CSE 599d - Quantum Computing
Introduction and Basics of Quantum Theory

Dave Bacon
Department of Computer Science & Engineering, University of Washington

I. COURSE ADMINISTRIVIA

Information about the structure of the course can be found in the course syllabus, handed out the first day of class and available on the course website at `http://www.cs.washington.edu/cse590d`.

II. WHENCE QUANTUM COMPUTING?

Digital machines dominate our everyday life to such a degree that it is hard to imagine a time when the idea of a programmable computer was but a twinkle in a few oddball's eyes. But that's the way it was, for example, way back in 1936 when Alan Turing wrote his famous paper "On computable numbers, with an application to the Entscheidungsproblem" where the notion of a universal Turing machine was first introduced. In the years that followed, the optimism of the earlier pioneers in computing must have seemed insane: machines which can execute billions of arithmetic operations per second? Crazy! Even after the invention of the transistor, few suspected that digital computers would be as ubiquitous and useful as they are today.

The force (or whatever you want to call it, especially if you aren't Luke Skywalker) behind this, of course, is Moore's Law (or Moore's self-fullfilling prophecy if you want to be a cynic, but I prefer not to get so cynical, especially at the beginning of a class): the feature size on silicon chips is cut in half approximately every two years. Since Gordon Moore (whose hand I got to shake...twice!) wrote down his famous prediction forty years ago, we have been in an era of unprecedented growth. But Moore's law is not really a law, but a statement about the rate of technological progress for computers. And if one thinks for a while, one begins to wonder when Moore's law will end (and perhaps, what will happen to your job when this happens!?) If we blindly extrapolate Moore's law into the future we see that somewhere around 2050, the feature size of computers would need to be the size of an atom. The size of an atom! Which makes us wonder (1) whether Moore's law will be able to reach the atomic size? and (2) what happens when we get to machines that are so small?

Of course we don't know the answer to either of these questions because foresight isn't 20/20, but we can, as engineers, physicists, chemists, and material scientists make some rough guesses about (1). What we know is that even today we can construct transistors which are molecular sized. Now these aren't very reliable, and are far from capable of being used in any sort of modern fab plant, on the other hand, they do indicate to us that there is a real possibility of designing computers whose individual components are a few atoms in size. As a side note, you might be wondering why should computers stop at the size of an atom. Well certainly I don't know the answer to this: it is conceivable that one could engineer nuclear matter such that it functions as a computer. However, I don't know how to do this without accessing energies which are considerably higher than the energies we find in atomic systems (atomic energies are in the range of electron Volts or smaller: remember the electron volt is the energy a single electron gets when moving through a volt of potential. Whereas nuclear reactions occur at much higher energies, around a MeV, or a million electron Volts.) So if you can deal with a computer that is a million times hotter and operates due to some bizarre nuclear reactions, then we might find some technology to build nuclear computers out of, but I don't see how to do this right now (and those of you who are easy daydreamers, I hope this early distraction doesn't keep you sidetracked during the rest of my lecture.)

The second question: what happens to computers when they become the size of atoms is also a very good question. One thing which could happen is just that we learn to engineer systems just like our modern computers at these scales. This is certainly the goal of those who design molecular transistors! But there is another possibility, one which will be the subject of this course, and which motivates a lot of interesting speculative thinking about the future of computing. This different possibility arises because of something strange that happens more easily to atomic sized systems than to large macroscopic systems: these atomic systems have the ability to function very precisely under the laws of quantum theory. This is not to say that modern computers don't function according to quantum theory: quantum theory is very essential in understanding the physics behind our modern silicon devices, but the important difference is that when we build devices that are atomic sized we can more easily exploit effects which are purely quantum mechanical in nature. The rough picture you might have is that large objects obey some laws, which we call the classical laws of physics and then as we get to smaller objects, the laws turn into quantum laws. The emergence

of the classical laws from the quantum laws is an interesting subject area in physics. This may seem a bit abstruse at this point. Don't worry, eventually we will sort this out and these sentences I utter will make complete sense (or at least I hope so!)

In the early 80s a few crazy oddballs began to think seriously about what would happen when you decided to build computers out of components which obeyed the laws of quantum physics. These were mostly physicists, among them the famous Nobel prize winning physicist Richard Feynman. What these physicists began to think about was how to build a computer which operating according to the laws of quantum theory. Part of the motivation of this was sparked by previous results about "reversible" computation. Reversible computers are basically computers which you can run backwards: each step the computer makes in the forward direction in time can be used to run the machine backwards. For example, the process which flips a single bit, $0 \rightarrow 1$, $1 \rightarrow 0$, is reversible (simply flip the value back), but the process which sets the bit to 0, $0 \rightarrow 0$, $1 \rightarrow 0$, is not reversible because we don't know whether to return the 0 to a 0 or a 1. Physicists became interested in reversible computers because there is a sense in which these computers do not dissipate heat. What the physicists who began to think about computers which operate according to quantum theory were doing was extending this type of work to the quantum realm. We'll learn that reversibility has an analogy in quantum theory which we call unitary and this motivated physicists to think about computers which evolved according to unitary evolutions. These computers were called quantum computers.

For a decade after people like Richard Feynman, Paul Benioff, and David Deutsch began to ponder quantum computers, not a ton happened in the field (I'm neglecting quantum cryptography at this point and the beginnings of quantum information, and instead focussing the idea of computing using quantum theory.) However, important seeds were planted in this time for some amazing results. Feynman pointed out that simulating quantum computers using a standard modern computer was very difficult. Physicists were very familiar with this problem because, ever since the earliest days of computing they had been using computers to simulate physical systems. And simulating quantum systems was always particularly hard. So, if you take Feynman's observation a bit seriously, that quantum computers are hard to simulate, does this mean that quantum computers can do things more efficiently than a classical computer? While Feynman never explicitly stated this idea (at least in the published literature) one can see it lurking around his early papers (sadly he died before the field of quantum computing really took off.) But someone else, the physicist David Deutsch did take serious such a question. Starting in 1985 Deutsch along with others began to think not just about how to build a quantum computer, but what exactly such a computer could compute and how efficiently the computer could do these computations. As often happens in science, a slow set of results began to trickle from a few brains.

These early results were all query complexity results, so I'll just briefly tell you want a query complexity result is (we will return to this latter.) Let f be a function, which for simplicity we will say takes as input a n bit number. We might imagine that this function is computed by some machine. In the classical world we can query this function by inputting some value to the machine and getting out the value of the function. This we count as a single query to the function. Now we may wish to determine some property of a function. For example we might want to know whether the function is always equal to zero, or some other such property. The query complexity of this problem is then the number of times that we have to query the function in order to solve this problem. In particular we are concerned with how the query complexity (number of queries) grows as a function of the size of the input. Further it is useful to note that we often deal with promise problems where the set of possible functions for the problem is not all possible functions but some restricted set. What the early research in quantum computers focused on was asking the question: what if the machine we are querying is a machine which operates according to quantum theory? If we require that the quantum machine, when it is used as a classical device, computes the function f, then we can ask the question of whether by exploiting the fact that we can operate the machine in a fully quantum mode we can perform less queries to the machine.

The first results along these lines were obtained by David Deutsch and Richard Jozsa. They showed that there is a property of a set of functions (not all functions: we call this restriction a promise problem) which took around 2^n queries to solve classically (where n is the number of bits inputted into f) but when you querried the function quantum mechanically you needed only a single query to solve the problem! Unfortunately, the problem these two considered was not robust under allowing probabilities of failure. In particular the problem could be solved using a few queries using a classical computer if one accepted some small probability of failing to solve the problem. The next big step was then taken by Ethan Bernstein and Umesh Vazirani. These two showed that there were promise problems for functions whose quantum query complexity was superpolynomially better than the corresponding classical query complexity, even when one allowed some probability of error. Superpolynomial means that the number of queries grows faster than a polynomial (but not necessarily exponential). Dan Simon, who is across the lake at Microsoft Research, then showed a problem for which the separation is exponential. Thus there were problems for which the query complexity of the quantum protocol was polynomial in n , but for which the classical protocol needed at least exponential in n. If your going to do something and you get an exponential speedup in the number of times you need to query a function to do that something, you're doing pretty good. But, it must be said, that even with these

nice results, at that point in time the problems these researchers were focussing on didn't really have any practical applications.

But then, in 1994, a bomb was dropped. In 1994 Peter Shor announced quantum computers could efficiently factor numbers and efficiently solve the discrete log problem. Many of you probably know that the RSA public key cryptography routine can be broken if you can efficiently factor numbers. No one knows a good classical algorithm for efficiently factoring numbers and this allows the RSA public key cryptography routine to be secure in the sense that there is no computer today (or if you make the key long enough, no conceivable computer) which could efficiently break the routine. But if quantum computers could efficiently factor numbers, then they could break RSA. And if they could break RSA, they would destroy the security that is most widely used to protect all sorts of important transfers of information. Needless to say, this sparked a huge amount of interesting in quantum computers. Especially among agencies with three letter acronyms.

This course owes its existence, in large part to Peter Shor's discovery. Following Shor's discovery, a large number of researchers began to think seriously about the implications of manipulating information in a quantum theoretic manner. The name of the new field which arose is "Quantum Information Science." People who work in "Quantum Information Science" come from all walks of scientific life–they are physicists, computer scientists, engineers, chemists, and mathematicians (but so far, really no biologists)–and the work spans a vast territory of research, from those working to build quantum devices to theoretical computer scientists designing new algorithms for the machines. Needless to say, to become an expert in the entire field of quantum information task is now, a little over a decade after its founding, a difficult task. In this class I hope to give you a glimpse of most of the main results in quantum information science. So you won't become an expert, but you'll understand what these main results are, why they matter, and a little of what might be possible in the future.

We'll begin the course by learning quantum physics. Don't worry this just sounds hard: we will take a viewpoint of only discussing quantum machines and therefore we don't really need to learn any physics, just the "quantum" stuff. Then we'll use this to discuss what a quantum computer is and begin to see that it is indeed something a little like a probabilistic computer. Having begun to assure ourselves that we're not chasing bogeyman, we'll then begin to discuss quantum algorithms, up to and including Peter Shor's famous algorithm. We'll also discuss another set of algorithms, developed by Lov Grover and also a little bit about how we can use quantum computers to solve the problem Feynman worried about: simulating quantum systems. Next we'll move to a subject area which has been around a lot longer than quantum information science, but which is, we are starting to find out, closely related to the reason that quantum computers can outperform classical computers: quantum entanglement. Quantum entanglement is such a strange and wonderful property that physicists have been avoiding thinking about it for a long long time. This is kind of sad, because if they had begun to think about entanglement and knew a little computer science they might have begun to discover quantum information science years ago. But they didn't and they have only their stubbornness to blame. When we study entanglement this will lead us naturally to studying communication complexity. Communication complexity is one of the more beautiful subdisciplines of theoretical computer science. I say this because in communication complexity we can often prove rigorous upper and lower bounds on problems and the theory behind doing this is very beautiful. We won't have time to go into this too deeply (there could be an entire course on this material alone), but we will see that when we talk about quantum communication complexity we again find that quantum systems outperform classical systems. Having motivated that building a quantum computer might be an interesting thing to do, we'll next delve into how you might actually build a quantum computer. In particular we will discuss the problems that make building a quantum computer a challenge. This will lead us naturally to perhaps the second biggest discovery after Peter Shor's algorithm: the theory of quantum error correction. Using the methods developed in quantum error correction, we'll show that building a real quantum computer is physically possible. Finally, having showed that we might possibly be able to build a computer which can break the RSA public key cryptosystem, we'll discuss how how quantum theory also allows us to build new types of cryptography.

A. Reasons for Getting to the End of this Class

Finally, before closing this introductory section, I'd like to list some of the exciting reasons for studying quantum computing coming from today's research. In this course you will learn quantum computing *lore*: what we already understand. But the real point of a course like this one, one which is on the boundary of a developing field of research, is to prepare yourself for understanding or participating in the future of the field. So here I'd like to list some interesting directions which quantum information science is currently taking. The details of these directions are beyond the scope of this course, but I hope they will serve to motivate *why* it is important to go through the pain of this class and what those who work in the field of quantum computing are most excited about. Further note that this list is my own personal list. There are certainly all sorts of other interesting directions occurring in quantum information science, but the things I've listed below are from the more speculative and, I think, exciting side of the

field.

The power of quantum computers and simulating physics As I have described above, one discovery of quantum computation has been that quantum computers appear to be more computationally powerful than classical computers. A question which people have been asking about this is "why" are quantum computers more powerful than classical computers. Recently a lot of progress has been made on this question and a consensus of sorts has been reached that entanglement is of vital importance in the computational speedups achieved by quantum computers. In particular it was realized that if a quantum system does not have a lot of entanglement then it would be easy to simulate this quantum system. This simple observation, motivated by studying quantum computation is now taking on a life of its own for those who are interested in studying many-body quantum systems (read: condensed matter theorists.) In particular, it has long been very difficult to simulate the low energy behavior of many-body quantum systems in two and higher dimensions. For example, a method called density matrix renormalization group, while working spectacularly for one dimensional quantum systems, has not been effective in higher dimensional quantum systems. What was realized by those in quantum information processing was that the reason for this was that in two dimensions, the density matrix renormalization group methods were not properly taking into account the scaling of the entanglement in these systems. By understanding the role of entanglement in these systems physicists are now busy writing algorithms which are efficient in two and higher dimensions. By understanding when quantum computers are not powerful, we can are learning about how to efficiently write classical algorithms for certain quantum systems!

Assume you have a quantum computer It is often remarked upon by mathematicians and physicists that there is a mysterious harmony between these two disciplines. Indeed both fields have benefitted immensely from each other. The same can not be said, I think, for computer science. A good question to ask is why this is true? One idea floating around in research land is that the reason for the lack of such deep connections has been that this is because the wrong model of computation has been studied. It makes sense that if the universe obeys the laws of quantum theory, then we shouldn't be surprised if a theory of computation based on classical theory doesn't make connection to the physics of these devices. And going one further step it also seems likely that if we use the theory of computation which is more relevant to the way our world works, then we just might be able to prove and reason about these computers in a simpler manner. A rough analogy might be that between real analysis and complex analysis: certainly those of you who have had a class on real analysis and complex analysis remember how "easier" everything is in complex analysis than in real analysis. Might quantum computers lead to a similar shift? Right now there are only a few results that I can point to along these directions, for example Scott Aaronson's has a very eloquent proof that the complexity class PP is closed under intersection which begins by first showing that PP is equal to a model of quantum computing in which one can post-selection the outcome of the quantum computation and then is extremely simple due to the properties of the quantum model of computation. Might not more of the theory of computational complexity be as easy if we focus on the model of quantum computation?

Quantum algorithms Although we will see that quantum computers can solve certain problems faster than classical computers, for exactly what tasks this can be achieved is a very open field of investigation. On the one hand, quantum computing is very spoiled by Shor's factoring algorithm. On the other hand, factoring efficiently is such a *big* result and all other progress in coming up with new quantum algorithms is measured in comparison to Shor's result. Thus there has been a severe disincentive to work on new quantum algorithms and the number of researchers working explicitly on new quantum algorithms is extremely small. This is beginning to change, but certainly if there is one direction of quantum computing that will serve the biggest research dividends, it is in the discovery of new quantum algorithms.

Quantum algorithms and the ultimate limits of computation If today someone was to prove that P does not equal NP for a classical computer, would we be satisfied? Well certainly we would be very excited and this would be the breakthrough of the century in computation, but because the universe is fundamentally quantum mechanical, would we really be satisfied that the intractability of NP-complete problems had been shown? Quantum computers opens up an entire bag of worrying about the foundations of computational complexity. It is dangerous to say this, of course: if this view is correct, then the hard work of classical computational theory might have been in vain. But if this is the correct view, then we need to begin weaning ourselves off of the classical model of computation.

New many-body quantum physics One of the things we will study in this class is the theory of fault-tolerant quantum computing. The remarkable result of this theory is that quantum computers are not analog computers (at least in the traditional definitions of such computers.) Indeed whereas the exceptional power of analog computers is not realizable because of real world issues of noise, the real world power of quantum computers

does not appear to suffer from similar problems. This means, remarkably again, that it is possible to robustly store coherent quantum information in the state of many thousand and even millions of quantum systems. But given this observation we can ask the question of whether there already exist or we can engineer many-body quantum systems which are able to robustly store quantum information. Surprisingly this idea, originally due to Kitaev, has led to fascinating results in quantum computing using exotic quasi-particles called non-abelian anyons. While ideas about such quasiparticles predate quantum computing, the focus of the quantum computing community on *engineering* such systems, along with the increased control of quantum systems achieved by quantum computing experimentalists has generated the exciting hope that a new era of quantum materials might be possible.

Quantum theory from quantum information A final strand of interesting research which is being carried out today concerns the interplay between the foundations of quantum theory and quantum information science proper. In particular, as you will see, quantum theory when viewed through the lens of information science has some very peculiar characteristics. Physicists and philosophers have long pondered the "meaning" of quantum theory and noted its remarkable harmony with the special theory of relativity. Recently there has been a growing movement to attempt to derive quantum theory from solely (or mostly) information theoretic considerations. Thus, just as it is possible to derive the deep consequences of the theory of special relativity from a few simple hypothesis, there is now a growing hope to derive quantum theory for a similar set of more self-evident postulates. For example, Robert Spekkens has developed a toy model of information processing which reproduces many of the properties of quantum theory but is derived from very simple postulates. Extending such a model to reproduce all of quantum theory is a current topic of great interest. Further, even if this task is not successful, it is motivating tremendous understanding of the resource trade-offs which occur in information processing in quantum and classical systems.

III. QUANTUM THEORY MADE EASY

Quantum theory is often presented as an abstruse and difficult theory. This isn't helped much by the legions of fine minds who have slandered quantum theory. The Nobel prize winning physicist Richard Feynman (Feynman will make more than one appearance in this class!) once said of the quantum theory "I think I can safely say that nobody understands quantum mechanics." But this is really a bit of hogwash. Certainly Feynman understood quantum theory perfectly well: for how else could he perform the quantum mechanical calculations upon which his fame so rests?! In fact, I will venture to say that learning quantum theory isn't that much more difficult than learning about the basic laws of probability. Quantum theory is not *difficult* per se, as much as it is *different*. In this lecture, I want to introduce to you, in as painless a way as possible the basics of quantum theory. This will involve drawing analogies between quantum theory and the evolution of a probabilistic computing device.

So what, anyway, is quantum theory? To put it in computer science terms, quantum theory is the machine language of the universe. In other words, all programs about the evolution of the universe are written in the lowest level language known as quantum theory. To put this in more concrete terms, you may have heard that there are four known fundamental forces in the universe: the electromagnetic force, the weak force (which governs some radioactive decays), the strong force (which is responsible for holding nuclei together), and the gravitational force. What physicists have discovered over the past hundred years is that three of these four forces obey the laws of quantum theory (and a big part of modern theoretical physics is spent trying to put that forth nasty force, gravity, into this same context.) One may as well think that these three forces form three different higher level languages, each of which is being executed by a universe which operates according to quantum theory. Physicists don't like to call these higher level languages, so they make up nice fancy names for them like quantum electrodynamics and quantum chromodynamics. But the main point is that in today's view of physics, there is quantum theory and then *on top* of the edifice of quantum theory one places a physical theory like electrodynamics or chromodynamics. Quantum theory is the base upon which we rest physical theories. In this sort of view of the universe, if quantum theory is the machine code, and the physical theories are the higher level languages, what are the programs? A physicist would say that the program is the initial conditions one sets up, with particles and fields and such in certain configurations. In a similar manner the program stored on your hard drive is the initial configuration which, when executed by your computer enacts a series of changes in the machine.

Okay, enough fillibustering about the spiritual configuration of quantum theory within the laws of the universe. Let's talk about what, exactly, these quantum rules of the universe are! To do this, we will begin by talking about a very simple classical information processing device.

A. A Probabilistic Information Processing Device

There are many different models of what, exactly, a classical computer is: a Turing machine, the lambda Calculus, the von Neumann architecture which forms in some ways the basis of our modern computer architecture, etc. We're going to begin by thinking about a system which is so simple, that you might even argue that it isn't really a computer. The system we will talk about is simply a system with some internal degrees of freedom, which we will call a memory, and rules for how this memory changes in time. This is a very barebones setup for a machine, so maybe we'll just call it an information processing device instead of a computer.

The device we are going to talk about has some very basic rules. We're going to write them in such a way that when we begin to talk about the quantum rules of a similar device, a quantum information processing device, the translation between the two notations should be fairly painless.

Rule 1: (Configuration Description) The internal memory of our system is described by a finite alphabet of symbols. These symbols label the different *configurations* of the machine. Our machine can be in one and only one configuration at any time. But, while the machine can only be in one configuration at one time, we may also want to talk about our description of the configuration of the machine. Our description of the configuration of the machine may be that the machine is in a particular configuration. But it may also be a configurationment like, "there is a fifty percent chance the system is in configuration 0 and a fifty percent chance the system is in the configuration 1". It is important, here, to not confuse our description of the configuration of the machine with the configuration of the machine itself. This will become important when we make the transition to a quantum information processing device.

So in general our description of the information processing system will be a set of probabilities that the machine will be in one of these particular configurations at a given time. We will denote these probabilities by a vector of probabilities (which we call a vector of probabilities or a probability vector interchangeably.) Thus, for example, we can choose some (arbitrary) order of the N configurations of the machine and call them 0, 1, etc. up to $N-1$. Then we can form a vector with its ith component corresponding to the probability of finding the system in the ith configuration. We will denote this by a column vector for good measure:

$$\begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{N-1} \end{bmatrix} \tag{1}$$

These are probabilities so they they must be positive, $p_i \geq 0$ and they must sum to unity $\sum_i p_i = 1$.

Example: our description of a three configuration system as being in configuration 0 with probability 30 percent, in configuration 1 with probability 70 percent, and in configuration 2 with 0 probability is the column vector

$$\begin{bmatrix} 0.3 \\ 0.7 \\ 0 \end{bmatrix}. \tag{2}$$

Rule 2: (Evolution) Whereas Rule 1 described the configurations and the description of the information processing machine, rule 2 has to do with the way in which our description of the configurations evolve. As I mentioned earlier, the model we want to talk about is the model of a probabilistic information processing device. One way of saying this is that given that the system is in the configuration i, the system in a given time step can evolve into any of the other configurations with a certain probability. To describe such a probabilistic evolution, we therefore need to specify these probabilities. If the system is in the configuration i, then we can specify the probability that the system is in the configuration j by the numbers $A_{j,i}$. Since these are probabilities, each of these numbers must be positive $A_{j,i} \geq 0$ and the sum over all the outcomes must be unity: $\sum_{j=0}^{N-1} A_{j,i} = 1$. For a fixed i there must be N of these numbers (where N is the number of configurations of our machine). So there are a total of N^2 such numbers. Now how does our description of the configuration evolve when we undergo such a probabilistic evolution? Let p be the vector of N probabilities with components p_i. Then the new probability vector, q, describing the configuration of the system has a probability of being in the configuration j given by

$$q_j = \sum_{i=1}^{N} A_{j,i} p_i. \tag{3}$$

Remember that we can think of p as a column vector. Similarly we can think of q, the new probability vector, as a column vector. Then we see that the above formula is just an expression of matrix multiplication:

$$q = Ap \tag{4}$$

where A is the matrix with components $A_{j,i}$. The properties of that this matrix must satisfy, $A_{i,j} \geq 0$ and $\sum_{j=1}^{N} A_{j,i} = 1$, define what is called a *stochastic* matrix.

So now we can describe Rule 2. The evolution of one step of our probabilistic information processing device is described by a stochastic matrix of dimension equal to the size of our alphabet of configurations of the system. If our description of system is given by a vector of probabilities p, then at the next time step, our description of the system has a new vector of probabilities, q given by Ap.

Example: Suppose that we have a two configuration machine which is described by a probability vector $p = [0.4\ 0.6]^T$ (T denotes transpose to make it a column vector). Suppose the evolution of the machine is such that, if the machine is in the configuration 0 it stays the same with probability 0.4 and it flips to 1 with probability 0.6 and if the machine is in the configuration 1 it stays there. Then the matrix describing this is given by

$$A = \begin{bmatrix} 0.4 & 0 \\ 0.6 & 1 \end{bmatrix} \tag{5}$$

and the configuration after this evolution is given by

$$Ap = \begin{bmatrix} 0.4 & 0 \\ 0.6 & 1 \end{bmatrix} \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 0.16 \\ 0.84 \end{bmatrix} \tag{6}$$

Rule 3: (Measurement) Now we are going to describe what happens when we "measure" our probabilistic computing device. Like we said before, for our machine, the system is always in some particular configuration, but our description of the system, is just a vector of probabilities. So given a description of our system in terms of the vector of probabilities we can calculate the probability that the system is in a particular configuration. Immediately after such a measurement, our description of the probabilities will "collapse" onto whatever outcome we measure. So our version of the measurement rule will be that if the system is in the configuration with probability vector p, the probability of a measurement finding the system in the configuration i is p_i. After the measurement, if we find the system in configuration i, then the new description of the system is given by a vector of probabilities q with $q_j = \delta_{j,i}$ (where $\delta_{j,i} = 1$ if $i = j$ and 0 otherwise.)

Rule 4: (Composite systems) The final rule is a bit hard to motivate for our probabilistic information processing device. It talks about what happens when we bring two of these devices together. Suppose we have two machines with N and M configurations. We may allow these machines to communicate back and forth and do all sorts of manipulations on their configurations. In fact, we can think about taking these two devices and making a new machine which is exactly like a probabilistic information processing device, but now with NM configurations. The fourth rule is all about the relationship between these original machines and the new machine. First it says that our description of the new machines is given by a probability vector in the *tensor* product of $\mathbb{R}^N \times \mathbb{R}^M$. What is the tensor product? Suppose you have a set of basis vectors for $\mathbb{R}^N$, e_i, $i = 1, \ldots, N$, and a set of basis vector for $\mathbb{R}^M$, f_i, $i = 1, \ldots, M$. Then the tensor product of these two vector spaces has a basis $e_i \otimes f_j$. So a probability vector in the new tensor product space is a vector with components in some basis which have two indices: $v = \sum_{i=1}^{N} \sum_{j=1}^{M} p_{ij} e_i \otimes f_j$.

Example: Suppose we have two machines, each which has two configurations and have the descriptions p and q. Then the description of the combined machine is

$$p \otimes q = \begin{bmatrix} p_1 \\ 1 - p_1 \end{bmatrix} \otimes \begin{bmatrix} q_1 \\ 1 - q_1 \end{bmatrix} = \begin{bmatrix} p_1 q_1 \\ p_1(1 - q_1) \\ (1 - p_1) q_1 \\ (1 - p_1)(1 - q_1) \end{bmatrix} \tag{7}$$

A word here about the labelling we have used. If p_1 is associated with the configuration 0 for the first machine and q_1 is associated with the second machine, then we see that for the tensor product we have labelled the four possible basis vectors in a natural fashion:

$$\begin{bmatrix} p_1 q_1 \\ p_1(1 - q_1) \\ (1 - p_1) q_1 \\ (1 - p_1)(1 - q_1) \end{bmatrix} \begin{matrix} 00 \\ 01 \\ 10 \\ 11 \end{matrix}. \tag{8}$$

We will be expressing the components of a tensor product space in this manner all the time, so it's best to get used to it. Also note that there are descriptions of the configuration of the new combined machine which are not expressible as a separable product $p \otimes q$. For example the description may be that we know the configuration of the combined

system is either 00 with probability $\frac{1}{2}$ or 11 with probability $\frac{1}{2}$. This configuration is denoted by

$$\begin{bmatrix} \frac{1}{2} \\ 0 \\ 0 \\ \frac{1}{2} \end{bmatrix} \tag{9}$$

Play around a bit and you will see that there is no way to express this as $p \otimes q$: we would say that the our description of the machine is such that the machine is correlated. Later we will see a similar effect in quantum theory and there, this will turn out to be one of the strangest effects every uncovered about nature.

Finally note that we can also define the tensor product for matrices which act on our tensor product of vector spaces. For example, we might think about a matrix which operates on the first machine with the action A and the second machine with the action B. We will denote this as $A \otimes B$. Its fun to get used to this notation. Suppose that

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \tag{10}$$

Then in the basis where our configurations are ordered 00, 01, 10, and 11, the tensor product of these two matrices is given by

$$A \otimes B = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{bmatrix} \tag{11}$$

Ack, big and ugly. But we'll be doing lots of this tensor producting, so it's best to get used to it.

So rule four was all about what happens when we build a new machine out of two smaller machine. We found that the description of the configuration lived in the tensor product of the two vector spaces for the original machine.

B. A Quantum Information Processing Device

Now we turn to the quantum analogy of the classical information processing device we described in the last subsection. Not surprisingly we are again going to have four rules (the game was rigged!)

Rule 1: (Configuration Description) For the classical device we first began by talking about the configuration of the system and then about a description of the system. The configuration of our quantum information processing device will again be one of a finite number of configurations. We will say there are N of these configurations. However, we will no long be describing our system by a set of probabilities! In quantum theory our description of the configuration of our system is no longer described by a vector of probabilities, but instead our description is given by a vector of *amplitudes*. More specifically we will describe our system with N configurations by a unit vector in the space $\mathbb{C}^N$. OK, lets take this one step at a time. What is $\mathbb{C}$? $\mathbb{C}$ are the complex numbers $a + bi$, $a, b \in \mathbb{R}$ and $i^2 = -1$. $\mathbb{C}^N$ is a vector of N complex numbers. So a description of the configuration of our machine, call it v is a vector of N complex numbers which can denote by, say, a column vector $v = [v_1 \ v_2 \ \ldots \ v_N]^T$, $v_i \in \mathbb{C}$. Now what does it mean that this vector is a unit vector? It means that the inner product of this vector with itself is one: $\langle v, v \rangle = 1$, or, in component notation $\sum_i v_i^* v_i = \sum_i |v_i|^2 = 1$. (Recall that the complex conjugate of a complex number $a + bi$ is $(a + bi)^* = a - bi$. The inner product is like the dot product except that we conjugate one set of components as done in our formula. We write a number times its complex conjugate as $|v|^2 = v^* v$.) Here is easy guide to the correspondences between the quantum machine and the classical machine

Classical		Quantum
N configurations	$\rightarrow$	N configurations
vector of probabilities	$\rightarrow$	vector of amplitudes
$p \in \mathbb{R}^N$	$\rightarrow$	$v \in \mathbb{C}^N$
$\sum_{i=1}^N p_i = 1$	$\rightarrow$	$\sum_{i=1}^N \lvert v_i \rvert^2 = 1$

At this point you may wonder whether I've gone crazy. But no, I haven't. You may ask yourself why in the world should our description of the machine be given by these crazy amplitude things and not by probabilities? ("You may ask yourself, how did I get here? You may ask yourself, this is not my beautiful wife") And to this I can only shrug

my shoulders and say "no one knows!" But apparently nature does not allow us to describe it by probabilities, but instead by these funny amplitudes.

Now it is nice to jump the gun here and describe Rule 3 which describes measurements. In particular if our machine is described by the vector v, then the probability that we observe the system in the configuration i is given by $|v_i|^2$. Thus we see in our table above that the prescription that v is a unit vector is really the requirement that these numbers are probabilities (also note that $|v_i|^2$ is always positive: $|a+bi|^2 = a^2+b^2 \geq 0$.)

So rule 1 is that the configuration of our system is one of N finite configurations and our description of this configuration is given by a unit vector of N complex amplitudes. We don't call these vectors of amplitudes, but we usually call this description the wave function or a quantum state. The reason we call it the wave function is that the original formulations of quantum theory dealt with cases unlike the case I'm describing to you where we have a finite number of configurations, but instead where our configuration is a continuous variable. Like, say, the position of a particle. Then the amplitudes are a function of the position and the equation which governs how this function evolves is a wave equation. So even though we aren't at the ocean and have only a finite number of configurations, we'll still call this thing a wave function. Actually what we will usually do is call the wave function the quantum state. This is kind of bad because the "state" of a system can mean its "configuration" but we'll just have to live with this.

Example: Suppose our system has two configurations 0 and 1. A possible description of the configuration of our system is given by the wave function

$$v = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{bmatrix} \tag{12}$$

We can check that this indeed is a unit vector $|v|^2 = |\frac{1}{\sqrt{2}}|^2 + |\frac{i}{\sqrt{2}}|^2 = 1$. Further, if we measure this system we will find that the probability of finding it in either configuration is $\frac{1}{2}$.

Just to recap, in rule 1 we have seen that again there are N configurations to our machine and our description of the configuration of our quantum machine is given by a wave funciton. This wave function is an element of a complex vector space with an inner product with the property that the wavefunction is of unit length.

Rule 2: (Evolution) We described evolution of our description of the configuration of our system in the probabilistic machine via a stochastic matrix. What happens for our quantum machine? What happens is that the requirement that the matrix is stochastic is changed to the requirement that the matrix is unitary. So again if we have an N configuration system, then our wave function will be described by a unitary matrix of size $N \times N$. Thus, if our vector is original v and then we evolve due to the unitary matrix U, then the new vector is $v' = Uv$: or in terms of components

$$v'_j = \sum_{i=1}^{N} U_{ji} v_i. \tag{13}$$

Now the requirement for our probabilistic machine was that the evolution matrix be stochastic. We did this because we required that the system respect the fact that we were evolving probability vectors. But now we are evolving these crazy amplitudes. But the complex vectors we evolve were required to be unit vectors. We don't want this to change under the evolution, and this leads to the requirement that our evolution matrix must be unitary (actually I'm fibbing a bit here, it could also be anti-unitary, but we'll leave out this possibility right now.) What is a unitary matrix? It is a matrix whose conjugate transpose is it's inverse. We write this as $U^\dagger U = I$. Let's take this one symbol at a time. What is the $\dagger$? It is the operation of complex conjugation and transpose. Thus if U has elements U_{ij} at row i and column j, then the elements of $U^\dagger$ have elements U^*_{ji} at row i and column j. Next what is I? I is the $N \times N$ identity matrix: the matrix with all 1s on its diagonal and all zeros elsewhere. So the requirement that U is unitary is the equation

$$\sum_{k=1}^{N} U^*_{ki} U_{kj} = \delta_{ij} \tag{14}$$

Example: Here is a unitary matrix:

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{i}{\sqrt{2}} & \frac{i}{\sqrt{2}} \end{bmatrix} \tag{15}$$

How do we know it is unitary? It's conjugate is

$$U^* = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} & -\frac{i}{\sqrt{2}} \end{bmatrix} \tag{16}$$

and the conjugate transpose is thus

$$U^\dagger = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{i}{\sqrt{2}} \end{bmatrix} \tag{17}$$

Matrix multiplication then shows that

$$\begin{aligned} U^\dagger U &= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{i}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{i}{\sqrt{2}} & \frac{i}{\sqrt{2}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}\frac{-i}{\sqrt{2}} & \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}\frac{i}{\sqrt{2}} \\ \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}} + \frac{-i}{\sqrt{2}}\frac{-i}{\sqrt{2}} & \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}} + \frac{-i}{\sqrt{2}}\frac{i}{\sqrt{2}} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I \end{aligned} \tag{18}$$

Suppose that our system has the wave function $v = [\frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}}]^T$. Now suppose that we evolve this wave function according to the above unitary evolution. Then the new wave function will be given by

$$\begin{aligned} Uv &= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{i}{\sqrt{2}} & \frac{i}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}} \\ \frac{-i}{\sqrt{2}}\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{aligned} \tag{19}$$

Notice that if we had measured the system as we describe in rule 1, then if we had measured the system before the evolution by U, the system would have a $\frac{1}{2}$ probability of being found in the 0 configuration and a $\frac{1}{2}$ probability of being found in the 1 configuration. If, however, we measure the system we have evolved by U, then there is a hundred percent probability that we find the system in the configuration 0.

Rule 3: (Measurement) As we described above, when we measure our quantum system which is described by the quantum state v, we find the configuration i with probability $|v_i|^2$. How do we update our description of the system after this measurement? If we made a measurement and found configuration i, then the new amplitudes of the quantum state satisfy $v'_j = \delta_{i,j}$: i.e. there is only one non-zero amplitude and it is equal to unity.

One thing to keep straight in quantum theory, and this is why I've presented things this way, is that evolving a system in time is a physical process and also measurement is a physical process. By this I mean that there is a certain configuration of physical systems which we arrange to carry out a unitary evolution and a different configuration which we arrange to perform a measurement. Now it is possible to treat both of these in a unified matter and we will do this later, but for now, thinking about them separately is very useful.

One final note about measurement is that actually what we've described here is usually called "measurement in the computational basis." In a bit you'll begin to understand why we call it this and what measurement in other basis than the "computational basis" means.

Example: Suppose our quantum device has a quantum state $v = \left[\frac{\sqrt{3}}{2} \ \frac{1}{2}\right]^T$. The probability of measuring the system and finding it in configuration 0 is $|\frac{\sqrt{3}}{2}|^2$. If this happens, then the new state of the system is $v' = [1 \ 0]^T$

Rule 4: (Combining systems) The rule for combining our quantum devices is nearly identical for what happens when we combining our classical devices.

Suppose we have two machines with N and M configurations. We may allow these machines to communicate back and forth and do all sorts of manipulations on their configurations. In fact, we can think about taking these two devices and making a new machine which is exactly like a quantum information processing device, but now with NM configurations. The fourth rule is all about the relationship between these original machines and the new machine. First it says that our description of the new machines is given by an amplitude vector in the *tensor* product of $\mathbb{C}^N \times \mathbb{C}^M$. What is the this tensor product? Suppose you have a set of basis vectors for $\mathbb{C}^N$, e_i, $i = 1, \ldots, N$, and a set of basis vector for $\mathbb{C}^M$, f_i, $i = 1, \ldots, M$. Then the tensor product of these two vector spaces has a basis $e_i \otimes f_j$. So an amplitude vector in the new tensor product space is a vector with components in some basis which have two indices: $v = \sum_{i=1}^N \sum_{j=1}^M v_{ij} e_i \otimes f_j$.

Tensor products are, like we saw for the classical computing device, a bit tricky at first. But with a little playing around (translation: with a little homework) you'll begin to get the hang of them and they won't both you any more.