfied if we choose $\rho^A \equiv \mathrm{tr}_B(\rho^{AB})$. In fact, the partial trace turns out to be the *unique* function having this property. To see this uniqueness property, let $f(\cdot)$ be any map of density operators on AB to density operators on A such that

$$\mathrm{tr}(M f(\rho^{AB})) = \mathrm{tr}((M \otimes I_B)\rho^{AB}), \tag{2.181}$$

for all observables M. Let M_i be an orthonormal basis of operators for the space of Hermitian operators with respect to the Hilbert–Schmidt inner product $(X, Y) \equiv \mathrm{tr}(XY)$ (compare Exercise 2.39 on page 76). Then expanding $f(\rho^{AB})$ in this basis gives

$$f(\rho^{AB}) = \sum_i M_i \mathrm{tr}(M_i f(\rho^{AB})) \tag{2.182}$$

$$= \sum_i M_i \mathrm{tr}((M_i \otimes I_B)\rho^{AB}). \tag{2.183}$$

It follows that f is uniquely determined by Equation (2.180). Moreover, the partial trace satisfies (2.180), so it is the unique function having this property.

quantum information from Alice to Bob, given that Alice and Bob share an EPR pair, and have a classical communications channel.

At first sight it appears as though teleportation can be used to do faster than light communication, a big no-no according to the theory of relativity. We surmised in Section 1.3.7 that what prevents faster than light communication is the need for Alice to communicate her measurement result to Bob. The reduced density operator allows us to make this rigorous.

Recall that immediately before Alice makes her measurement the quantum state of the three qubits is (Equation (1.32)):

$$|\psi_2\rangle = \frac{1}{2}\Big[|00\rangle\left(\alpha|0\rangle + \beta|1\rangle\right) + |01\rangle\left(\alpha|1\rangle + \beta|0\rangle\right) + |10\rangle\left(\alpha|0\rangle - \beta|1\rangle\right) + |11\rangle\left(\alpha|1\rangle - \beta|0\rangle\right)\Big]. \tag{2.192}$$

Measuring in Alice's computational basis, the state of the system after the measurement is:

$$|00\rangle\left[\alpha|0\rangle + \beta|1\rangle\right] \text{ with probability } \frac{1}{4} \tag{2.193}$$

$$|01\rangle\left[\alpha|1\rangle + \beta|0\rangle\right] \text{ with probability } \frac{1}{4} \tag{2.194}$$

$$|10\rangle\left[\alpha|0\rangle - \beta|1\rangle\right] \text{ with probability } \frac{1}{4} \tag{2.195}$$

$$|11\rangle\left[\alpha|1\rangle - \beta|0\rangle\right] \text{ with probability } \frac{1}{4}. \tag{2.196}$$

The density operator of the system is thus

$$\rho = \frac{1}{4}\Big[|00\rangle\langle 00|(\alpha|0\rangle + \beta|1\rangle)(\alpha^*\langle 0| + \beta^*\langle 1|) + |01\rangle\langle 01|(\alpha|1\rangle + \beta|0\rangle)(\alpha^*\langle 1| + \beta^*\langle 0|) + |10\rangle\langle 10|(\alpha|0\rangle - \beta|1\rangle)(\alpha^*\langle 0| - \beta^*\langle 1|) + |11\rangle\langle 11|(\alpha|1\rangle - \beta|0\rangle)(\alpha^*\langle 1| - \beta^*\langle 0|)\Big]. \tag{2.197}$$

Tracing out Alice's system, we see that the reduced density operator of Bob's system is

$$\rho^B = \frac{1}{4}\Big[(\alpha|0\rangle + \beta|1\rangle)(\alpha^*\langle 0| + \beta^*\langle 1|) + (\alpha|1\rangle + \beta|0\rangle)(\alpha^*\langle 1| + \beta^*\langle 0|) + (\alpha|0\rangle - \beta|1\rangle)(\alpha^*\langle 0| - \beta^*\langle 1|) + (\alpha|1\rangle - \beta|0\rangle)(\alpha^*\langle 1| - \beta^*\langle 0|)\Big] \tag{2.198}$$

$$= \frac{2(|\alpha|^2 + |\beta|^2)|0\rangle\langle 0| + 2(|\alpha|^2 + |\beta|^2)|1\rangle\langle 1|}{4} \tag{2.199}$$

$$= \frac{|0\rangle\langle 0| + |1\rangle\langle 1|}{2} \tag{2.200}$$

$$= \frac{I}{2}, \tag{2.201}$$

where we have used the completeness relation in the last line. Thus, the state of Bob's system *after* Alice has performed the measurement but *before* Bob has learned the measurement result is $I/2$. This state has no dependence upon the state $|\psi\rangle$ being teleported, and thus any measurements performed by Bob will contain no information about $|\psi\rangle$, thus preventing Alice from using teleportation to transmit information to Bob faster than light.

2.5 The Schmidt decomposition and purifications

Density operators and the partial trace are just the beginning of a wide array of tools useful for the study of composite quantum systems, which are at the heart of quantum computation and quantum information. Two additional tools of great value are the *Schmidt decomposition* and *purifications*. In this section we present both these tools, and try to give the flavor of their power.

Theorem 2.7: **(Schmidt decomposition)** Suppose $|\psi\rangle$ is a pure state of a composite system, AB. Then there exist orthonormal states $|i_A\rangle$ for system A, and orthonormal states $|i_B\rangle$ of system B such that

$$|\psi\rangle = \sum_i \lambda_i |i_A\rangle |i_B\rangle, \tag{2.202}$$

where λ_i are non-negative real numbers satisfying $\sum_i \lambda_i^2 = 1$ known as *Schmidt co-efficients*.

This result is very useful. As a taste of its power, consider the following consequence: let $|\psi\rangle$ be a pure state of a composite system, AB. Then by the Schmidt decomposition $\rho^A = \sum_i \lambda_i^2 |i_A\rangle\langle i_A|$ and $\rho^B = \sum_i \lambda_i^2 |i_B\rangle\langle i_B|$, so the eigenvalues of ρ^A and ρ^B are identical, namely λ_i^2 for both density operators. Many important properties of quantum systems are completely determined by the eigenvalues of the reduced density operator of the system, so for a pure state of a composite system such properties will be the same for both systems. As an example, consider the state of two qubits, $(|00\rangle + |01\rangle + |11\rangle)/\sqrt{3}$. This has no obvious symmetry property, yet if you calculate $\mathrm{tr}\left((\rho^A)^2\right)$ and $\mathrm{tr}\left((\rho^B)^2\right)$ you will discover that they have the same value, $7/9$ in each case. This is but one small consequence of the Schmidt decomposition.

Proof

We give the proof for the case where systems A and B have state spaces of the same dimension, and leave the general case to Exercise 2.76. Let $|j\rangle$ and $|k\rangle$ be any fixed orthonormal bases for systems A and B, respectively. Then $|\psi\rangle$ can be written

$$|\psi\rangle = \sum_{jk} a_{jk} |j\rangle |k\rangle, \tag{2.203}$$

for some matrix a of complex numbers a_{jk}. By the singular value decomposition, $a = udv$, where d is a diagonal matrix with non-negative elements, and u and v are unitary matrices. Thus

$$|\psi\rangle = \sum_{ijk} u_{ji} d_{ii} v_{ik} |j\rangle |k\rangle. \tag{2.204}$$

Defining $|i_A\rangle \equiv \sum_j u_{ji} |j\rangle$, $|i_B\rangle \equiv \sum_k v_{ik} |k\rangle$, and $\lambda_i \equiv d_{ii}$, we see that this gives

$$|\psi\rangle = \sum_i \lambda_i |i_A\rangle |i_B\rangle. \tag{2.205}$$

It is easy to check that $|i_A\rangle$ forms an orthonormal set, from the unitarity of u and the orthonormality of $|j\rangle$, and similarly that the $|i_B\rangle$ form an orthonormal set. □

Exercise 2.76: Extend the proof of the Schmidt decomposition to the case where A and B may have state spaces of different dimensionality.

Exercise 2.77: Suppose ABC is a three component quantum system. Show by example that there are quantum states $|\psi\rangle$ of such systems which can not be written in the form

$$|\psi\rangle = \sum_i \lambda_i |i_A\rangle |i_B\rangle |i_C\rangle, \tag{2.206}$$

where λ_i are real numbers, and $|i_A\rangle, |i_B\rangle, |i_C\rangle$ are orthonormal bases of the respective systems.

The bases $|i_A\rangle$ and $|i_B\rangle$ are called the *Schmidt bases* for A and B, respectively, and the number of non-zero values λ_i is called the *Schmidt number* for the state $|\psi\rangle$. The Schmidt number is an important property of a composite quantum system, which in some sense quantifies the 'amount' of entanglement between systems A and B. To get some idea of why this is the case, consider the following obvious but important property: the Schmidt number is preserved under unitary transformations on system A or system B alone. To see this, notice that if $\sum_i \lambda_i |i_A\rangle |i_B\rangle$ is the Schmidt decomposition for $|\psi\rangle$ then $\sum_i \lambda_i (U|i_A\rangle)|i_B\rangle$ is the Schmidt decomposition for $U|\psi\rangle$, where U is a unitary operator acting on system A alone. Algebraic invariance properties of this type make the Schmidt number a very useful tool.

Exercise 2.78: Prove that a state $|\psi\rangle$ of a composite system AB is a product state if and only if it has Schmidt number 1. Prove that $|\psi\rangle$ is a product state if and only if ρ^A (and thus ρ^B) are pure states.

A second, related technique for quantum computation and quantum information is *purification*. Suppose we are given a state ρ^A of a quantum system A. It is possible to introduce another system, which we denote R, and *define* a *pure state* $|AR\rangle$ for the joint system AR such that $\rho^A = \mathrm{tr}_R(|AR\rangle\langle AR|)$. That is, the pure state $|AR\rangle$ reduces to ρ^A when we look at system A alone. This is a purely mathematical procedure, known as *purification*, which allows us to associate pure states with mixed states. For this reason we call system R a *reference* system: it is a fictitious system, without a direct physical significance.

To prove that purification can be done for *any* state, we explain how to construct a system R and purification $|AR\rangle$ for ρ^A. Suppose ρ^A has orthonormal decomposition $\rho^A = \sum_i p_i |i^A\rangle\langle i^A|$. To purify ρ^A we introduce a system R which has the same state space as system A, with orthonormal basis states $|i^R\rangle$, and define a pure state for the combined system

$$|AR\rangle \equiv \sum_i \sqrt{p_i} |i^A\rangle |i^R\rangle. \tag{2.207}$$

We now calculate the reduced density operator for system A corresponding to the state $|AR\rangle$:

$$\mathrm{tr}_R(|AR\rangle\langle AR|) = \sum_{ij} \sqrt{p_i p_j} |i^A\rangle\langle j^A| \, \mathrm{tr}(|i^R\rangle\langle j^R|) \tag{2.208}$$

$$= \sum_{ij} \sqrt{p_i p_j} |i^A\rangle\langle j^A| \, \delta_{ij} \tag{2.209}$$

$$= \sum_i p_i |i^A\rangle\langle i^A| \tag{2.210}$$

$$= \rho^A. \tag{2.211}$$

Thus $|AR\rangle$ is a purification of ρ^A.

Notice the close relationship of the Schmidt decomposition to purification: the procedure used to purify a mixed state of system A is to define a pure state whose Schmidt basis for system A is just the basis in which the mixed state is diagonal, with the Schmidt coefficients being the square root of the eigenvalues of the density operator being purified.

In this section we've explained two tools for studying composite quantum systems, the Schmidt decomposition and purifications. These tools will be indispensable to the study of quantum computation and quantum information, especially quantum information, which is the subject of Part III of this book.

Exercise 2.79: Consider a composite system consisting of two qubits. Find the Schmidt decompositions of the states

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}; \quad \frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2}; \text{ and } \quad \frac{|00\rangle + |01\rangle + |10\rangle}{\sqrt{3}}. \tag{2.212}$$

Exercise 2.80: Suppose $|\psi\rangle$ and $|\varphi\rangle$ are two pure states of a composite quantum system with components A and B, with identical Schmidt coefficients. Show that there are unitary transformations U on system A and V on system B such that $|\psi\rangle = (U \otimes V)|\varphi\rangle$.

Exercise 2.81: (Freedom in purifications) Let $|AR_1\rangle$ and $|AR_2\rangle$ be two purifications of a state ρ^A to a composite system AR. Prove that there exists a unitary transformation U_R acting on system R such that $|AR_1\rangle = (I_A \otimes U_R)|AR_2\rangle$.

Exercise 2.82: Suppose $\{p_i, |\psi_i\rangle\}$ is an ensemble of states generating a density matrix $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ for a quantum system A. Introduce a system R with orthonormal basis $|i\rangle$.

(1) Show that $\sum_i \sqrt{p_i}|\psi_i\rangle|i\rangle$ is a purification of ρ.

(2) Suppose we measure R in the basis $|i\rangle$, obtaining outcome i. With what probability do we obtain the result i, and what is the corresponding state of system A?

(3) Let $|AR\rangle$ be *any* purification of ρ to the system AR. Show that there exists an orthonormal basis $|i\rangle$ in which R can be measured such that the corresponding post-measurement state for system A is $|\psi_i\rangle$ with probability p_i.

2.6 EPR and the Bell inequality

Anybody who is not shocked by quantum theory has not understood it.
– Niels Bohr

> *I recall that during one walk Einstein suddenly stopped, turned to me and asked whether I really believed that the moon exists only when I look at it. The rest of this walk was devoted to a discussion of what a physicist should mean by the term 'to exist'.*
> – Abraham Pais

> *...quantum phenomena do not occur in a Hilbert space, they occur in a laboratory.*
> – Asher Peres

> *...what is proved by impossibility proofs is lack of imagination.*
> – John Bell

This chapter has focused on introducing the tools and mathematics of quantum mechanics. As these techniques are applied in the following chapters of this book, an important recurring theme is the unusual, *non-classical* properties of quantum mechanics. But what exactly is the difference between quantum mechanics and the classical world? Understanding this difference is vital in learning how to perform information processing tasks that are difficult or impossible with classical physics. This section concludes the chapter with a discussion of the Bell inequality, a compelling example of an essential difference between quantum and classical physics.

When we speak of an object such as a person or a book, we assume that the physical properties of that object have an existence independent of observation. That is, measurements merely act to *reveal* such physical properties. For example, a tennis ball has as one of its physical properties its *position*, which we typically measure using light scattered from the surface of the ball. As quantum mechanics was being developed in the 1920s and 1930s a strange point of view arose that differs markedly from the classical view. As described earlier in the chapter, according to quantum mechanics, an unobserved particle does not possess physical properties that exist independent of observation. Rather, such physical properties arise as a consequence of measurements performed upon the system. For example, according to quantum mechanics a qubit does not possess definite properties of 'spin in the z direction, σ_z', and 'spin in the x direction, σ_x', each of which can be revealed by performing the appropriate measurement. Rather, quantum mechanics gives a set of rules which specify, given the state vector, the probabilities for the possible measurement outcomes when the observable σ_z is measured, or when the observable σ_x is measured.

Many physicists rejected this new view of Nature. The most prominent objector was Albert Einstein. In the famous 'EPR paper', co-authored with Nathan Rosen and Boris Podolsky, Einstein proposed a thought experiment which, he believed, demonstrated that quantum mechanics is not a complete theory of Nature.

The essence of the EPR argument is as follows. EPR were interested in what they termed 'elements of reality'. Their belief was that any such element of reality *must* be represented in any complete physical theory. The goal of the argument was to show that quantum mechanics is not a complete physical theory, by identifying elements of reality that were not included in quantum mechanics. The way they attempted to do this was by introducing what they claimed was a *sufficient condition* for a physical property to

be an element of reality, namely, that it be possible to predict with certainty the value that property will have, immediately before measurement.

Box 2.7: Anti-correlations in the EPR experiment

Suppose we prepare the two qubit state

$$|\psi\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}, \tag{2.213}$$

a state sometimes known as the *spin singlet* for historical reasons. It is not difficult to show that this state is an entangled state of the two qubit system. Suppose we perform a measurement of spin along the $\vec{v}$ axis on both qubits, that is, we measure the observable $\vec{v} \cdot \vec{\sigma}$ (defined in Equation (2.116) on page 90) on each qubit, getting a result of +1 or -1 for each qubit. It turns out that no matter what choice of $\vec{v}$ we make, the results of the two measurements are always opposite to one another. That is, if the measurement on the first qubit yields +1, then the measurement on the second qubit will yield -1, and vice versa. It is as though the second qubit knows the result of the measurement on the first, no matter how the first qubit is measured. To see why this is true, suppose $|a\rangle$ and $|b\rangle$ are the eigenstates of $\vec{v} \cdot \vec{\sigma}$. Then there exist complex numbers $\alpha, \beta, \gamma, \delta$ such that

$$|0\rangle = \alpha|a\rangle + \beta|b\rangle \tag{2.214}$$

$$|1\rangle = \gamma|a\rangle + \delta|b\rangle. \tag{2.215}$$

Substituting we obtain

$$\frac{|01\rangle - |10\rangle}{\sqrt{2}} = (\alpha\delta - \beta\gamma)\frac{|ab\rangle - |ba\rangle}{\sqrt{2}}. \tag{2.216}$$

But $\alpha\delta - \beta\gamma$ is the determinant of the unitary matrix $\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$, and thus is equal to a phase factor $e^{i\theta}$ for some real θ. Thus

$$\frac{|01\rangle - |10\rangle}{\sqrt{2}} = \frac{|ab\rangle - |ba\rangle}{\sqrt{2}}, \tag{2.217}$$

up to an unobservable global phase factor. As a result, if a measurement of $\vec{v} \cdot \vec{\sigma}$ is performed on both qubits, then we can see that a result of +1 (-1) on the first qubit implies a result of -1 (+1) on the second qubit.

Consider, for example, an entangled pair of qubits belonging to Alice and Bob, respectively:

$$\frac{|01\rangle - |10\rangle}{\sqrt{2}}. \tag{2.218}$$

Suppose Alice and Bob are a long way away from one another. Alice performs a measurement of spin along the $\vec{v}$ axis, that is, she measures the observable $\vec{v} \cdot \vec{\sigma}$ (defined in Equation (2.116) on page 90). Suppose Alice receives the result +1. Then a simple quantum mechanical calculation, given in Box 2.7, shows that she can predict with certainty

that Bob will measure -1 on his qubit if he also measures spin along the $\vec{v}$ axis. Similarly, if Alice measured -1, then she can predict with certainty that Bob will measure $+1$ on his qubit. Because it is always possible for Alice to predict the value of the measurement result recorded when Bob's qubit is measured in the $\vec{v}$ direction, that physical property must correspond to an element of reality, by the EPR criterion, and should be represented in any complete physical theory. However, standard quantum mechanics, as we have presented it, merely tells one how to calculate the probabilities of the respective measurement outcomes if $\vec{v} \cdot \vec{\sigma}$ is measured. Standard quantum mechanics certainly does not include any fundamental element intended to represent the value of $\vec{v} \cdot \vec{\sigma}$, for all unit vectors $\vec{v}$.

The goal of EPR was to show that quantum mechanics is incomplete, by demonstrating that quantum mechanics lacked some essential 'element of reality', by their criterion. They hoped to force a return to a more classical view of the world, one in which systems could be ascribed properties which existed independently of measurements performed on those systems. Unfortunately for EPR, most physicists did not accept the above reasoning as convincing. The attempt to impose on Nature *by fiat* properties which she must obey seems a most peculiar way of studying her laws.

Indeed, Nature has had the last laugh on EPR. Nearly thirty years after the EPR paper was published, an *experimental test* was proposed that could be used to check whether or not the picture of the world which EPR were hoping to force a return to is valid or not. It turns out that Nature *experimentally invalidates* that point of view, while agreeing with quantum mechanics.

The key to this experimental invalidation is a result known as *Bell's inequality*. Bell's inequality is *not* a result about quantum mechanics, so the first thing we need to do is momentarily *forget* all our knowledge of quantum mechanics. To obtain Bell's inequality, we're going to do a thought experiment, which we will analyze using our common sense notions of how the world works – the sort of notions Einstein and his collaborators thought Nature ought to obey. After we have done the common sense analysis, we will perform a quantum mechanical analysis which we can show *is not consistent with the common sense analysis*. Nature can then be asked, by means of a real experiment, to decide between our common sense notions of how the world works, and quantum mechanics.

Imagine we perform the following experiment, illustrated in Figure 2.4. Charlie prepares two particles. It doesn't matter how he prepares the particles, just that he is capable of repeating the experimental procedure which he uses. Once he has performed the preparation, he sends one particle to Alice, and the second particle to Bob.

Once Alice receives her particle, she performs a measurement on it. Imagine that she has available two different measurement apparatuses, so she could choose to do one of two different measurements. These measurements are of physical properties which we shall label P_Q and P_R, respectively. Alice doesn't know in advance which measurement she will choose to perform. Rather, when she receives the particle she flips a coin or uses some other random method to decide which measurement to perform. We suppose for simplicity that the measurements can each have one of two outcomes, $+1$ or -1. Suppose Alice's particle has a value Q for the property P_Q. Q is assumed to be an *objective property* of Alice's particle, which is merely revealed by the measurement, much as we imagine the position of a tennis ball to be revealed by the particles of light being scattered off it. Similarly, let R denote the value revealed by a measurement of the property P_R.

Similarly, suppose that Bob is capable of measuring one of two properties, P_S or P_T, once again revealing an objectively existing value S or T for the property, each taking value +1 or −1. Bob does not decide beforehand which property he will measure, but waits until he has received the particle and then chooses randomly. The timing of the experiment is arranged so that Alice and Bob do their measurements *at the same time* (or, to use the more precise language of relativity, in a causally disconnected manner). Therefore, the measurement which Alice performs cannot disturb the result of Bob's measurement (or vice versa), since physical influences cannot propagate faster than light.

Figure 2.4. Schematic experimental setup for the Bell inequalities. Alice can choose to measure either Q or R, and Bob chooses to measure either S or T. They perform their measurements simultaneously. Alice and Bob are assumed to be far enough apart that performing a measurement on one system can not have any effect on the result of measurements on the other.

We are going to do some simple algebra with the quantity $QS + RS + RT - QT$. Notice that

$$QS + RS + RT - QT = (Q + R)S + (R - Q)T. \tag{2.219}$$

Because $R, Q = \pm 1$ it follows that either $(Q + R)S = 0$ or $(R - Q)T = 0$. In either case, it is easy to see from (2.219) that $QS + RS + RT - QT = \pm 2$. Suppose next that $p(q, r, s, t)$ is the probability that, before the measurements are performed, the system is in a state where $Q = q, R = r, S = s$, and $T = t$. These probabilities may depend on how Charlie performs his preparation, and on experimental noise. Letting $\mathbf{E}(\cdot)$ denote the mean value of a quantity, we have

$$\begin{aligned} \mathbf{E}(QS + RS + RT - QT) &= \sum_{qrst} p(q, r, s, t)(qs + rs + rt - qt) && (2.220)\\ &\leq \sum_{qrst} p(q, r, s, t) \times 2 && (2.221)\\ &= 2. && (2.222) \end{aligned}$$

Also,

$$\begin{aligned} \mathbf{E}(QS + RS + RT - QT) &= \sum_{qrst} p(q, r, s, t)qs + \sum_{qrst} p(q, r, s, t)rs \\ &\quad + \sum_{qrst} p(q, r, s, t)rt - \sum_{qrst} p(q, r, s, t)qt && (2.223)\\ &= \mathbf{E}(QS) + \mathbf{E}(RS) + \mathbf{E}(RT) - \mathbf{E}(QT). && (2.224) \end{aligned}$$

Comparing (2.222) and (2.224) we obtain the *Bell inequality*,

$$\mathbf{E}(QS) + \mathbf{E}(RS) + \mathbf{E}(RT) - \mathbf{E}(QT) \leq 2. \tag{2.225}$$

This result is also often known as the *CHSH inequality* after the initials of its four discoverers. It is part of a larger set of inequalities known generically as Bell inequalities, since the first was found by John Bell.

By repeating the experiment many times, Alice and Bob can determine each quantity on the left hand side of the Bell inequality. For example, after finishing a set of experiments, Alice and Bob get together to analyze their data. They look at all the experiments where Alice measured P_Q and Bob measured P_S. By multiplying the results of their experiments together, they get a sample of values for QS. By averaging over this sample, they can estimate $\mathbf{E}(QS)$ to an accuracy only limited by the number of experiments which they perform. Similarly, they can estimate all the other quantities on the left hand side of the Bell inequality, and thus check to see whether it is obeyed in a real experiment.

It's time to put some quantum mechanics back in the picture. Imagine we perform the following quantum mechanical experiment. Charlie prepares a quantum system of two qubits in the state

$$|\psi\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}. \tag{2.226}$$

He passes the first qubit to Alice, and the second qubit to Bob. They perform measurements of the following observables:

$$Q = Z_1 \qquad S = \frac{-Z_2 - X_2}{\sqrt{2}} \tag{2.227}$$

$$R = X_1 \qquad T = \frac{Z_2 - X_2}{\sqrt{2}}. \tag{2.228}$$

Simple calculations show that the average values for these observables, written in the quantum mechanical $\langle \cdot \rangle$ notation, are:

$$\langle QS \rangle = \frac{1}{\sqrt{2}};\ \langle RS \rangle = \frac{1}{\sqrt{2}};\ \langle RT \rangle = \frac{1}{\sqrt{2}};\ \langle QT \rangle = -\frac{1}{\sqrt{2}}. \tag{2.229}$$

Thus,

$$\langle QS \rangle + \langle RS \rangle + \langle RT \rangle - \langle QT \rangle = 2\sqrt{2}. \tag{2.230}$$

Hold on! We learned back in (2.225) that the average value of QS plus the average value of RS plus the average value of RT minus the average value of QT can never exceed two. Yet here, quantum mechanics predicts that this sum of averages yields $2\sqrt{2}$!

Fortunately, we can ask Nature to resolve the apparent paradox for us. Clever experiments using photons – particles of light – have been done to check the prediction (2.230) of quantum mechanics versus the Bell inequality (2.225) which we were led to by our common sense reasoning. The details of the experiments are outside the scope of the book, but the results were resoundingly in favor of the quantum mechanical prediction. The Bell inequality (2.225) is *not* obeyed by Nature.

What does this mean? It means that one or more of the assumptions that went into the derivation of the Bell inequality must be incorrect. Vast tomes have been written analyzing the various forms in which this type of argument can be made, and analyzing the subtly different assumptions which must be made to reach Bell-like inequalities. Here we merely summarize the main points.

There are two assumptions made in the proof of (2.225) which are questionable:

(1) The assumption that the physical properties P_Q, P_R, P_S, P_T have definite values Q, R, S, T which exist independent of observation. This is sometimes known as the assumption of *realism*.
(2) The assumption that Alice performing her measurement does not influence the result of Bob's measurement. This is sometimes known as the assumption of *locality*.

These two assumptions together are known as the assumptions of *local realism*. They are certainly intuitively plausible assumptions about how the world works, and they fit our everyday experience. Yet the Bell inequalities show that at least one of these assumptions is not correct.

What can we learn from Bell's inequality? For physicists, the most important lesson is that their deeply held commonsense intuitions about how the world works are wrong. The world is *not* locally realistic. Most physicists take the point of view that it is the assumption of realism which needs to be dropped from our worldview in quantum mechanics, although others have argued that the assumption of locality should be dropped instead. Regardless, Bell's inequality together with substantial experimental evidence now points to the conclusion that either or both of locality and realism must be dropped from our view of the world if we are to develop a good intuitive understanding of quantum mechanics.

What lessons can the fields of quantum computation and quantum information learn from Bell's inequality? Historically the most useful lesson has perhaps also been the most vague: there is something profoundly 'up' with entangled states like the EPR state. A lot of mileage in quantum computation and, especially, quantum information, has come from asking the simple question: 'what would some entanglement buy me in this problem?' As we saw in teleportation and superdense coding, and as we will see repeatedly later in the book, by throwing some entanglement into a problem we open up a new world of possibilities unimaginable with classical information. The bigger picture is that Bell's inequality teaches us that entanglement is a fundamentally new resource in the world that goes essentially *beyond* classical resources; iron to the classical world's bronze age. A major task of quantum computation and quantum information is to exploit this new resource to do information processing tasks impossible or much more difficult with classical resources.

Problem 2.1: (Functions of the Pauli matrices) Let $f(\cdot)$ be any function from complex numbers to complex numbers. Let $\vec{n}$ be a normalized vector in three dimensions, and let θ be real. Show that

$$f(\theta \vec{n} \cdot \vec{\sigma}) = \frac{f(\theta) + f(-\theta)}{2} I + \frac{f(\theta) - f(-\theta)}{2} \vec{n} \cdot \vec{\sigma}. \tag{2.231}$$

Problem 2.2: (Properties of the Schmidt number) Suppose $|\psi\rangle$ is a pure state of a composite system with components A and B.

(1) Prove that the Schmidt number of $|\psi\rangle$ is equal to the rank of the reduced density matrix $\rho_A \equiv \mathrm{tr}_B(|\psi\rangle\langle\psi|)$. (Note that the rank of a Hermitian operator is equal to the dimension of its support.)
(2) Suppose $|\psi\rangle = \sum_j |\alpha_j\rangle|\beta_j\rangle$ is a representation for $|\psi\rangle$, where $|\alpha_j\rangle$ and $|\beta_j\rangle$ are (un-normalized) states for systems A and B, respectively. Prove that the

number of terms in such a decomposition is greater than or equal to the Schmidt number of $|\psi\rangle$, $\mathrm{Sch}(\psi)$.

(3) Suppose $|\psi\rangle = \alpha|\varphi\rangle + \beta|\gamma\rangle$. Prove that

$$\mathrm{Sch}(\psi) \geq |\mathrm{Sch}(\varphi) - \mathrm{Sch}(\gamma)| . \tag{2.232}$$

Problem 2.3: (Tsirelson's inequality) Suppose $Q = \vec{q} \cdot \vec{\sigma}, R = \vec{r} \cdot \vec{\sigma}, S = \vec{s} \cdot \vec{\sigma}, T = \vec{t} \cdot \vec{\sigma}$, where $\vec{q}, \vec{r}, \vec{s}$ and $\vec{t}$ are real unit vectors in three dimensions. Show that

$$(Q \otimes S + R \otimes S + R \otimes T - Q \otimes T)^2 = 4I + [Q, R] \otimes [S, T]. \tag{2.233}$$

Use this result to prove that

$$\langle Q \otimes S \rangle + \langle R \otimes S \rangle + \langle R \otimes T \rangle - \langle Q \otimes T \rangle \leq 2\sqrt{2}, \tag{2.234}$$

so the violation of the Bell inequality found in Equation (2.230) is the maximum possible in quantum mechanics.

History and further reading

There are an enormous number of books on linear algebra at levels ranging from High School through to Graduate School. Perhaps our favorites are the two volume set by Horn and Johnson[HJ85, HJ91], which cover an extensive range of topics in an accessible manner. Other useful references include Marcus and Minc[MM92], and Bhatia[Bha97]. Good introductions to linear algebra include Halmos[Hal58], Perlis[Per52], and Strang[Str76].

There are many excellent books on quantum mechanics. Unfortunately, most of these books focus on topics of tangential interest to quantum information and computation. Perhaps the most relevant in the existing literature is Peres' superb book[Per93]. Beside an extremely clear exposition of elementary quantum mechanics, Peres gives an extensive discussion of the Bell inequalities and related results. Good introductory level texts include Sakurai's book[Sak95], Volume III of the superb series by Feynman, Leighton, and Sands[FLS65a], and the two volume work by Cohen-Tannoudji, Diu and Laloë[CTDL77a, CTDL77b]. All three of these works are somewhat closer in spirit to quantum computation and quantum information than are most other quantum mechanics texts, although the great bulk of each is still taken up by applications far removed from quantum computation and quantum information. As a result, none of these texts need be read in detail by someone interested in learning about quantum computation and quantum information. However, any one of these texts may prove handy as a reference, especially when reading articles by physicists. References for the history of quantum mechanics may be found at the end of Chapter 1.

Many texts on quantum mechanics deal only with projective measurements. For applications to quantum computing and quantum information it is more convenient – and, we believe, easier for novices – to start with the general description of measurements, of which projective measurements can be regarded as a special case. Of course, ultimately, as we have shown, the two approaches are equivalent. The theory of generalized measurements which we have employed was developed between the 1940s and 1970s. Much of the history can be distilled from the book of Kraus[Kra83]. Interesting discussion of quantum measurements may be found in Section 2.2 of Gardiner[Gar91], and in the book by Braginsky and Khahili[BK92]. The POVM measurement for distinguishing

non-orthogonal states described in Section 2.2.6 is due to Peres[Per88]. The extension described in Exercise 2.64 appeared in Duan and Guo[DG98].

Superdense coding was invented by Bennett and Wiesner[BW92]. An experiment implementing a variant of superdense coding using entangled photon pairs was performed by Mattle, Weinfurter, Kwiat, and Zeilinger[MWKZ96].

The density operator formalism was introduced independently by Landau[Lan27] and by von Neumann[von27]. The unitary freedom in the ensemble for density matrices, Theorem 2.6, was first pointed out by Schrödinger[Sch36], and was later rediscovered and extended by Jaynes[Jay57] and by Hughston, Jozsa and Wootters[HJW93]. The result of Exercise 2.73 is from the paper by Jaynes, and the results of Exercises 2.81 and 2.82 appear in the paper by Hughston, Jozsa and Wootters. The class of probability distributions which may appear in a density matrix decomposition for a given density matrix has been studied by Uhlmann[Uhl70] and by Nielsen[Nie99b]. Schmidt's eponymous decomposition appeared in[Sch06]. The result of Exercise 2.77 was noted by Peres[Per95].

The EPR thought experiment is due to Einstein, Podolsky and Rosen[EPR35], and was recast in essentially the form we have given here by Bohm[Boh51]. It is sometimes misleadingly referred to as the EPR 'paradox'. The Bell inequality is named in honour of Bell[Bel64], who first derived inequalities of this type. The form we have presented is due to Clauser, Horne, Shimony, and Holt[CHSH69], and is often known as the CHSH inequality. This inequality was derived independently by Bell, who did not publish the result.

Part 3 of Problem 2.2 is due to Thapliyal (private communication). Tsirelson's inequality is due to Tsirelson[Tsi80].

3 Introduction to computer science

In natural science, Nature has given us a world and we're just to discover its laws. In computers, we can stuff laws into it and create a world.
– Alan Kay

Our field is still in its embryonic stage. It's great that we haven't been around for 2000 years. We are still at a stage where very, very important results occur in front of our eyes.
– Michael Rabin, on computer science

Algorithms are the key concept of computer science. An algorithm is a precise recipe for performing some task, such as the elementary algorithm for adding two numbers which we all learn as children. This chapter outlines the modern theory of algorithms developed by computer science. Our fundamental model for algorithms will be the *Turing machine*. This is an idealized computer, rather like a modern personal computer, but with a simpler set of basic instructions, and an idealized unbounded memory. The apparent simplicity of Turing machines is misleading; they are very powerful devices. We will see that they can be used to execute any algorithm whatsoever, even one running on an apparently much more powerful computer.

The fundamental question we are trying to address in the study of algorithms is: what resources are required to perform a given computational task? This question splits up naturally into two parts. First, we'd like to understand what computational tasks are possible, preferably by giving explicit algorithms for solving specific problems. For example, we have many excellent examples of algorithms that can quickly sort a list of numbers into ascending order. The second aspect of this question is to demonstrate *limitations* on what computational tasks may be accomplished. For example, lower bounds can be given for the number of operations that must be performed by any algorithm which sorts a list of numbers into ascending order. Ideally, these two tasks – the finding of algorithms for solving computational problems, and proving limitations on our ability to solve computational problems – would dovetail perfectly. In practice, a significant gap often exists between the best techniques known for solving a computational problem, and the most stringent limitations known on the solution. The purpose of this chapter is to give a broad overview of the tools which have been developed to aid in the analysis of computational problems, and in the construction and analysis of algorithms to solve such problems.

Why should a person interested in *quantum* computation and *quantum* information spend time investigating *classical* computer science? There are three good reasons for this effort. First, classical computer science provides a vast body of concepts and techniques which may be reused to great effect in quantum computation and quantum information. Many of the triumphs of quantum computation and quantum information have come by combining existing ideas from computer science with novel ideas from quantum

mechanics. For example, some of the fast algorithms for quantum computers are based upon the Fourier transform, a powerful tool utilized by many classical algorithms. Once it was realized that quantum computers could perform a type of Fourier transform much more quickly than classical computers this enabled the development of many important quantum algorithms.

Second, computer scientists have expended great effort understanding what resources are required to perform a given computational task on a classical computer. These results can be used as the basis for a comparison with quantum computation and quantum information. For example, much attention has been focused on the problem of finding the prime factors of a given number. On a classical computer this problem is believed to have no 'efficient' solution, where 'efficient' has a meaning we'll explain later in the chapter. What is interesting is that an efficient solution to this problem *is* known for quantum computers. The lesson is that, for this task of finding prime factors, there appears to be a *gap* between what is possible on a classical computer and what is possible on a quantum computer. This is both intrinsically interesting, and also interesting in the broader sense that it suggests such a gap may exist for a wider class of computational problems than merely the finding of prime factors. By studying this specific problem further, it may be possible to discern features of the problem which make it more tractable on a quantum computer than on a classical computer, and then act on these insights to find interesting quantum algorithms for the solution of other problems.

Third, and most important, there is learning to *think* like a computer scientist. Computer scientists think in a rather different style than does a physicist or other natural scientist. Anybody wanting a deep understanding of quantum computation and quantum information must learn to think like a computer scientist at least some of the time; they must instinctively know what problems, what techniques, and most importantly what problems are of greatest interest to a computer scientist.

The structure of this chapter is as follows. In Section 3.1 we introduce two models for computation: the Turing machine model, and the circuit model. The Turing machine model will be used as our fundamental model for computation. In practice, however, we mostly make use of the circuit model of computation, and it is this model which is most useful in the study of quantum computation. With our models for computation in hand, the remainder of the chapter discusses resource requirements for computation. Section 3.2 begins by overviewing the computational tasks we are interested in as well as discusing some associated resource questions. It continues with a broad look at the key concepts of *computational complexity*, a field which examines the time and space requirements necessary to solve particular computational problems, and provides a broad classification of problems based upon their difficulty of solution. Finally, the section concludes with an examination of the energy resources required to perform computations. Surprisingly, it turns out that the energy required to perform a computation can be made vanishingly small, provided one can make the computation reversible. We explain how to construct reversible computers, and explain some of the reasons they are important both for computer science and for quantum computation and quantum information. Section 3.3 concludes the chapter with a broad look at the entire field of computer science, focusing on issues of particular relevance to quantum computation and quantum information.

3.1 Models for computation

> *...algorithms are concepts that have existence apart from any programming language.*
> – Donald Knuth

What does it mean to have an *algorithm* for performing some task? As children we all learn a procedure which enables us to add together any two numbers, no matter how large those numbers are. This is an example of an algorithm. Finding a mathematically precise formulation of the concept of an algorithm is the goal of this section.

Historically, the notion of an algorithm goes back centuries; undergraduates learn Euclid's two thousand year old algorithm for finding the greatest common divisor of two positive integers. However, it wasn't until the 1930s that the fundamental notions of the modern theory of algorithms, and thus of computation, were introduced, by Alonzo Church, Alan Turing, and other pioneers of the computer era. This work arose in response to a profound challenge laid down by the great mathematician David Hilbert in the early part of the twentieth century. Hilbert asked whether or not there existed some algorithm which could be used, in principle, to solve all the problems of mathematics. Hilbert expected that the answer to this question, sometimes known as the *entscheidungsproblem*, would be yes.

Amazingly, the answer to Hilbert's challenge turned out to be no: there is no algorithm to solve all mathematical problems. To prove this, Church and Turing had to solve the deep problem of capturing in a mathematical definition what we mean when we use the intuitive concept of an algorithm. In so doing, they laid the foundations for the modern theory of algorithms, and consequently for the modern theory of computer science.

In this chapter, we use two ostensibly different approaches to the theory of computation. The first approach is that proposed by Turing. Turing defined a class of machines, now known as *Turing machines*, in order to capture the notion of an algorithm to perform a computational task. In Section 3.1.1, we describe Turing machines, and then discuss some of the simpler variants of the Turing machine model. The second approach is via the *circuit model* of computation, an approach that is especially useful as preparation for our later study of quantum computers. The circuit model is described in Section 3.1.2. Although these models of computation appear different on the surface, it turns out that they are equivalent. Why introduce more than one model of computation, you may ask? We do so because different models of computation may yield different insights into the solution of specific problems. Two (or more) ways of thinking about a concept are better than one.

3.1.1 Turing machines

The basic elements of a Turing machine are illustrated in Figure 3.1. A Turing machine contains four main elements: (a) a *program*, rather like an ordinary computer; (b) a *finite state control*, which acts like a stripped-down microprocessor, co-ordinating the other operations of the machine; (c) a *tape*, which acts like a computer memory; and (d) a *read-write tape-head*, which points to the position on the tape which is currently readable or writable. We now describe each of these four elements in more detail.

The *finite state control* for a Turing machine consists of a finite set of *internal states*,

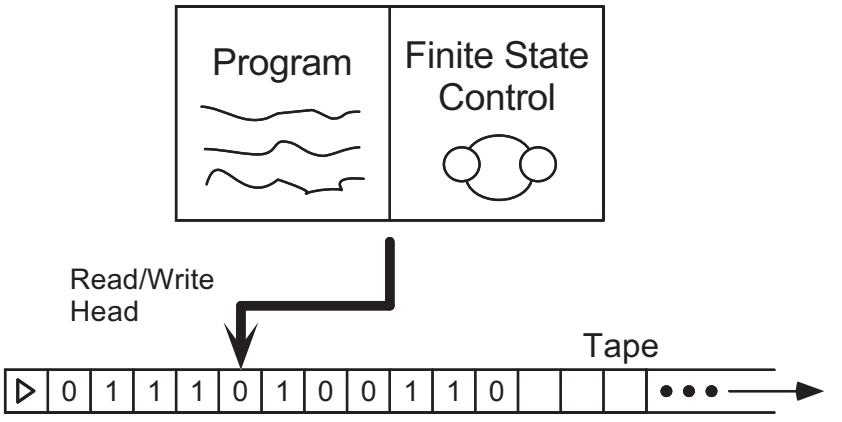


Figure 3.1. Main elements of a Turing machine. In the text, blanks on the tape are denoted by a 'b'. Note the ▷ marking the left hand end of the tape.

$q_1, \ldots, q_m$. The number m is allowed to be varied; it turns out that for m sufficiently large this does not affect the power of the machine in any essential way, so without loss of generality we may suppose that m is some fixed constant. The best way to think of the finite state control is as a sort of microprocessor, co-ordinating the Turing machine's operation. It provides temporary storage off-tape, and a central place where all processing for the machine may be done. In addition to the states $q_1, \ldots, q_m$, there are also two special internal states, labelled $q_{\rm s}$ and $q_{\rm h}$. We call these the *starting state* and the *halting state*, respectively. The idea is that at the beginning of the computation, the Turing machine is in the starting state $q_{\rm s}$. The execution of the computation causes the Turing machine's internal state to change. If the computation ever finishes, the Turing machine ends up in the state $q_{\rm h}$ to indicate that the machine has completed its operation.

The Turing machine *tape* is a one-dimensional object, which stretches off to infinity in one direction. The tape consists of an infinite sequence of *tape squares*. We number the tape squares $0, 1, 2, 3, \ldots$. The tape squares each contain one symbol drawn from some *alphabet*, Γ, which contains a finite number of distinct symbols. For now, it will be convenient to assume that the alphabet contains four symbols, which we denote by 0, 1, b (the 'blank' symbol), and ▷, to mark the left hand edge of the tape. Initially, the tape contains a ▷ at the left hand end, a finite number of 0s and 1s, and the rest of the tape contains blanks. The *read-write tape-head* identifies a single square on the Turing machine tape as the square that is currently being accessed by the machine.

Summarizing, the machine starts its operation with the finite state control in the state $q_{\rm s}$, and with the read-write head at the leftmost tape square, the square numbered 0. The computation then proceeds in a step by step manner according to the *program*, to be defined below. If the current state is $q_{\rm h}$, then the computation has halted, and the *output* of the computation is the current (non-blank) contents of the tape.

A *program* for a Turing machine is a finite ordered list of *program lines* of the form $\langle q, x, q', x', s\rangle$. The first item in the program line, q, is a state from the set of internal states of the machine. The second item, x, is taken from the alphabet of symbols which may appear on the tape, Γ. The way the program works is that on each machine cycle, the Turing machine looks through the list of program lines in order, searching for a line $\langle q, x, \cdot, \cdot, \cdot\rangle$, such that the current internal state of the machine is q, and the symbol

being read on the tape is x. If it doesn't find such a program line, the internal state of the machine is changed to $q_{\rm h}$, and the machine halts operation. If such a line is found, then that program line is *executed*. Execution of a program line involves the following steps: the internal state of the machine is changed to q'; the symbol x on the tape is overwritten by the symbol x', and the tape-head moves left, right, or stands still, depending on whether s is -1, $+1$, or 0, respectively. The only exception to this rule is if the tape-head is at the leftmost tape square, and $s = -1$, in which case the tape-head stays put.

Now that we know what a Turing machine is, let's see how it may be used to compute a simple function. Consider the following example of a Turing machine. The machine starts with a binary number, x, on the tape, followed by blanks. The machine has three internal states, q_1, q_2, and q_3, in addition to the starting state $q_{\rm s}$ and halting state $q_{\rm h}$. The program contains the following program lines (the numbers on the left hand side are for convenience in referring to the program lines in later discussion, and do not form part of the program):

$$\begin{aligned}
1:&\ \langle q_{\rm s}, \triangleright, q_1, \triangleright, +1\rangle \\
2:&\ \langle q_1, 0, q_1, b, +1\rangle \\
3:&\ \langle q_1, 1, q_1, b, +1\rangle \\
4:&\ \langle q_1, b, q_2, b, -1\rangle \\
5:&\ \langle q_2, b, q_2, b, -1\rangle \\
6:&\ \langle q_2, \triangleright, q_3, \triangleright, +1\rangle \\
7:&\ \langle q_3, b, q_{\rm h}, 1, 0\rangle.
\end{aligned}$$

What function does this program compute? Initially the machine is in the state $q_{\rm s}$ and at the left-most tape position so line 1, $\langle q_{\rm s}, \triangleright, q_1, \triangleright, +1\rangle$, is executed, which causes the tape-head to move right without changing what is written on the tape, but changing the internal state of the machine to q_1. The next three lines of the program ensure that while the machine is in the state q_1 the tape-head will continue moving right while it reads either 0s (line 2) or 1s (line 3) on the tape, over-writing the tape contents with blanks as it goes and remaining in the state q_1, until it reaches a tape square that is already blank, at which point the tape-head is moved one position to the left, and the internal state is changed to q_2 (line 4). Line 5 then ensures that the tape-head keeps moving left while blanks are being read by the tape-head, without changing the contents of the tape. This keeps up until the tape-head returns to its starting point, at which point it reads a $\triangleright$ on the tape, changes the internal state to q_3, and moves one step to the right (line 6). Line 7 completes the program, simply printing the number 1 onto the tape, and then halting.

The preceding analysis shows that this program computes the constant function $f(x) = 1$. That is, regardless of what number is input on the tape the number 1 is output. More generally, a Turing machine can be thought of as computing functions from the non-negative integers to the non-negative integers; the initial state of the tape is used to represent the input to the function, and the final state of the tape is used to represent the output of the function.

It seems as though we have gone to a very great deal of trouble to compute this simple function using our Turing machines. Is it possible to build up more complicated functions using Turing machines? For example, could we construct a machine such that when two numbers, x and y, are input on the tape with a blank to demarcate them, it will

output the sum $x + y$ on the tape? More generally, what class of functions is it possible to compute using a Turing machine?

It turns out that the Turing machine model of computation can be used to compute an enormous variety of functions. For example, it can be used to do all the basic arithmetical operations, to search through text represented as strings of bits on the tape, and many other interesting operations. Surprisingly, it turns out that a Turing machine can be used to simulate all the operations performed on a modern computer! Indeed, according to a thesis put forward independently by Church and by Turing, the Turing machine model of computation *completely captures* the notion of computing a function using an algorithm. This is known as the *Church–Turing thesis*:

The class of functions computable by a Turing machine corresponds exactly to the class of functions which we would naturally regard as being computable by an algorithm.

The Church–Turing thesis asserts an equivalence between a rigorous mathematical concept – function computable by a Turing machine – and the intuitive concept of what it means for a function to be computable by an algorithm. The thesis derives its importance from the fact that it makes the study of real-world algorithms, prior to 1936 a rather vague concept, amenable to rigorous mathematical study. To understand the significance of this point it may be helpful to consider the definition of a *continuous function* from real analysis. Every child can tell you what it means for a line to be continuous on a piece of paper, but it is far from obvious how to capture that intuition in a rigorous definition. Mathematicians in the nineteenth century spent a great deal of time arguing about the merits of various definitions of continuity before the modern definition of continuity came to be accepted. When making fundamental definitions like that of continuity or of computability it is important that good definitions be chosen, ensuring that one's intuitive notions closely match the precise mathematical definition. From this point of view the Church–Turing thesis is simply the assertion that the Turing machine model of computation provides a good foundation for computer science, capturing the intuitive notion of an algorithm in a rigorous definition.

A priori it is not obvious that every function which we would intuitively regard as computable by an algorithm can be computed using a Turing machine. Church, Turing and many other people have spent a great deal of time gathering evidence for the Church–Turing thesis, and in sixty years no evidence to the contrary has been found. Nevertheless, it is possible that in the future we will discover in Nature a process which computes a function not computable on a Turing machine. It would be wonderful if that ever happened, because we could then harness that process to help us perform new computations which could not be performed before. Of course, we would also need to overhaul the definition of computability, and with it, computer science.

Exercise 3.1: (**Non-computable processes in Nature**) How might we recognize that a process in Nature computes a function not computable by a Turing machine?

Exercise 3.2: (**Turing numbers**) Show that single-tape Turing machines can each be given a number from the list $1, 2, 3, \ldots$ in such a way that the number uniquely specifies the corresponding machine. We call this number the *Turing number* of the corresponding Turing machine. (*Hint*: Every positive integer has

a unique prime factorization $p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$, where p_i are distinct prime numbers, and $a_1, \ldots, a_k$ are non-negative integers.)

In later chapters, we will see that quantum computers also obey the Church–Turing thesis. That is, quantum computers can compute the same class of functions as is computable by a Turing machine. The difference between quantum computers and Turing machines turns out to lie in the *efficiency* with which the computation of the function may be performed – there are functions which can be computed much more efficiently on a quantum computer than is believed to be possible with a classical computing device such as a Turing machine.

Demonstrating in complete detail that the Turing machine model of computation can be used to build up all the usual concepts used in computer programming languages is beyond the scope of this book (see 'History and further reading' at the end of the chapter for more information). When specifying algorithms, instead of explicitly specifying the Turing machine used to compute the algorithm, we shall usually use a much higher level *pseudocode*, trusting in the Church–Turing thesis that this pseudocode can be translated into the Turing machine model of computation. We won't give any sort of rigorous definition for pseudocode. Think of it as a slightly more formal version of English or, if you like, a sloppy version of a high-level programming language such as C++ or BASIC. Pseudocode provides a convenient way of expressing algorithms, without going into the extreme level of detail required by a Turing machine. An example use of pseudocode may be found in Box 3.2 on page 130; it is also used later in the book to describe quantum algorithms.

There are many variants on the basic Turing machine model. We might imagine Turing machines with different kinds of tapes. For example, one could consider two-way infinite tapes, or perhaps computation with tapes of more than one dimension. So far as is presently known, it is not possible to change any aspect of the Turing model in a way that is physically reasonable, and which manages to extend the class of functions computable by the model.

As an example consider a Turing machine equipped with multiple tapes. For simplicity we consider the two-tape case, as the generalization to more than two tapes is clear from this example. Like the basic Turing machine, a two-tape Turing machine has a finite number of internal states $q_1, \ldots, q_m$, a start state $q_{\rm s}$, and a halt state $q_{\rm h}$. It has two tapes, each of which contain symbols from some finite alphabet of symbols, Γ. As before we find it convenient to assume that the alphabet contains four symbols, $0, 1, b$ and $\triangleright$, where $\triangleright$ marks the left hand edge of each tape. The machine has two tape-heads, one for each tape. The main difference between the two-tape Turing machine and the basic Turing machine is in the program. Program lines are of the form $\langle q, x_1, x_2, q', x_1', x_2', s_1, s_2 \rangle$, meaning that if the internal state of the machine is q, tape one is reading x_1 at its current position, and tape two is reading x_2 at its current position, then the internal state of the machine should be changed to q', x_1 overwritten with x_1', x_2 overwritten with x_2', and the tape-heads for tape one and tape two moved according to whether s_1 or s_2 are equal to $+1, -1$ or 0, respectively.

In what sense are the basic Turing machine and the two-tape Turing machine equivalent models of computation? They are equivalent in the sense that each computational model is able to *simulate* the other. Suppose we have a two-tape Turing machine which takes as input a bit string x on the first tape and blanks on the remainder of both tapes,

except the endpoint marker $\triangleright$. This machine computes a function $f(x)$, where $f(x)$ is defined to be the contents of the first tape after the Turing machine has halted. Rather remarkably, it turns out that given a two-tape Turing machine to compute f, there exists an equivalent single-tape Turing machine that is also able to compute f. We won't explain how to do this explicitly, but the basic idea is that the single-tape Turing machine *simulates* the two-tape Turing machine, using its single tape to store the contents of both tapes of the two-tape Turing machine. There is some computational overhead required to do this simulation, but the important point is that in principle it can always be done. In fact, there exists a Universal Turing machine (see Box 3.1) which can simulate any other Turing machine!

Another interesting variant of the Turing machine model is to introduce randomness into the model. For example, imagine that the Turing machine can execute a program line whose effect is the following: if the internal state is q and the tape-head reads x, then flip an unbiased coin. If the coin lands heads, change the internal state to $q_{i_{\rm H}}$, and if it lands tails, change the internal state to $q_{i_{\rm T}}$, where $q_{i_{\rm H}}$ and $q_{i_{\rm T}}$ are two internal states of the Turing machine. Such a program line can be represented as $\langle q, x, q_{i_{\rm H}}, q_{i_{\rm T}} \rangle$. However, even this variant doesn't change the essential power of the Turing machine model of computation. It is not difficult to see that we can simulate the effect of the above algorithm on a deterministic Turing machine by explicitly 'searching out' all the possible computational paths corresponding to different values of the coin tosses. Of course, this deterministic simulation may be far less efficient than the random model, but the key point for the present discussion is that the class of functions computable is not changed by introducing randomness into the underlying model.

Exercise 3.3: (Turing machine to reverse a bit string) Describe a Turing machine which takes a binary number x as input, and outputs the bits of x in reverse order. (*Hint*: In this exercise and the next it may help to use a multi-tape Turing machine and/or symbols other than $\triangleright, 0, 1$ and the blank.)

Exercise 3.4: (Turing machine to add modulo 2) Describe a Turing machine to add two binary numbers x and y modulo 2. The numbers are input on the Turing machine tape in binary, in the form x, followed by a single blank, followed by a y. If one number is not as long as the other then you may assume that it has been padded with leading 0s to make the two numbers the same length.

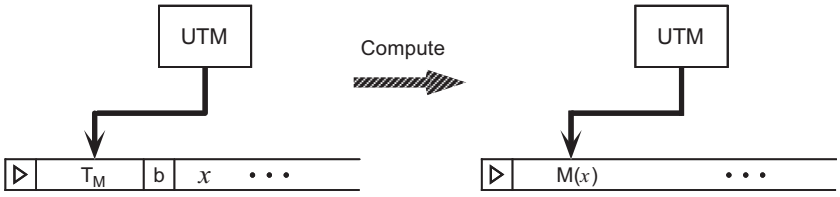
Let us return to Hilbert's entscheidungsproblem, the original inspiration for the founders of computer science. Is there an algorithm to decide all the problems of mathematics? The answer to this question was shown by Church and Turing to be no. In Box 3.2, we explain Turing's proof of this remarkable fact. This phenomenon of *undecidability* is now known to extend far beyond the examples which Church and Turing constructed. For example, it is known that the problem of deciding whether two topological spaces are topologically equivalent ('homeomorphic') is undecidable. There are simple problems related to the behavior of dynamical systems which are undecidable, as you will show in Problem 3.4. References for these and other examples are given in the end of chapter 'History and further reading'.

Besides its intrinsic interest, undecidability foreshadows a topic of great concern in computer science, and also to quantum computation and quantum information: the dis-

Box 3.1: The Universal Turing Machine

We've described Turing machines as containing three elements which may vary from machine to machine – the initial configuration of the tape, the internal states of the finite state control, and the program for the machine. A clever idea known as the *Universal Turing Machine* (UTM) allows us to fix the program and finite state control once and for all, leaving the initial contents of the tape as the only part of the machine which needs to be varied.

The Universal Turing Machine (see the figure below) has the following property. Let M be any Turing machine, and let T_M be the Turing number associated to machine M. Then on input of the binary representation for T_M followed by a blank, followed by any string of symbols x on the remainder of the tape, the Universal Turing Machine gives as output whatever machine M would have on input of x. Thus, the Universal Turing Machine is capable of simulating any other Turing machine!



The Universal Turing Machine is similar in spirit to a modern programmable computer, in which the action to be taken by the computer – the 'program' – is stored in memory, analogous to the bit string T_M stored at the beginning of the tape by the Universal Turing Machine. The data to be processed by the program is stored in a separate part of memory, analogous to the role of x in the Universal Turing Machine. Then some fixed hardware is used to run the program, producing the output. This fixed hardware is analogous to the internal states and the (fixed) program being executed by the Universal Turing Machine.

Describing the detailed construction of a Universal Turing Machine is beyond the scope of this book. (Though industrious readers may like to attempt the construction.) The key point is the existence of such a machine, showing that a single fixed machine can be used to run any algorithm whatsoever. The existence of a Universal Turing Machine also explains our earlier statement that the number of internal states in a Turing machine does not matter much, for provided that number m exceeds the number needed for a Universal Turing Machine, such a machine can be used to simulate a Turing machine with any number of internal states.

tinction between problems which are easy to solve, and problems which are hard to solve. Undecidability provides the ultimate example of problems which are hard to solve – so hard that they are in fact impossible to solve.

Exercise 3.5: (Halting problem with no inputs) Show that given a Turing

machine M there is no algorithm to determine whether M halts when the input to the machine is a blank tape.

Exercise 3.6: (Probabilistic halting problem) Suppose we number the probabilistic Turing machines using a scheme similar to that found in Exercise 3.2 and define the probabilistic halting function $h_p(x)$ to be 1 if machine x halts on input of x with probability at least $1/2$ and 0 if machine x halts on input of x with probability less than $1/2$. Show that there is no probabilistic Turing machine which can output $h_p(x)$ with probability of correctness strictly greater than $1/2$ for all x.

Exercise 3.7: (Halting oracle) Suppose a *black box* is made available to us which takes a non-negative integer x as input, and then outputs the value of $h(x)$, where $h(\cdot)$ is the halting function defined in Box 3.2 on page 130. This type of black box is sometimes known as an *oracle* for the halting problem. Suppose we have a regular Turing machine which is augmented by the power to call the oracle. One way of accomplishing this is to use a two-tape Turing machine, and add an extra program instruction to the Turing machine which results in the oracle being called, and the value of $h(x)$ being printed on the second tape, where x is the current contents of the second tape. It is clear that this model for computation is more powerful than the conventional Turing machine model, since it can be used to compute the halting function. Is the halting problem for this model of computation undecidable? That is, can a Turing machine aided by an oracle for the halting problem decide whether a program for the Turing machine with oracle will halt on a particular input?

3.1.2 Circuits

Turing machines are rather idealized models of computing devices. Real computers are *finite* in size, whereas for Turing machines we assumed a computer of unbounded size. In this section we investigate an alternative model of computation, the *circuit model*, that is equivalent to the Turing machine in terms of computational power, but is more convenient and realistic for many applications. In particular the circuit model of computation is especially important as preparation for our investigation of quantum computers.

A circuit is made up of *wires* and *gates*, which carry information around, and perform simple computational tasks, respectively. For example, Figure 3.2 shows a simple circuit which takes as input a single bit, a. This bit is passed through a NOT gate, which flips the bit, taking 1 to 0 and 0 to 1. The wires before and after the NOT gate serve merely to carry the bit to and from the NOT gate; they can represent movement of the bit through space, or perhaps just through time.

More generally, a circuit may involve many input and output bits, many wires, and many logical gates. A *logic gate* is a function $f : \{0,1\}^k \rightarrow \{0,1\}^l$ from some fixed number k of *input bits* to some fixed number l of *output bits*. For example, the NOT gate is a gate with one input bit and one output bit which computes the function $f(a) = 1 \oplus a$, where a is a single bit, and $\oplus$ is modulo 2 addition. It is also usual to make the convention that no loops are allowed in the circuit, to avoid possible instabilities, as illustrated in Figure 3.3. We say such a circuit is *acyclic*, and we adhere to the convention that circuits in the circuit model of computation be acyclic.

Box 3.2: The halting problem

In Exercise 3.2 you showed that each Turing machine can be uniquely associated with a number from the list $1, 2, 3, \ldots$. To solve Hilbert's problem, Turing used this numbering to pose the *halting problem*: does the machine with Turing number x halt upon input of the number y? This is a well posed and interesting mathematical problem. After all, it is a matter of some considerable interest to us whether our algorithms halt or not. Yet it turns out that there is no algorithm which is capable of solving the halting problem. To see this, Turing asked whether there is an algorithm to solve an even more specialized problem: does the machine with Turing number x halt upon input of the same number x? Turing defined the *halting function*,

$$h(x) \equiv \begin{cases} 0 & \text{if machine number } x \text{ does not halt upon input of } x \\ 1 & \text{if machine number } x \text{ halts upon input of } x. \end{cases}$$

If there is an algorithm to solve the halting problem, then there surely is an algorithm to evaluate $h(x)$. We will try to reach a contradiction by supposing such an algorithm exists, denoted by HALT(x). Consider an algorithm computing the function TURING(x), with pseudocode

```
TURING(x)

y = HALT(x)
if y = 0 then
        halt
else
        loop forever
end if
```

Since HALT is a valid program, TURING must also be a valid program, with some Turing number, t. By definition of the halting function, $h(t) = 1$ if and only if TURING halts on input of t. But by inspection of the program for TURING, we see that TURING halts on input of t if and only if $h(t) = 0$. Thus $h(t) = 1$ if and only if $h(t) = 0$, a contradiction. Therefore, our initial assumption that there is an algorithm to evaluate $h(x)$ must have been wrong. We conclude that there is no algorithm allowing us to solve the halting problem.

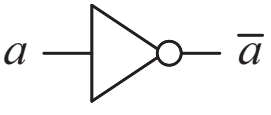


Figure 3.2. Elementary circuit performing a single NOT gate on a single input bit.

There are many other elementary logic gates which are useful for computation. A partial list includes the AND gate, the OR gate, the XOR gate, the NAND gate, and the NOR gate. Each of these gates takes two bits as input, and produces a single bit as output. The AND gate outputs 1 if and only if both of its inputs are 1. The OR gate outputs 1 if

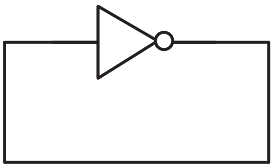


Figure 3.3. Circuits containing cycles can be unstable, and are not usually permitted in the circuit model of computation.

and only if at least one of its inputs is 1. The XOR gate outputs the sum, modulo 2, of its inputs. The NAND and NOR gates take the AND and OR, respectively, of their inputs, and then apply a NOT to whatever is output. The action of these gates is illustrated in Figure 3.4.

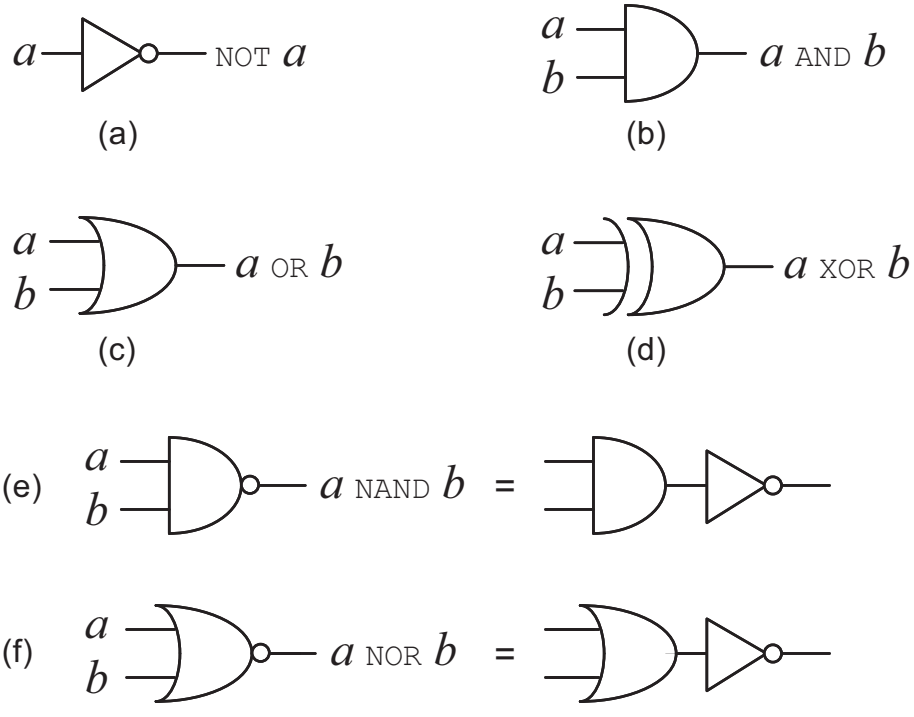


Figure 3.4. Elementary circuits performing the AND, OR, XOR, NAND, and NOR gates.

There are two important 'gates' missing from Figure 3.4, namely the FANOUT gate and the CROSSOVER gate. In circuits we often allow bits to 'divide', replacing a bit with two copies of itself, an operation referred to as FANOUT. We also allow bits to CROSSOVER, that is, the value of two bits are interchanged. A third operation missing from Figure 3.4, not really a logic gate at all, is to allow the preparation of extra *ancilla* or *work* bits, to allow extra working space during the computation.

These simple circuit elements can be put together to perform an enormous variety of computations. Below we'll show that these elements can be used to compute any function whatsoever. In the meantime, let's look at a simple example of a circuit which adds two n bit integers, using essentially the same algorithm taught to school-children around the

world. The basic element in this circuit is a smaller circuit known as a *half-adder*, shown in Figure 3.5. A half-adder takes two bits, x and y, as input, and outputs the sum of the bits $x \oplus y$ modulo 2, together with a carry bit set to 1 if x and y are both 1, or 0 otherwise.

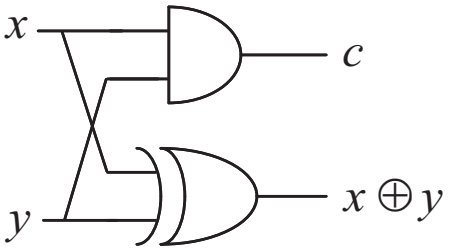


Figure 3.5. Half-adder circuit. The carry bit c is set to 1 when x and y are both 1, otherwise it is 0.

Two cascaded half-adders may be used to build a *full-adder*, as shown in Figure 3.6. A full-adder takes as input three bits, x, y, and c. The bits x and y should be thought of as data to be added, while c is a carry bit from an earlier computation. The circuit outputs two bits. One output bit is the modulo 2 sum, $x \oplus y \oplus c$ of all three input bits. The second output bit, c', is a carry bit, which is set to 1 if two or more of the inputs is 1, and is 0 otherwise.

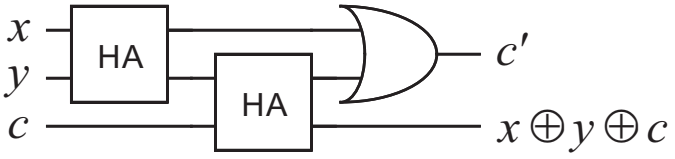


Figure 3.6. Full-adder circuit.

By cascading many of these full-adders together we obtain a circuit to add two n-bit integers, as illustrated in Figure 3.7 for the case $n = 3$.

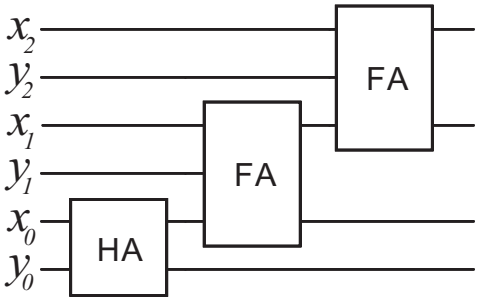


Figure 3.7. Addition circuit for two three-bit integers, $x = x_2x_1x_0$ and $y = y_2y_1y_0$, using the elementary algorithm taught to school-children.

We claimed earlier that just a few *fixed* gates can be used to compute *any* function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ whatsoever. We will now prove this for the simplified case of a function $f : \{0,1\}^n \rightarrow \{0,1\}$ with n input bits and a single output bit. Such a function

is known as a *Boolean function*, and the corresponding circuit is a *Boolean circuit*. The general universality proof follows immediately from the special case of Boolean functions. The proof is by induction on n. For $n = 1$ there are four possible functions: the identity, which has a circuit consisting of a single wire; the bit flip, which is implemented using a single NOT gate; the function which replaces the input bit with a 0, which can be obtained by ANDing the input with a work bit initially in the 0 state; and the function which replaces the input with a 1, which can be obtained by ORing the input with a work bit initially in the 1 state.

To complete the induction, suppose that any function on n bits may be computed by a circuit, and let f be a function on $n + 1$ bits. Define n-bit functions f_0 and f_1 by $f_0(x_1, \ldots, x_n) \equiv f(0, x_1, \ldots, x_n)$ and $f_1(x_1, \ldots, x_n) \equiv f(1, x_1, \ldots, x_n)$. These are both n-bit functions, so by the inductive hypothesis there are circuits to compute these functions.

It is now an easy matter to design a circuit which computes f. The circuit computes both f_0 and f_1 on the last n bits of the input. Then, depending on whether the first bit of the input was a 0 or a 1 it outputs the appropriate answer. A circuit to do this is shown in Figure 3.8. This completes the induction.

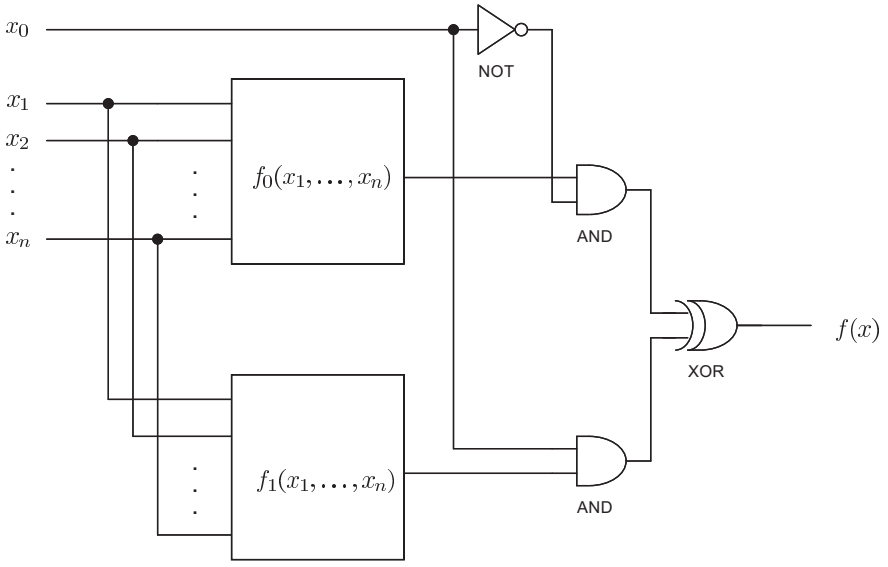


Figure 3.8. Circuit to compute an arbitrary function f on $n + 1$ bits, assuming by induction that there are circuits to compute the n-bit functions f_0 and f_1.

Five elements may be identified in the universal circuit construction: (1) wires, which preserve the states of the bits; (2) ancilla bits prepared in standard states, used in the $n = 1$ case of the proof; (3) the FANOUT operation, which takes a single bit as input and outputs two copies of that bit; (4) the CROSSOVER operation, which interchanges the value of two bits; and (5) the AND, XOR, and NOT gates. In Chapter 4 we'll define the quantum circuit model of computation in a manner analogous to classical circuits. It is interesting to note that many of these five elements pose some interesting challenges when extending to the quantum case: it is not necessarily obvious that good quantum wires for the preservation of qubits can be constructed, even in principle, the FANOUT

operation cannot be performed in a straightforward manner in quantum mechanics, due to the no-cloning theorem (as explained in Section 1.3.5), and the AND and XOR gates are not invertible, and thus can't be implemented in a straightforward manner as unitary quantum gates. There is certainly plenty to think about in defining a quantum circuit model of computation!

Exercise 3.8: (Universality of NAND) Show that the NAND gate can be used to simulate the AND, XOR and NOT gates, provided wires, ancilla bits and FANOUT are available.

Let's return from our brief quantum digression, to the properties of classical circuits. We claimed earlier that the Turing machine model is equivalent to the circuit model of computation. In what sense do we mean the two models are equivalent? On the face of it, the two models appear quite different. The unbounded nature of a Turing machine makes them more useful for abstractly specifying what it is we mean by an algorithm, while circuits more closely capture what an actual physical computer does.

The two models are connected by introducing the notion of a *uniform circuit family*. A *circuit family* consists of a collection of circuits, $\{C_n\}$, indexed by a positive integer n. The circuit C_n has n input bits, and may have any finite number of extra work bits, and output bits. The output of the circuit C_n, upon input of a number x of at most n bits in length, is denoted by $C_n(x)$. We require that the circuits be *consistent*, that is, if $m < n$ and x is at most m bits in length, then $C_m(x) = C_n(x)$. The function computed by the circuit family $\{C_n\}$ is the function $C(\cdot)$ such that if x is n bits in length then $C(x) = C_n(x)$. For example, consider a circuit C_n that squares an n-bit number. This defines a family of circuits $\{C_n\}$ that computes the function, $C(x) = x^2$, where x is any positive integer.

It's not enough to consider unrestricted families of circuits, however. In practice, we need an algorithm to build the circuit. Indeed, if we don't place any restrictions on the circuit family then it becomes possible to compute all sorts of functions which we do not expect to be able to compute in a reasonable model of computation. For example, let $h_n(x)$ denote the halting function, restricted to values of x which are n bits in length. Thus h_n is a function from n bits to 1 bit, and we have proved there exists a circuit C_n to compute $h_n(\cdot)$. Therefore the circuit family $\{C_n\}$ computes the halting function! However, what prevents us from using this circuit family to solve the halting problem is that we haven't specified an algorithm which will allow us to build the circuit C_n for all values of n. Adding this requirement results in the notion of a uniform circuit family.

That is, a family of circuits $\{C_n\}$ is said to be a *uniform circuit family* if there is some algorithm running on a Turing machine which, upon input of n, generates a *description* of C_n. That is, the algorithm outputs a description of what gates are in the circuit C_n, how those gates are connected together to form a circuit, any ancilla bits needed by the circuit, FANOUT and CROSSOVER operations, and where the output from the circuit should be read out. For example, the family of circuits we described earlier for squaring n-bit numbers is certainly a uniform circuit family, since there is an algorithm which, given n, outputs a description of the circuit needed to square an n-bit number. You can think of this algorithm as the means by which an engineer is able to generate a description of (and thus build) the circuit for any n whatsoever. By contrast, a circuit family that is not uniform is said to be a *non-uniform* circuit family. There is no algorithm to construct

the circuit for arbitrary n, which prevents our engineer from building circuits to compute functions like the halting function.

Intuitively, a uniform circuit family is a family of circuits that can be generated by some reasonable algorithm. It can be shown that the class of functions computable by uniform circuit families is exactly the same as the class of functions which can be computed on a Turing machine. With this uniformity restriction, results in the Turing machine model of computation can usually be given a straightforward translation into the circuit model of computation, and vice versa. Later we give similar attention to issues of uniformity in the quantum circuit model of computation.

3.2 The analysis of computational problems

The analysis of computational problems depends upon the answer to three fundamental questions:

(1) **What is a computational problem?** Multiplying two numbers together is a computational problem; so is programming a computer to exceed human abilities in the writing of poetry. In order to make progress developing a general theory for the analysis of computational problems we are going to isolate a special class of problems known as *decision problems*, and concentrate our analysis on those. Restricting ourselves in this way enables the development of a theory which is both elegant and rich in structure. Most important, it is a theory whose principles have application far beyond decision problems.

(2) **How may we design algorithms to solve a given computational problem?** Once a problem has been specified, what algorithms can be used to solve the problem? Are there general techniques which can be used to solve wide classes of problems? How can we be sure an algorithm behaves as claimed?

(3) **What are the minimal resources required to solve a given computational problem?** Running an algorithm requires the consumption of *resources*, such as time, space, and energy. In different situations it may be desirable to minimize consumption of one or more resource. Can we classify problems according to the resource requirements needed to solve them?

In the next few sections we investigate these three questions, especially questions 1 and 3. Although question 1, 'what is a computational problem?', is perhaps the most fundamental of the questions, we shall defer answering it until Section 3.2.3, pausing first to establish some background notions related to resource quantification in Section 3.2.1, and then reviewing the key ideas of *computational complexity* in Section 3.2.2.

Question 2, how to design good algorithms, is the subject of an enormous amount of ingenious work by many researchers. So much so that in this brief introduction we cannot even begin to describe the main ideas employed in the design of good algorithms. If you are interested in this beautiful subject, we refer you to the end of chapter 'History and further reading'. Our closest direct contact with this subject will occur later in the book, when we study quantum algorithms. The techniques involved in the creation of quantum algorithms have typically involved a blend of deep existing ideas in algorithm design for classical computers, and the creation of new, wholly quantum mechanical techniques for algorithm design. For this reason, and because the spirit of quantum algorithm design

is so similar in many ways to classical algorithm design, we encourage you to become familiar with at least the basic ideas of algorithm design.

Question 3, what are the minimal resources required to solve a given computational problem, is the main focus of the next few sections. For example, suppose we are given two numbers, each n bits in length, which we wish to multiply. If the multiplication is performed on a single-tape Turing machine, how many computational steps must be executed by the Turing machine in order to complete the task? How much space is used on the Turing machine while completing the task?

These are examples of the type of resource questions we may ask. Generally speaking, computers make use of many different kinds of resources, however we will focus most of our attention on time, space, and energy. Traditionally in computer science, time and space have been the two major resource concerns in the study of algorithms, and we study these issues in Sections 3.2.2 through 3.2.4. Energy has been a less important consideration; however, the study of energy requirements motivates the subject of reversible classical computation, which in turn is a prerequisite for quantum computation, so we examine energy requirements for computation in some considerable detail in Section 3.2.5.

3.2.1 How to quantify computational resources

Different models of computation lead to different resource requirements for computation. Even something as simple as changing from a single-tape to a two-tape Turing machine may change the resources required to solve a given computational problem. For a computational task which is extremely well understood, like addition of integers, for example, such differences between computational models may be of interest. However, for a first pass at understanding a problem, we would like a way of quantifying resource requirements that is independent of relatively trivial changes in the computational model. One of the tools which has been developed to do this is the *asymptotic notation*, which can be used to summarize the *essential* behavior of a function. This asymptotic notation can be used, for example, to summarize the essence of how many time steps it takes a given algorithm to run, without worrying too much about the exact time count. In this section we describe this notation in detail, and apply it to a simple problem illustrating the quantification of computational resources – the analysis of algorithms for sorting a list of names into alphabetical order.

Suppose, for example, that we are interested in the number of gates necessary to add together two n-bit numbers. Exact counts of the number of gates required obscure the big picture: perhaps a specific algorithm requires $24n + 2\lceil \log n \rceil + 16$ gates to perform this task. However, in the limit of large problem size the only term which matters is the $24n$ term. Furthermore, we disregard constant factors as being of secondary importance to the analysis of the algorithm. The *essential* behavior of the algorithm is summed up by saying that the number of operations required scales like n, where n is the number of bits in the numbers being added. The asymptotic notation consists of three tools which make this notion precise.

The O ('big O') notation is used to set *upper bounds* on the behavior of a function. Suppose $f(n)$ and $g(n)$ are two functions on the non-negative integers. We say '$f(n)$ is in the class of functions $O(g(n))$', or just '$f(n)$ is $O(g(n))$', if there are constants c and n_0 such that for all values of n greater than n_0, $f(n) \leq cg(n)$. That is, for sufficiently large n, the function $g(n)$ is an upper bound on $f(n)$, up to an unimportant constant

factor. The big O notation is particularly useful for studying the worst-case behavior of *specific* algorithms, where we are often satisfied with an upper bound on the resources consumed by an algorithm.

When studying the behaviors of a *class* of algorithms – say the entire class of algorithms which can be used to multiply two numbers – it is interesting to set lower bounds on the resources required. For this the Ω ('big Omega') notation is used. A function $f(n)$ is said to be $\Omega(g(n))$ if there exist constants c and n_0 such that for all n greater than n_0, $cg(n) \leq f(n)$. That is, for sufficiently large n, $g(n)$ is a lower bound on $f(n)$, up to an unimportant constant factor.

Finally, the Θ ('big Theta') notation is used to indicate that $f(n)$ behaves the same as $g(n)$ asymptotically, up to unimportant constant factors. That is, we say $f(n)$ is $\Theta(g(n))$ if it is both $O(g(n))$ and $\Omega(g(n))$.

Asymptotic notation: examples

Let's consider a few simple examples of the asymptotic notation. The function $2n$ is in the class $O(n^2)$, since $2n \leq 2n^2$ for all positive n. The function 2^n is $\Omega(n^3)$, since $n^3 \leq 2^n$ for sufficiently large n. Finally, the function $7n^2 + \sqrt{n}\log(n)$ is $\Theta(n^2)$, since $7n^2 \leq 7n^2 + \sqrt{n}\log(n) \leq 8n^2$ for all sufficiently large values of n. In the following few exercises you will work through some of the elementary properties of the asymptotic notation that make it a useful tool in the analysis of algorithms.

Exercise 3.9: Prove that $f(n)$ is $O(g(n))$ if and only if $g(n)$ is $\Omega(f(n))$. Deduce that $f(n)$ is $\Theta(g(n))$ if and only if $g(n)$ is $\Theta(f(n))$.

Exercise 3.10: Suppose $g(n)$ is a polynomial of degree k. Show that $g(n)$ is $O(n^l)$ for any $l \geq k$.

Exercise 3.11: Show that $\log n$ is $O(n^k)$ for any $k > 0$.

Exercise 3.12: ($n^{\log n}$ **is super-polynomial**) Show that n^k is $O(n^{\log n})$ for any k, but that $n^{\log n}$ is never $O(n^k)$.

Exercise 3.13: ($n^{\log n}$ **is sub-exponential**) Show that c^n is $\Omega(n^{\log n})$ for any $c > 1$, but that $n^{\log n}$ is never $\Omega(c^n)$.

Exercise 3.14: Suppose $e(n)$ is $O(f(n))$ and $g(n)$ is $O(h(n))$. Show that $e(n)g(n)$ is $O(f(n)h(n))$.

An example of the use of the asymptotic notation in quantifying resources is the following simple application to the problem of sorting an n element list of names into alphabetical order. Many sorting algorithms are based upon the 'compare-and-swap' operation: two elements of an n element list are compared, and swapped if they are in the wrong order. If this compare-and-swap operation is the only means by which we can access the list, how many such operations are required in order to ensure that the list has been correctly sorted?

A simple compare-and-swap algorithm for solving the sorting problem is as follows: (note that `compare-and-swap(j,k)` compares the list entries numbered j and k, and swaps them if they are out of order)

```
for j = 1 to n-1
    for k = j+1 to n
        compare-and-swap(j,k)
    end k
end j
```

It is clear that this algorithm correctly sorts a list of n names into alphabetical order. Note that the number of compare-and-swap operations executed by the algorithm is $(n-1) + (n-2) + \cdots + 1 = n(n-1)/2$. Thus the number of compare-and-swap operations used by the algorithm is $\Theta(n^2)$. Can we do better than this? It turns out that we can. Algorithms such as 'heapsort' are known which run using $O(n \log n)$ compare-and-swap operations. Furthermore, in Exercise 3.15 you'll work through a simple counting argument that shows any algorithm based upon the compare-and-swap operation requires $\Omega(n \log n)$ such operations. Thus, the sorting problem requires $\Theta(n \log n)$ compare-and-swap operations, in general.

Exercise 3.15: (Lower bound for compare-and-swap based sorts) Suppose an n element list is sorted by applying some sequence of compare-and-swap operations to the list. There are $n!$ possible initial orderings of the list. Show that after k of the compare-and-swap operations have been applied, at most 2^k of the possible initial orderings will have been sorted into the correct order. Conclude that $\Omega(n \log n)$ compare-and-swap operations are required to sort all possible initial orderings into the correct order.

3.2.2 Computational complexity

The idea that there won't be an algorithm to solve it – this is something fundamental that won't ever change – that idea appeals to me.
– Stephen Cook

Sometimes it is good that some things are impossible. I am happy there are many things that nobody can do to me.
– Leonid Levin

It should not come as a surprise that our choice of polynomial algorithms as the mathematical concept that is supposed to capture the informal notion of 'practically efficient computation' is open to criticism from all sides. [...] Ultimately, our argument for our choice must be this: ***Adopting polynomial worst-case performance as our criterion of efficiency results in an elegant and useful theory that says something meaningful about practical computation, and would be impossible without this simplification.***
– Christos Papadimitriou

What time and space resources are required to perform a computation? In many cases these are the most important questions we can ask about a computational problem. Problems like addition and multiplication of numbers are regarded as efficiently solvable because we have *fast* algorithms to perform addition and multiplication, which consume

little *space* when running. Many other problems have no known fast algorithm, and are effectively impossible to solve, not because we can't find an algorithm to solve the problem, but because all known algorithms consume such vast quantities of space or time as to render them practically useless.

Computational complexity is the study of the time and space resources required to solve computational problems. The task of computational complexity is to prove *lower bounds* on the resources required by the best possible algorithm for solving a problem, even if that algorithm is not explicitly known. In this and the next two sections, we give an overview of computational complexity, its major concepts, and some of the more important results of the field. Note that computational complexity is in a sense complementary to the field of algorithm design; ideally, the most efficient algorithms we could design would match perfectly with the lower bounds proved by computational complexity. Unfortunately, this is often not the case. As already noted, in this book we won't examine classical algorithm design in any depth.

One difficulty in formulating a theory of computational complexity is that different computational models may require different resources to solve the same problem. For instance, multiple-tape Turing machines can solve many problems substantially faster than single-tape Turing machines. This difficulty is resolved in a rather coarse way. Suppose a problem is specified by giving n bits as input. For instance, we might be interested in whether a particular n-bit number is prime or not. The chief distinction made in computational complexity is between problems which can be solved using resources which are bounded by a *polynomial* in n, or which require resources which grow faster than any polynomial in n. In the latter case we usually say that the resources required are *exponential* in the problem size, abusing the term exponential, since there are functions like $n^{\log n}$ which grow faster than any polynomial (and thus are 'exponential' according to this convention), yet which grow slower than any true exponential. A problem is regarded as *easy*, *tractable* or *feasible* if an algorithm for solving the problem using polynomial resources exists, and as *hard*, *intractable* or *infeasible* if the best possible algorithm requires exponential resources.

As a simple example, suppose we have two numbers with binary expansions $x_1 \ldots x_{m_1}$ and $y_1 \ldots y_{m_2}$, and we wish to determine the sum of the two numbers. The total size of the input is $n \equiv m_1 + m_2$. It's easy to see that the two numbers can be added using a number of elementary operations that scales as $\Theta(n)$; this algorithm uses a polynomial (indeed, linear) number of operations to perform its tasks. By contrast, it is believed (though it has never been proved!) that the problem of factoring an integer into its prime factors is intractable. That is, the belief is that there is no algorithm which can factor an arbitrary n-bit integer using $O(p(n))$ operations, where p is some fixed polynomial function of n. We will later give many other examples of problems which are believed to be intractable in this sense.

The polynomial versus exponential classification is rather coarse. In practice, an algorithm that solves a problem using $2^{n/1000}$ operations is probably more useful than one which runs in n^{1000} operations. Only for very large input sizes ($n \approx 10^8$) will the 'efficient' polynomial algorithm be preferable to the 'inefficient' exponential algorithm, and for many purposes it may be more practical to prefer the 'inefficient' algorithm.

Nevertheless, there are many reasons to base computational complexity primarily on the polynomial versus exponential classification. First, historically, with few exceptions, polynomial resource algorithms have been much faster than exponential algorithms. We

might speculate that the reason for this is lack of imagination: coming up with algorithms requiring n, n^2 or some other low degree polynomial number of operations is often much easier than finding a natural algorithm which requires n^{1000} operations, although examples like the latter do exist. Thus, the predisposition for the human mind to come up with relatively simple algorithms has meant that in practice polynomial algorithms usually do perform much more efficiently than their exponential cousins.

A second and more fundamental reason for emphasizing the polynomial versus exponential classification is derived from the *strong Church–Turing thesis*. As discussed in Section 1.1, it was observed in the 1960s and 1970s that probabilistic Turing machines appear to be the strongest 'reasonable' model of computation. More precisely, researchers consistently found that if it was possible to compute a function using k elementary operations in some model that was *not* the probabilistic Turing machine model of computation, then it was always possible to compute the same function in the probabilistic Turing machine model, using at most $p(k)$ elementary operations, where $p(\cdot)$ is some *polynomial* function. This statement is known as the *strong Church–Turing thesis*:

> **Strong Church–Turing thesis**: *Any model of computation can be simulated on a probabilistic Turing machine with at most a polynomial increase in the number of elementary operations required.*

The strong Church–Turing thesis is great news for the theory of computational complexity, for it implies that attention may be restricted to the probabilistic Turing machine model of computation. After all, if a problem has no polynomial resource solution on a probabilistic Turing machine, then the strong Church–Turing thesis implies that it has no efficient solution on any computing device. Thus, the strong Church–Turing thesis implies that the entire theory of computational complexity will take on an elegant, model-independent form if the notion of efficiency is identified with polynomial resource algorithms, and this elegance has provided a strong impetus towards acceptance of the identification of 'solvable with polynomial resources' and 'efficiently solvable'. Of course, one of the prime reasons for interest in quantum computers is that they cast into doubt the strong Church–Turing thesis, by enabling the efficient solution of a problem which is believed to be intractable on all classical computers, including probabilistic Turing machines! Nevertheless, it is useful to understand and appreciate the role the strong Church–Turing thesis has played in the search for a model-independent theory of computational complexity.

Finally, we note that, in practice, computer scientists are not only interested in the polynomial versus exponential classification of problems. This is merely the first and coarsest way of understanding how difficult a computational problem is. However, it is an exceptionally important distinction, and illustrates many broader points about the nature of resource questions in computer science. For most of this book, it will be our central concern in evaluating the efficiency of a given algorithm.

Having examined the merits of the polynomial versus exponential classification, we now have to confess that the theory of computational complexity has one remarkable outstanding failure: it seems very hard to prove that there are interesting classes of problems which require exponential resources to solve. It is quite easy to give non-constructive proofs that *most* problems require exponential resources (see Exercise 3.16, below), and furthermore many interesting problems are *conjectured* to require exponential resources for their solution, but rigorous proofs seem very hard to come by, at least with the present

state of knowledge. This failure of computational complexity has important implications for quantum computation, because it turns out that the computational power of quantum computers can be related to some major open problems in *classical* computational complexity theory. Until these problems are resolved, it cannot be stated with certainty how computationally powerful a quantum computer is, or even whether it is more powerful than a classical computer!

Exercise 3.16: (Hard-to-compute functions exist) Show that there exist Boolean functions on n inputs which require at least $2^n/\log n$ logic gates to compute.

3.2.3 Decision problems and the complexity classes P and NP

Many computational problems are most cleanly formulated as *decision problems* – problems with a yes or no answer. For example, is a given number m a prime number or not? This is the *primality* decision problem. The main ideas of computational complexity are most easily and most often formulated in terms of decision problems, for two reasons: the theory takes its simplest and most elegant form in this form, while still generalizing in a natural way to more complex scenarios; and historically computational complexity arose primarily from the study of decision problems.

Although most decision problems can easily be stated in simple, familiar language, discussion of the general properties of decision problems is greatly helped by the terminology of *formal languages*. In this terminology, a *language* L over the *alphabet* Σ is a subset of the set Σ^* of all (finite) strings of symbols from Σ. For example, if $\Sigma = \{0, 1\}$, then the set of binary representations of even numbers, $L = \{0, 10, 100, 110, \ldots\}$ is a language over Σ.

Decision problems may be encoded in an obvious way as problems about languages. For instance, the primality decision problem can be encoded using the binary alphabet $\Sigma = \{0, 1\}$. Strings from Σ^* can be interpreted in a natural way as non-negative integers. For example, 0010 can be interpreted as the number 2. The language L is defined to consist of all binary strings such that the corresponding number is prime.

To solve the primality decision problem, what we would like is a Turing machine which, when started with a given number n on its input tape, eventually outputs some equivalent of 'yes' if n is prime, and outputs 'no' if n is not prime. To make this idea precise, it is convenient to modify our old Turing machine definition (of Section 3.1.1) slightly, replacing the halting state $q_{\rm h}$ with two states $q_{\rm Y}$ and $q_{\rm N}$ to represent the answers 'yes' and 'no' respectively. In all other ways the machine behaves in the same way, and it still halts when it enters the state $q_{\rm Y}$ or $q_{\rm N}$. More generally, a language L is *decided* by a Turing machine if the machine is able to decide whether an input x on its tape is a member of the language of L or not, eventually halting in the state $q_{\rm Y}$ if $x \in L$, and eventually halting in the state $q_{\rm N}$ if $x \notin L$. We say that the machine has *accepted* or *rejected* x depending on which of these two cases comes about.

How quickly can we determine whether or not a number is prime? That is, what is the fastest Turing machine which decides the language representing the primality decision problem? We say that a problem is in **TIME**$(f(n))$ if there exists a Turing machine which decides whether a candidate x is in the language in time $O(f(n))$, where n is the length of x. A problem is said to be solvable in *polynomial time* if it is in **TIME**(n^k) for some finite k. The collection of all languages which are in **TIME**(n^k), for some k, is denoted **P**. **P** is our first example of a *complexity class*. More generally, a complexity

class is defined to be a collection of languages. Much of computational complexity theory is concerned with the definition of various complexity classes, and understanding the relationship between different complexity classes.

Not surprisingly, there are problems which cannot be solved in polynomial time. Unfortunately, proving that any given problem can't be solved in polynomial time seems to be very difficult, although conjectures abound! A simple example of an interesting decision problem which is believed not to be in **P** is the *factoring decision problem*:

> FACTORING: Given a composite integer m and $l < m$, does m have a non-trivial factor less than l?

An interesting property of factoring is that if somebody claims that the answer is 'yes, m does have a non-trivial factor less than l' then they can establish this by exhibiting such a factor, which can then be efficiently checked by other parties, simply by doing long-division. We call such a factor a *witness* to the fact that m has a factor less than l. This idea of an easily checkable witness is the key idea in the definition of the complexity class **NP**, below. We have phrased factoring as a decision problem, but you can easily verify that the decision problem is equivalent to finding the factors of a number:

Exercise 3.17: Prove that a polynomial-time algorithm for finding the factors of a number m exists if and only if the factoring decision problem is in **P**.

Factoring is an example of a problem in an important complexity class known as **NP**. What distinguishes problems in **NP** is that 'yes' instances of a problem can easily be verified with the aid of an appropriate witness. More rigorously, a language L is in **NP** if there is a Turing machine M with the following properties:

(1) If $x \in L$ then there exists a witness string w such that M halts in the state $q_{\rm Y}$ after a time polynomial in $|x|$ when the machine is started in the state x-blank-w.
(2) If $x \notin L$ then for all strings w which attempt to play the role of a witness, the machine halts in state $q_{\rm N}$ after a time polynomial in $|x|$ when M is started in the state x-blank-w.

There is an interesting asymmetry in the definition of **NP**. While we have to be able to quickly decide whether a possible witness to $x \in L$ is truly a witness, there is no such need to produce a witness to $x \notin L$. For instance, in the factoring problem, we have an easy way of proving that a given number has a factor less than m, but exhibiting a witness to prove that a number has no factors less than m is more daunting. This suggests defining **coNP**, the class of languages which have witnesses to 'no' instances; obviously the languages in **coNP** are just the complements of languages in **NP**.

How are **P** and **NP** related? It is clear that **P** is a subset of **NP**. The most famous open problem in computer science is *whether or not there are problems in* **NP** *which are not in* **P**, often abbreviated as the **P** $\neq$ **NP** problem. Most computer scientists believe that **P** $\neq$ **NP**, but despite decades of work nobody has been able to prove this, and the possibility remains that **P** = **NP**.

Exercise 3.18: Prove that if **coNP** $\neq$ **NP** then **P** $\neq$ **NP**.

Upon first acquaintance it's tempting to conclude that the conjecture **P** $\neq$ **NP** ought to be pretty easy to resolve. To see why it's actually rather subtle it helps to see couple of

examples of problems that are in **P** and **NP**. We'll draw the examples from *graph theory*, a rich source of decision problems with surprisingly many practical applications. A *graph* is a finite collection of *vertices* $\{v_1, \ldots, v_n\}$ connected by *edges*, which are pairs (v_i, v_j) of vertices. For now, we are only concerned with *undirected graphs*, in which the order of the vertices (in each edge pair) does not matter; similar ideas can be investigated for *directed graphs* in which the order of vertices does matter. A typical graph is illustrated in Figure 3.9.

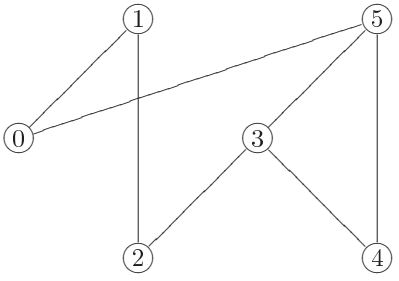


Figure 3.9. A graph.

A *cycle* in a graph is a sequence $v_1, \ldots, v_m$ of vertices such that each pair (v_j, v_{j+1}) is an edge, as is (v_1, v_m). A *simple cycle* is a cycle in which none of the vertices is repeated, except for the first and last vertices. A *Hamiltonian cycle* is a simple cycle which visits every vertex in the graph. Examples of graphs with and without Hamiltonian cycles are shown in Figure 3.10.

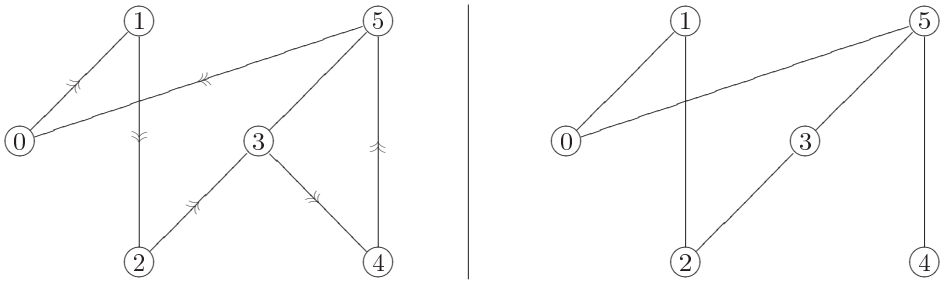


Figure 3.10. The graph on the left contains a Hamiltonian cycle, $0, 1, 2, 3, 4, 5, 0$. The graph on the right contains no Hamiltonian cycle, as can be verified by inspection.

The *Hamiltonian cycle problem* (HC) is to determine whether a given graph contains a Hamiltonian cycle or not. HC is a decision problem in **NP**, since if a given graph has a Hamiltonian cycle, then that cycle can be used as an easily checkable witness. Moreover, HC has no known polynomial time algorithm. Indeed, HC is a problem in the class of so-called **NP**-complete problems, which can be thought of as the 'hardest' problems in **NP**, in the sense that solving HC in time t allows any other problem in **NP** to be solved in time $O(\text{poly}(t))$. This also means that if any **NP**-complete problem has a polynomial time solution then it will follow that **P** = **NP**.

There is a problem, the Euler cycle decision problem, which is superficially similar to HC, but which has astonishingly different properties. An *Euler cycle* is an ordering of the edges of a graph G so that every edge in the graph is visited exactly once. The Euler

cycle decision problem (EC) is to determine, given a graph G on n vertices, whether that graph contains an Euler cycle or not. EC is, in fact, exactly the same problem as HC, only the path visits edges, rather than vertices. Consider the following remarkable theorem, to be proven in Exercise 3.20:

Theorem 3.1: (**Euler's theorem**) A connected graph contains an Euler cycle if and only if every vertex has an even number of edges incident upon it.

Euler's theorem gives us a method for efficiently solving EC. First, check to see whether the graph is connected; this is easily done with $O(n^2)$ operations, as shown in Exercise 3.19. If the graph is not connected, then obviously no Euler cycle exists. If the graph is connected then for each vertex check whether there is an even number of edges incident upon the vertex. If a vertex is found for which this is not the case, then there is no Euler cycle, otherwise an Euler cycle exists. Since there are n vertices, and at most $n(n-1)/2$ edges, this algorithm requires $O(n^3)$ elementary operations. Thus EC is in **P**! Somehow, there is a structure present in the problem of visiting each edge that can be exploited to provide an efficient algorithm for EC, yet which does not seem to be reflected in the problem of visiting each vertex; it is not at all obvious why such a structure should be present in one case, but not in the other, if indeed it is absent for the HC problem.

Exercise 3.19: The REACHABILITY problem is to determine whether there is a path between two specified vertices in a graph. Show that REACHABILITY can be solved using $O(n)$ operations if the graph has n vertices. Use the solution to REACHABILITY to show that it is possible to decide whether a graph is connected in $O(n^2)$ operations.

Exercise 3.20: (**Euler's theorem**) Prove Euler's theorem. In particular, if each vertex has an even number of incident edges, give a constructive procedure for finding an Euler cycle.

The equivalence between the factoring decision problem and the factoring problem proper is a special instance of one of the most important ideas in computer science, an idea known as *reduction*. Intuitively, we know that some problems can be viewed as special instances of other problems. A less trivial example of reduction is the reduction of HC to the *traveling salesman* decision problem (TSP). The traveling salesman decision problem is as follows: we are given n cities $1, 2, \ldots, n$ and a non-negative integer distance d_{ij} between each pair of cities. Given a distance d the problem is to determine if there is a tour of all the cities of distance less than d.

The reduction of HC to TSP goes as follows. Suppose we have a graph containing n vertices. We turn this into an instance of TSP by thinking of each vertex of the graph as a 'city' and defining the distance d_{ij} between cities i and j to be one if vertices i and j are connected, and the distance to be two if the vertices are unconnected. Then a tour of the cities of distance less than $n+1$ must be of distance n, and be a Hamiltonian cycle for the graph. Conversely, if a Hamiltonian cycle exists then a tour of the cities of distance less than $n+1$ must exist. In this way, given an algorithm for solving TSP, we can convert it into an algorithm for solving HC without much overhead. Two consequences can be inferred from this. First, if TSP is a tractable problem, then HC is also tractable. Second, if HC is hard then TSP must also be hard. This is an example of a general technique known

as *reduction*: we've reduced the problem HC to the problem TSP. This is a technique we will use repeatedly throughout this book.

A more general notion of reduction is illustrated in Figure 3.11. A language B is said to be *reducible* to another language A if there exists a Turing machine operating in polynomial time such that given as input x it outputs $R(x)$, and $x \in B$ if and only if $R(x) \in A$. Thus, if we have an algorithm for deciding A, then with a little extra overhead we can decide the language B. In this sense, the language B is essentially no more difficult to decide than the language A.

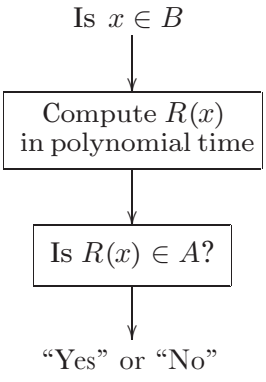


Figure 3.11. Reduction of B to A.

Exercise 3.21: (Transitive property of reduction) Show that if a language L_1 is reducible to the language L_2 and the language L_2 is reducible to L_3 then the language L_1 is reducible to the language L_3.

Some complexity classes have problems which are *complete* with respect to that complexity class, meaning there is a language L in the complexity class which is the 'most difficult' to decide, in the sense that every other language in the complexity class can be reduced to L. Not all complexity classes have complete problems, but many of the complexity classes we are concerned with do have complete problems. A trivial example is provided by **P**. Let L be any language in **P** which is not empty or equal to the set of all words. That is, there exists a string x_1 such that $x_1 \notin L$ and a string x_2 such that $x_2 \in L$. Then any other language L' in **P** can be reduced to L using the following reduction: given an input x, use the polynomial time decision procedure to determine whether $x \in L'$ or not. If it is not, then set $R(x) = x_1$, otherwise set $R(x) = x_2$.

Exercise 3.22: Suppose L is complete for a complexity class, and L' is another language in the complexity class such that L reduces to L'. Show that L' is complete for the complexity class.

Less trivially, **NP** also contains complete problems. An important example of such a problem and the prototype for all other **NP**-complete problems is the *circuit satisfiability* problem or CSAT: given a Boolean circuit composed of AND, OR and NOT gates, is there an assignment of values to the inputs to the circuit that results in the circuit outputting 1, that is, is the circuit *satisfiable* for some input? The **NP**-completeness of CSAT is known as the *Cook–Levin theorem*, for which we now outline a proof.

Theorem 3.2: **(Cook–Levin)** CSAT is **NP**-complete.

Proof

The proof has two parts. The first part of the proof is to show that CSAT is in **NP**, and the second part is to show that any language in **NP** can be reduced to CSAT. Both parts of the proof are based on *simulation* techniques: the first part of the proof is essentially showing that a Turing machine can efficiently simulate a circuit, while the second part of the proof is essentially showing that a circuit can efficiently simulate a Turing machine. Both parts of the proof are quite straightforward; for the purposes of illustration we give the second part in some detail.

The first part of the proof is to show that CSAT is in **NP**. Given a circuit containing n circuit elements, and a potential witness w, it is obviously easy to check in polynomial time on a Turing machine whether or not w satisfies the circuit, which establishes that CSAT is in **NP**.

The second part of the proof is to show that any language $L \in$ **NP** can be reduced to CSAT. That is, we aim to show that there is a polynomial time computable reduction R such that $x \in L$ if and only if $R(x)$ is a satisfiable circuit. The idea of the reduction is to find a circuit which simulates the action of the machine M which is used to check instance-witness pairs, (x, w), for the language L. The input variables for the circuit will represent the witness; the idea is that finding a witness which satisfies the circuit is equivalent to M accepting (x, w) for some specific witness w. Without loss of generality we may make the following assumptions about M to simplify the construction:

(1) M's tape alphabet is $\triangleright$,0,1 and the blank symbol.
(2) M runs using time at most $t(n)$ and total space at most $s(n)$ where $t(n)$ and $s(n)$ are polynomials in n.
(3) Machine M can actually be assumed to run using time *exactly* $t(n)$ for all inputs of size n. This is done by adding the lines $\langle q_{\rm Y}, x, q_{\rm Y}, x, 0\rangle$, and $\langle q_{\rm N}, x, q_{\rm N}, x, 0\rangle$ for each of $x = \triangleright, 0, 1$ and the blank, artificially halting the machine after exactly $t(n)$ steps.

The basic idea of the construction to simulate M is outlined in Figure 3.12. Each internal state of the Turing machine is represented by a single bit in the circuit. We name the corresponding bits $\tilde{q}_{\rm s}, \tilde{q}_1, \ldots, \tilde{q}_m, \tilde{q}_{\rm Y}, \tilde{q}_{\rm N}$. Initially, $\tilde{q}_{\rm s}$ is set to one, and all the other bits representing internal states are set to zero. Each square on the Turing machine tape is represented by three bits: two bits to represent the letter of the alphabet ($\triangleright, 0, 1$ or blank) currently residing on the tape, and a single 'flag' bit which is set to one if the read-write head is pointing to the square, and set to zero otherwise. We denote the bits representing the tape contents by $(u_1, v_1), \ldots, (u_{s(n)}, v_{s(n)})$ and the corresponding flag bits by $f_1, \ldots, f_{s(n)}$. Initially the u_j and v_j bits are set to represent the inputs x and w, as appropriate, while $f_1 = 1$ and all other $f_j = 0$. There is also a lone extra 'global flag' bit, F, whose function will be explained later. F is initially set to zero. We regard all the bits input to the circuit as fixed, except for those representing the witness w, which are the variable bits for the circuit.

The action of M is obtained by repeating $t(n)$ times a 'simulation step' which simulates the execution of a single program line for the Turing machine. Each simulation step may be broken up into a sequence of steps corresponding in turn to the respective program lines, with a final step which resets the global flag F to zero, as

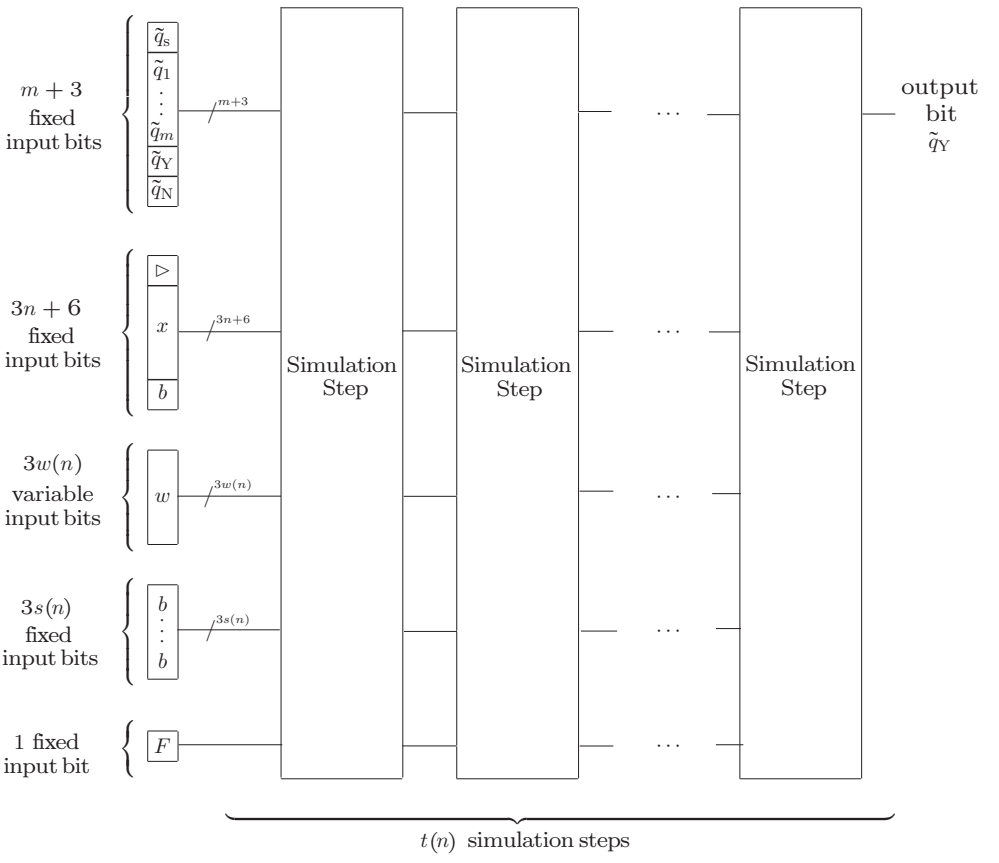


Figure 3.12. Outline of the procedure used to simulate a Turing machine using a circuit.

illustrated in Figure 3.13. To complete the simulation, we only need to simulate a program line of the form $\langle q_i, x, q_j, x', s\rangle$. For convenience, we assume $q_i \neq q_j$, but a similar construction works in the case when $q_i = q_j$. The procedure is as follows:

(1) Check to see that $\tilde{q}_i = 1$, indicating that the current state of the machine is q_i.
(2) For each tape square:
    (a) Check to see that the global flag bit is set to zero, indicating that no action has yet been taken by the Turing machine.
    (b) Check that the flag bit is set to one, indicating that the tape head is at this tape square.
    (c) Check that the simulated tape contents at this point are x.
    (d) If all conditions check out, then perform the following steps:
        1. Set $\tilde{q}_i = 0$ and $\tilde{q}_j = 1$.
        2. Update the simulated tape contents at this tape square to x'.
        3. Update the flag bit of this and adjacent 'squares' as appropriate, depending on whether $s = +1, 0, -1$, and whether we are at the left hand end of the tape.
        4. Set the global flag bit to one, indicating that this round of computation has been completed.

This is a fixed procedure which involves a constant number of bits, and by the universality result of Section 3.1.2 can be performed using a circuit containing a constant number of gates.

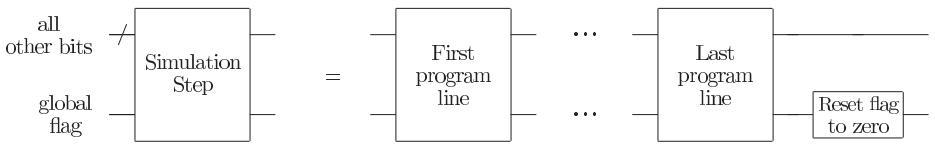


Figure 3.13. Outline of the simulation step used to simulate a Turing machine using a circuit.

The total number of gates in the entire circuit is easily seen to be $O(t(n)(s(n) + n))$, which is polynomial in size. At the end of the circuit, it is clear that $\tilde{q}_Y = 1$ if and only if the machine M accepts (x, w). Thus, the circuit is satisfiable if and only if there exists w such that machine M accepts (x, w), and we have found the desired reduction from L to CSAT. □

CSAT gives us a foot in the door which enables us to easily prove that many other problems are **NP**-complete. Instead of directly proving that a problem is **NP**-complete, we can instead prove that it is in **NP** and that CSAT reduces to it, so by Exercise 3.22 the problem must be **NP**-complete. A small sample of **NP**-complete problems is discussed in Box 3.3. An example of another **NP**-complete problem is the *satisfiability* problem (SAT), which is phrased in terms of a Boolean formula. Recall that a Boolean formula φ is composed of the following elements: a set of Boolean variables, $x_1, x_2, \ldots$; Boolean connectives, that is, a Boolean function with one or two inputs and one output, such as $\wedge$ (AND), $\vee$ (OR), and $\neg$ (NOT); and parentheses. The truth or falsity of a Boolean formula for a given set of Boolean variables is decided according to the usual laws of Boolean algebra. For example, the formula $\varphi = x_1 \vee \neg x_2$ has the satisfying assignment $x_1 = 0$ and $x_2 = 0$, while $x_1 = 0$ and $x_2 = 1$ is not a satisfying assignment. The satisfiability problem is to determine, given a Boolean formula φ, whether or not it is satisfiable by any set of possible inputs.

Exercise 3.23: Show that SAT is **NP**-complete by first showing that SAT is in **NP**, and then showing that CSAT reduces to SAT. (*Hint*: for the reduction it may help to represent each distinct wire in an instance of CSAT by different variables in a Boolean formula.)

An important restricted case of SAT is also **NP**-complete, the 3-satisfiability problem (3SAT), which is concerned with formulae in *3-conjunctive normal form*. A formula is said to be in *conjunctive normal form* if it is the AND of a collection of *clauses*, each of which is the OR of one or more *literals*, where a literal is an expression is of the form x or $\neg x$. For example, the formula $(x_1 \vee \neg x_2) \wedge (x_2 \vee x_3 \vee \neg x_4)$ is in conjunctive normal form. A formula is in *3-conjunctive normal form* or *3-CNF* if each clause has exactly three literals. For example, the formula $(\neg x_1 \vee x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (x_2 \vee x_3 \vee x_4)$ is in 3-conjunctive normal form. The 3-satisfiability problem is to determine whether a formula in 3-conjunctive normal form is satisfiable or not.

The proof that 3SAT is **NP**-complete is straightforward, but is a little too lengthy to justify inclusion in this overview. Even more than CSAT and SAT, 3SAT is in some sense

the **NP**-complete problem, and it is the basis for countless proofs that other problems are **NP**-complete. We conclude our discussion of **NP**-completeness with the surprising fact that 2SAT, the analogue of 3SAT in which every clause has two literals, can be solved in polynomial time:

Exercise 3.24: (2SAT **has an efficient solution**) Suppose φ is a Boolean formula in conjunctive normal form, in which each clause contains only two literals.

(1) Construct a (directed) graph $G(\varphi)$ with directed edges in the following way: the vertices of G correspond to variables x_j and their negations $\neg x_j$ in φ. There is a (directed) edge (α, β) in G if and only if the clause $(\neg\alpha \vee \beta)$ or the clause $(\beta \vee \neg\alpha)$ is present in φ. Show that φ is not satisfiable if and only if there exists a variable x such that there are paths from x to $\neg x$ and from $\neg x$ to x in $G(\varphi)$.

(2) Show that given a directed graph G containing n vertices it is possible to determine whether two vertices v_1 and v_2 are connected in polynomial time.

(3) Find an efficient algorithm to solve 2SAT.

Box 3.3: A zoo of NP-complete problems

The importance of the class **NP** derives, in part, from the enormous number of computational problems that are known to be **NP**-complete. We can't possibly hope to survey this topic here (see 'History and further reading'), but the following examples, taken from many distinct areas of mathematics, give an idea of the delicious melange of problems known to be **NP**-complete.

- CLIQUE (*graph theory*): A clique in an undirected graph G is a subset of vertices, each pair of which is connected by an edge. The *size* of a clique is the number of vertices it contains. Given an integer m and a graph G, does G have a clique of size m?
- SUBSET-SUM (*arithmetic*): Given a finite collection S of positive integers and a *target* t, is there any subset of S which sums to t?
- 0-1 INTEGER PROGRAMMING (*linear programming*): Given an integer $m \times n$ matrix A and an m-dimensional vector y with integer values, does there exist an n-dimensional vector x with entries in the set $\{0, 1\}$ such that $Ax \leq y$?
- VERTEX COVER (*graph theory*): A *vertex cover* for an undirected graph G is a set of vertices V' such that every edge in the graph has one or both vertices contained in V'. Given an integer m and a graph G, does G have a vertex cover V' containing m vertices?

Assuming that **P** $\neq$ **NP** it is possible to prove that there is a *non-empty* class of problems **NPI** (**NP** intermediate) which are neither solvable with polynomial resources, nor are **NP**-complete. Obviously, there are no problems known to be in **NPI** (otherwise we would know that **P** $\neq$ **NP**) but there are several problems which are regarded as being likely candidates. Two of the strongest candidates are the factoring and graph isomorphism problems:

GRAPH ISOMORPHISM: Suppose G and G' are two undirected graphs over the vertices $V \equiv \{v_1, \ldots, v_n\}$. Are G and G' isomorphic? That is, does there exist a one-to-one function $\varphi : V \rightarrow V$ such that the edge (v_i, v_j) is contained in G if and only if $(\varphi(v_i), \varphi(v_j))$ is contained in G?

Problems in **NPI** are interesting to researchers in quantum computation and quantum information for two reasons. First, it is desirable to find fast quantum algorithms to solve problems which are not in **P**. Second, many suspect that quantum computers will not be able to efficiently solve all problems in **NP**, ruling out **NP**-complete problems. Thus, it is natural to focus on the class **NPI**. Indeed, a fast quantum algorithm for factoring has been discovered (Chapter 5), and this has motivated the search for fast quantum algorithms for other problems suspected to be in **NPI**.

3.2.4 A plethora of complexity classes

We have investigated some of the elementary properties of some important complexity classes. A veritable pantheon of complexity classes exists, and there are many non-trivial relationships known or suspected between these classes. For quantum computation and quantum information, it is not necessary to understand all the different complexity classes that have been defined. However, it is useful to have some appreciation for the more important of the complexity classes, many of which have natural analogues in the study of quantum computation and quantum information. Furthermore, if we are to understand how powerful quantum computers are, then it behooves us to understand how the class of problems solvable on a quantum computer fits into the zoo of complexity classes which may be defined for classical computers.

There are essentially three properties that may be varied in the definition of a complexity class: the resource of interest (time, space, ...), the type of problem being considered (decision problem, optimization problem, ...), and the underlying computational model (deterministic Turing machine, probabilistic Turing machine, quantum computer, ...). Not surprisingly, this gives us an enormous range to define complexity classes. In this section, we briefly review a few of the more important complexity classes and some of their elementary properties. We begin with a complexity class defined by changing the resource of interest from time to *space*.

The most natural space-bounded complexity class is the class **PSPACE** of decision problems which may be solved on a Turing machine using a polynomial number of working bits, with no limitation on the amount of time that may be used by the machine (see Exercise 3.25). Obviously, **P** is included in **PSPACE**, since a Turing machine that halts after polynomial time can only traverse polynomially many squares, but it is also true that **NP** is a subset of **PSPACE**. To see this, suppose L is any language in **NP**. Suppose problems of size n have witnesses of size at most $p(n)$, where $p(n)$ is some polynomial in n. To determine whether or not the problem has a solution, we may sequentially test all $2^{p(n)}$ possible witnesses. Each test can be run in polynomial time, and therefore polynomial space. If we erase all the intermediate working between tests then we can test all the possibilities using polynomial space.

Unfortunately, at present it is not even known whether **PSPACE** contains problems which are not in **P**! This is a pretty remarkable situation – it seems fairly obvious that having unlimited time and polynomial spatial resources must be more powerful than having only a polynomial amount of time. However, despite considerable effort and in-

genuity, this has never been shown. We will see later that the class of problems solvable on a quantum computer in polynomial time is a subset of **PSPACE**, so proving that a problem efficiently solvable on a quantum computer is not efficiently solvable on a classical computer would establish that **P** $\neq$ **PSPACE**, and thus solve a major outstanding problem of computer science. An optimistic way of looking at this result is that ideas from quantum computation might be useful in proving that **P** $\neq$ **PSPACE**. Pessimistically, one might conclude that it will be a long time before anyone rigorously proves that quantum computers can be used to efficiently solve problems that are intractable on a classical computer. Even more pessimistically, it is possible that **P = PSPACE**, in which case quantum computers offer no advantage over classical computers! However, very few (if any) computational complexity theorists believe that **P = PSPACE**.

Exercise 3.25: (**PSPACE** $\subseteq$ **EXP**) The complexity class **EXP** (for *exponential time*) contains all decision problems which may be decided by a Turing machine running in exponential time, that is time $O(2^{n^k})$, where k is any constant. Prove that **PSPACE** $\subseteq$ **EXP**. (*Hint*: If a Turing machine has l internal states, an m letter alphabet, and uses space $p(n)$, argue that the machine can exist in one of at most $lm^{p(n)}$ different states, and that if the Turing machine is to avoid infinite loops then it must halt before revisiting a state.)

Exercise 3.26: (**L** $\subseteq$ **P**) The complexity class **L** (for *logarithmic space*) contains all decision problems which may be decided by a Turing machine running in logarithmic space, that is, in space $O(\log(n))$. More precisely, the class **L** is defined using a two-tape Turing machine. The first tape contains the problem instance, of size n, and is a read-only tape, in the sense that only program lines which don't change the contents of the first tape are allowed. The second tape is a working tape which initially contains only blanks. The logarithmic space requirement is imposed on the second, working tape only. Show that **L** $\subseteq$ **P**.

Does allowing more time or space give greater computational power? The answer to this question is yes in both cases. Roughly speaking, the *time hierarchy theorem* states that **TIME**$(f(n))$ is a proper subset of **TIME**$(f(n)\log^2(f(n)))$. Similarly, the *space hierarchy theorem* states that **SPACE**$(f(n))$ is a proper subset of **SPACE**$(f(n)\log(f(n)))$, where **SPACE**$(f(n))$ is, of course, the complexity class consisting of all languages that can be decided with spatial resources $O(f(n))$. The hierarchy theorems have interesting implications with respect to the equality of complexity classes. We know that

$$\mathbf{L} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}. \tag{3.1}$$

Unfortunately, although each of these inclusions is widely believed to be strict, none of them has ever been proved to be strict. However, the time hierarchy theorem implies that **P** is a strict subset of **EXP**, and the space hierarchy theorem implies that **L** is a strict subset of **PSPACE**! So we can conclude that at least one of the inclusions in (3.1) must be strict, although we do not know which one.

What should we do with a problem once we know that it is **NP**-complete, or that some other hardness criterion holds? It turns out that this is far from being the end of the story in problem analysis. One possible line of attack is to identify special cases of the problem which may be amenable to attack. For example, in Exercise 3.24 we saw that the 2SAT problem has an efficient solution, despite the **NP**-completeness of SAT.

Another approach is to change the type of problem which is being considered, a tactic which typically results in the definition of new complexity classes. For example, instead of finding exact solutions to an **NP**-complete problem, we can instead try to find good algorithms for finding *approximate* solutions to a problem. For example, the VERTEX COVER problem is an **NP**-complete problem, yet in Exercise 3.27 we show that it is possible to efficiently find an approximation to the minimal vertex cover which is correct to within a factor two! On the other hand, in Problem 3.6 we show that it is not possible to find approximations to solutions of TSP correct to within any factor, *unless* **P = NP**!

Exercise 3.27: (Approximation algorithm for VERTEX COVER) Let $G = (V, E)$ be an undirected graph. Prove that the following algorithm finds a vertex cover for G that is within a factor two of being a minimal vertex cover:

$VC = \emptyset$
$E' = E$
do until $E' = \emptyset$
    let (α, β) be any edge of E'
    $VC = VC \cup \{\alpha, \beta\}$
    remove from E' every edge incident on α or β
return VC.

Why is it possible to approximate the solution of one **NP**-complete problem, but not another? After all, isn't it possible to efficiently transform from one problem to another? This is certainly true, however it is not necessarily true that this transformation preserves the notion of a 'good approximation' to a solution. As a result, the computational complexity theory of approximation algorithms for problems in **NP** has a structure that goes beyond the structure of **NP** proper. An entire complexity theory of approximation algorithms exists, which unfortunately is beyond the scope of this book. The basic idea, however, is to define a notion of reduction that corresponds to being able to efficiently reduce one approximation problem to another, in such a way that the notion of good approximation is preserved. With such a notion, it is possible to define complexity classes such as **MAXSNP** by analogy to the class **NP**, as the set of problems for which it is possible to efficiently verify approximate solutions to the problem. Complete problems exist for **MAXSNP**, just as for **NP**, and it is an interesting open problem to determine how the class **MAXSNP** compares to the class of approximation problems which are efficiently solvable.

We conclude our discussion with a complexity class that results when the underlying model of computation itself is changed. Suppose a Turing machine is endowed with the ability to flip coins, using the results of the coin tosses to decide what actions to take during the computation. Such a Turing machine may only accept or reject inputs with a certain probability. The complexity class **BPP** (for *bounded-error probabilistic time*) contains all languages L with the property that there exists a probabilistic Turing machine M such that if $x \in L$ then M accepts x with probability at least $3/4$, and if $x \notin L$, then M rejects x with probability at least $3/4$. The following exercise shows that the choice of the constant $3/4$ is essentially arbitrary:

Exercise 3.28: (Arbitrariness of the constant in the definition of BPP) Suppose k is a fixed constant, $1/2 < k \leq 1$. Suppose L is a language such that there exists a Turing machine M with the property that whenever $x \in L$, M accepts x with probability at least k, and whenever $x \notin L$, M rejects x with probability at least k. Show that $L \in \mathbf{BPP}$.

Indeed, the *Chernoff bound*, discussed in Box 3.4, implies that with just a few repetitions of an algorithm deciding a language in **BPP** the probability of success can be amplified to the point where it is essentially equal to one, for all intents and purposes. For this reason, **BPP** even more than **P** is the class of decision problems which is usually regarded as being efficiently solvable on a classical computer, and it is the quantum analogue of **BPP**, known as **BQP**, that is most interesting in our study of quantum algorithms.

3.2.5 Energy and computation

Computational complexity studies the amount of time and space required to solve a computational problem. Another important computational resource is *energy*. In this section, we study the energy requirements for computation. Surprisingly, it turns out that computation, both classical and quantum, can in principle be done without expending any energy! Energy consumption in computation turns out to be deeply linked to the *reversibility* of the computation. Consider a gate like the NAND gate, which takes as input two bits, and produces a single bit as output. This gate is intrinsically *irreversible* because, given the output of the gate, the input is not uniquely determined. For example, if the output of the NAND gate is 1, then the input could have been any one of $00, 01$, or 10. On the other hand, the NOT gate is an example of a *reversible* logic gate because, given the output of the NOT gate, it is possible to infer what the input must have been.

Another way of understanding irreversibility is to think of it in terms of information erasure. If a logic gate is irreversible, then some of the information input to the gate is lost irretrievably when the gate operates – that is, some of the information has been erased by the gate. Conversely, in a reversible computation, no information is ever erased, because the input can always be recovered from the output. Thus, saying that a computation is reversible is equivalent to saying that no information is erased during the computation.

What is the connection between energy consumption and irreversibility in computation? *Landauer's principle* provides the connection, stating that, in order to erase information, it is necessary to dissipate energy. More precisely, Landauer's principle may be stated as follows:

> **Landauer's principle (first form)**: Suppose a computer erases a single bit of information. The amount of energy dissipated into the environment is *at least* $k_B T \ln 2$, where k_B is a universal constant known as *Boltzmann's constant*, and T is the temperature of the environment of the computer.

According to the laws of thermodynamics, Landauer's principle can be given an alternative form stated not in terms of energy dissipation, but rather in terms of entropy:

> **Landauer's principle (second form)**: Suppose a computer erases a single bit of information. The entropy of the environment increases by *at least* $k_B \ln 2$, where k_B is Boltzmann's constant.

Justifying Landauer's principle is a problem of physics that lies beyond the scope of this

Box 3.4: BPP and the Chernoff bound

Suppose we have an algorithm for a decision problem which gives the correct answer with probability $1/2 + \epsilon$, and the wrong answer with probability $1/2 - \epsilon$. If we run the algorithm n times, then it seems reasonable to guess that the correct answer is whichever appeared most frequently. How reliably does this procedure work? The *Chernoff bound* is a simple result from elementary probability which answers this question.

Theorem 3.3: (**The Chernoff bound**) Suppose $X_1, \ldots, X_n$ are independent and identically distributed random variables, each taking the value 1 with probability $1/2 + \epsilon$, and the value 0 with probability $1/2 - \epsilon$. Then

$$p\left(\sum_{i=1}^{n} X_i \leq n/2\right) \leq e^{-2\epsilon^2 n}. \tag{3.2}$$

Proof
Consider any sequence $(x_1, \ldots, x_n)$ containing at most $n/2$ ones. The probability of such a sequence occurring is maximized when it contains $\lfloor n/2 \rfloor$ ones, so

$$p(X_1 = x_1, \ldots, X_n = x_n) \leq \left(\frac{1}{2} - \epsilon\right)^{\frac{n}{2}} \left(\frac{1}{2} + \epsilon\right)^{\frac{n}{2}} \tag{3.3}$$

$$= \frac{(1 - 4\epsilon^2)^{\frac{n}{2}}}{2^n}. \tag{3.4}$$

There can be at most 2^n such sequences, so

$$p\left(\sum_i X_i \leq n/2\right) \leq 2^n \times \frac{(1 - 4\epsilon^2)^{\frac{n}{2}}}{2^n} = (1 - 4\epsilon^2)^{\frac{n}{2}}. \tag{3.5}$$

Finally, by calculus, $1 - x \leq \exp(-x)$, so

$$p\left(\sum_i X_i \leq n/2\right) \leq e^{-4\epsilon^2 n/2} = e^{-2\epsilon^2 n}. \tag{3.6}$$

□

What this tells us is that for fixed ϵ, the probability of making an error decreases *exponentially quickly* in the number of repetitions of the algorithm. In the case of **BPP** we have $\epsilon = 1/4$, so it takes only a few hundred repetitions of the algorithm to reduce the probability of error below 10^{-20}, at which point an error in one of the computer's components becomes much more likely than an error due to the probabilistic nature of the algorithm.

book – see the end of chapter 'History and further reading' if you wish to understand why Landauer's principle holds. However, if we accept Landauer's principle as given, then it raises a number of interesting questions. First of all, Landauer's principle only provides a *lower bound* on the amount of energy that must be dissipated to erase information.

How close are existing computers to this lower bound? Not very, turns out to be the answer – computers circa the year 2000 dissipate roughly $500 k_B T \ln 2$ in energy for each elementary logical operation.

Although existing computers are far from the limit set by Landauer's principle, it is still an interesting problem of principle to understand how much the energy consumption can be reduced. Aside from the intrinsic interest of the problem, a practical reason for the interest follows from Moore's law: if computer power keeps increasing then the amount of energy dissipated must also increase, unless the energy dissipated per operation drops at least as fast as the rate of increase in computing power.

If all computations could be done reversibly, then Landauer's principle would imply no lower bound on the amount of energy dissipated by the computer, since no bits at all are erased during a reversible computation. Of course, it is possible that some other physical principle might require that energy be dissipated during the computation; fortunately, this turns out not to be the case. But is it possible to perform universal computation without erasing any information? Physicists can cheat on this problem to see in advance that the answer to this question *must* be yes, because our present understanding of the laws of physics is that they are fundamentally reversible. That is, if we know the final state of a closed physical system, then the laws of physics allow us to work out the initial state of the system. If we believe that those laws are correct, then we must conclude that hidden in the irreversible logic gates like AND and OR, there must be some underlying reversible computation. But where is this hidden reversibility, and can we use it to construct manifestly reversible computers?

We will use two different techniques to give reversible circuit-based models capable of universal computation. The first model, a computer built entirely of billiard balls and mirrors, gives a beautiful concrete realization of the principles of reversible computation. The second model, based on a reversible logic gate known as the *Toffoli gate* (which we first encountered in Section 1.4.1), is a more abstract view of reversible computation that will later be of great use in our discussion of quantum computation. It is also possible to build reversible Turing machines that are universal for computation; however, we won't study these here, since the reversible circuit models turn out to be much more useful for quantum computation.

The basic idea of the billiard ball computer is illustrated in Figure 3.14. Billiard ball 'inputs' enter the computer from the left hand side, bouncing off mirrors and each other, before exiting as 'outputs' on the right hand side. The presence or absence of a billiard ball at a possible input site is used to indicate a logical 1 or a logical 0, respectively. The fascinating thing about this model is that it is manifestly reversible, insofar as its operation is based on the laws of classical mechanics. Furthermore, this model of computation turns out to be *universal* in the sense that it can be used to simulate an arbitrary computation in the standard circuit model of computation.

Of course, if a billiard ball computer were ever built it would be highly unstable. As any billiards player can attest, a billiard ball rolling frictionlessly over a smooth surface is easily knocked off course by small perturbations. The billiard ball model of computation depends on perfect operation, and the absence of external perturbations such as those caused by thermal noise. Periodic corrections can be performed, but information gained by doing this would have to be erased, requiring work to be performed. Expenditure of energy thus serves the purpose of reducing this susceptibility to noise, which is necessary for a practical, real-world computational machine. For the purposes of this introduction,

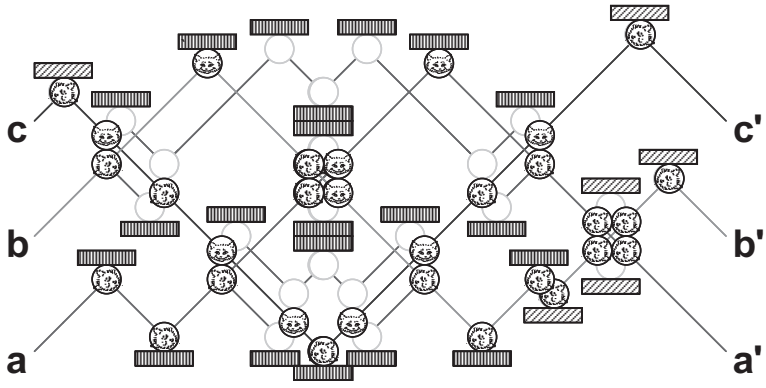


Figure 3.14. A simple billiard ball computer, with three input bits and three output bits, shown entering on the left and leaving on the right, respectively. The presence or absence of a billiard ball indicates a 1 or a 0, respectively. Empty circles illustrate potential paths due to collisions. This particular computer implements the Fredkin classical reversible logic gate, discussed in the text.

we will ignore the effects of noise on the billiard ball computer, and concentrate on understanding the essential elements of reversible computation.

The billiard ball computer provides an elegant means for implementing a reversible universal logic gate known as the *Fredkin gate*. Indeed, the properties of the Fredkin gate provide an informative overview of the general principles of reversible logic gates and circuits. The Fredkin gate has three input bits and three output bits, which we refer to as a, b, c and a', b', c', respectively. The bit c is a *control bit*, whose value is not changed by the action of the Fredkin gate, that is, $c' = c$. The reason c is called the control bit is because it controls what happens to the other two bits, a and b. If c is set to 0 then a and b are left alone, $a' = a$, $b' = b$. If c is set to 1, a and b are swapped, $a' = b$, $b' = a$. The explicit truth table for the Fredkin gate is shown in Figure 3.15. It is easy to see that the Fredkin gate is reversible, because given the output a', b', c', we can determine the inputs a, b, c. In fact, to recover the original inputs a, b and c we need only apply another Fredkin gate to a', b', c':

Exercise 3.29: (**Fredkin gate is self-inverse**) Show that applying two consecutive Fredkin gates gives the same outputs as inputs.

Examining the paths of the billiard balls in Figure 3.14, it is not difficult to verify that this billiard ball computer implements the Fredkin gate:

Exercise 3.30: Verify that the billiard ball computer in Figure 3.14 computes the Fredkin gate.

In addition to reversibility, the Fredkin gate also has the interesting property that the number of 1s is *conserved* between the input and output. In terms of the billiard ball computer, this corresponds to the number of billiard balls going into the Fredkin gate being equal to the number coming out. Thus, it is sometimes referred to as being a *conservative* reversible logic gate. Such reversibility and conservative properties are interesting to a physicist because they can be motivated by fundamental physical princi-

Inputs			Outputs		
a	b	c	a'	b'	c'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	0	1	1
1	1	0	1	1	0
1	1	1	1	1	1

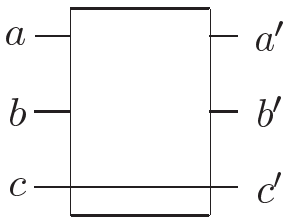


Figure 3.15. Fredkin gate truth table and circuit representation. The bits a and b are swapped if the control bit c is set, and otherwise are left alone.

ples. The laws of Nature are reversible, with the possible exception of the measurement postulate of quantum mechanics, discussed in Section 2.2.3 on page 84. The conservative property can be thought of as analogous to properties such as conservation of mass, or conservation of energy. Indeed, in the billiard ball model of computation the conservative property corresponds exactly to conservation of mass.

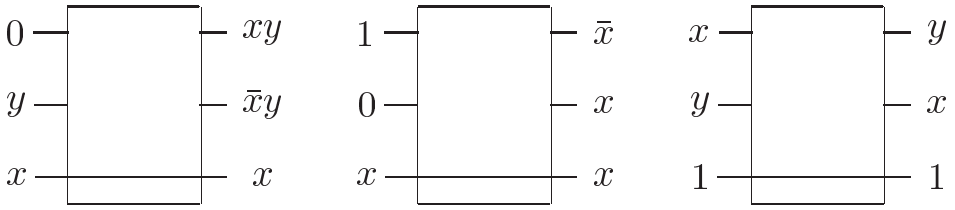


Figure 3.16. Fredkin gate configured to perform the elementary gates AND (left), NOT (middle), and a primitive routing function, the CROSSOVER (right). The middle gate also serves to perform the FANOUT operation, since it produces two copies of x at the output. Note that each of these configurations requires the use of extra 'ancilla' bits prepared in standard states – for example, the 0 input on the first line of the AND gate – and in general the output contains 'garbage' not needed for the remainder of the computation.

The Fredkin gate is not only reversible and conservative, it's a universal logic gate as well! As illustrated in Figure 3.16, the Fredkin gate can be configured to simulate AND, NOT, CROSSOVER and FANOUT functions, and thus can be cascaded to simulate any classical circuit whatsoever.

To simulate irreversible gates such as AND using the Fredkin gate, we made use of two ideas. First, we allowed the input of 'ancilla' bits to the Fredkin gate, in specially prepared states, either 0 or 1. Second, the output of the Fredkin gate contained extraneous 'garbage' not needed for the remainder of the computation. These ancilla and garbage bits are not directly important to the computation. Their importance lies in the fact that they make the computation reversible. Indeed the irreversibility of gates like the AND and OR may be viewed as a consequence of the ancilla and garbage bits being 'hidden'. Summarizing, given any classical circuit computing a function $f(x)$, we can build a reversible circuit made entirely of Fredkin gates, which on input of x, together with some ancilla bits

in a standard state a, computes $f(x)$, together with some extra 'garbage' output, $g(x)$. Therefore, we represent the action of the computation as $(x, a) \rightarrow (f(x), g(x))$.

We now know how to compute functions reversibly. Unfortunately, this computation produces unwanted garbage bits. With some modifications it turns out to be possible to perform the computation so that any garbage bits produced are in a *standard state*. This construction is crucial for quantum computation, because garbage bits whose value depends upon x will in general destroy the interference properties crucial to quantum computation. To understand how this works it is convenient to assume that the NOT gate is available in our repertoire of reversible gates, so we may as well assume that the ancilla bits a all start out as 0s, with NOT gates being added where necessary to turn the ancilla 0s into 1s. It will also be convenient to assume that the classical controlled-NOT gate is available, defined in a manner analogous to the quantum definition of Section 1.3.2, that is, the inputs (c, t) are taken to $(c, t \oplus c)$, where $\oplus$ denotes addition modulo 2. Notice that $t = 0$ gives $(c, 0) \rightarrow (c, c)$, so the controlled-NOT can be thought of as a reversible copying gate or FANOUT, which leaves no garbage bits at the output.

With the additional NOT gates appended at the beginning of the circuit, the action of the computation may be written as $(x, 0) \rightarrow (f(x), g(x))$. We could also have added CNOT gates to the beginning of the circuit, in order to create a copy of x which is not changed during the subsequent computation. With this modification, the action of the circuit may be written

$$(x, 0, 0) \rightarrow (x, f(x), g(x)) . \tag{3.7}$$

Equation (3.7) is a very useful way of writing the action of the reversible circuit, because it allows an idea known as *uncomputation* to be used to get rid of the garbage bits, for a small cost in the running time of the computation. The idea is the following. Suppose we start with a four register computer in the state $(x, 0, 0, y)$. The second register is used to store the result of the computation, and the third register is used to provide workspace for the computation, that is, the garbage bits $g(x)$. The use of the fourth register is described shortly, and we assume it starts in an arbitrary state y.

We begin as before, by applying a reversible circuit to compute f, resulting in the state $(x, f(x), g(x), y)$. Next, we use CNOTs to add the result $f(x)$ bitwise to the fourth register, leaving the machine in the state $(x, f(x), g(x), y \oplus f(x))$. However, all the steps used to compute $f(x)$ were reversible and did not affect the fourth register, so by applying the reverse of the circuit used to compute f we come to the state $(x, 0, 0, y \oplus f(x))$. Typically, we omit the ancilla 0s from the description of the function evaluation, and just write the action of the circuit as

$$(x, y) \rightarrow (x, y \oplus f(x)) . \tag{3.8}$$

In general we refer to this modified circuit computing f as *the* reversible circuit computing f, even though in principle there are many other reversible circuits which could be used to compute f.

What resource overhead is involved in doing reversible computation? To analyze this question, we need to count the number of extra ancilla bits needed in a reversible circuit, and compare the gate counts with classical models. It ought to be clear that the number of gates in a reversible circuit is the same as in an irreversible circuit to within the constant factor which represents the number of Fredkin gates needed to simulate a single element of the irreversible circuit, and an additional factor of two for uncomputation, with an

overhead for the extra CNOT operations used in reversible computation which is linear in the number of bits involved in the circuit. Similarly, the number of ancilla bits required scales at most linearly with the number of gates in the irreversible circuit, since each element in the irreversible circuit can be simulated using a constant number of ancilla bits. As a result, natural complexity classes such as **P** and **NP** are the same no matter whether a reversible or irreversible model of computation is used. For more elaborate complexity classes like **PSPACE** the situation is not so immediately clear; see Problem 3.9 and 'History and further reading' for a discussion of some such subtleties.

Exercise 3.31: (Reversible half-adder) Construct a reversible circuit which, when two bits x and y are input, outputs $(x, y, c, x \oplus y)$, where c is the carry bit when x and y are added.

The Fredkin gate and its implementation using the billiard ball computer offers a beautiful paradigm for reversible computation. There is another reversible logic gate, the *Toffoli gate*, which is also universal for classical computation. While the Toffoli gate does not have quite the same elegant physical simplicity as the billiard ball implementation of the Fredkin gate, it will be more useful in the study of quantum computation. We have already met the Toffoli gate in Section 1.4.1, but for convenience we review its properties here.

The Toffoli gate has three input bits, a, b and c. a and b are known as the first and second *control bits*, while c is the *target bit*. The gate leaves both control bits unchanged, flips the target bit if both control bits are set, and otherwise leaves the target bit alone. The truth table and circuit representation for the Toffoli gate are shown in Figure 3.17.

Inputs			Outputs		
a	b	c	a'	b'	c'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

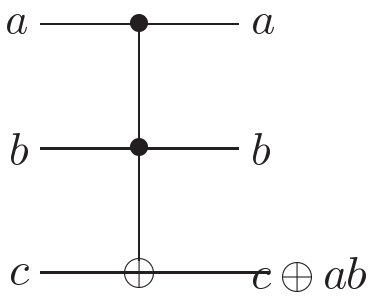


Figure 3.17. Truth table and circuit representation of the Toffoli gate.

How can the Toffoli gate be used to do universal computation? Suppose we wish to NAND the bits a and b. To do this using the Toffoli gate, we input a and b as control bits, and send in an ancilla bit set to 1 as the target bit, as shown in Figure 3.18. The NAND of a and b is output as the target bit. As expected from our study of the Fredkin gate, the Toffoli gate simulation of a NAND requires the use of a special ancilla input, and some of the outputs from the simulation are garbage bits.

The Toffoli gate can also be used to implement the FANOUT operation by inputting an ancilla 1 to the first control bit, and a to the second control bit, producing the output $1, a, a$. This is illustrated in Figure 3.19. Recalling that NAND and FANOUT are together

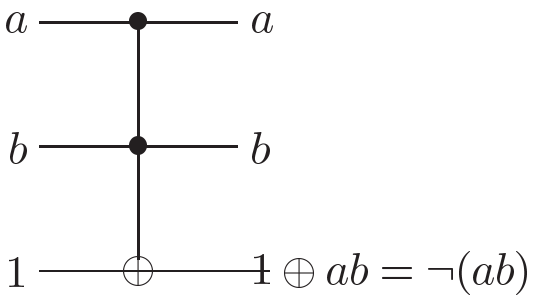


Figure 3.18. Implementing a NAND gate using a Toffoli gate. The top two bits represent the input to the NAND, while the third bit is prepared in the standard state 1, sometimes known as an *ancilla* state. The output from the NAND is on the third bit.

universal for computation, we see that an arbitrary circuit can be efficiently simulated using a reversible circuit consisting only of Toffoli gates and ancilla bits, and that useful additional techniques such as uncomputation may be achieved using the same methods as were employed with the Fredkin gate.

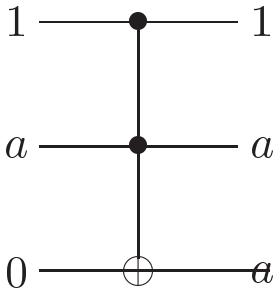


Figure 3.19. FANOUT with the Toffoli gate, with the second bit being the input to the FANOUT, and the other two bits standard ancilla states. The output from FANOUT appears on the second and third bits.

Our interest in reversible computation was motivated by our desire to understand the energy requirements for computation. It is clear that the noise-free billiard ball model of computation requires no energy for its operation; what about models based upon the Toffoli gate? This can only be determined by examining specific models for the computation of the Toffoli gate. In Chapter 7, we examine several such implementations, and it turns out that, indeed, the Toffoli gate can be implemented in a manner which does not require the expenditure of energy.

There is a significant caveat attached to the idea that computation can be done without the expenditure of energy. As we noted earlier, the billiard ball model of computation is highly sensitive to noise, and this is true of many other models of reversible computation. To nullify the effects of noise, some form of error-correction needs to be done. Such error-correction typically involves the performance of measurements on the system to determine whether the system is behaving as expected, or if an error has occurred. Because the computer's memory is finite, the bits used to store the measurement results utilized in error-correction must eventually be erased to make way for new measurement results. According to Landauer's principle, this erasure carries an associated energy cost

that must be accounted for when tallying the total energy cost of the computation. We analyze the energy cost associated with error-correction in more detail in Section 12.4.4.

What can we conclude from our study of reversible computation? There are three key ideas. First, reversibility stems from keeping track of every bit of information; irreversibility occurs only when information is lost or erased. Second, by doing computation reversibly, we obviate the need for energy expenditure during computation. All computations can be done, in principle, for zero cost in energy. Third, reversible computation can be done efficiently, without the production of garbage bits whose value depends upon the input to the computation. That is, if there is an irreversible circuit computing a function f, then there is an efficient simulation of this circuit by a reversible circuit with action $(x, y) \rightarrow (x, y \oplus f(x))$.

What are the implications of these results for physics, computer science, and for quantum computation and quantum information? From the point of view of a physicist or hardware engineer worried about heat dissipation, the good news is that, in principle, it is possible to make computation dissipation-free by making it reversible, although in practice energy dissipation is required for system stability and immunity from noise. At an even more fundamental level, the ideas leading to reversible computation also lead to the resolution of a century-old problem in the foundations of physics, the famous problem of *Maxwell's demon*. The story of this problem and its resolution is outlined in Box 3.5 on page 162. From the point of view of a computer scientist, reversible computation validates the use of irreversible elements in models of computation such as the Turing machine (since using them or not gives polynomially equivalent models). Moreover, since the physical world is fundamentally reversible, one can argue that complexity classes based upon reversible models of computation are more natural than complexity classes based upon irreversible models, a point revisited in Problem 3.9 and 'History and further reading'. From the point of view of quantum computation and quantum information, reversible computation is enormously important. To harness the full power of quantum computation, any classical subroutines in a quantum computation must be performed reversibly and without the production of garbage bits depending on the classical input.

Exercise 3.32: (From Fredkin to Toffoli and back again) What is the smallest number of Fredkin gates needed to simulate a Toffoli gate? What is the smallest number of Toffoli gates needed to simulate a Fredkin gate?

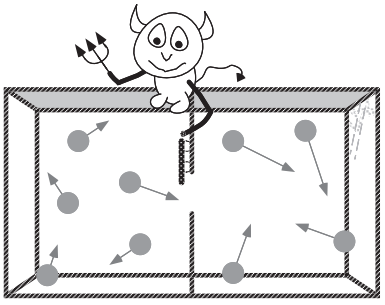
3.3 Perspectives on computer science

In a short introduction such as this chapter, it is not remotely possible to cover in detail all the great ideas of a field as rich as computer science. We hope to have conveyed to you something of what it means to *think* like a computer scientist, and provided a basic vocabulary and overview of some of the fundamental concepts important in the understanding of computation. To conclude this chapter, we briefly touch on some more general issues, in order to provide some perspective on how quantum computation and quantum information fits into the overall picture of computer science.

Our discussion has revolved around the Turing machine model of computation. How does the computational power of unconventional models of computation such as massively parallel computers, DNA computers and analog computers compare with the standard

Box 3.5: Maxwell's demon

The laws of thermodynamics govern the amount of work that can be performed by a physical system at thermodynamic equilibrium. One of these laws, the second law of thermodynamics, states that the *entropy* in a closed system can never decrease. In 1871, James Clerk Maxwell proposed the existence of a machine that apparently violated this law. He envisioned a miniature little 'demon', like that shown in the figure below, which could reduce the entropy of a gas cylinder initially at equilibrium by individually separating the fast and slow molecules into the two halves of the cylinder. This demon would sit at a little door at the middle partition. When a fast molecule approaches from the left side the demon opens a door between the partitions, allowing the molecule through, and then closes the door. By doing this many times the total entropy of the cylinder can be *decreased*, in apparent violation of the second law of thermodynamics.



The resolution to the Maxwell's demon paradox lies in the fact that the demon must perform *measurements* on the molecules moving between the partitions, in order to determine their velocities. The result of this measurement must be stored in the demon's memory. Because any memory is finite, the demon must eventually begin erasing information from its memory, in order to have space for new measurement results. By Landauer's principle, this act of erasing information increases the total entropy of the combined system – demon, gas cylinder, and their environments. In fact, a complete analysis shows that Landauer's principle implies that the entropy of the combined system is increased *at least as much* by this act of erasing information as the entropy of the combined system is decreased by the actions of the demon, thus ensuring that the second law of thermodynamics is obeyed.

Turing machine model of computation and, implicitly, with quantum computation? Let's begin with parallel computing architectures. The vast majority of computers in existence are serial computers, processing instructions one at a time in some central processing unit. By contrast, parallel computers can process more than one instruction at a time, leading to a substantial savings in time and money for some applications. Nevertheless, parallel processing does not offer any fundamental advantage over the standard Turing machine model when issues of efficiency are concerned, because a Turing machine can simulate a parallel computer with polynomially equivalent total physical resources – the total space and time used by the computation. What a parallel computer gains in time,

it loses in the total spatial resources required to perform the computation, resulting in a net of no essential change in the power of the computing model.

An interesting specific example of massively parallel computing is the technique of *DNA computing*. A strand of DNA, deoxyribonucleic acid, is a molecule composed of a sequence (a polymer) of four kinds of nucleotides distinguished by the bases they carry, denoted by the letter A (adenine), C (cytosine), G (guanine) and T (thymine). Two strands, under certain circumstances, can anneal to form a double strand, if the respective base pairs form complements of each other (A matches T and G matches C). The ends are also distinct and must match appropriately. Chemical techniques can be used to amplify the number of strands beginning or ending with specific sequences (polymerase chain reaction), separate the strands by length (gel electrophoresis), dissolve double strands into single strands (changing temperature and pH), read the sequence on a strand, cut strands at a specific position (restriction enzymes), and detect if a certain sequence of DNA is in a test tube. The procedure for using these mechanisms in a robust manner is rather involved, but the basic idea can be appreciated from an example.

The directed Hamiltonian path problem is a simple and equivalently hard variant of the Hamiltonian cycle problem of Section 3.2.2, in which the goal is to determine if a path exists or not between two specified vertices j_1 and j_N in a directed graph G of N vertices, entering each vertex exactly once, and following only allowed edge directions. This problem can be solved with a DNA computer using the following five steps, in which x_j are chosen to be unique sequences of bases (and $\bar{x}_j$ their complements), DNA strands $x_j x_k$ encode edges, and strands $\bar{x}_j \bar{x}_j$ encode vertices. (1) Generate random paths through G, by combining a mixture of all possible vertex and edge DNA strands, and waiting for the strands to anneal. (2) Keep only the paths beginning with j_1 and ending with j_N, by amplifying only the double strands beginning with $\bar{x}_{j_1}$ and ending with $\bar{x}_{j_N}$. (3) Select only paths of length N, by separating the strands according to their length. (4) Select only paths which enter each vertex at least once, by dissolving the DNA into single strands, and annealing with all possible vertex strands one at a time and filtering out only those strands which anneal. And (5) detect if any strands have survived the selection steps; if so, then a path exists, and otherwise, it does not. To ensure the answer is correct with sufficiently high probability, x_j may be chosen to contain many (≈ 30) bases, and a large number ($\approx 10^{14}$ or more are feasible) of strands are used in the reaction.

Heuristic methods are available to improve upon this basic idea. Of course, exhaustive search methods such as this only work as long as all possible paths can be generated efficiently, and thus the number of molecules used must grow exponentially as the size of the problem (the number of vertices in the example above). DNA molecules are relatively small and readily synthesized, and the huge number of DNA combinations one can fit into a test tube can stave off the exponential complexity cost increase for a while – up to a few dozen vertices – but eventually the exponential cost limits the applicability of this method. Thus, while DNA computing offers an attractive and physically realizable model of computation for the solution of certain problems, it is a classical computing technique and offers no essential improvement in principle over a Turing machine.

Analog computers offer a yet another paradigm for performing computation. A computer is analog when the physical representation of information it uses for computation is based on continuous degrees of freedom, instead of zeroes and ones. For example, a thermometer is an analog computer. Analog circuitry, using resistors, capacitors, and amplifiers, is also said to perform analog computation. Such machines have an infinite

resource to draw upon in the ideal limit, since continuous variables like position and voltage can store an unlimited amount of information. But this is only true in the absence of noise. The presence of a finite amount of noise reduces the number of *distinguishable states* of a continuous variable to a finite number – and thus restricts analog computers to the representation of a finite amount of information. In practice, noise reduces analog computers to being no more powerful than conventional digital computers, and through them Turing machines. One might suspect that quantum computers are just analog computers, because of the use of continuous parameters in describing qubit states; however, it turns out that the effects of noise on a quantum computer can effectively be *digitized*. As a result, their computational advantages remain even in the presence of a finite amount of noise, as we shall see in Chapter 10.

What of the effects of noise on digital computers? In the early days of computation, noise was a very real problem for computers. In some of the original computers a vacuum tube would malfunction every few minutes. Even today, noise is a problem for computational devices such as modems and hard drives. Considerable effort was devoted to the problem of understanding how to construct reliable computers from unreliable components. It was proven by von Neumann that this is possible with only a polynomial increase in the resources required for computation. Ironically, however, modern computers use none of those results, because the components of modern computers are fantastically reliable. Failure rates of 10^{-17} and even less are common in modern electronic components. For this reason, failures happen so rarely that the extra effort required to protect against them is not regarded as being worth making. On the other hand, we shall find that quantum computers are very delicate machines, and will likely require substantial application of error-correction techniques.

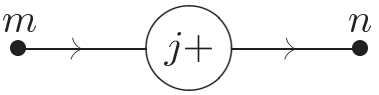
Different architectures may change the effects of noise. For example, if the effect of noise is ignored, then changing to a computer architecture in which many operations are performed in parallel may not change the number of operations which need to be done. However, a parallel system may be substantially more resistant to noise, because the effects of noise have less time to accumulate. Therefore, in a realistic analysis, the parallel version of an algorithm may have some substantial advantages over a serial implementation. Architecture design is a well developed field of study for classical computers. Hardly anything similar has been developed along the same lines for quantum computers, but the study of noise already suggests some desirable traits for future quantum computer architectures, such as a high level of parallelism.

A fourth model of computation is *distributed computation*, in which two or more spatially separated computational units are available to solve a computational problem. Obviously, such a model of computation is no more powerful than the Turing machine model in the sense that it can be efficiently simulated on a Turing machine. However, distributed computation gives rise to an intriguing new resource challenge: how best to utilize multiple computational units when the cost of *communication* between the units is high. This problem of distributed computation becomes especially interesting as computers are connected through high speed networks; although the total computational capacity of all the computers on a network might be extremely large, utilization of that potential is difficult. Most interesting problems do not divide easily into independent chunks that can be solved separately, and may frequently require global communication between different computational subsystems to exchange intermediate results or synchronize status. The field of *communication complexity* has been developed to address such issues, by

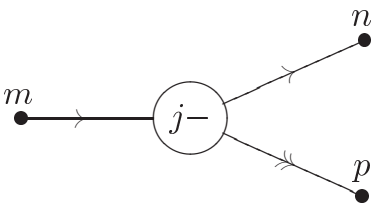
quantifying the cost of communication requirements in solving problems. When quantum resources are available and can be exchanged between distributed computers, the communication costs can sometimes be greatly reduced.

A recurring theme through these concluding thoughts and through the entire book is that despite the traditional independence of computer science from physical constraints, ultimately physical laws have tremendous impact not only upon how computers are realized, but also the class of problems they are capable of solving. The success of quantum computation and quantum information as a physically reasonable alternative model of computation questions closely held tenets of computer science, and thrusts notions of computer science into the forefront of physics. The task of the remainder of this book is to stir together ideas from these disparate fields, and to delight in what results!

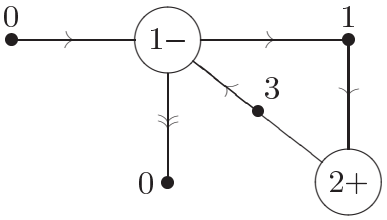
Problem 3.1: (Minsky machines) A *Minsky machine* consists of a finite set of *registers*, $r_1, r_2, \ldots, r_k$, each capable of holding an arbitrary non-negative integer, and a *program*, made up of *orders* of one of two types. The first type has the form:



The interpretation is that at point m in the program register r_j is incremented by one, and execution proceeds to point n in the program. The second type of order has the form:



The interpretation is that at point m in the program, register r_j is decremented if it contains a positive integer, and execution proceeds to point n in the program. If register r_j is zero then execution simply proceeds to point p in the program. The *program* for the Minsky machine consists of a collection of such orders, of a form like:



The starting and all possible halting points for the program are conventionally labeled zero. This program takes the contents of register r_1 and adds them to register r_2, while decrementing r_1 to zero.

(1) Prove that all (Turing) computable functions can be computed on a Minsky machine, in the sense that given a computable function $f(\cdot)$ there is a Minsky machine program that when the registers start in the state $(n, 0, \ldots, 0)$ gives as output $(f(n), 0, \ldots, 0)$.
(2) Sketch a proof that any function which can be computed on a Minsky machine, in the sense just defined, can also be computed on a Turing machine.

Problem 3.2: (Vector games) A *vector game* is specified by a finite list of vectors, all of the same dimension, and with integer co-ordinates. The game is to start with a vector x of non-negative integer co-ordinates and to add to x the first vector from the list which preserves the non-negativity of all the components, and to repeat this process until it is no longer possible. Prove that for any computable function $f(\cdot)$ there is a vector game which when started with the vector $(n, 0, \ldots, 0)$ reaches $(f(n), 0, \ldots, 0)$. (*Hint*: Show that a vector game in $k + 2$ dimensions can simulate a Minsky machine containing k registers.)

Problem 3.3: (Fractran) A *Fractran* program is defined by a list of positive rational numbers $q_1, \ldots, q_n$. It acts on a positive integer m by replacing it by $q_i m$, where i is the least number such that $q_i m$ is an integer. If there is ever a time when there is no i such that $q_i m$ is an integer, then execution stops. Prove that for any computable function $f(\cdot)$ there is a Fractran program which when started with 2^n reaches $2^{f(n)}$ without going through any intermediate powers of 2. (*Hint*: use the previous problem.)

Problem 3.4: (Undecidability of dynamical systems) A Fractran program is essentially just a very simple dynamical system taking positive integers to positive integers. Prove that there is no algorithm to decide whether such a dynamical system ever reaches 1.

Problem 3.5: (Non-universality of two bit reversible logic) Suppose we are trying to build circuits using only one and two bit reversible logic gates, and ancilla bits. Prove that there are Boolean functions which cannot be computed in this fashion. Deduce that the Toffoli gate cannot be simulated using one and two bit reversible gates, even with the aid of ancilla bits.

Problem 3.6: (Hardness of approximation of TSP) Let $r \geq 1$ and suppose that there is an approximation algorithm for TSP which is guaranteed to find the shortest tour among n cities to within a factor r. Let $G = (V, E)$ be any graph on n vertices. Define an instance of TSP by identifying cities with vertices in V, and defining the distance between cities i and j to be 1 if (i, j) is an edge of G, and to be $\lceil r \rceil |V| + 1$ otherwise. Show that if the approximation algorithm is applied to this instance of TSP then it returns a Hamiltonian cycle for G if one exists, and otherwise returns a tour of length more than $\lceil r \rceil |V|$. From the **NP**-completeness of HC it follows that no such approximation algorithm can exist unless **P = NP**.

Problem 3.7: (Reversible Turing machines)

(1) Explain how to construct a reversible Turing machine that can compute the same class of functions as is computable on an ordinary Turing machine. (*Hint*: It may be helpful to use a multi-tape construction.)

(2) Give general space and time bounds for the operation of your reversible Turing machine, in terms of the time $t(x)$ and space $s(x)$ required on an ordinary single-tape Turing machine to compute a function $f(x)$.

Problem 3.8: (Find a hard-to-compute class of functions (Research)) Find a natural class of functions on n inputs which requires a super-polynomial number of Boolean gates to compute.

Problem 3.9: (Reversible PSPACE = PSPACE) It can be shown that the problem 'quantified satisfiability', or QSAT, is **PSPACE**-complete. That is, every other language in **PSPACE** can be reduced to QSAT in polynomial time. The language QSAT is defined to consist of all Boolean formulae φ in n variables $x_1, \ldots, x_n$, and in conjunctive normal form, such that:

$$\exists_{x_1} \forall_{x_2} \exists_{x_3} \ldots \forall_{x_n} \ \varphi \ \text{ if } n \text{ is even;} \tag{3.9}$$

$$\exists_{x_1} \forall_{x_2} \exists_{x_3} \ldots \exists_{x_n} \ \varphi \ \text{ if } n \text{ is odd.} \tag{3.10}$$

Prove that a reversible Turing machine operating in polynomial space can be used to solve QSAT. Thus, the class of languages decidable by a computer operating reversibly in polynomial space is equal to **PSPACE**.

Problem 3.10: (Ancilla bits and efficiency of reversible computation) Let p_m be the mth prime number. Outline the construction of a reversible circuit which, upon input of m and n such that $n > m$, outputs the product $p_m p_n$, that is $(m, n) \rightarrow (p_m p_n, g(m, n))$, where $g(m, n)$ is the final state of the ancilla bits used by the circuit. Estimate the number of ancilla qubits your circuit requires. Prove that if a polynomial (in $\log n$) size reversible circuit can be found that uses $O(\log(\log n))$ ancilla bits then the problem of factoring a product of two prime numbers is in **P**.

History and further reading

Computer science is a huge subject with many interesting subfields. We cannot hope for any sort of completeness in this brief space, but instead take the opportunity to recommend a few titles of general interest, and some works on subjects of specific interest in relation to topics covered in this book, with the hope that they may prove stimulating.

Modern computer science dates to the wonderful 1936 paper of Turing[Tur36]. The Church–Turing thesis was first stated by Church[Chu36] in 1936, and was then given a more complete discussion from a different point of view by Turing. Several other researchers found their way to similar conclusions at about the same time. Many of these contributions and a discussion of the history may be found in a volume edited by Davis[Dav65]. Provocative discussions of the Church–Turing thesis and undecidability may be found in Hofstadter[Hof79] and Penrose[Pen89].

There are many excellent books on algorithm design. We mention only three. First, there is the classic series by Knuth[Knu97, Knu98a, Knu98b] which covers an enormous portion of computer science. Second, there is the marvelous book by Cormen, Leiserson, and Rivest[CLR90]. This huge book contains a plethora of well-written material on many areas

of algorithm design. Finally, the book of Motwani and Raghavan[MR95] is an excellent survey of the field of randomized algorithms.

The modern theory of computational complexity was especially influenced by the papers of Cook[Coo71] and Karp[Kar72]. Many similar ideas were arrived at independently in Russia by Levin[Lev73], but unfortunately took time to propagate to the West. The classic book by Garey and Johnson[GJ79] has also had an enormous influence on the field. More recently, Papadimitriou[Pap94] has written a beautiful book that surveys many of the main ideas of computational complexity theory. Much of the material in this chapter is based upon Papadimitriou's book. In this chapter we considered only one type of reducibility between languages, polynomial time reducibility. There are many other notions of reductions between languages. An early survey of these notions was given by Ladner, Lynch and Selman[LLS75]. The study of different notions of reducibility later blossomed into a subfield of research known as *structural complexity*, which has been reviewed by Balcázar, Diaz, and Gabarró[BDG88a, BDG88b].

The connection between information, energy dissipation, and computation has a long history. The modern understanding is due to a 1961 paper by Landauer[Lan61], in which Landauer's principle was first formulated. A paper by Szilard[Szi29] and a 1949 lecture by von Neumann[von66] (page 66) arrive at conclusions close to Landauer's principle, but do not fully grasp the essential point that it is the *erasure* of information that requires dissipation.

Reversible Turing machines were invented by Lecerf[Lec63] and later, but independently, in an influential paper by Bennett[Ben73]. Fredkin and Toffoli[FT82] introduced reversible circuit models of computation. Two interesting historical documents are Barton's May, 1978 MIT 6.895 term paper[Bar78], and Ressler's 1981 Master's thesis[Res81], which contain designs for a reversible PDP-10! Today, reversible logic is potentially important in implementations of low-power CMOS circuitry[YK95].

Maxwell's demon is a fascinating subject, with a long and intricate history. Maxwell proposed his demon in 1871[Max71]. Szilard published a key paper in 1929[Szi29] which anticipated many of the details of the final resolution of the problem of Maxwell's demon. In 1965 Feynman[FLS65b] resolved a special case of Maxwell's demon. Bennett, building on Landauer's work[Lan61], wrote two beautiful papers on the subject[BBBW82, Ben87] which completed the resolution of the problem. An interesting book about the history of Maxwell's demon and its exorcism is the collection of papers by Leff and Rex[LR90].

DNA computing was invented by Adleman, and the solution of the directed Hamiltonian path problem we describe is his[Adl94]. Lipton has also shown how 3SAT and circuit satisfiability can be solved in this model[Lip95]. A good general article is Adleman's *Scientific American* article[Adl98]; for an insightful look into the universality of DNA operations, see Winfree[Win98]. An interesting place to read about performing reliable computation in the presence of noise is the book by Winograd and Cowan[WC67]. This topic will be addressed again in Chapter 10. A good textbook on computer architecture is by Hennessey, Goldberg, and Patterson.[HGP96].

Problems 3.1 through 3.4 explore a line of thought originated by Minsky (in his beautiful book on computational machines[Min67]) and developed by Conway[Con72, Con86]. The Fractran programming language is certainly one of the most beautiful and elegant universal computational models known, as demonstrated by the following example, known

as PRIMEGAME[Con86]. PRIMEGAME is defined by the list of rational numbers:

$$\frac{17}{91}; \frac{78}{85}; \frac{19}{51}; \frac{23}{38}; \frac{29}{33}; \frac{77}{29}; \frac{95}{23}; \frac{77}{19}; \frac{1}{17}; \frac{11}{13}; \frac{13}{11}; \frac{15}{2}; \frac{1}{7}; \frac{55}{1}. \tag{3.11}$$

Amazingly, when PRIMEGAME is started at 2, the other powers of 2 that appear, namely, $2^2, 2^3, 2^5, 2^7, 2^{11}, 2^{13}, \ldots$, are precisely the prime powers of 2, with the powers stepping through the prime numbers, in order. Problem 3.9 is a special case of the more general subject of the spatial requirements for reversible computation. See the papers by Bennett[Ben89], and by Li, Tromp and Vitanyi[LV96, LTV98].