Learning Objectives:

- Main aspects in which quantum mechanics differs from local deterministic theories.
- Model of quantum circuits, or equivalently QNAND programs
- Simon's Algorithm: an example of potential exponential speedup using quantum computers that predated and inspired Shor's factoring algorithm.

23

Quantum computing

> *"We always have had (secret, secret, close the doors!) ... a great deal of difficulty in understanding the world view that quantum mechanics represents ... It has not yet become obvious to me that there's no real problem. ... Can I learn anything from asking this question about computers–about this may or may not be mystery as to what the world view of quantum mechanics is?"* , Richard Feynman, 1981

> *"The only difference between a probabilistic classical world and the equations of the quantum world is that somehow or other it appears as if the probabilities would have to go negative"*, Richard Feynman, 1981

There were two schools of natural philosophy in ancient Greece. *Aristotle* believed that objects have an *essence* that explains their behavior, and a theory of the natural world has to refer to the *reasons* (or "final cause" to use Aristotle's language) as to why they exhibit certain phenonmena. *Democritus* believed in a purely mechanistic explanation of the world. In his view, the universe was ultimately composed of elementary particles (or *Atoms*) and our observed phenomena arise from the interactions between these particles according to some local rules. Modern science (arguably starting with Newton) has embraced Democritus' point of view, of a mechanistic or "clockwork" universe of particles and forces acting upon them.

While the classification of particles and forces evolved with time, to a large extent the "big picture" has not changed from Newton till Einstein. In particular it was held as an axiom that if we knew fully the current *state* of the universe (i.e., the particles and their properties

such as location and velocity) then we could predict its future state at any point in time. In computational language, in all these theories the state of a system with n particles could be stored in an array of $O(n)$ numbers, and predicting the evolution of the system can be done by running some efficient (e.g., $poly(n)$ time) computation on this array.

Alas, in the beginning of the 20th century, several experimental results were calling into question the "billiard ball" theory of the world. One such experiment is the famous double slit experiment. One way to describe it is as following. Suppose that we buy one of those baseball pitching machines, and aim it at a soft plastic wall. If we shoot baseballs at the wall, then we will dent it. Now, if we use another machine, aimed at a slightly different part of the wall, and interleave between shooting at the wall with both machines, then we will now make *two* dents in the wall. Obviously, we expect the level of "denting" in any particular position of the wall to only be bigger when we shoot at it with two machines than when we shoot at it with one. (See Fig. 23.1)

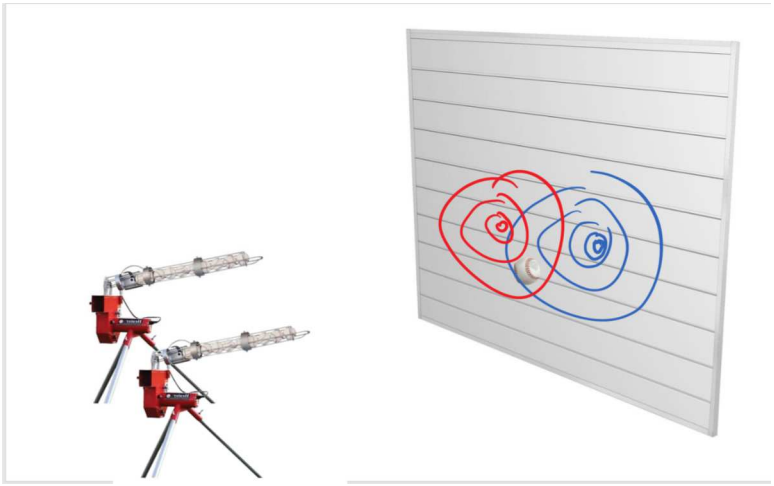

Figure 23.1: In the "double baseball experiment" we shoot baseballs from two guns at a wall. There is only "constructive interference" in the sense that the dent in each position in the wall when both guns operate is the sum of the dents when each gun operates on its own.

The above is (to my knowledge) an accurate description of what happens when we shoot baseballs at a wall. However, this is not the same when we shoot *photons*. Amazingly, if we shoot with two "photon guns" (i.e., lasers) at a wall equipped with photon detectors, then some of the detectors will see *fewer* hits when the two lasers operate than when only one of them does.[1].] In particular there are positions in the wall that are hit when the first gun is turned on, and when the second gone is turned on, but are *not hit at all when both guns are turned on!*. It's almost as if the photons from both guns are

[1] Normally rather than shooting with two lasers, people use a single laser with a barrier between the laser and the detectors that has either one or two *slits* open in it, hence the name "double slit experiemnt", see Fig. 23.2 and Fig. 23.3. The variant of the experiment we describe was first performed by Pfleegor and Mandel in 1967. A nice illustrated description of the double slit experiment appears in this video

aware of each other's existence, and behave differently when they know that in the future a photon would be shot from another gun. (Indeed, we stress that we can modulate the rate of firing so that photons are *not* fired at the same time, and so there is no chance of "collision".)

This and other experiments ultimately forced scientists to accept the following picture of the world. Let us go back to the baseball experiment, and consider a particular position in the wall. Suppose that the probability that a ball shot from the first machine hits that position is p, and the probability that a ball shot from the second machine hits the same position is q. Then, if we shoot N balls out of each gun, we expect that this position will be hit $(p+q)N$ times. In the quantum world, for photons, we have almost the same picture, except that the probabilities can be *negative*. In particular, it can be the case that $p+q=0$, in which case the position would be hit with nonzero probability when gun number one is operating, and with nonzero probability when gun number two is operating, but with zero probability when both of them are operating. If we try to "catch photons in the act" and place detectors right next to the mouth of each gun so we can see exactly the path that the photons took then something even more bizzare happens. The mere fact that we *measure* the path changes the photon's behavior, and now this "destructive interference" pattern is gone and the detector will be hit $p+q$ fraction of the time.

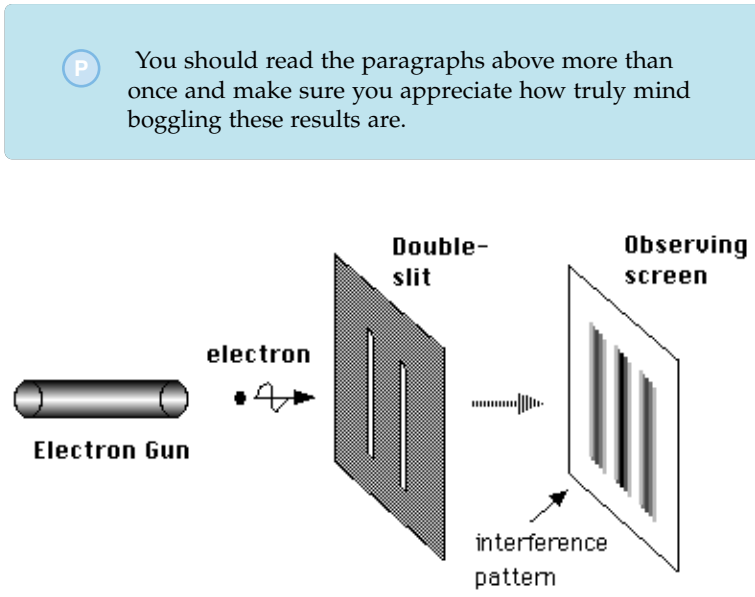

Figure 23.2: The setup of the double slit experiment

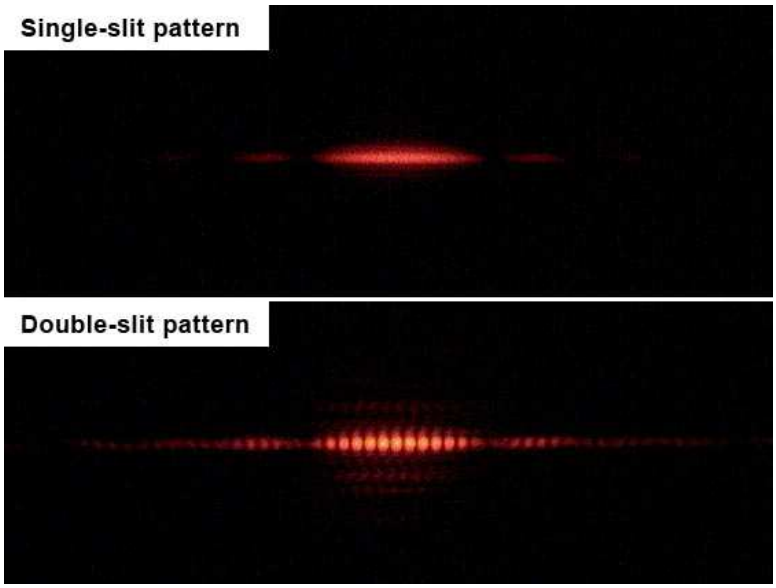

Figure 23.3: Results of the double slit experiement.

23.1 Negative probabilities

What does it mean for a probability to be negative? The physicists' answer is that it does not mean much in isolation, but it can cause *interference* when a positive and negative probability interact. Specifically, let's consider the simplest system of all: one that can be in only one of two states, call them "red" and "blue". (If you have some physics background then you can think of an electron that can be in either "spin up" or "spin down" state.) In classical probability terms, we would model the state of such a system by a pair of two non-negative numbers p, q such that $p + q = 1$. If we observe (or *measure*, to use quantum mechanical parlance), the color of the system, we will see that it is red with probability p and blue with probability q.

In quantum mechanics, we model the state of such a system by a pair of two (potentially negative) numbers α, β such that $\alpha^2 + \beta^2 = 1$. If we measure the color of the system then we will see that it is red with probability α^2 and blue with probability β^2.[2] In isolation, these negative numbers don't matter much, since we anyway square them to obtain probabilities. But as we mention above, the interaction of positive and negative probabilities can result in surprising *cancellations* where somehow combining two scenarios where a system is "blue" with positive probability results in a scenario where it is never blue.

Quantum mechanics is a mathematical theory that allows us to

calculate and predict the results of the double-slit and many other experiments. If you think of quantum as an explanation as to what "really" goes on in the world, it can be rather confusing. However, if you simply "shut up and calculate" then it works amazingly well at predicting the experimental results. In particular, in the double slit experiment, for any position in the wall, we can compute numbers α and β such that photons from the first and second gun hit that position with probabilities α^2 and β^2 respectively. When we activate both guns, the probability that the position will be hit is proportional to $(\alpha + \beta)^2$, and so in particular, if $\alpha = -\beta$ then it will be the case that, despite being hit when *either* gun one or gun two are working, the position is *not hit at all* when they both work. If you haven't seen it before, it may seem like complete nonsense and at this point I'll have to politely point you back to the part where I said we should not question quantum mechanics but simply "shut up and calculate".

23.1.1 Bell's Inequality

There is something weird about quantum mechanics. In 1935 Einstein, Podolsky and Rosen (EPR) tried to pinpoint this issue by highlighting a previously unrealized corollary of this theory. People already understood that the fact that quantum measurement collapses a state to a definite state yields the *uncertainty principle*, whereby if you measure a quantum system in one orthogonal basis, you will not know how it would have measured in an incohrent basis to it (such as position vs. momentum). What EPR showed was that quantum mechanics results in so called "spooky action at a distance" where if you have a system of two particles then measuring one of them would instantenously effect the state of the other. Since this "state" is just a mathematical description, as far as I know the EPR paper was considered to be a thought experiment showing troubling aspects of quantum mechanics, without bearing on experiment. This changed when in 1965 John Bell showed an actual experiment to test the predictions of EPR and hence pit intuitive common sense against the predictions of quantum mechanics. Quantum mechanics won. Nonetheless, since the results of these experiments are so obviously wrong to anyone that has ever sat in an armchair, that there are still a number of Bell denialists arguing that quantum mechanics is wrong in some way.

So, what is this Bell's Inequality? Suppose that Alice and Bob try to convince you they have telepathic ability, and they aim to prove it via the following experiment. Alice and Bob will be in separate closed rooms.[3] You will interrogate Alice and your associate will in-

[2] I should warn that we are making here many simplifications. In particular in quantum mechanics the "probabilities" can actually be *complex* numbers, though essentially all of the power and subtleties of quantum mechanics and quantum computing arise from allowing *negative* real numbers, and the generalization from real to complex numbers is much less important. We will also be focusing on so called "pure" quantum states, and ignore the fact that generally the states of a quantum subsystem are *mixed* states that are a convex combination of pure states and can be described by a so called *density matrix*. This issue does not arise as much in quantum algorithms precisely because the goal is for a quantum computer is to be an isolated system that can evolve to continue to be in a pure state; in real world quantum computers however there will be interference from the outside world that causes the state to become mixed and increase its so called "von Neumann entropy". Fighting this interference and the second law of thermodynamics is much of what the challenge of building quantum computers is all about . More generally, this lecture is not meant to be a complete or accurate description of quantum mechanics, quantum information theory, or quantum computing, but rather just give a sense of the main points where it differs from classical computing.

[3] If you are extremely paranoid about Alice and Bob communicating with one another, you can coordinate with your assistant to perform the experiment exactly at the same time, and make sure that the rooms are sufficiently far apart (e.g., are on two sides of the world, or maybe even one is on the moon and another is on earth) so that Alice and Bob couldn't communicate to each other in time the results of the coin even if they do so at the speed of light.

terrogate Bob. You choose a random bit $x \in \{0,1\}$ and your associate chooses a random $y \in \{0,1\}$. We let a be Alice's response and b be Bob's response. We say that Alice and Bob win this experiment if $a \oplus b = x \wedge y$. In other words, Alice and Bob need to output two bits that *disagree* if $x = y = 1$ and *agree* otherwise.

Now if Alice and Bob are not telepathic, then they need to agree in advance on some strategy. It's not hard for Alice and Bob to succeed with probability $3/4$: just always output the same bit. However, by doing some case analysis, we can show that no matter what strategy they use, Alice and Bob cannot succeed with higher probability than that:

Theorem 23.1 — Bell's Inequality. For every two functions $f, g : \{0,1\} \to \{0,1\}$, $\mathbb{P}_{x,y \in \{0,1\}}[f(x) \oplus g(y) = x \wedge y] \leq 3/4$.

Proof. Since the probability is taken over all four choices of $x, y \in \{0,1\}$, the theorem can only be violated if there exist f, g that satisfy $f(x) \oplus g(y) = x \wedge y$ $(*)$ for every $x, y \in \{0,1\}^2$. In other words, we assume for the sake of contradiction that $f(x) = (x \wedge y) \oplus g(y)$ $(*)$ for every $x, y \in \{0,1\}^2$. So if $y = 0$ it must be that $f(x) = (x \wedge 0) \oplus g(0) = 0 \oplus g(0) = g(0)$ for both $x = 0$ and $x = 1$ and in particular $f(0) = f(1)$. On the other hand, if $y = 1$ then plugging $x = 0$ and $x = 1$ to $(*)$ implies that $f(0) = g(1)$ and $f(1) = 1 \oplus g(1)$. Yet this implies that $f(0) \neq f(1)$, containing a contradiction. ■

An amazing experimentally verified fact is that quantum mechanics allows for telepathy.[4] Specifically, it has been shown that using the weirdness of quantum mechanics, there is in fact a strategy for Alice and Bob to succeed in this game with probability at least 0.8 (see Lemma 23.2).

[4] More accurately, one either has to give up on a "billiard ball type" theory of the universe or believe in telepathy (believe it or not, some scientists went for the latter option).

23.1.2 *Quantum weirdness*

Some of the counterintuitive properties that arise from these negative probabilities include:

- **Interference** - As we see here, probabilities can "cancel each other out".
- **Measurement** - The idea that probabilities are negative as long as "no one is looking" and "collapse" (by squaring them) to positive probabilities when they are *measured* is deeply disturbing. Indeed, people have shown that it can yield to various strange outcomes such as "spooky actions at a distance", where we can measure

an object at one place and instantaously (faster than the speed of light) cause a difference in the results of a measurements in a place far removed. Unfortunately (or fortunately?) these strange outcomes have been confirmed experimentally.

- **Entanglement** - The notion that two parts of the system could be connected in this weird way where measuring one will affect the other is known as *quantum entanglement*.

Again, as counter-intuitive as these concepts are, they have been experimentally confirmed, so we just have to live with them.

> R **More on quantum** The discussion in this lecture is quite brief and somewhat superficial. The chapter on quantum computation in my book with Arora (see draft here) is one relatively short resource that contains essentially everything we discuss here and more. See also this blog post of Aaronson for a high level explanation of Shor's algorithm which ends with links to several more detailed expositions. This lecture of Aaronson for a great discussion of the feasibility of quantum computing (Aaronson's course lecture notes and the book that they spawned are fantastic reads as well). The videos of Umesh Variani'z EdX course are an accessible and recommended introduction to quantum computing. See the "bibliographical notes" section at the end of this lecture for more resources.

23.2 *Quantum computing and computation - an executive summary.*

One of the strange aspects of the quantum-mechanical picture of the world is that unlike in the billiard ball example, there is no obvious algorithm to simulate the evolution of n particles over t time periods in $poly(n,t)$ steps. In fact, the natural way to simulate n quantum particles will require a number of steps that is *exponential* in n. This is a huge headache for scientists that actually need to do these calculations in practice.

In the 1981, physicist Richard Feynman proposed to "turn this lemon to lemonade" by making the following almost tautological observation:

> If a physical system cannot be simulated by a computer in T steps, the system can be considered

as performing a computation that would take more than T steps.

So, he asked whether one could design a quantum system such that its outcome y based on the initial condition x would be some function $y = f(x)$ such that **(a)** we don't know how to efficiently compute in any other way, and **(b)** is actually useful for something.[5] In 1985, David Deutsch formally suggested the notion of a quantum Turing machine, and the model has been since refined in works of Detusch and Josza and Bernstein and Vazirani. Such a system is now known as a *quantum computer*.

For a while these hypothetical quantum computers seemed useful for one of two things. First, to provide a general-purpose mechanism to simulate a variety of the real quantum systems that people care about. Second, as a challenge to the *Extended Church Turing hypothesis* which says that every physically realizable computation device can be modeled (up to polynomial overhead) by Turing machines (or equivalently, NAND++ / NAND« programs). However, (unless you care about quantum chemistry) it seemed like a challenge that might have little bearing to practice, given that this theoretical "extra power" of quantum computer seemed to offer little advantage in the majority of the problems people want to solve in areas such as combinatorial optimization, machine learning, data structures, etc..

To a significant extent, this is still true today. We have no real evidence that quantum computers, if built, will offer truly significant[6] advantage in 99% of the applications of computing. In particular, as far as we know, quantum computers will *not* help us solve **NP** complete problems in polynomial or even sub-exponential time.[7] However, there is one cryptography-sized exception: In 1994 Peter Shor showed that quantum computers can solve the integer factoring and discrete logarithm in polynomial time. This result has captured the imagination of a great many people, and completely energized research into quantum computing.

This is both because the hardness of these particular problems provides the foundations for securing such a huge part of our communications (and these days, our economy), as well as it was a powerful demonstration that quantum computers could turn out to be useful for problems that a-priori seemd to have nothing to do with quantum physics. As we'll discuss later, at the moment there are several intensive efforts to construct large scale quantum computers. It seems safe to say that, as far as we know, in the next five years or so there will not be a quantum computer large enough to factor, say,

[5] As its title suggests, Feynman's lecture was actually focused on the other side of simulating physics with a computer. However, he mentioned that as a "side remark" one could wonder if it's possible to simulate physics with a new kind of computer - a "quantum computer" which would "not [be] a Turing machine, but a machine of a different kind". As far as I know, Feynman did not suggest that such a computer could be useful for computations completely outside the domain of quantum simulation, and in fact he found the question of whether quantum mechanics could be simulated by a classical computer to be more interesting.

[6] I am using the theorist' definition of conflating "significant" with "super-polynomial". Grover's algorithm does offer a very generic *quadratic* advantage in computation. Whether that quadratic advantage will ever be good enough to offset in practice the significant overhead in building a quantum computer remains an open question. We also don't have evidence that super-polynomial speedups *can't* be achieved for some problems outside the Factoring/Dlog or quantum simulation domains, and there is at least one company banking on such speedups actually being feasible.

[7] This "99 percent" is a figure of speech, but not completely so. It seems that for many web servers, the TLS protocol (which based on the current non-lattice based systems would be completely broken by quantum computing) is responsible for about 1 percent of the CPU usage.

a 1024 bit number, but there it is quite possible that some quantum computer will be built that is strong enough to achieve some task that is too inefficient to achieve with a non-quantum or "classical" computer (or at least requires far more resources classically than it would for this computer). When and if such a computer is built that can break reasonable parameters of Diffie Hellman, RSA and elliptic curve cryptography is anybody's guess. It could also be a "self destroying prophecy" whereby the existence of a small-scale quantum computer would cause everyone to shift away to lattice-based crypto which in turn will diminish the motivation to invest the huge resources needed to build a large scale quantum computer.[8]

[8] Of course, given that we're still hearing of attacks exploiting "export grade" cryptography that was supposed to disappear in 1990's, I imagine that we'll still have products running 1024 bit RSA when everyone has a quantum laptop.

R **Quantum computing and NP** Despite popular accounts of quantum computers as having variables that can take "zero and one at the same time" and therefore can "explore an exponential number of possiblities simultaneously", their true power is much more subtle and nuanced. In particular, as far as we know, quantum computers do *not* enable us to solve **NP** complete problems such as 3SAT. In particular it is believed that for every large enough n, the restriction of 3SAT to length n cannot be solved by quantum circuits (or equivalently QNAND programs) of polynomial, or even subexponential size. However, Grover's search algorithm does give a more modest advantage (namely, quadratic) for quantum computers over classical ones for problems in **NP**. In particular, due to Grover's search algorithm, we know that the k-SAT problem for n variables can be solved in time $O(2^{n/2}poly(n))$ on a quantum computer for every k. In contrast, the best known algorithms for k-SAT on a classical computer take roughly $2^{(1-\frac{1}{k})n}$ steps.

23.3 *The computational model*

Before we talk about *quantum* computing, let us recall how we physically realize "vanilla" or *classical* computing. We model a *logical bit* that can equal 0 or a 1 by some physical system that can be in one of two states. For example, it might be a wire with high or low voltage, charged or uncharged capacitor, or even (as we saw) a pipe with or without a flow of water, or the presence or absence of a soldier crab. A *classical* system of n bits is composed of n such "basic systems", each of which can be in either a "zero" or "one" state. We can model the state of such a system by a string $s \in \{0,1\}^n$. If we perform an

operation such as writing to the 17-th bit the NAND of the 3rd and 5th bits, this corresponds to applying a *local* function to s such as setting $s_{17} = 1 - s_3 \cdot s_5$.

In the *probabilistic* setting, we would model the state of the system by a *distribution*. For an individual bit, we could model it by a pair of non-negative numbers α, β such that $\alpha + \beta = 1$, where α is the probability that the bit is zero and β is the probability that the bit is one. For example, applying the *negation* (i.e., NOT) operation to this bit corresponds to mapping the pair (α, β) to (β, α) since the probability that $NOT(\sigma)$ is equal to 1 is the same as the probability that σ is equal to 0. This means that we can think of the NOT function as the linear map $N : \mathbb{R}^2 \to \mathbb{R}^2$ such that $N(\alpha, \beta) = (\beta, \alpha)$.

If we think of the n-bit system as a whole, then since the n bits can take one of 2^n possible values, we model the state of the system as a vector p of 2^n probabilities $p_0, \ldots, p_{2^n}$, where for every $s \in \{0,1\}^n$, p_s denotes the probability that the system is in the state s, identifying $\{0,1\}^n$ with $[2^n]$. Applying the operation above of setting the 17-th bit to the NAND of the 3rd and 5th bits, corresponds to transforming the vector p to the vector Fp where $F : \mathbb{R}^{2^n} \to \mathbb{R}^{2^n}$ is the map such that

$$F(p)_s = \begin{cases} 0 & s_{17} \neq 1 - s_3 \cdot s_5 \\ p_s + p_{s'} & \text{otherwise} \end{cases} \tag{23.1}$$

where s' is the string that agrees with s on all but the 17th coordinate.

P It might not be immediate to see why Eq. (23.1) describes the progression of such a system, so you should pause here and make sure you understand that. Understanding evolution of probabilistic systems is a prerequisite to understanding evolution of quantum systems. You should also make sure that you understand why the function $F : \mathbb{R}^{2^n} \to F : \mathbb{R}^{2^n}$ described in Eq. (23.1) is *linear*, in the sense that for every pair of vectors $p, q \in \mathbb{R}^{2^n}$ and numbers $a, b \in R$, $F(ap + bq) = aF(p) + bF(q)$.
If your linear algebra is a bit rusty, now would be a good time to review it, and in particular make sure you are comfortable with the notions of *matrices*, *vectors*, (orthogonal and orthonormal) *bases*, and *norms*.

23.3.1 Quantum probabilities

In the quantum setting, the state of an individual bit (or "qubit", to use quantum parlance) is modeled by a pair of numbers (α, β) such that $\alpha^2 + \beta^2 = 1$. As before, we think of α^2 as the probability that the bit equals 0 and β^2 as the probability that the bit equals 1. Therefore, as before, we can model the NOT operation by the map $N : \mathbb{R}^2 \to \mathbb{R}^2$ where $N(\alpha, \beta) = (\beta, \alpha)$.

Following quantum tradition, we will denote the vector $(1, 0)$ by $|0\rangle$ and the vector $(0, 1)$ by $|1\rangle$ (and moreover, think of these as column vectors). So NOT is the unique linear map $N : \mathbb{R}^2 \to \mathbb{R}^2$ that satisfies $N(|0\rangle) = |1\rangle$ and $N(|1\rangle) = |0\rangle$. (This is known as the Dirac "ket" notation.)

In classical computation, we typically think that there are only two operations that we can do on a single bit: keep it the same or negate it. In the quantum setting, a single bit operation corresponds to any linear map $OP : \mathbb{R}^2 \to \mathbb{R}^2$ that is *norm preserving* in the sense that for every α, β such that $\alpha^2 + \beta^2 = 1$, if $(\alpha', \beta') = OP(\alpha, \beta)$ then $\alpha'^2 + \beta'^2 = 1$. Such a linear map OP corresponds to a unitary two by two matrix.[9] Keeping the bit the same corresponds to the matrix $I = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ and the NOT operations corresponds to the matrix $N = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$. But there are other operations we can use as well. One such useful operation is the *Hadamard* operation, which corresponds to the matrix $H = \frac{1}{\sqrt{2}} \left(\begin{smallmatrix} +1 & +1 \\ +1 & -1 \end{smallmatrix}\right)$. In fact it turns out that Hadamard is all that we need to add to a classical universal basis to achieve the full power of quantum computing.

[9] As we mentioned, quantum mechanics actually models states as vectors with *complex* coordinates. However, this does not make any qualitative difference to our discussion.

23.3.2 Quantum circuits and QNAND

A *quantum circuit* is analogous to a Boolean circuit, and can be described as a directed acyclic graph. One crucial difference that the *out degree* of every vertex in a quantum circuit is at most one. This is because we cannot "reuse" quantum states without *measuring* them (which collapses their "probabilities"). Therefore, we cannot use the same bit as input for two different gates.[10] Another more technical difference is that to express our operations as unitary matrices, we will need to make sure all our gates are *reversible*. This is not hard to ensure. For example, in the quantum context, instead of thinking of $NAND$ as a (non reversible) map from $\{0,1\}^2$ to $\{0,1\}$, we will think of it as the reversible map on *three* qubits that maps a, b, c to $a, b, c \oplus NAND(a, b)$. Equivalently, the NAND operation corresponds to the unitary map U_{NAND} on $\mathbb{R}^{2^3}$ such that (identi-

[10] This is known as the No Cloning Theorem.

fying $\{0,1\}^3$ with $[8]$) for every $a,b,c \in \{0,1\}$, if $|abc\rangle$ is the basis element with 1 in the abc-th coordinate and zero elsewhere, then $U(|abc\rangle) = |ab(c \oplus NAND(a,b))\rangle$.[11]

[11] U_{NAND} is a close variant of the so called Toffoli gate and so QNAND programs correspond to quantum circuits with the Hadamard and Toffoli gates.

Just like in the classical case, there is an equivalence between circuits and straightline programs, and so we can define the programming language QNAND that is the quantum analog of our NAND programming language. To do so, we only add a single operation: `HAD(foo)` which applies the single-bit operation H to the variable `foo`. We also use the following interpretation to make `NAND` reversible: `foo := bar NAND blah` means that we modify `foo` to be the XOR of its original value and the NAND of `bar` and `blah`. (In other words, apply the 8 by 8 unitary transformation U_{NAND} defined above to the three qubits corresponding to `foo`, `bar` and `blah`.) If `foo` is initialized to zero then this makes no difference.

If P is a QNAND program with n input variables, ℓ workspace variables, and m output variables, then running it on the input $x \in \{0,1\}^n$ corresponds to setting up a system with $n+m+\ell$ qubits and performing the following process:

1. We initialize the input variables to $x_0,\ldots,x_{n-1}$ and all other variables to 0.
2. We execute the program line by line, applying the corresponding physical operation H or U_{NAND} to the qubits that are referred to by the line.
3. We *measure* the output variables `y_0`, ..., `y_`$\langle m-1 \rangle$ and output the result.

This seems quite simple, but maintaining the qubits in a way that we can apply the operations on one hand, but we don't accidentally measure them or corrupt them in another way, is a significant engineering challenge.

R **Uniformity** Just like NAND, QNAND is a *non uniform* model of computation, where we consider a different program for every input step. One can define quantum Turing machines or QNAND++ programs to capture the notion of *uniform* quantum computation where a single algorithm is specified for all input lengths. The quantum analog of **P**, known as **BQP** is defined using such notions. However, for the sake of simplicity, we omit the discussion of those models in this lecture. However, all of the discussion in this section holds equally well for the uniform and non-uniform models.

23.3.3 *Analyzing QNAND execution*

The *state* of an $n + \ell + m$-qubit system can be modeled, as we discussed, by a vector in $\mathbb{R}^{2^{n+\ell+m}}$. For an input x, the initial state corresponds to the fact that we initialize the system to $x0^{\ell+m}$ (i.e., the inputs variable get the values of x and all other variables get the value zero). We can think of this initial state as saying that if we measured all variables in the system then we'll get the value $x0^{\ell+m}$ with probability one, and hence it is modeled by the vector $s^0 \in \mathbb{R}^{2^{n+\ell+m}}$ such that (identifying the coordinates with strings of length $n + \ell + m$) $s^0_{x0^{\ell+m}} = 1$ and $s^0_z = 1$ for every $z \neq x0^{n+\ell+m}$. (Please pause here and see that you understand what the notation above means.)

Every line in the program corresponds to applying a (rather simple) unitary map on $\mathbb{R}^{2^{n+\ell+m}}$, and so if L_i is the map corresponding to the i-th line, then $s_{i+1} = L_i(s_i)$. If the program has t lines then $s^* = L_{t-1}(L_{t-2}(\cdots L_0(s^0)))$ is the final state of the system. At the end of the process we *measure* the bits, and so we get a particular Boolean assignment z for its variables with probability $(s^*_z)^2$. Since we output the bits corresponding to the output variables, for every string $y \in \{0,1\}^m$, the output will equal y with probability $\sum_{z \in S_y} (s^*_z)^2$ where S_y is the set of all $z \in \{0,1\}^{n+\ell+m}$ that agree with y in the last m coordinates.

R **The obviously exponential fallacy** A priori it might seem "obvious" that quantum computing is exponentially powerful, since to perform a quantum computation on n bits we need to maintain the 2^n dimensional state vector and apply $2^n \times 2^n$ matrices to it. Indeed popular descriptions of quantum computing (too) often say something along the lines that the difference between quantum and classical computer is that a classic bit can either be zero or one while a qubit can be in both states at once, and so in many qubits a quantum computer can perform exponentially many computations at once.

Depending on how you interpret it, this description is either false or would apply equally well to *probabilistic computation*, even though we've already seen that every randomized algorithm can be simulated by a similar-sized circuit, and in fact we conjecture that $\mathbf{BPP} = \mathbf{P}$.

Moreover, this "obvious" approach for simulating a quantum computation will take not just exponential time but *exponential space* as well, while it is not hard to show that using a simple recursive formula one can calculate the final quantum state using *polynomial space* (in physics this is known as "Feynman

path integrals"). So, the exponentially long vector description by itself does not imply that quantum computers are exponentially powerful. Indeed, we cannot *prove* that they are (i.e., as far as we know, every QNAND program could be simulated by a NAND program with polynomial overhead), but we do have some problems (integer factoring most prominently) for which they do provide exponential speedup over the currently best *known* classical (deterministic or probabilistic) algorithms.

23.3.4 Complexity classes

If $F : \{0,1\}^n \to \{0,1\}$ is a finite function and $s \in \mathbb{N}$ then we say that $F \in QSIZE(s)$ if there exists a QNAND program P of at most s lines that comptues F, in the sense that for every $x \in \{0,1\}^n$, $\mathbb{P}[P(x) = F(x)] \geq 2/3$.[12] Equivalently, $F \in QSIZE(S)$ if there is a quantum circuit of at most s gates that computes it.[13] For an *infinite* function $F : \{0,1\}^* \to \{0,1\}$, we say that $F \in \mathbf{BQP}_{/\mathbf{poly}}$ if there is some polynomial $p : \mathbb{N} \to \mathbb{N}$ and a sequence $\{Q_n\}_{n\in\mathbb{N}}$ of QNAND programs such that for every $n \in \mathbb{N}$, Q_n has less than $p(n)$ lines and Q_n computes the restriction of F to inputs in $\{0,1\}^n$. We can also define the class **BQP** to be the uniform analog of $\mathbf{BQP}_{/\mathbf{poly}}$. It can be defined using QNAND++ programs, but also has the following equivalent definition: $F : \{0,1\}^* \to \{0,1\}$ is in **BQP** if there is polynomial-time (classical) NAND++ program P such that for every $n \in \mathbb{N}$, $P(1^n)$ is a string representing a QNAND program Q_n such that Q_n computes the restriction of F to inputs in $\{0,1\}^n$.

[12] The number 2/3 is arbitrary. As in the case of **BPP**, we can amplify success probability of a quantum algorithm to our liking.

[13] Recall that we use circuits over the basis consisting of the Hadamard gate and the "reversible NAND" or "shifted Toffoli" gate $abc \mapsto ab(c \oplus (1-ab))$. However, using any other universal basis only changes the number of gates by a constant factor.

P Parsing the above definitions can take a bit of time, but they are ultimately not very deep. One way to verify that you've understood these definitions it to see that you can prove **(1)** $\mathbf{P} \subseteq \mathbf{BQP}$ and in fact the stronger statement $\mathbf{BPP} \subseteq \mathbf{BQP}$, **(2)** $\mathbf{BQP} \subseteq \mathbf{EXP}$, and **(3)** For every **NP**-complete function F, if $F \in \mathbf{BQP}$ then $\mathbf{NP} \subseteq \mathbf{BQP}$.

The relation between **NP** and **BQP** is not known. It is believed that they are incomprable, in the sense that $\mathbf{NP} \not\subseteq \mathbf{BQP}$ (and in particular no **NP**-complete function belongs to **BQP**) but also $\mathbf{BQP} \not\subseteq \mathbf{NP}$ (and there are some interesting candidates for such problems).

It can be shown that $QNANDEVAL$ (evluating a quantum circuit on an input) is computable by a polynomial size QNAND program, and moreover this program can even be generated *uniformly* and

hence *QNANDEVAL* is in **BQP**. This allows us to "port" many of the results of classical computational complexity into the quantum realm as well.

23.3.5 *Physically realizing quantum computation*

To realize quantum computation one needs to create a system with n independent binary states (i.e., "qubits"), and be able to manipulate small subsets of two or three of these qubits to change their state. While by the way we defined operations above it might seem that one needs to be able to perform arbitrary unitary operations on these two or three qubits, it turns out that there several choices for *universal sets* - a small constant number of gates that generate all others. The biggest challenge is how to keep the system from being measured and *collapsing* to a single classical combination of states. This is sometimes known as the *coherence time* of the system. The threshold theorem says that there is some absolute constant level of errors τ so that if errors are created at every gate at rate smaller than τ then we can recover from those and perform arbitrary long computations. (Of course there are different ways to model the errors and so there are actually several threshold *theorems* corresponding to various noise models).

There have been several proposals to build quantum computers:[14]

- Superconducting quantum computers use super-conducting electric circuits to do quantum computation. Recent works have shown one can keep these superconducting qubits fairly robust to the point one can do some error correction on them (see also here).
- Trapped ion quantum computers Use the states of an ion to simulate a qubit. People have made some recent advances on these computers too. While it's not clear that's the right measuring yard, the current best implementation of Shor's algorithm (for factoring 15) is done using an ion-trap computer.
- Topological quantum computers use a different technology, which is more stable by design but arguably harder to manipulate to create quantum computers.

These approaches are not mutually exclusive and it could be that ultimately quantum computers are built by combining all of them together. I am not at all an expert on this matter, but it seems that progress has been slow but steady and it is quite possible that we'll see a 50 qubit computer sometime in the next 5 years.

[14] The text below was written in early 2016 and likely needs to be updated.

23.4 *Analysis of Bell's Inequality*

Now that we have the notation in place, we can show the strategy for showing "quantum telepathy".

Lemma 23.2 There is a 2-qubit quantum state $s \in \mathbb{R}^4$ so that if Alice has access to the first qubit of s, can manipulate and measure it and output $a \in \{0,1\}$ and Bob has access to the second qubit of s and can manipulate and measure it and output $b \in \{0,1\}$ then $\mathbb{P}[a \oplus b = x \wedge y] \geq 0.8$.

Proof. The main idea is for Alice and Bob to first prepare a 2-qubit quantum system in the state (up to normalization) $|00\rangle + |11\rangle$ (this is known as an *EPR pair*). Alice takes the first qubit in this system to her room, and Bob takes the qubit to his room. Now, when Alice receives x if $x = 0$ she does nothing and if $x = 1$ she applies the unitary map $R_{\pi/8}$ to her qubit where $R_\theta = \left(\begin{smallmatrix} cos\theta & \sin -\theta \\ \sin\theta & \cos\theta \end{smallmatrix}\right)$ is the unitary operation corresponding to rotation in the plane with angle θ. When Bob receives y, if $y = 0$ he does nothing and if $y = 1$ he applies the unitary map $R_{-\pi/8}$ to his qubit. Then each one of them measures their qubit and sends this as their response. Recall that to win the game Bob and Alice want their outputs to be more likely to differ if $x = y = 1$ and to be more likely to agree otherwise.

If $x = y = 0$ then the state does not change and Alice and Bob always output either both 0 or both 1, and hence in both case $a \oplus b = x \wedge y$. If $x = 0$ and $y = 1$ then after Alice measures her bit, if she gets 0 then Bob's state is equal to $-\cos(\pi/8)|0\rangle - \sin(\pi/8)|1\rangle$ which will equal 0 with probability $\cos^2(\pi/8)$. The case that Alice gets 1, or that $x = 1$ and $y = 0$, is symmetric, and so in all the cases where $x \neq y$ (and hence $x \wedge y = 0$) the probability that $a = b$ will be $\cos^2(\pi/8) \geq 0.85$. For the case that $x = 1$ and $y = 1$, direct calculation via trigonomertic identities yields that all four options for (a,b) are equally likely and hence in this case $a = b$ with probability 0.5. The overall probability of winning the game is at least $\frac{1}{4} \cdot 1 + \frac{1}{2} \cdot 0.85 + \frac{1}{4} \cdot 0.5 = 0.8$. ■

R **Quantum vs probabilistic strategies** It is instructive to understand what is it about quantum mechanics that enabled this gain in Bell's Inequality. For this, consider the following analogous probabilistic strategy for Alice and Bob. They agree that each one of them output 0 if he or she get 0 as input and outputs 1 with probability p if they get 1 as input. In this case one

can see that their success probability would be $\frac{1}{4} \cdot 1 + \frac{1}{2}(1-p) + \frac{1}{4}[2p(1-p)] = 0.75 - 0.5p^2 \leq 0.75$. The quantum strategy we described above can be thought of as a variant of the probabilistic strategy for parameter p set to $\sin^2(\pi/8) = 0.15$. But in the case $x = y = 1$, instead of disagreeing only with probability $2p(1-p) = 1/4$, because we can use these negative probabilities in the quantum world and rotate the state in opposite directions, and hence the probability of disagreement ends up being $\sin^2(\pi/4) = 0.5$.

23.5 *Shor's Algorithm*

Bell's Inequality is powerful demonstration that there is something very strange going on with quantum mechanics. But could this "strangeness" be of any use to solve computational problems not directly related to quantum systems? A priori, one could guess the answer is *no*. In 1994 Peter Shor showed that one would be wrong:

Theorem 23.3 — Shor's Algorithm. Let $FACT_n : \{0,1\}^n \to \{0,1\}^*$ be the function that on input a number M (represented in base two), outputs the prime factorization of M. Then $FACT_n$ can be computed by a QNAND program of $O(n^3)$ lines.

This is an exponential improvement over the best known classical algorithms, which take roughly $2^{\tilde{O}(n^{1/3})}$ time, where the $\tilde{O}$ notation hides factors that are polylogarithmic in n. In the rest of this section we will sketch some of the ideas behinf Shor's algorithm.

R **Group theory** While we will define the concepts we use, some background in group or number theory might be quite helpful for fully understanding this section.

We will not use anything more than the basic properties of finite Abelian groups. Specifically we use the following notions: A finite *group* $\mathbb{G}$ can be thought of as simply a set of elements and some *binary operation* $\star$ on these elements (i.e., if $g, h \in \mathbb{G}$ then $g \star h$ is an element of $\mathbb{G}$ as well). The operation satisfies the sort of properties that a product operation does. It is associative (i.e., $(g \star h) \star f = g \star (h \star f)$) and there is some element 1 such that $g \star 1 = g$ for all g, where for every $g \in \mathbb{G}$ there exists an element g^{-1} such that $g \star g^{-1} = 1$. Moreover, we assume the group is *Abelian* or *commutative*, which means

that $g \star h = h \star g$ for all $g, h \in \mathbb{G}$. We denote by g^2 the element $g \star g$, by g^3 the element $g \star g \star g$, and so on and so forth. The *order* of $g \in \mathbb{G}$ is the smallest natural number a such that $g^a = 1$. (It can be shown that such a number exists for every g in a finite group, and moreover that it is always smaller than the size of the group.)

23.5.1 *Period finding*

The heart of Shor's algorithm is the following result:

Lemma 23.4 For every (efficiently presented) Abliean group $\mathbb{G}$, there is a quantum polynomial time algorithm that given a *periodic* function $f : \mathbb{G} \to \{0,1\}^*$ finds a period of f.

If $\mathbb{G}$ is an Abelian group with operation $\star$ and $z \in \mathbb{G}$ is not the 1 element, then a function $f : \mathbb{G} \to \{0,1\}^*$ is z *periodic* if $f(x \star z) = f(x)$ for every $x \in \mathbb{G}$. The algorithm of Lemma 23.4 is "given" the group $\mathbb{G}$ in the form of a classical algorithm (e.g., a NAND program) that computes the function $\star : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, where n is the number of bits that are required to represent an element of the group (which is logarithmic in the size of the group itself). Similarly, it is given the function f in the form of a NAND program computing it.

23.5.2 *From order finding to factoring and discrete log*

Using the function $f(a) = g^a$ one can use period finding (for the group of $\mathbb{Z}_{|\mathbb{G}|} = \{0,1,2,\ldots,|\mathbb{G}|-1\}$ with modular addition) to find the *order* of any element in a group $\mathbb{G}$. We can then use order finding to both factor integers in polynomial time and solve the discrete logarithm over arbitrary Abelian groups. This shows that quantum computers will break not RSA, Diffie Hellman and Elliptic Curve Cryptography, which are by far the most widely deployed public key cryptosystems today. We merely sketch how one reduces the factoring and discrete logarithm problems to order finding: (see some of the sources above for the full details)

- For **factoring**, let us restrict to the case $m = pq$ for distinct p, q. It turns out that in this case the multiplicative group modulo m (known as $\mathbb{Z}_m^*$) has size $(p-1)(q-1) = m - p - q - 1$ and moreover finding this size can be used to efficiently obtain the factors of m. One can show that if we pick a few random x's in

$\mathbb{Z}_m^*$ and compute their order, the least common multiplier of these orders is likely to be the group size.

- For **discrete log** in a group $\mathbb{G}$, if we get $X = g^x$ and need to recover x, we can compute the order of various elements of the form $X^a g^b$. The order of such an element is a number c satisfying $c(xa + b) = 0 \pmod{|\mathbb{G}|}$. Again, with a few random examples we will get a non trivial equation on x (where c is not zero modulo $|\mathbb{G}|$) and would be to use our knowledge of a, b, c to recover x.

23.5.3 *Finding periods of a function: Simon's Algorithm*

How do we find the period of a function? Let us consider the simplest case, where f is a function from $\mathbb{R}$ to $\mathbb{R}$ that is h^* periodic for some number h^*, in the sense that f repeats itself on the intervals $[0, h^*]$, $[h^*, 2h^*]$, $[2h^*, 3h^*]$, etc.. How do we find this number h^*? A standard technique in finding the period of a function f is to transform f from the *time* to the *frequency* domain. That is, we use the Fourier transform to represent f as a sum of *wave functions*. In this representation, wavelengths that divide the period h^* would get significant mass, while wavelengths that don't "cancel out".

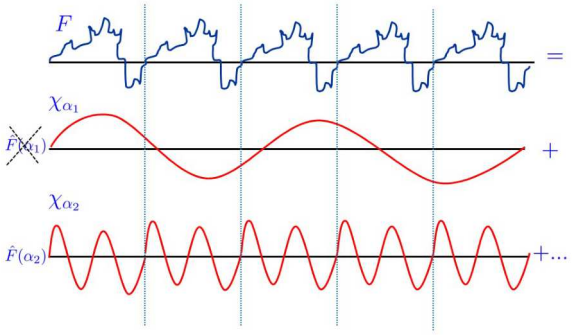

Figure 23.4: If f is a periodic function then when we represent it in the Fourier transform, we expect the coefficients corresponding to wavelengths that do not evenly divide the period to be very small, as they would tend to "cancel out".

Similarly, the main idea behind Shor's algorithm is to use a tool known as the quantum fourier transform that given a circuit computing the function $f : \mathbb{H} \to \mathbb{R}$, creates a quantum state over roughly $\log |\mathbb{H}|$ qubits (and hence dimension $|\mathbb{H}|$) that corresponds to the Fourier transform of f. Hence when we measure this state, we get a group element h with probability proportional to the square of the corresponding Fourier coefficient. One can show that if f is

h^*-periodic then we can recover h^* from this distribution.

Shor carried out this approach for the group $\mathbb{H} = \mathbb{Z}_m^*$ for some m,[15] but we will show this for the group $\mathbb{H} = \{0,1\}^n$ with the XOR operation. This case is known as *Simon's algorithm* (given by Dan Simon in 1994) and actually preceded (and inspired) Shor's algorithm:

[15] For a number $m \in \mathbb{N}$, the group $\mathbb{Z}_m^*$ is set $\{k \in [m] \mid gcd(k,m) = 1\}$ with the operation being multiplication modulo m. It is known as the *multiplicative group* modulo m.

Theorem 23.5 — Simon's Algorithm. If $f : \{0,1\}^n \rightarrow \{0,1\}^*$ is polynomial time computable and satisfies the property that $f(x) = f(y)$ iff $x \oplus y = h^*$ then there exists a quantum polynomial-time algorithm that outputs a random $h \in \{0,1\}^n$ such that $\sum_{i=0}^{n-1} h_i h_i^* = 0 \mod 2$.

Note that given $O(n)$ such samples, we can recover h^* with high probability by solving the corresponding linear equations.

Proof Idea: The idea behind the proof is that the *Hadamard* operation corresponds to the *Fourier transform* over the group $\{0,1\}^n$ (with the XOR operations). We can use that to create quantum state over n qubits where the probability of obtaining some value h is proportional to the coefficient corresponding to h in the Fourier transform of (a real-valued function related to) f. We can show that this coefficient will be zero if h is not orthogonal to the period h^* modulo 2, and hence when we measuer this state we will obtain some h satisfying $\sum_{i=0}^{n-1} h_i h_i^* = 0 \mod 2$.

Proof of Theorem 23.5. We can express the Hadamard operation HAD as follows:

$$HAD|a\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^a|1\rangle) \; . \tag{23.2}$$

Given the state $|0^{n+m}\rangle$ we can apply this map to each one of the first n qubits to get the state

$$2^{-n/2} \sum_{x \in \{0,1\}^n} |x\rangle|0^m\rangle \tag{23.3}$$

and then we can apply the gates of f to map this to the state

$$2^{-n/2} \sum_{x \in \{0,1\}^n} |x\rangle|f(x)\rangle \; . \tag{23.4}$$

Now suppose that we apply the HAD operation again to the first

n qubits. We can see that we get the state

$$2^{-n} \sum_{x \in \{0,1\}^n} \prod_{i=0}^{n-1} (|0\rangle + (-1)^{x_i}|1\rangle)|f(x)\rangle \; . \tag{23.5}$$

We can use the distributive law and express a product of the form $\varphi(x) = \prod_{i=0}^{n-1}(|0\rangle + (-1)^{x_i}|1\rangle)|f(x)\rangle$ as the sum of 2^n terms, where each term corresponds to picking either $|0\rangle$ or $(-1)^{x_i}|1\rangle$. Another way to say it is that this product $\varphi(x)$ is equal to

$$\sum_{y \in \{0,1\}^n} \prod_{i=0}^{n-1} (-1)^{x_i y_i}|y_i\rangle|f(x)\rangle \; . \tag{23.6}$$

Using the equality $(-1)^a(-1)^b = (-1)^{a+b}$ (and the fact that when raising -1 to an integer, we only care if it's odd or even) we get that

$$\varphi(x) = \sum_{y \in \{0,1\}^n} (-1)^{\sum_{i=0}^{n-1} x_i y_i \mod 2}|y\rangle|f(x)\rangle \tag{23.7}$$

and therefore the overall state is equal to

$$2^{-n} \sum_{x \in \{0,1\}^n} \varphi(x) = 2^{-n} \sum_{x \in \{0,1\}^n} \sum_{y \in \{0,1\}^n} (-1)^{\sum_{i=0}^{n-1} x_i y_i \mod 2}|y\rangle|f(x)\rangle \; . \tag{23.8}$$

Now under our assumptions for every particular z in the image of f, there exist exactly two preimages x and $x \oplus h^*$ such that $f(x) = f(x + h^*) = z$. So, if $\sum_{i=0}^{n-1} h_i^* y_i = 0 \mod 2$, we get that $|(-1)^{\sum_{i=0}^{n-1} x_i y_i = 0 \mod 2} + (-1)^{\sum_{i=0}^{n-1}(x_i + h_i^*)y_i = 0 \mod 2}| = 2$ (positive interference) and otherwise we get $|(-1)^{\sum_{i=0}^{n-1} x_i y_i = 0 \mod 2} + (-1)^{\sum_{i=0}^{n-1}(x_i + h_i^*)y_i = 0 \mod 2}| = 0$ (negative interference, i.e., cancellation). Therefore, if measure the end state then with probability one we the first n bits will be a string y such that $\sum_{i=0}^{n-1} y_i h_i^* = 0 \mod 2$. ■

Simon's algorithm seems to really use the special bit-wise structure of the group $\{0,1\}^n$, so one could wonder if it has any relevance for the group $\mathbb{Z}_m^*$ for some exponentially large m. It turns out that the same insights that underlie the well known Fast Fourier Transform (FFT) algorithm can be used to essentially follow the same strategy for this group as well.

23.6 *Lecture summary*

23.7 *Exercises*

23.8 *Bibliographical notes*

Chapters 9 and 10 in the book *Quantum Computing Since Democritus* give an informal but highly informative introduction to the topics of this lecture and much more. Shor's and Simon's algorithms are also covered in Chapter 10 of my book with Arora on computational complexity.

There are many excellent videos available online covering some of these materials. The Fourier transform is covered in this videos of Dr. Chris Geoscience, Clare Zhang and Vi Hart. More specifically to quantum computing, the videos of Umesh Vazirani on the Quantum Fourier Transform and Kelsey Houston-Edwards on Shor's Algorithm are very recommended.

Chapter 10 in Avi Wigderson's book gives a high level overview of quantum computing. Andrew Childs' lecture notes on quantum algorithms, as well as the lecture notes of Umesh Vazirani, John Preskill, and John Watrous

Regarding quantum mechanics in general, this video illustrates the double slit experiment, this Scientific American video is a nice exposition of Bell's Theorem. This talk and panel moderated by Brian Greene discusses some of the philosophical and technical issues around quantum mechanics and its so called "measurement problem". The Feynmann lecture on the Fourier Transform and quantum mechanics in general are very much worth reading.

The Fast Fourier Transform, used as a component in Shor's algorithm, is one of the most useful algorithms across many applications areas. The stories of its discovery by Gauss in trying to calculate astroid orbits and rediscovery by Tukey during the cold war are fascinating as well.

23.9 *Further explorations*

Some topics related to this lecture that might be accessible to advanced students include: (to be completed)

23.10 Acknowledgements