Quantum Computation

March 20, 2009

The subject of computer science has succeeded by abstracting the notion of "computation" from its physical implementation - thus, rather than electrons and chips, we speak of bits and logical operations. Nevertheless, it is important to consider what is possible in the real, physical world. For example, computer science studies primarily deterministic, "Turing" computation, and for a while it seemed this was the "best possible" model of computation. However, it emerges that probabilistic computers are useful for some applications, and analogue computation also appears worthy of some consideration (although having received said attention, it seems to be useless in practice). Finally, we consider quantum computation, the subject of this course; the lecturer believes it is very much useful.

It must be said that of the various approaches to building a quantum computer, none has advanced very far. Therefore while our initial motivation was the physical reality, we shall return rapidly to theory, and consider quantum computers far beyond that which can currently be built in the lab.

If we had such a "full" quantum computer, the obvious applications are much faster codebreaking, and also database search. More appealing to the mathematician is the ability to simulate complicated quantum systems in reasonable time; of course, the best applications likely remain to be discovered.

We shall return briefly to classical computation to understand where we are; there are several universal architectures for classical computation, e.g. Turing machines or cellular automata. We shall consider circuit models: the bits, 0 or 1, are input on wires, and pass through a series of gates to give an output. Gates are usually (at least initially) described by truth tables. E.g. we have the AND gate, whose output is given by the table:

Input	Output
00	0
01	0
10	0
11	1

. Other examples would be the NOT gate, and the CNOT or "controlled NOT" gate, which takes two inputs, the first of which acts as a control on a NOT gate for the second input (and is also outputted unchanged):

Input	Output
00	00
01	01
10	11
11	10

.

Truth tables are hard to manipulate mathematically. So, we will instead associate each classical bit with a 2D real vector: $0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, 1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. We will

use the Dirac bra-ket notation: vectors are kets $|v\rangle$, dual vectors are bras $\langle v|$, and the inner product is $\langle u|v\rangle$; states are orthogonal if this is 0. So for example $\langle 0|1\rangle = \begin{pmatrix}1 & 0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} = 0$. The norm is $||v\rangle| = \sqrt{\langle v|v\rangle}$; we say $|v\rangle$ is normalised if this is 1. A set of states is orthonormal if each is normalised and pairs are orthogonal, e.g. $|0\rangle, |1\rangle$.

Multiple bits are given by tensor products: if we have two bits input with values $|0\rangle, |1\rangle$ respectively, $|v\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}1\begin{pmatrix}0\\1\end{pmatrix}\\0\begin{pmatrix}0\\1\end{pmatrix}\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$. We will sometimes write $|u\rangle \otimes |v\rangle$ as $|u\rangle|v\rangle$ or even just $|uv\rangle$.

Note that the tensor product of vector spaces of dimension d_1, d_2 is of dimension $d_1 d_2$, so the space of states of n bits is 2^n-dimensional, quite large.

Gates are represented by operators or matrices, e.g. the AND gate $A = |0\rangle(\langle 00| + \langle 10| + \langle 01|) + |1\rangle\langle 11| = \begin{pmatrix}1 & 1 & 1 & 0\\0 & 0 & 0 & 1\end{pmatrix}$; similarly we have the NOT gate X and the CNOT gate C_X. The identity gate I corresponds to an empty wire.

When applying gates in series, we take their ordinary (operator or matrix) product; when applying gates to distinct wires in parallel, we take their tensor product. We can use this approach to derive the output of any classical circuit, e.g. if we have three wires inputting, the top two pass into an AND gate, then the bottom through a NOT gate, and then the two wires we now have through a CNOT gate, then when the input bits are $|1\rangle, |1\rangle, |0\rangle$, the output will be $C_X(I \otimes X)(A \otimes I)|110\rangle$, which we can calculate to be $|10\rangle$. The beauty of this approach is that rather than calculating a huge truth table, we only have a product of operators. Note that the gates in the circuit diagram appear left-to-right, but operators are written right-to-left.

Now, on to quantum. Quantum theory is a general framework for describing physical systems and their evolution. The central part is: the physical state of a closed system is given by a normalised complex vector (within that system's Hilbert space), which evolves unitarily. Our Hilbert spaces will always be finite-dimensional, i.e. ordinary complex vector spaces. We form the quantum version of a bit, the qubit, by simply allowing the vectors in our previous formalism to take any normalised 2-dimensional complex vector as their state, i.e. $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha\\\beta\end{pmatrix}$ for any $|\alpha|^2 + |\beta|^2 = 1$, rather than just $|0\rangle$ and $|1\rangle$. If $\alpha\beta \neq 0$ we say this state is a superposition of $|0\rangle$ and $|1\rangle$.

Note that the corresponding bra is the Hermitian conjugate $\langle\psi| = (|\psi\rangle)^\dagger = \begin{pmatrix}\alpha^\star & \beta^\star\end{pmatrix}$. The states $|0\rangle, |1\rangle$ form a basis for single-qubit states; we call them the computational basis states; for n-qubit states the computational basis consists of all tensor products of $|0\rangle$ and $|1\rangle$.

Unitary evolution is linear, reversible, and norm-preserving. As in the classical case, any unitary evolution or gate can be represented by an operator $|\psi\rangle \to U|\psi\rangle$. However, such an operator is unitary iff $U^\dagger U = UU^\dagger = I$, or equivalently, if its rows/columns as a matrix form an orthonormal basis.

Of the gates we considered earlier, the NOT and CNOT gates are unitary; these are widely used in quantum computation. However, the AND gate is

not unitary; it is irreversible, and changes the number of qubits, so cannot be used in our circuits (In classical physics, even though the dynamics are reversible, we build AND gates (which act by dumping information into the environment). It is possible to construct a similar gate in quantum theory, which will be represented by a super-operator, but this generally takes pure states to mixed states and is generally bad, and not useful for computation). There are various new gates which are not possible in the classical theory, e.g. the Hadamard gate $H = \frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$; it is helpful to define another orthonormal basis $|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$, then $H = |+\rangle\langle 0| + |-\rangle\langle 1|$; also note $X = |+\rangle\langle +| - |-\rangle\langle -|$. Also, we define the Z gate: $Z = |0\rangle\langle 0| - |1\rangle\langle 1|$. This is one of a family of phase gates $R_\phi = |0\rangle\langle 0| + e^{i\phi}|1\rangle\langle 1|$. Finally, the controlled-Z gate on 2 qubits, $C_Z = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes Z$.

We can combine these gates into a quantum circuit, e.g. H followed by Z gives ZH; on input $|0\rangle$ the output state is $ZH|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. But how should we interpret such a superposition state? To understand this, we include ourselves in the quantum state: measuring the output qubit will be some physical process, therefore modelled by a unitary evolution U: say the qubit state is $\alpha|0\rangle + \beta|1\rangle$ and we are in some about-to-measure state $|a\rangle$. In order to recover the classical case, we require $U|0\rangle \otimes |a\rangle = |0\rangle \otimes |b\rangle, U|1\rangle \otimes |a\rangle = |1\rangle \otimes |c\rangle$ where $|b\rangle, |c\rangle$ are the states where we see a 0 or a 1 respectively. We can do this; however, linearity now fixes U in general: $U(\alpha|0\rangle + \beta|1\rangle) \otimes |a\rangle = \alpha|0\rangle|b\rangle + \beta|1\rangle|c\rangle$ - now we are also in the superposition.

This is known as the measurement problem, and is not fully resolved; some popular interpretations are the many-worlds approach, which states that the universe really behaves like this; some form of "hidden variables" notion, which claims that quantum theory is merely an approximation to some deeper underlying theory, or some notion of "collapse" of superpositions of sufficiently large objects. However, all these interpretations agree on the resulting experience: with probability $|\alpha|^2$ you see 0 and the qubit state becomes $|0\rangle$, while with probability $|\beta|^2$ you see 1 and the qubit state becomes 1.

Most approaches tend to introduce axioms for dealing with measurements. Theoretically this is deeply unsatisfying; there is no physical distinction between a measurement and an ordinary interaction. However, pragmatically, we see what happens in the lab. A measurement is described by a set of measurement operators M_k satisfying the completeness relation $\sum_k M_k^\dagger M_k = I$. The measurement gives result k with probability $|M_k|\psi\rangle|^2$, and the resulting state is $\frac{M_k|\psi\rangle}{|M_k|\psi\rangle|}$. The constraint essentially just ensures that probability is conserved: the probability of outcome k is $|M_k|\psi\rangle|^2 = (M_k|\psi\rangle)^\dagger(M_k|\psi\rangle = \langle\psi|M_k^\dagger M_k|\psi\rangle$, so $\sum_k \text{prob}(k) = \langle\psi|(\sum_k M_k^\dagger M_k)|\psi\rangle$ which should always be equal to $\langle\psi|\psi\rangle = 1$. This choice of formalism gives us the best set of "useful" measurements; the simplified "observables" seen in basic QM are certainly insufficient for some cases we will want to cover. Those correspond to projective measurement, where each measurement operator is a projector satisfying $M = M^2 = M^\dagger$, e.g. our computational basis measurement on a single qubit: $M_0 = |0\rangle\langle 0|, M_1 = |1\rangle\langle 1|$. (But note that any of these measurements can be done by interacting with an ancilla and then performing a projective measurement). Note that global phase factors are unmeasurable.

We can now perform a full quantum computation, e.g. if we input two qubits in state $|0\rangle$ and pass the first through a Hadamard gate, then the two through a CNOT gate with the first as control and finally the second through a Z gate, the output state will be $(I \otimes Z)C_X(H \otimes I)|00\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$; thus, measuring the output yields (0 0) or (1 1) with probability $\frac{1}{2}$ each.

Universality

Note that for a general quantum circuit we allow CNOT gates to have the controller and controllee bits non-adjacent; however, restricting these to adjacent bits only makes a polynomial difference to the compute time. (It is possible to construct a "swap" gate by three CNOT gates in succession with the middle one "upside down"). We prepare the input state $|\psi_{\text{in}}\rangle = |0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle$, then apply a series of unitary operations to the state at successive times: $|\psi_{\text{out}}\rangle = U_T U_{T-1} \dots U_1 |\psi_{\text{in}}\rangle$. T is called the circuit depth; each U_t is a tensor product of 1- and 2-qubit gate unitaries, acting in parallel. Finally, we measure the output qubits in the computational basis; we obtain the result $(x_1, \dots, x_n) \in \{0,1\}^N$ with probability $|\langle x_1 x_2 \dots x_n | \psi_{\text{out}}\rangle|^2$.

In the classical model, some simple sets of gates are sufficient to generate all logical operations, e.g. NOT and AND will suffice. If we restrict ourselves to unitary gates we require a three-bit gate, e.g. the Toffoli gate, a "controlled controlled NOT", which is sufficient by itself.

In the quantum case, it is less obvious that we can generate all operations from some simple set of gates, since there is now a continuous space of possible unitary operations, However, we can in fact generate all possible unitary operations on the state space simply by all phase gates $\begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$ (a continuous family), the Hadamard gate $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, and the CNOT gate $|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X$. We shall prove this in four steps: 1) we can construct all 1-bit unitaries using phase gates and Hadamards, 2) We can use these and CNOT gates to form Controlled-U gates, for arbitrary unitary U and arbitrary numbers of control bits. 3) We can modify controlled-U gates to form V which apply a unitary operator U to the subspace spanned by any two computational basis states, and are the identity everywhere else (think about it: a controlled-U gate is such a gate where the two computational basis states are $|1 \dots 10\rangle, |1 \dots 11\rangle$). 4) Any unitary matrix can be expressed as a sequence of V gates each of which acts nontrivially only on a 2D subspace.

For single qubit universality, consider the gates: α phase, then Hadamard, then β phase, then Hadamard, then γ phase: this forms $U = \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 0 & e^{i\gamma} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & e^{i\beta} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix}$ $e^{i\frac{\beta}{2}} \begin{pmatrix} \cos\frac{\beta}{2} & ie^{i\alpha}\sin\frac{\beta}{2} \\ ie^{i\gamma}\sin\frac{\beta}{2} & e^{i(\alpha+\gamma)}\cos\frac{\beta}{2} \end{pmatrix}$. We can see that up to an irrelevant global phase factor, this represents a general single bit unitary: for a unitary matrix, the first column must form a complex vector of norm 1, which the given matrix contains in complete generality, and then the second column is determined up to its relative phase α.

For a controlled-U gate, we first generate a controlled-phase gate: noting that X followed by $-\phi$ phase followed by X is a ϕ phase gawe with a overall phase

factor of $e^{-i\phi}$, $X(|0\rangle\langle 0| + e^{-i\phi}|1\rangle\langle 1|)X = (|1\rangle\langle 1| + e^{-i\phi}|0\rangle\langle 0|) = e^{-i\phi}(|0\rangle\langle 0| + e^{i\phi}|1\rangle\langle 1|)$, we see if we have: $\frac{\phi}{2}$ phase gate on the second qubit, CNOT, $\frac{\phi}{2}$ phase gate on the first qubit and $-\frac{\phi}{2}$ phase bit on the second qubit, CNOT gate, this corresponds to a controlled ϕ phase gate; note the necessity of the phase gate on the top to ensure that the phase factors cancel. (While a global phase does not matter, a phase change only when the CNOT gate is "triggered" would be important).

Since $H^2 = I$ we can now generate any Controlled-U gate analagously to the single-qubit case: controlled-α, Hadamard on the second qubit, controlled-β, Hadamard on the second qubit, controlled-γ, δ on the first qubit gives a general controlled-U. Note the need for the aditional phase gate δ on the control qubit; this allows us to modify the global phase of U (a controlled-U followed by a δ on the control qubit is equivalent to a controlled=$e^{i\delta}U$), which unlike in the single-qubit case, is important, since the U is only triggered some of the time.

We can generate a U-gate controlled by k qubits (i.e. triggered only if all the k control qubits are 1), which we shall call a C_U^k gate, inductively: suppose we have C_U^k gates, then say we have k "top" control lines, a "middle" control line, and a "bottom" controlled line. Then we have a controlled-$\sqrt{U}$ gate, on the bottom, controlled by the middle, followed by a C_X^k gate, controlled by the top, on the middle, then a controlled-$(\sqrt{U})^\dagger$ gate, on the bottom, controlled by the middle, a C_X^k gate, constrolled by the top, on the middle, and finally a $C_{\sqrt{U}}^k$ gate, on the bottom, controlled by the top; the reader may check this always gives the correct outcomes. $\sqrt{U}$ always exists, non-uniquely, as U is unitary: its eigenvalues are always phases, so can be halved.

These controlled-U gates act nontrivially only on two computational basis states, $|1\ldots10\rangle, |1\ldots11\rangle$; in matrix form e.g. a C_U^2 looks like: $\begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & U_{00} & U_{01} \\ & & & & & & U_{10} & U_{11} \end{pmatrix}$.

By permuting the basis states before and after the gate, we can build gates that act as an arbitrary unitary on the subspace spanned by any two basis states. We form our permutation P by: map the first basis state to $|1\ldots11\rangle$ by putting X gates on the bits which are currently $|0\rangle$. Then we move the second basis state from wherever it has ended up to $|1\ldots10\rangle$ by permutations that swap two basis states by a single bit flip and leave all others unchanged (and so we repeatedly swap our basis state with adjacent ones until we reach $|1\ldots10\rangle$, without passing through $|1\ldots11\rangle$). We do these permutations by n-way CNOT gates with X gates surrounding the approprate controls: thus e.g. to swap $|1000\rangle, |1010\rangle$, we place a controlled NOT gate on the third qubit with controls on all three other bits, and X gates on each side of the control on the second and fourth bits. Simply reversing the order of the circuit P gives us the inverse $P^\dagger$. Then the circuit $PC_U^{(n-1)}P^\dagger$ acts by U on the desired subspace and not at all on the other vectors, e.g. for the 3-bit case where we have chosen $P|100\rangle = |111\rangle, P|001\rangle = |110\rangle$, the

combined circuit in matrix form is $V = \begin{pmatrix} 1 & & & & & & & \\ & U_{00} & & & U_{01} & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & U_{10} & & & U_{11} & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{pmatrix}$.

Finally, we can express any unitary U as a product of V-operations, each of which acts nontrivially only on a 2D subspace: given a matrix U, we can form a V gate such that VU has a zero in any one position on the left hand column, and not changing any other entry of the left hand column other than the top left, e.g. $\frac{1}{\sqrt{|u_{00}|^2+|u_{20}|^2}} \begin{pmatrix} u_{00}^\star & 0 & u_{20}^\star & 0 \\ 0 & 1 & 0 & 0 \\ u_{20} & 0 & -u_{00} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_{00} & u_{01} & u_{02} & u_{03} \\ u_{10} & u_{11} & u_{12} & u_{13} \\ u_{20} & u_{21} & u_{22} & u_{23} \\ u_{30} & u_{31} & u_{32} & u_{33} \end{pmatrix} = \begin{pmatrix} v_{00} & v_{01} & v_{02} & v_{03} \\ u_{10} & u_{11} & u_{12} & u_{13} \\ 0 & v_{21} & v_{22} & v_{23} \\ u_{30} & u_{31} & u_{32} & u_{33} \end{pmatrix}$; also the top left entry of the resulting matrix is real positive. By a sequence of such operations we can change all elements of the first column to 0 other than the top left which is now 1; by unitarity, the rows and columns are all normalized vectors; thus, the top row is now all 0 other than the first element. Then, recurse on the $(n-1) \times (n-1)$ submatrix in the bottom right; by induction we eventually obtain a sequence of K V-gates with $V_K V_{K-1} \dots V_1 U = I$; then $U = V_1^\dagger \dots V_K^\dagger$.

Thus, we can construct an arbitrary unitary in terms of such V-gates, and hence from only phase gates, Hadamards and CNOTs. Note this construction is not very efficient; for N qubits we require $O(4^n)$ gates.

Note that our universal gate set is not unique; proving the universality of other possible gate sets is now much easier, since we only need to construct this set from them. For example, CZ, phase gates and Hadamards form a universal set since we can construct a CNOT by a CZ with Hadamards on each side on one qubit.

It is more practical to consider finite sets of gates; for such a set, it is impossible to perfectly recreate every unitary, since there are only countably many possible circuits but uncountably many unitary operations. However, we will consider a set of gates as universal - "approximate universality" - if they can be used to approximate any unitary operation to an arbitrary accuracy, i.e. given U, ϵ we can find a finite sequence of gates $G_M \dots G_1$ such that $\|U - G_M \dots G_1\| \leq \epsilon$, where we use the operator norm $\|A\| = \sup\{|A|\psi\rangle| : |\psi\rangle| = 1\}$. Given an error in the unitary of ϵ, the difference in probability for obtaining any measurement result is at most 2ϵ: $|MU|\psi\rangle|^2 - |MV|\psi\rangle|^2 = \langle\psi|U^\dagger M^\dagger MU|\psi\rangle - \langle\psi|V^\dagger M^\dagger MV|\psi\rangle = \langle\psi|(U-V)^\dagger M^\dagger MU + V^\dagger M^\dagger M(U-V)|\psi\rangle \leq \|U-V\|\|M^\dagger M\|(\|U\| + \|V\|)$; $\|M^\dagger M\|$ is ≤ 1 by completeness, and $\|U\|, \|V\|$ are ≤ 1 by unitarity, so this is $\leq 2\|U-V\|$.

Furthermore, if we approximate each gate independently, the errors grow at most linearly in the number of gates: $\|U_2U_1 - V_2V_1\| = \|U_2(U_1-V_1) + (U_2-V_2)V_1\| \leq \|U_2\|\|U_1-V_1\| + \|U_2-V_2\|\|V\| \leq \|U_1-V_1\| + \|V_2\|$. The main point about this is that it is not exponential; thus, this method is (relatively)

practical. To obtain an overall error of ϵ in a circuit of K gates, we only need to ensure the error on each is less than $\frac{\epsilon}{K}$.

A finite universal set can be obtained by using the previous universal gate set with just a single phase gate (with an angle that is not a rational multiple of π, e.g. 1 radian); by using the phase gate M times we can get arbitrarily close to any other phase gate, with error of order $\frac{1}{M}$; thus obtaining the desired accuracy requires an overhead of $M \sim \frac{K}{\epsilon}$. A better universal set is obtained by choosing our phase gate to have $\phi = \frac{\pi}{4}$; as is proven in Nielsen and Cheung[?], combining this with Hadamards allows us to approximate any phase gate needing only $M \sim \log \frac{K}{\epsilon}$.

Deutsch's Problem

This presents an example where quantum computation is more efficient than classical. As always we are concerned with asymptotic performance; we characterise a problem by n, usually the number of input bits; then we write $g(n) = O(f(n)$ if $\exists c, m$ such that $|g(n)| \leq |cf(n)| \forall n > m$; $g(n) = \Omega(f(n)$ if $\exists c, m$ such that $|g(n)| \geq |cf(n)| \forall n > m$, and $g(n) = \Theta(f(n))$ if $g(n) = O(f(n)), g(n) = \Omega(f(n))$.

An oracle is a "black box" which takes a "question" in the form of an n-bit string and returns an m-bit "answer" which is some function of the input; here we shall restrict ourselves to 1-bit output - "yes/no" answers. To consider oracles in quantum circuits we need to consider a "reversible" oracle; say our oracle inputs $x_1, \ldots, x_n, y$ and outputs $x_1, \ldots, x_n, y \oplus f(\boldsymbol{x})$ where by $a \oplus b$ we mean $(a + b) \mod 2$. This can then be made quantum by replacing the bits with qubits; the oracle will be a unitary transformation $U_f = \sum_{\boldsymbol{x}} \sum_y |\boldsymbol{x}\rangle|y \oplus f(\boldsymbol{x})\rangle\langle\boldsymbol{x}|\langle y|$.

Classically, we would always input $y = |0\rangle$ and a "question" in the $\boldsymbol{x}$ bits, so the oracle's "answer" is given on the final bit which becomes simply $f(\boldsymbol{x})$. However, for the quantum oracle we can use a "trick": set the y qubit to $|-\rangle$, then our oracle U_f will act as a phase gate: $U_f|\boldsymbol{x}\rangle \otimes |-\rangle = (-1)^{f(\boldsymbol{x})}|\boldsymbol{x}\rangle \otimes |-\rangle$. But since phase is a global factor we can view this as acting on the "question" qubits and leaving the "answer" unchanged. Of course, a global phase factor is undetectable; this will induce a detectable change in the output state only when the input is a superposition. The same trick can be used on any controlled gate, e.g. if we feed $|\psi\rangle \otimes |-\rangle$ into a CNOT gate, the output can be seen as $Z|\psi\rangle \otimes |-\rangle$.

The examples in this lecture are "artifical" cases which nevertheless serve as a proof of principle that quantum computation can have an advantage over classical; we shall move on to "real-world" applications later. The first example was an oracle problem found by Deutsch: suppose we're given a reversible function for an unknown 1-bit function f, $U_f = \sum_{x,y \in \{0,1\}} |x\rangle\langle x| \otimes |y \oplus f(x)\rangle\langle y|$. We wish to determine whether $f(x)$ is constant, $f(0) = f(1)$, or "balanced" $(f(0) \neq f(1))$, with the minimum number of oracle queries. The best classical solution to this problem requries two queries; we simply apply f to 0 and 1, and from the results we can calculate $f(0) \oplus f(1)$, which tells us our result; note that we have "done too much work" here, since we can also read off the values of $f(0), f(1)$. Thus we have obtained two bits of information about f where we only really wanted 1.

Using quantum computation, we can solve the problem using only one oracle query: $|\psi_{\text{out}}\rangle = (H \otimes I)U_f(H \otimes H)(I \otimes X)|0\rangle|0\rangle$ will give an output of $(-1)^{f(0)}|f(0) \oplus f(1)\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, i.e. our result can be read off from the first qubit. Note that we still only learn one bit of information about f from one query; we cannot determine $f(0)$ or $f(1)$ without further queries.

We can generalise this to an n-qubit boolean function: we are given the promise that either f is constant, or balanced ($f(\boldsymbol{x}) = 0$ for exactly half the possible $\boldsymbol{x}$s, 1 otherwise), and must determine which (Deutsch-Jozsa). We first apply a NOT gate on the final ("answer") qubit and then Hadamards on all qubits; this is a frequent way to start, because it generates an equal superposition of all inputs followed by a $|-\rangle$: under a set of Hadamards in parallel, $|\mathbf{0}\rangle$ becomes $\frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)\ldots$. Then we apply f to this state, and then Hadamard all our "question" qubits and measure them; if all the outputs are 0 the function is constant, in any other case it must be balanced. The final qubit, the "answer", will always remain $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, so there is nothing to be gained by measuring it. To see that this works, we have that $|\psi_{\text{out}}\rangle$, which we take to be the state immediately before the final measurement, is $(H^{\otimes n} \otimes I)U_f(H^{\otimes n} \otimes H)(I \otimes X)|0\rangle^{\otimes n}|0\rangle = (H^{\otimes n} \otimes I)U_f(H^{\otimes n} \otimes H)|\mathbf{0}\rangle|1\rangle = \frac{1}{\sqrt{2^{n+1}}}(H^{\otimes n} \otimes I)U_f(\sum_{\boldsymbol{x}\in\{0,1\}^n}|\boldsymbol{x}\rangle)(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2^{n+1}}}(H^{\otimes n} \otimes I)(\sum_{\boldsymbol{x}\in\{0,1\}^n}(-1)^{f(\boldsymbol{x})}|\boldsymbol{x}\rangle)(|0\rangle - |1\rangle) = (H^{\otimes n}\frac{1}{\sqrt{2^n}}\sum_{\boldsymbol{x}\in\{0,1\}^n}(-1)^{f(\boldsymbol{x})}|\boldsymbol{x}\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Whilst we could compute this state fully, it is more convenient to now consider measurement: the probability of measuring 0 on all of the first n qubits can be found using the measurement operator $M = |\mathbf{0}\rangle\langle\mathbf{0}| \otimes I$ (we could form a complete set of measurement operators by e.g. $M_{\boldsymbol{x}} = \bigotimes_i |x_i\rangle\langle x_i| \otimes I$, but we only actually need to consider $M_{\mathbf{0}}$ here). The probability of this outcome is $|M|\psi_{\text{out}}\rangle|^2 = |(|\mathbf{0}\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle))(\langle\mathbf{0}|H^{\otimes n}\frac{1}{\sqrt{2^n}}\sum_{\boldsymbol{x}\in\{0,1\}^n}(-1)^{f(\boldsymbol{x})}|\boldsymbol{x}\rangle)|^2$; the first large bracket here is a normalised vector, and so applying the Hs this is $|(\frac{1}{\sqrt{2^n}}\sum_{\boldsymbol{y}\in\{0,1\}^n}\langle\boldsymbol{y}|)(\frac{1}{\sqrt{2^n}}\sum_{\boldsymbol{x}\in\{0,1\}^n}(-1)^{f(\boldsymbol{x})}|\boldsymbol{x}\rangle)|^2 = \frac{1}{2^n}\sum_{\boldsymbol{y}}\sum_{\boldsymbol{x}}(-1)^{f(\boldsymbol{x})}\delta_{\boldsymbol{x}\boldsymbol{y}}|^2 = |\frac{1}{2^n}\sum_{\boldsymbol{x}}(-1)^{f(\boldsymbol{x})}|^2$, which is 1 for f constant and 0 for f balanced.

For a classical oracle, to reliably determine whether f is constant or balanced requires $2^{n-1} + 1$ queries (sock problem); the quantum case requires only one query. However, in the classical case after sampling k random values of x, the probability of an error, ϵ, is exponentially small: the only way we can be wrong is if we declare the function to be constant when it is in fact balanced, which happens with probability $\epsilon = \frac{2^{n-1}-1}{2^n-1}\frac{2^{n-1}-2}{2^n-2}\ldots\frac{2^{n-1}-k+1}{2^n-k+1} \leq (\frac{1}{2})^{k-1}$. So if we permit a small error probability ϵ, our classical circuit will require only $1 + \log_2\frac{1}{\epsilon} = \Omega(1)$ queries.

For a problem where there is a substantial difference between the quantum and even a probabilistic classical solution, we have the Bernstein-Vazirani problem: we are given an oracle for an n-bit boolean function $f(\boldsymbol{x})$ and the promise that $f(\boldsymbol{x}) = \boldsymbol{a}\cdot\boldsymbol{x}$ (modulo 2), and must find $\boldsymbol{a}$ in the least number of queries. For the quantum case the exact same circuit as for Deutsch-Jozsa outputs $\boldsymbol{a}$ in the first n qubits (consider a measurement operator $M = |a_1\rangle\langle a_1 \otimes \cdots \otimes |a_n\rangle\langle a_n| \otimes I$, then since $H^{\otimes n}|\boldsymbol{a}\rangle = \sum_{\boldsymbol{y}\in\{0,1\}^n}(-1)^{\boldsymbol{a}\cdot\boldsymbol{x}}|\boldsymbol{y}\rangle$, we can calculate $|M|\psi_{\text{out}}\rangle|^2 = |(\frac{1}{\sqrt{2^n}}\sum_{\boldsymbol{y}\in\{0,1\}^n}(-1)^{\boldsymbol{a}\cdot\boldsymbol{x}}\langle\boldsymbol{y}|)(\sum_{\boldsymbol{x}\in\{0,1\}^n}(-1)^{\boldsymbol{a}\cdot\boldsymbol{x}}|\boldsymbol{x}\rangle)|^2 = |\frac{1}{2^n}\sum_{\boldsymbol{x}}1|^2 = 1$. In the classical case, each oracle query gives us 1 bit of information; we cannot do any better than trying $\boldsymbol{x} = (1, 0, \ldots, 0), (0, 1, 0, \ldots, 0)$ etc. Thus we require n queries

to determine $\boldsymbol{a}$ with certainty; even in the probabilistic case, if we are permitted k queries the best we can do is to discover k bits of $\boldsymbol{a}$; we shall then have to guess the remaining $n-k$ bits, meaning the error probability $\epsilon = (\frac{1}{2})^{n-k}$; thus even if we allowed some fixed error probability ϵ we still need $k \geq n - \log \frac{1}{\epsilon} = \Omega(n)$ uses of the oracle.

Grover's Search Algorithm

Consider a database of 2^n entries. Searching a sorted database can be done in $O(n)$ steps (by binary chop); however, searching an unsorted database is much harder, requiring $O(2^n)$ entries. E.g. it is easy to find a given name in the 'phone book, but very hard to find a given telephone number.

Suppose $\boldsymbol{x}$ indexes the entries of our database, and we are seeking to find a record for which a particular field $F(\boldsymbol{x}) = A$. For now suppose there is a unique record $\boldsymbol{a}$ for which this holds. We can consider the search as an oracle problem: each query asks "is $\boldsymbol{x}$ the correct entry", returning 1 if $\boldsymbol{x} = \boldsymbol{a}$, 0 otherwise. As always, we wish to discover $\boldsymbol{a}$ in as few oracle queries as possible.

Grover's search algorithm circuit consists of: we have first a NOT gate on the last "answer" qubit, then a row of Hadamards, as usual. Then we have several iterations of a Grover operation G, before measuring all the "question" qubits. Each G consists of one call to the oracle f_a, and then a particular phase gate $V_{|\psi\rangle}$ on the "question" qubits. Therefore we require one oracle query for each G. This is a probabilistic algorithm; the final measurement will give $\boldsymbol{a}$ with high probability, but not certainly.

Set $|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{\boldsymbol{x} \in \{0,1\}^n} |\boldsymbol{x}\rangle$, the equally weighted superposition over all basis states. $V_{|\psi\rangle} = 2|\psi\rangle\langle\psi| - I$ - it leaves $|\psi\rangle$ invariant and gives all states orthogonal to it a -1 phase factor. Since the "answer" qubit of the oracle is always in a state $|-\rangle$, its action on the "question" qubits can be described by a similar phase gate: $U_{f_a} = I - 2|\boldsymbol{a}\rangle\langle\boldsymbol{a}| = -V_{|\boldsymbol{a}\rangle}$ (it applies a -1 phase factor to $|\boldsymbol{a}\rangle$, for which the oracle response is 1, and leaves orthogonal states, for which the oracle response is 0, unchanged).

Thus, if we ignore the "answer" qubit for now, the effective circuit on the remaining qubits is a row of Hs, then iterated $G = V_{|\psi\rangle}(-V_{|\boldsymbol{a}\rangle})$, then finally measurement. Observe that $G = (2|\psi\rangle\langle\psi| - I)(I - 2|\boldsymbol{a}\rangle\langle\boldsymbol{a}|)$ preserves the 2D subspace spanned by $|\psi\rangle, |\boldsymbol{a}\rangle$; it does not move states out of this subspace. Remember $|\psi\rangle, |\boldsymbol{a}\rangle$ are not orthogonal; therefore, define $|\boldsymbol{a}^\perp\rangle$ to be such that $|\boldsymbol{a}\rangle, |\boldsymbol{a}^\perp\rangle$ are an ON basis for the aforementioned subspace: $|\boldsymbol{a}^\perp\rangle = \frac{1}{\sqrt{2^n - 1}} \sum_{\boldsymbol{x} \neq \boldsymbol{a}} |\boldsymbol{x}\rangle$. Note $|\psi\rangle = \frac{1}{\sqrt{2^n}}|\boldsymbol{a}\rangle + \sqrt{\frac{2^n-1}{2^n}}|\boldsymbol{a}^\perp\rangle$.

If we restrict our attention to the subspace, $-V_{|\boldsymbol{a}\rangle} = V_{|\boldsymbol{a}^\perp\rangle}$. Our states will always take the form $|\phi\rangle = \alpha|\boldsymbol{a}\rangle + \beta|\boldsymbol{a}^\perp\rangle$ with α, β real, so we can view this subspace as the 2D real plane. Then, geometrically, $V_{|\boldsymbol{a}^\perp\rangle}$ represents a reflection in the $|\boldsymbol{a}^\perp\rangle$ axis; similarly, $V_{|\psi\rangle}$ is a reflection in the $|\psi\rangle$ axis. Thus their combined effect on states in the subspace is a rotation by angle 2θ, where θ is the angle between $|\psi\rangle, |\boldsymbol{\alpha}^\perp\rangle$; $\theta = \sin^{-1} \frac{1}{\sqrt{2^n}}$ (recall $|\psi\rangle = \frac{1}{\sqrt{2^n}}|\boldsymbol{a}\rangle + \sqrt{\frac{2^n-1}{2^n}}|\boldsymbol{a}^\perp\rangle$; thus if we let the $|\boldsymbol{a}^\perp\rangle$ axis be the "x" axis, $|\psi\rangle$ points mostly to the right and slightly up.

Thus, iterating the Grover operation, we rotate the initial state $|\psi\rangle$ "up"

towards the $|\boldsymbol{a}\rangle$ axis, in steps of 2θ. For the final result to be as close as possible to $|\boldsymbol{a}\rangle$, we iterate G k = closest integer to $\frac{\frac{\pi}{2}-\theta}{2\theta} \leq \frac{\pi}{4}\sqrt{2^n} = O(2^{\frac{n}{2}})$ times. The state immediately before measurement is then at angle $\alpha \leq \theta$ from the $|\boldsymbol{a}\rangle$ state. Measuring in the computational basis our outcomes are $|\boldsymbol{a}\rangle$ with probability $\cos^2\alpha$, or any other state with equal probability $\frac{\sin^2\alpha}{2^n-1}$ (recall $|\boldsymbol{\alpha}^\perp\rangle = \sum_{\boldsymbol{x}\neq\boldsymbol{a}}|\boldsymbol{x}\rangle$). Thus the probability of an error $\epsilon \leq \sin^2\theta = \frac{1}{2^n}$.

Grover's algorithm requires $O(2^{\frac{n}{2}})$ queries (even if we require a smaller ϵ, we can just repeat k times and take a "majority vote"), while the best classical search algorithm requires $O(2^n)$, even if we allow it to be probabilistic; thus this achieves a quadratic speedup.

If there are M different solutions $A = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_M\}$ satisfying our search criteria, then our search oracle is f_A, where $f_A(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in A$, 0 otherwise. We apply Grover's algorithm identically, except that in place of $|\boldsymbol{a}\rangle, |\boldsymbol{a}^\perp\rangle$ we have $|A\rangle = \frac{1}{\sqrt{M}}\sum_{\boldsymbol{x}\in A}|\boldsymbol{x}\rangle, |A^\perp\rangle = \frac{1}{\sqrt{2^n-M}}\sum_{\boldsymbol{x}\notin A}|\boldsymbol{x}\rangle$ - evenly weighted superpositions of correct and incorrect answers. The only difference from before is that the angle of the state $|\psi\rangle$ from $|A^\perp\rangle$ changes; now $\theta = \sin^{-1}\sqrt{\frac{M}{2^n}}$; thus the required number of Grover operations k is different. When we measure the "question" qubits we will obtain a random element of A, i.e. a random "correct" answer, except for an error probability $\epsilon \leq \frac{M}{2^n}$. If M is unknown we might think this would cause problems, since θ and hence k depend on M, but there is a "quantum counting" algorithm, which we may see later in the course, which can determine M in $O(2^{\frac{n}{2}})$ oracle queries, so our quadratic speedup is retained.

Although we have phrased this section in terms of databases, actually building a database which functioned like this would be a hard task - it is necessary to be able to query the database without it measuring the query $|\boldsymbol{x}\rangle$, as that would collapse the superposition. We can imagine a database which worked with e.g. photons and beam splitters to perform such a measurement, but this is not yet technically feasible. However, such a "database" is a good model for an interesting class of mathematical problems - in particular, problems where it is hard to find a solution, but easy to check whether a particular candidate is a solution or not, so the best approach to finding a solution really is by testing all possibilities. E.g. if we want to factor a number J which is a product of two large primes, we can see this as an oracle problem where the oracle is the (quite simple) circuit which determines whether x is a factor of J; to find a factor classically in the naive way, one would have to "query" this "oracle" $O(J^{\frac{1}{2}})$ times, to test all possible factors up to $J^{\frac{1}{2}}$, wheras using Grover's algorithm we can do so with $O(J^{\frac{1}{4}})$ queries. Of course, this is in fact worse than the best classical approaches to the problem, and we shall see later there is a far better quantum method for factorisation.

Classical Computational Complexity

Thus far we have considered oracle problems; they make it easy to demonstrate a separation between quantum and classical approaches, but give slightly artificial problems, and cannot be used to express the difficulty of solving more general mathematical problems.

Complexity classes are usually defined in terms of Turing machines. There are several different but equivalent definitions of such; we take it to consist of

a tape, infinite in both directions, each "square" of which contains a symbol from a finite alphabet A (with a distinguished element "blank"), a "read-write head" pointing towards a particular square on the tape, a "state register" which contains states from a finite set S, one of which is a "starting" state and one or more of which are "halting" states, and a "table" $(S, A) \to (S, \{R, L\}, A)$, which maps from the machine's current state and the symbol "read" from the current square to a new symbol to "write" to the current square (possibly blank), a direction "left" or "right" for the tape to be shifted by one square, and a new state of the machine. The machine starts in the start state and with the head pointing towards a particular position on the tape; the input to the algorithm consists of finitely many non-blank squares on the tape. The "rules" in the table are applied repeatedly until the machine reaches a "halt" state; at this point it stops, and its output is considered to be whatever is now written on the tape.

A Turing machine of this form would usually be constructed to solve a specific problem, but it is relatively easy to construct a universal Turing machine, which reads as input a description of another Turing machine along with its input, and simulates that Turing machine, outputting what it would output.

The Church-Turing Thesis is that any function we consider as being computable by an algorithm is computable by a Turing machine. Quantum theory does not challenge this; we only claim that it is possible to compute computable functions more efficiently using quantum computation.

We want a notion of how "hard" a problem is to solve on a computer which is indepedent of the precise implementation details and the specific computer used. We consider a problem to be solvable efficiently if the quantity of resources required scales polynomially with the problem size: $f(n) = \text{poly}(n)$ if there is a k such that $f(n) = O(n^k)$. This is obviously a coarse measure (e.g. for practical problem sizes, $O(2^{\frac{n}{1000}})$ might be preferable to $O(n^{1000})$), but in practice works well as a classification. Its great advantage is that it is independent of the precise details of the computational model; different computational models can generally simulate each other with polynomial overhead; we say they are "equivalent" if so. In a slight abuse of notation, we tend to call any problem which requires greater than polynomial resources "exponential", even though some functions e.g. $O(n^{\log n})$ are not truly exponential.

For examples of the universality of this definition, Turing machines appear able to simulate any other classical deterministic algorithm with an overhead at most polynomial in the number of elementary operations, e.g. they can provably simulate classical circuits with such efficiency. Turing machines with two tapes give only a polynomial speedup over a one-tape machine; only allowing gates to act "locally" (i.e. between neighbouring bits) in a circuit only incurs a polynomial overhead. It is believed to be false that a classical Turing machine can simulate a quantum circuit in polynomial time, but as with many problems in this field this is not proven - see later.

Many computational tasks can be formulated as decision problems, which are clean and easy to state. In these, there is a variable-sized input, but the answer is always a "yes" or "no". We can describe them more formally using languages: a language L is a subset of all possible strings composed of characters from some finite alphabet A, e.g. in the binary alphabet $A = \{0, 1\}$, the language of odd numbers is $L = \{1, 01, 11, 101, \dots\} \subset \{0, 1, 00, 01, 10, 11, \dots\}$. The language represents those inputs for which the answer to the decision problem

(in this case, "is x odd?") is "yes".

The complexity class P is defined to be the set of languages which can be decided by a Turing machine in polynomial time: a language L is decided by a Turing machine if, when given x as input, it reaches the halting state "accept" if $x \in L$ and the halting state "reject" if $x \notin L$. A langiage is decided in polynomial time if for all strings x, the Turing machine halts in the correct state after poly1($|x|$) steps. Intuitively, P represents those decision problems which can be solved efficiently on a classical deterministic computer; it is hard to prove a problem does not lie in P, since we need to prove that no possible algorithm could solve it in polynomial time.

As a demonstration of the model independence of complexity classes, we could also define P based on classical circuits; this is somewhat clunkier, because we need different circuits for different input sizes. A language L (with alphabet $\{0,1\}$; this is not a real problem as languages with larger alphabets can be encoded in binary with polynomial overhead) is in P if there is a uniformly-generated family of polynomial-size classical circuits $\{C_n\}$ such that when x is input into the circuit $C_{|x|}$, it outputs 1 if $x \in L$ and 0 if $x \notin L$. (A circuit C_n is polynomial-size if it contains poly(n) gates, each of which acts on $O(1)$ bits. A circuit family $\{C_n\}$ is uniformly-generated if there is a turing machine which inputs (a representation of) n and outputs (a description of) the circuit C_n in polynomial time. For compatibility with reversible/quantum computers (where we don't have the "fanout" gate where an input $|\psi\rangle \mapsto |\psi\rangle \otimes |\psi\rangle$ (in fact this is impossible by the no cloning theorem)), the circuit is also allowed poly(n) "working" bits prepared in some fixed computational basis state independent of x (usually 0), and permitted to output some "garbage" bits alongside the "true" output). (To see that the uniformity condition is important, consider the decision problem: is x the middle prime in the range $1, 2^{|}x|$? This appears "hard" to solve, at least in terms of a Turing machine, but if we were allowed to generate our circuit for $|x|$ arbitrarily, we could easily solve it in polynomial time by "precomputing" the middle prime in that range, so the circuit would simply be "compare this number with x, if they are equal output yes, otherwise output no".)

Somewhat surprisingly, randomness seems to be useful in solving computational problems. Thus we define the complexity class BPP: bounded error, probabilistic, polynomial time. This is the set of language which can be decided in polynomial time on a probabilistic Turing machine (i.e. one which is permitted to have state updates conditioned on a coin toss; of course there is an equivalent definition in terms of circuits permitted to input some random bits) with error probability at most $\frac{1}{3}$ (this is arbitrary; any ϵ in the range $0 < \epsilon < \frac{1}{2}$ does not change the class, since we may use "majority voting" to obtain a smaller error probability with only a polynomial overhead (in fact only a constant factor's difference; the probability of an error drops like $2^{-\Omega(k)}$, the "Chernoff bound"). Intuitively, it contains those problems which can be solved efficiently on a probabilistic classical computer. The "strong Church Turing thesis" claims that "all effectively solvable problems are in BPP": any model of computation can be simulated by a probabilistic turing machine with at most a polynomial overhead. Quantum theory does seem to violate this.

There are some languages believed to be outside BPP; however, this cannot be proven. For example, the factoring decision problem: given two n-bit integers x and $y < x$, does x have a nontrivial factor less than y (the input is a composite

of two integers, but we're grown-up enough to handle that). This is a somewhat contrived version of the problem in order to have the form of a decision problem, but notice that if we had an efficient solution to this, we could also factor numbers efficiently, by using binary search, which only incurs a polynomial overhead. Thus "we get the full power of the problem form the decision version", and this is a general phenomenon.

It is believed that this decision problem is hard to solve classically, but it is easy to prove that the answer is "yes": given a factor, verifying that it divides exactly is an easy decision problem, in P. We call the factor a witness to the decision problem. The complexity class of decision problems with witnesses checkable in polynomial time is called NP (for non-deterministic polynomial time; it is possible to define it in terms of non-deterministic Turing machines, although we shall not do so in this course). Formally, a language L lies in NP if there is a Turing machine such that if $x \in L$ there exists a witness w such that if we input "x-blank-w" it reaches the "accept" halting state in time poly($|x|$), and if $x \notin L$ then for all strings w, when we input "x-blank-w" it reaches the "reject" halting state in time poly($|x|$). (Note that it is implicit that our witness must have size poly($|x|$), since the Turing machine can only read one bit in each timestep).

$PSPACE$ is the set of languages decidable by a Turing machine using only polynomial space on the tape: for an input x, the algorithm must not move the tape more than poly($|x|$) squares from its starting position. Intuitively, this corresponds to those decision problems which can be solved in finite time on a deterministic polynomial-sized computer. We would like to also define this for a circuit by saying that the conditions are as for P, but with no constraint on the total number of gates; however, at any given time there may be at most poly(n) bits in the circuit. However, while this is an intuitively correct definition, we cannot make it formally valid since the uniformity condition breaks down.

To summarise the known relations between the complexity classes: P is contained in $PSPACE$, as a Turing machine only moves one square per timestep, so any computation done in poly(n) timesteps will necessarily use only poly(n) space. P is contained in NP as we could just ignore the witness input; NP is contained in PSPACE as we could try out all possible witnesses sequentially (we shall not prove this in detail at this stage, since it is very similar to the quantum case, which we shall cover later). P is contained in BPP as we could just ignore the random inputs (in the circuit model; in the Turing machine model we would just not use any probabilistic state changes). BPP is contained in PSPACE as we could try out each possible value for the random variables sequentially and count the number which accept. The relation between BPP and NP is not proven; nor is it proven that any of the mentioned complexity classes are distinct. It is generally believed that P does not equal PSPACE; it is thought that BPP is probably equal to P since efficient (polynomial-time) good pseudorandom number generators seem to exist, and so any probabilistic polynomial-time algorithm can be made deterministic with only polynomial overhead by combining in with a pseudorandom number generator, but this is not proven. The most famous open problem is whether P=NP; there is a million dollar prize on offer for any proof one way or the other on this.

Quantum Computational Complexity

The complexity class BQP - bounded error, quantum, polynomial-time - describes those decision problems which can be solved efficiently [with high probability] on a quantum computer; the formal definition is usually given using the quantum circuit model, although it can also be defined in terms of Turing machines: A language L (with alphabet $\{0,1\}$) is in BQP if there exists a uniform family of polynomial-sized quantum circuits $\{C_n\}$ such that when $|\boldsymbol{x}\rangle|\mathbf{0}\rangle$ is inupt to the circuit $C_{|\boldsymbol{x}|}$, the probability of obtaining 1 in a computational basis measurement of the first output is $\geq \frac{2}{3}$ if $\boldsymbol{x} \in L, \leq \frac{1}{3}$ if $\boldsymbol{x} \notin L$. Circuits are composed of gates from some universal set; we require the matrix elements of all gates must be efficiently computable (the nth gate must be obtainable by a Turing machine in time poly(n). As with BPP, the choice of $\frac{2}{3}$ as the required probability is arbitrary.

For a Quantum Turing Machine, we associate each configuration of a normal Turing machine with a different orthonormal vector. Evolution is given by a unitary operator U, which acts on the internal state and current square, and moves the tape head by one step. We also require the entries of U to be efficiently computable. After each update, we partially measure the internal state to see whethere it is a halting state (and whether it accepts or rejects); it is essential that we do not fully measure it with a basis as that would collapse this to a probabilistic classical Turing machine. At a general time-step the machine will be in a complex superposition of classical states. We can define BQP as the set of languages which can be decided in polynomial time on a quantum Turing machine with error probability at most $\frac{1}{3}$.

Clearly, P is a subset of BQP: we can restrict to reversible gates in the circuit definition of P without changing the complexity class by using ancillas; these reversible gates can be used in the quantum circuit, then we simply use qubits in computational basis states.

BPP is also a subset of BQP: circuits in BPP are permitted poly(n) random bits as input. We simulate this by adding poly(n) ancillas and passing these through Hadamard gates, then connecting these to the circuit's random inputs. This does give the same results: the input state corresponding to inputs of $\boldsymbol{x}$, working bits, and random bits is $|\boldsymbol{x}\rangle|\mathbf{0}\rangle(\frac{1}{\sqrt{2^k}}\sum_{\boldsymbol{r}\in\{0,1\}^k}|\boldsymbol{r}\rangle)$. The classical reversible circuit U then permutes the computational basis states such that $U|\boldsymbol{x}\rangle|\mathbf{0}\rangle|\boldsymbol{r}\rangle$ has 1 as its first qubit iff $\boldsymbol{r}$ is one of the random inputs for which the circuit would give an output of 1, 0 otherwise; when we measure, the probability of obtaining the result 1 is thus precisely the proportion of possible random inputs for which the classical circuit would output 1, and similarly for 0.

BQP is contained in PSPACE. The naive way to simulate a computation - storing the components of the state vector in the computational basis, $\langle\boldsymbol{x}|\psi\rangle$, and updating these each time a gate is applied, could, interestingly, be done efficiently in terms of time, if we had a parallel architecture (e.g. a circuit model rather than a Turing machine): each gate update can be done in constant time, hence the simulation may run in poly(n) time. However, to simulate an n-bit quantum computation we must store 2^n complex numbers (the coefficients of each possible state), and hence require $O(2^n)$ bits of working memory, thus this does not show that BQP is in PSPACE.

We can simulate quantum computation using only polynomial memory but

taking exponentially long time. Consider a quantum circuit on n qubits with depth T; the amplitude for obtaining the final computational basis state $|\boldsymbol{y}_T\rangle$, if we start from the basis state $|\boldsymbol{y}_0\rangle$, is $\langle \boldsymbol{y}_T|U_T U_{T-1}\dots U_1|\boldsymbol{y}_0\rangle$; expanding in the computational basis between each pair of unitaries this is $\langle \boldsymbol{y}_T|U_T \sum_{\boldsymbol{y}_{T-1}} |\boldsymbol{y}_{T-1}\rangle\langle \boldsymbol{y}_{T-1}|U_{T-1}\cdots \sum_{\boldsymbol{y}_1} |\boldsymbol{y}_1\rangle\langle \boldsymbol{y}_1|U_1$ which we can express as a "sum over paths" $\sum_{\boldsymbol{y}_1,\dots,\boldsymbol{y}_{T-1}} \langle \boldsymbol{y}_T|U_T|\boldsymbol{y}_{T-1}\rangle\langle \boldsymbol{y}_{T-1}|U_{T-1}|\boldsymbol{y}_{T-2}\rangle\dots\langle \boldsymbol{y}_1|U_1|\boldsymbol{y}_0\rangle$; the sum is over all possible values of each of the $\boldsymbol{y}_j$. The complex number $\langle \boldsymbol{y}_k|U_k|\boldsymbol{y}_{k-1}\rangle$ can be easily computed in polynomial space by using the matrix representation of U_k; by computing each of these in turn and multiplying them in an accumulator, we can compute the product $\langle \boldsymbol{y}_T|U_T|\boldsymbol{y}_{T-1}\rangle\langle \boldsymbol{y}_{T-1}|U_{T-1}|\boldsymbol{y}_{T-2}\rangle\dots\langle \boldsymbol{y}_1|U_1|\boldsymbol{y}_0\rangle$ corresponding to a given "path" in space poly(n). We can store the sequence $(\boldsymbol{y}_1,\dots,\boldsymbol{y}_T)$ in space $nT = \text{poly}(n)$, so we can calculate our amplitude for a given final state $\langle \boldsymbol{y}_T|\psi\rangle$ by summing over all possible $(\boldsymbol{y}_1,\dots,\boldsymbol{y}_{T-1})$ (the calculation for any fixed sequence can be done in polynomial space, then we add the result to an accumulator and reuse the same working bits for the next value of the sequence). Finally, we can iterate through all possible values of $\boldsymbol{y}_T$ for which the first qubit is in the state $|\boldsymbol{1}\rangle$, summing $|\langle \boldsymbol{y}_T|\psi\rangle|^2$ in an accumulator, and thus obtain the probability of the measurement of the first qubit giving output 1.

The time required for this is dominated by running through all values of $\boldsymbol{y}_1,\boldsymbol{y}_2,\dots,\boldsymbol{y}_{T-1}$; there are 2^{nT} possible values so this will take time $O(2^{nT})$ and is thus exponential in the circuit size. The space requirement is actually dominated by the space taken to store $\boldsymbol{y}_1,\boldsymbol{y}_2,\dots,\boldsymbol{y}_{T-1}$, which is polynomial in n; thus BQP is contained in PSPACE. This also shows that anything computable on a quantum computer is computable (perhaps inefficiently) on a classical computer, thus supporting the Church-Turing thesis.

The relationship between BQP and NP is not proven. BQP contains some problems (e.g. factoring) which are known to be in NP but thought to lie outside BPP. However, it is thought that BQP does not contain NP. An "easy" way to prove $NP \subset BQP$ would be to show that an NP-complete problem lay in BQP; there is a large class of "NP-complete" problems for which any problem in NP can be efficiently (i.e. in polynomial time) transformed into that problem. Proving $NP \not\subseteq BQP$ would be a strong result, since it would imply $P \neq NP$ and thus $PSPACE \neq P$.

Simon's Algorithm and the Quantum Fourier Transform

Simon's algorithm is a solution to an oracle problem, and the first to demonstrate an exponential speedup in quantum over (probabilistic) classical computation. It provides the inspiration for Shor's algorithm. Oracle problems of course "do not tell the whole story", but do suggest a separation between complexity classes.

Simon's problem: We are given the oracle for an n-bit to $(n-1)$-bit function $f(\boldsymbol{x})$ and the promise that $f(\boldsymbol{x}) = f(\boldsymbol{y}) \Leftrightarrow \boldsymbol{y} = \boldsymbol{x} \oplus \boldsymbol{a}$ for some fixed $\boldsymbol{a}$. Determine $\boldsymbol{a}$ in the minimal number of oracle queries. We consider the oracle reversible as before, but with n "question" and $n-1$ "answer" bits.

To determine $\boldsymbol{a}$ classically, we have no better option than: sample $f(\boldsymbol{x})$ for various $\boldsymbol{x}$, until we find two for which the answers are the same, i.e. $f(\boldsymbol{x}) = f(\boldsymbol{y})$;

then we declare $\boldsymbol{a} = \boldsymbol{x} \oplus \boldsymbol{y}$. Unfortunately, this requires $\Omega(2^{\frac{n}{2}})$ queries: note for every pair of values $\boldsymbol{x}, \boldsymbol{y}$ with $f(\boldsymbol{x}) \neq f(\boldsymbol{y})$, all this tells us is that $\boldsymbol{a} \neq \boldsymbol{x} \oplus \boldsymbol{y}$; thus after k non-matching queries we can eliminate less than k^2 possible values of $\boldsymbol{a}$, leaving over $2^n - k^2$ possibilities. The probability of obtaining a match on the $(k+1)$th query is then $P_{k+1} \leq \frac{k}{2^n - k^2}$. Summing these, plus the probability of guessing $\boldsymbol{a}$ correctly if there is no match, the probability of correctly determining $\boldsymbol{a}$ in k queries is $P \leq \sum_{i=1}^{k} P_i + \frac{1}{2^n - k^2} \leq \frac{\frac{k(k-1)}{2}+1}{2^n - k^2} \leq \frac{k^2}{2^n - k^2}$; thus for a probability of success greater than $1 - \epsilon$ we require $k \geq 2^{\frac{n}{2}} \sqrt{\frac{1-\epsilon}{2-\epsilon}} = \Omega(2^{\frac{n}{2}})$.

Using a quantum method we can solve the problem (with high probability) in only $O(n)$ queries: consider the circuit where we input all $|0\rangle$s, then Hadamard the "question" qubits. We pass through the oracle, then Hadamard the "question" qubits again, finally measuring the "question" qubits only. (Note that unlike in previous cases we do <u>not</u> X and Hadamard the "answer" qubits to place them in state $|-\rangle$; rather, we let them remain in state $|0\rangle$). The final state $|\psi\rangle = (H^{\otimes n} \otimes I) U_f (H^{\otimes n} \otimes I) |\mathbf{0}\rangle|\mathbf{0}\rangle = \frac{1}{\sqrt{2^n}} (H^{\otimes n} \otimes I) U_f \sum_{\boldsymbol{x} \in \{0,1\}^n} |\boldsymbol{x}\rangle|\mathbf{0}\rangle = \frac{1}{\sqrt{2^n}} (H^{\otimes n} \otimes I) \sum_{\boldsymbol{x}} |\boldsymbol{x}\rangle|f(\boldsymbol{x})\rangle = \frac{1}{2^n} \sum_{\boldsymbol{x}} \sum_{\boldsymbol{k} \in \{0,1\}^n} (-1)^{\boldsymbol{x} \cdot \boldsymbol{k}} |\boldsymbol{k}\rangle|f(\boldsymbol{x})\rangle$.

The probability of measuring the result $\boldsymbol{r}$ on the first n bits is found using $M_r = |\boldsymbol{r}\rangle\langle\boldsymbol{r}| \otimes I$; $|M_r|\psi\rangle|^2 = \langle\psi|M_r|\psi\rangle = \frac{1}{2^{2n}} \sum_{\boldsymbol{x}', \boldsymbol{k}' \in \{0,1\}^n} (-1)^{\boldsymbol{x}' \cdot \boldsymbol{k}'} \langle \boldsymbol{k}'|\langle f(\boldsymbol{x}')|(|\boldsymbol{r}\rangle\langle\boldsymbol{r}| \otimes I) \sum_{\boldsymbol{x}, \boldsymbol{k}} (-1)^{\boldsymbol{x} \cdot \boldsymbol{k}} |\boldsymbol{k}\rangle|f(\boldsymbol{x})\rangle = \frac{1}{2)^{2n}} \sum_{\boldsymbol{x}, \boldsymbol{x}'} (-1)^{\boldsymbol{x}' \cdot \boldsymbol{r} + \boldsymbol{x} \cdot \boldsymbol{r}} \langle f(\boldsymbol{x}')|f(\boldsymbol{x})\rangle = \frac{1}{2^{2n}} \sum_{\boldsymbol{x}} (-1)^{\boldsymbol{x} \cdot \boldsymbol{r} + \boldsymbol{x} \cdot \boldsymbol{r}} + (-1)^{(\boldsymbol{x} \oplus \boldsymbol{a}) \cdot \boldsymbol{r} + \boldsymbol{x} \cdot \boldsymbol{r}} = \frac{1}{2^{2n}} \sum_{\boldsymbol{x}} 1 + (-1)^{\boldsymbol{a} \cdot \boldsymbol{r}} = \frac{1}{2^{n-1}}$ for $\boldsymbol{a} \cdot \boldsymbol{r} = 0$, 0 otherwise. Thus we measure a random bitstring $\boldsymbol{r}$ satisfying $\boldsymbol{a} \cdot \boldsymbol{r} = 0$.

Each such $\boldsymbol{r}$ tells us that some sum of the bits in $\boldsymbol{a}$ is 0; with $(n-1)$ linearly independent such $\boldsymbol{r}$, we can uniquely determine $\boldsymbol{a} \neq \mathbf{0}$. We can obtain this information in $O(n)$ queries with high probability:

Given k lineraly indepedent vectors, their span is 2^k vectors. Since there are 2^{n-1} vectors for which $\boldsymbol{a} \cdot \boldsymbol{r} = 0$, the probability that the $k+1$th vector is linearly indepedent from the rest is $p_{k+1} = 1 - \frac{2^k}{2^{n-1}}$. Thus the probability that the first $n-1$ vectors are all linearly independent is $p = (1 - \frac{1}{2^{n-1}})(1 - \frac{1}{2^{n-2}}) \ldots (1 - \frac{1}{2}) \geq (1 - (\frac{1}{2^{n-1}} + \frac{1}{2^{n-2}} + \cdots + \frac{1}{2^2}))\frac{1}{2} \geq \frac{1}{4}$. Since we can easily determine whether we have found $n-1$ linearly independent vectors, we can achieve success probabilities arbitrarily close to 1 with $O(n)$ oracle queries - even naively repeating the entire process if there is some linear dependence among our $n-1$ vectors suffices to show this works. (In fact, if we are a little more careful in our proof, $(n-1) + \log \frac{1}{\epsilon}$ queries will obtain $n-1$ linearly independent vectors with probability $1 - \epsilon$).

Since the remainder of the quantum algorithm can be done in polynomial time, we can solve Simon's problem with a small probability of error in polynomially many oracle queries on a quantum computer, while a classical approach takes $\Omega(2^{\frac{n}{2}})$ queries.

Simon's work inspired the period-finding algorithm which forms the heart of Shor's algorithm; this finds the value of $\boldsymbol{a}$ if we are given $f(\boldsymbol{x}) = f(\boldsymbol{x} + \boldsymbol{a})$.

The quantum Fourier transform on an N-dimensional Hilbert space is given by $U_{\text{FT}} = \frac{1}{\sqrt{N}} \sum_{x,k=0}^{N-1} e^{\frac{2\pi i x k}{N}} |k\rangle\langle x|$. If we interpret a bitstring $\boldsymbol{x} = x_1 \ldots x_n$ as an integer in binary form with x_1 the MSB, $x = 2^{n-1} x_1 + \cdots + 2^0 x_n$, the action of the QFT on k qubits is given by $U_{\text{FT}} = \frac{1}{\sqrt{2^n}} \sum_{\boldsymbol{x}, \boldsymbol{k}} e^{\frac{2\pi i x k}{2^n}} |\boldsymbol{k}\rangle\langle\boldsymbol{x}| = \frac{1}{\sqrt{2^n}} \sum_{\boldsymbol{x}, \boldsymbol{k}} e^{i \omega_n x k} |\boldsymbol{k}\rangle\langle\boldsymbol{x}|$ where $\omega_n := \frac{2\pi}{2^n}$.

To see that we cam implement this U_{FT} efficiently using controlled phase

gates and Hadamards, we expanding k in binary, $U_{\mathrm{FT}} = \frac{1}{\sqrt{2^n}} \sum_x (\sum_{k_1 \in \{0,1\}} e^{i\omega_1 x k_1} |k_1\rangle)(\sum_{k_2 \in \{0,1\}} e^{i\omega_2 x k_2} |k_2\rangle)$ $\frac{1}{\sqrt{2^n}} \sum_x (|0\rangle + e^{i\omega_1 x}|1\rangle)(|0\rangle + e^{i\omega_2 x}|1\rangle) \ldots (|0\rangle + e^{i\omega_n x}|1\rangle)\langle \boldsymbol{x}|$. Then expanding x in binary and using $2^{k-1}\omega_k = \pi$ we have $U_{\mathrm{FT}} = \frac{1}{\sqrt{2^n}} \sum_x (|0\rangle + (-1)^{x_n}|1\rangle)(|0\rangle + (-1)^{x_{n-1}} e^{i\omega_2 x_n}|1\rangle) \ldots (|0\rangle + (-1)^{x_1} e^{i(\omega_2 x_2 + \cdots + \omega_n x_n)}|1\rangle)\langle \boldsymbol{x}|$.

It is marginally easier to implement the "mirror image" of U_{FT}, whose output has the MSB last. This is done by e.g. for $n = 4$, the first qubit has a Hadamard gate followed by controled phase gates with phases $\omega_2, \omega_3, \omega_4$ controlled by qubits 2,3,4 respectively, then we act on the second qubit with a Hadamard, then a ω_2 phase gate controlled by qubit 3, then an ω_3 phase gate controlled by qubit 4, then we act on qubit 3 by a Hadamard followed by a ω_2 phase gate controlled by qubit 4, and finally we act on qubit 4 by a Hadamard. We see that this circuit requires $O(n^2)$, i.e. polynomially many, gates.

Period- and order-finding

This is an actual, practical use for quantum computation - an efficient algorithm for factoring (in fact, for solving the discrete logarithm problem, which this also does) would enable rapid cracking of many importart cryptosystems in use today. It would also go some way to suggesting a "real" separation between quantum and classical computation, since a lot of effort has been put into efficient classical factoring, with (so far) no success. The heart of Shor's algorithm is a period-finding circuit inspired by that in Simon's algorithm, using the Quantum Fourier Transform.

We consider the problem: given the oracle for an m-bit to n-bit function $f(\boldsymbol{x})$, with $m = O(n)$ but $m > 2n$, and the promise that there is a nonzero period a such that $f(\boldsymbol{x}) = f(\boldsymbol{a}) \Leftrightarrow y = x + ka$ for some integer k, determine a in the minimum number of queries (notation: non-boldface a means we are considering a as a number rather than a bitstring). Since there are only 2^n possible outputs to f, we have $a \leq 2^n$; the requirement that $m > 2n$ ensures that our range of x contains at least 2^n periods a. As always, we consider a reversible oracle; there are m "question" and n "answer" bits.

Our quantum circuit is quite simple: first, we Hadamard all the "question" qubits, then pass our bits to the oracle f, then we apply the QFT $U_{FT} = \frac{1}{\sqrt{2^m}} \sum_{x,k} e^{i\frac{2\pi x k}{2^m}} |\boldsymbol{k}\rangle\langle \boldsymbol{x}|$ to the "question" qubits (some treatments will equivalently use the inverse Fourier transform $U_{FT}^\dagger$), before measuring them. The pre-measurement state is $(U_{FT} \otimes I) U_f (H^{\otimes m} \otimes I)|\boldsymbol{0}\rangle|\boldsymbol{0}\rangle = \frac{1}{\sqrt{2^m}} (U_{FT} \otimes I) \sum_x |\boldsymbol{x}\rangle|f(\boldsymbol{x})\rangle = \frac{1}{2^m} \sum_k \sum_x e^{i\frac{2\pi x k}{2^m}} |\boldsymbol{k}\rangle|f(\boldsymbol{x})\rangle$.

If we were to measure the answer qubits, we would obtain some bitstring $\boldsymbol{y}_0$, and then the question qubits immediately after the oracle would be in an equal superposition of all the $|\boldsymbol{x}\rangle$ states with $f(\boldsymbol{x}) = \boldsymbol{y}_0$. But x_0 is random here, so measuring the question qubits immediately after the oracle would give very little information about a. By taking the Fourier transformation we can find information about a, ignoring x_0, y_0.

We shall first consider the case when $b = \frac{2^m}{a}$ is an integer, i.e. the range of x covers an integer number of periods; the general case is similar but somewhat more complicated. The pre-measurement state can be expressed as $\frac{1}{2^m} \sum_k \sum_{\widehat{x}=0}^{a-1} \sum_{l=0}^{b-1} e^{i\frac{2\pi(\widehat{x}+la)k}{2^m}} |\boldsymbol{k}\rangle|f(\boldsymbol{x})\rangle = \frac{1}{2^m} \sum_{k=0}^{2^m-1} \sum_{\widehat{x}=0}^{a-1} e^{i\frac{2\pi x k}{2^m}} (\sum_{l=0}^{b-1} e^{i\frac{2\pi l k}{b}}) |\boldsymbol{k}\rangle|f(\boldsymbol{x})\rangle$, since by the promise, $f(\boldsymbol{x})$ is in-

dependent of l. Note that the bracketed term is zero unless $k = jb$ for some integer $0 \leq j \leq a-1$; the probability of obtaining any particular value of j is uniform, so $= \frac{1}{a}$. Given $k = jb$ we will attempt to determine a using continued fractions (this method generalises to b non-integral; there are better methods for this special case).

This is now entirely classical: any real number $x \in (0,1)$ is the limit of a sequence of rationals $x_1 = \frac{1}{a_1}, x_2 = \frac{1}{a_1 + \frac{1}{a_2}}, x_3 = \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3}}}, \ldots$; we can define these by $\alpha_0 = x$ and a_k, α_k are the integral and fractional parts of $\frac{1}{\alpha_{k-1}}$ respectively. Two important properties: 1) if x is rational, this sequence terminates at the simplest (i.e. with smallest denominator) rational representation of x 2) if x is within $\frac{1}{2r^2}$ of $\frac{j}{r}$, then the simplest version of $\frac{j}{r}$ will appear as one of the x_k, after $O((\log_2 r)^3)$ (i.e. polynomially many) operations.

Given $k = jb$ we have $\frac{k}{2^m} = \frac{j}{a}$; applying the continued fractions algorithm to $\frac{k}{2^m}$ therefore gives $\frac{j}{a}$ in its simplest form, say as $\frac{j'}{a'}$. If j is prime, clearly no simplification of $\frac{j}{a}$ is possible, and so $a = a'$. Since j is a uniform integer from $(0, a-1)$, by bounds on the density of primes, the probability of obtaining a prime j is greater than $\frac{1}{2\log_2 a} > \frac{1}{2n}$; we can easily check whether a' is the true period using two oracle queries (we have $a' \leq a$, so we may just input $x = 0, x = a'$ and see whether the outputs match), so we can obtain a with high probability using $O(n)$ repetitions. We can in fact do better than this (although we don't actually "need" to): by performing the algorithm $O(1)$ primes we will obtain with high probability $\frac{j_1'}{a_1'}, \frac{j_2'}{a_2'}$ with $(j_1', j_2') = 1$, then $a = \frac{a_1' a_2'}{(a_1', a_2')}$. (We can efficiently compute the GCD of two n-bit integers in $O(n^2)$ steps using Euclid's algorithm).

For the general case, $\frac{2^m}{a}$ will not be an integer, so the range of x will not contain an exact number of periods; however, since $m > 2n$, the previous approach will still succeed with high probability. Define $b(x)$ to be the number of times $f(x)$ is achieved in the complete range of x; so $|b(x) - \frac{2^m}{a}| < 1$. Then the pre-measurement state is $\frac{1}{2^m} \sum_k \sum_{\widehat{x}=0}^{a-1} e^{i\frac{2\pi xk}{2^m}} (\sum_{l=0}^{b(\widehat{x})-1} e^{i\frac{2\pi ilk}{2^m}}) |\boldsymbol{k}\rangle |f(\boldsymbol{x})\rangle$. Then the probability of measuring a particular value of k is $\frac{1}{2^{2m}} \sum_{\widehat{x}=0}^{a-1} |\sum_{l=0}^{b(\widehat{x})-1} e^{i\frac{2\pi alk}{2^m}}|^2$, which will be close to 0 unless $k \approx j\frac{2^m}{a}$ for some integer j.

The probability that there is some j with $|k - j\frac{2^m}{a}| < 2^{m-2n-1}$ is $O(1)$ for $m > 2n$, and approaches 1 rapidly as m increases; to achieve success probability $1 - \epsilon$ requires $m = 2n + O(\log \frac{1}{\epsilon})$. Given that we have obtained a k with $|k - j\frac{2^m}{a}| < 2^{m-2n-1}$ we have $|\frac{k}{2^m} - \frac{j}{a}| < 2^{-2n-1} < \frac{1}{2a^2}$, so it can be proven that the continued fractions algorithm applied to $\frac{k}{2^m}$ will return $\frac{j'}{a'}$, the simplest form of $\frac{j}{a}$, as one of the approximations. We can tell we have a valid j', a' when we have $a' \leq 2^n, |\frac{k}{2^m} - \frac{j'}{a'}| < 2^{-2n-1}$; as before, we can find a using $O(1)$ such values of $\frac{j'}{a'}$. Therefore, we can find periods efficiently on a quantum computer with high probability, using $O(1)$ oracle queries and $O(n^3)$ additional operations; no efficient classical algorithm is known (and indeed, for this particular oracle, it can be proven to be impossible).

We can use the quantum period-finding algorithm to efficiently order find: given positive integers $r, N, r < N$ and promised that $(r, N) = 1$, the order of r modulo N is the least integer a (existence ensured by $(r, N) = 1$)) such that $r^a \mod N = 1$. To use the previous problem, define $f(x) = r^x \mod N$; then this

is periodic with period a, $f(x) = f(y)$ iff $x = y + ka$ for integer k. Thus, given an "oracle" for $f(x) = r^x \mod N$, we can use the period finding algorithm to efficiently determine a, the order of $r \mod N$. Since this is not an oracle problem, however, we shall need to build our "oracle" ourselves, and it must be efficient.

Let n be the smallest integer such that $2^n \geq N$. Our circuit first prepares $m = O(n)$ n-bit ancillas containing the binary representations of $r \mod N, r^2 \mod N, r^4 \mod N, \ldots$; we can do this efficiently by repeated squaring modulo N (clearly, a circuit to input x and output x^2 (in reversible fashion) can be easily constructed). We then compute $r^x \mod N$ by a series of multiplications controlled by the bits of x, accumulating the result in another ancilla: $r^x \mod N = (r^{2^{m-1}} \mod N)^{x_1}(r^{2^{m-2}} \mod N)^{x_2} \ldots (r^{2^0} \mod N)^{x_n} \mod N$ (again, we can easily implement multiplication modulo N reversible). We follow this with the unitary $|y \oplus r^x \mod N\rangle\langle y|$ on the answer register, so that the answer register now has the answer, as it would were this an oracle. Finally, we reverse our process of computing $r^x \mod N$ so that we do not have "junk" registers which remain entangled with the x (input) register. If we did not do this, we would have decoherence - see later.

The process as a whole requires $O(n^3)$ gates, and is thus polynomial; we can find the order in $O(n^3)$ steps.

Shor's Factoring Algorithm

The factoring problem is: given a composite number N, find one of its nontrivial factors. Shor's algorithm has two parts: a classical algorithm for efficiently factoring a number, given the ability to do order-finding efficiently, and the previous quantum algorithm for efficient order-finding.

The classical algorithm is: 1. Check whether N is even; if so, output 2. 2. Check whether $N = p^d$ for some $p, d > 1$; if so, output p. 3. Pick a random positive integer $r < N$ (i.e. $r \in \{1, 2, \ldots, N-1\}$) 4. Compute the GCD (r, N). 5. If this GCD is > 1, output it and halt. 6. Compute the order of r modulo N, i.e. the least integer a such that $r^a \mod N = 1$ 7. If a is odd, go back to step 3. 8. If $r^{\frac{a}{2}} \mod N = N - 1$, go back to step 3. 9. Output $(r^{\frac{a}{2}} - 1, N)$.

Explanation: the first two steps are just eliminating some special cases which would otherwise cause problems later on. We can perform step 2. in $O(n^3)$ operations by simply, for each $1 < d < n$, calculating a close approximation to $p = 2^{\frac{\log d}{n}}$ (in $O(2)$ operations) and checking whether the integers close to this work (since we can never have $N = p^d$ for $d > n$).

Steps 4-5 are again essentially a check; it is unlikely that the GCD is > 1, but possible; if so, it would cause problems later on, but we now have a factor as required. Knowing that the GCD is 1 means that we can guarantee the order-finding problem has a solution.

Step 6 is the essential quantum step; no efficient classical algorithm for order finding is known. Note that we can of course incorporate the rest of the (classical probabilistic) steps into a quantum circuit efficiently, in the same way we did when proving $BQP \supset BPP$.

In steps 7 and 8 we reject some cases where the final step doesn't work. Given that N is odd and has at least two distinct prime factors (as we ensured in steps 1 and 2) we can show that the probability of failing one of these steps

is no more than $\frac{1}{2}$ (so our success probability becomes $1-\epsilon$ with only $O(\log_2 \frac{1}{\epsilon})$ repetitions of the loop), using number theory; see Nielsen and Schrang for the complete proof. Essentially, if we decompose N as a product of prime powers $p_1^{d_1} \dots p_k^{d_k}$, let a_i be the order of r modulo $p_i^{d_i}$ and define f_i as the largest integer such that $a_i \mod 2^{f_i} = 0$. Then steps 7. and 8. only fail if the f_i are all the same (step 7 if $f_1 = 0 \forall i$, step 8 if they're > 0). The probability for f_i to take any particular value is at most $\frac{1}{2}$, so the probability of falure is at most $\frac{1}{2^{k-1}} \leq \frac{1}{2}$ (Note that if N is an odd prime power p^d, the algorithm always fails at step 8).

If we reach step 9 we have a even and $r^{\frac{a}{2}} \mod N \neq N-1$; then we can guarantee that $(r^{\frac{a}{2}}-1, N)$ is a factor: we have $r^a \mod N = 1$. Let $x = r^{\frac{a}{2}} \mod N$ (valid since a even); we have $(x^2-1) \mod N = 0 \Rightarrow (x-1)(x+1) \mod N = 0$; since $x+1 \neq N$, both $(x+1), (x-1)$ are smaller than N, but their product is divisible by N; thus each contains at least one nontrivial common factor with N.

In total, Shor's algorithm takes $O(n^3)$ elementary operations to return a factor with arbitrarily high probability, wheras the best known classical approaches take exponentially many. By repeatedly applying the algorithm we can obtain a complete factorisation of N: there are efficient [hah!] classical algorithms for determining primality, so we know when we have abtained a prime factor, and any n-bit number has at most n prime factors. Each application of Shor's algorithm partitions the factors into two sets (those in the output f and those in $\frac{N}{f}$), so after $(n-1)$ iterations, we identify all factors; thus a complete factorisation can be done in poly(n) time. This of course also allows efficient solution of the factoring decision problem; thus factoring is in BQP.

[The lecturer here gave a description of RSA, which I imagine to be well known to most readers so shall not repeat here]. Note that quantum cryptography provides a provably secure (though non-publickey) cryptosystem, even in a world with quantum computation.

The hidden subgroup problem

Interestingly, both the cases where we've seen an exponential quantum speedup - Simon's algorithm and Shor's period-finding algorithm - can be cast as examples of a more general problem, the hidden subgroup problem. Given an oracle for a function $f : G \to A$ from a group to some finite set, and the proomise that there is a subgroup $H \subset G$ such that $f(g_1) = f(g_2) \Leftrightarrow g_1 H = g_2 H$, find (a generating set for) H in the minimum number of oracle queries. Simon's problem, where our oracle $f : \{0,1\}^n \to \{0,1\}^{n-1}$ has $f(\boldsymbol{x}) = f(\boldsymbol{y}) \Leftrightarrow \boldsymbol{x} = \boldsymbol{y}$ or $\boldsymbol{a}$; we can express this as $G = (\mathbb{Z}_2)^n, H = \{\boldsymbol{0}, \boldsymbol{a}\}$.

Shor's period-finding algorithm is strictly only a hidden subgroup for the case where $b = \frac{2^m}{a}$ is an integer. Then $G = \mathbb{Z}_{2^m}, H = \{0, a, 2a, \dots, (b-1)a\}$. More generally, we can consider the period finding problem as defined for an infinite range of inputs (rather than just m bits); $G = \mathbb{Z}, H = a\mathbb{Z}$.

In both these examples, the group G is abelian; in fact, using a quantum computer we can efficiently solve any hidden subgroup problem where G is finite abelian (and we can extend this to any finitely generated abelian group); the key to this result is that any such gorup is isomorphic to a product of groups $\mathbb{Z}_N$, where each N is a power of a prime.

Imagine we are given a reversible oracle for $f : G \to A$ for such a G, where

group elements $g = (x_1, \ldots, x_m) \in G = \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_m}$ are input by inputting each of the x_i in the form: if N_i is a power of 2, x_i is represented by $\log N_i$ qubits; otherwise x_i is represented y $n_i = \text{poly}(\log N_i) >> \log N_i$ qubits (this is to ensure that an even superposition of input qubits really is an (approximately) even superposition of the group elements; we interpret x_i as modulo N_i). In addition to these "question" qubits there are, of course, more than $\log |A|$ "answer" qubits; if the inputs are $x_1, \ldots, x_m, y$, the outputs are $x_1, \ldots, x_m, \boldsymbol{y} \oplus f(x_1 \mod N_1, \ldots, x_m \mod N_m)$.

The quantum circuit for this problem is simple: first we Hadamard each "question" qubit, then input all qubits to the oracle, then we FT each x_i block of "question" qubits (note we perform a separate FT for each i, not a single large FT) before measuring the "question" qubits, obtaining result k_i for each x_i. By a slight generalization of our previous techniques it can be shown that we measure a value of k such that $h \circ k = \sum_{i=1}^{m} \frac{h_i k_i}{2^{n_i}}$ is close to an integer for all $h \in H$; if all N_i are powers of 2, it will be exactly an integer for all h. Once we have $\text{poly}(\log |G|)$ different values of k satisfying this linear relation, we can find a generating set for H with high probability using generalizations of the techniques used in Simon's and Shor's algorithms.

In fact, these two present good examples of the two "extreme" cases: where G is a product of many subgroups, or a single group; a general G will be "in between" and we have to use a combination of both techniques (linear algebra and continued fractions). In Simon's algorithm, $G = (\mathbb{Z}_2)^n \therefore n_1 = n_2 = \cdots = n_m = 1$; each Fourier transform on the output acts only on a single qubit, and is thus just a Hadamard; we have the same circuit before. $h \circ k$ being an integer $\forall h$ means $a \cdot k = \sum_i a_i k_i \mod 2 = 0$ (trivially $0 \cdot k$ is an integer) - we have the same condition as before.

In Shor's algorithm $G = \mathbb{Z}_{2^m}$; there is a single m-qubit FT on the output. $h \circ k$ is always a multiple of $\frac{ak}{2^m}$, so to say it is always close to an integer is the same as saying $\sum a_i k_i = 2^m j$ for some integer j, which gives $\frac{k}{2^m} \approx \frac{j}{a}$ as before.

Some other interesting problems which can be cast as abelian hidden subgroup problems include the discrete logarithm problem (given $a, b = a^s$ in some finite cyclic group, find s) and finding the order of a permutation (given P, the smallest n such that $P^n = 1$).

The definition of hidden subgroup problems remains valid when the group G is not abelian; an example of a problem which can be cast in this form is the graph isomorphism problem: given two graphs, is there an isomorphism between them? Finding a (efficient) quantum algorithm for this is an open problem; no general quantum algorithm for efficiently solving the hidden subgroup problem in the nonabelian case is known. There are some efforts to implement cryptosystems based on these problems, which may offer the possibility of secure classical cryptography even assuming the existence of quantum computers.

Measurement-based quantum computing

The models of quantum computation we have discussed so far have been directly analagous to their classical equivalents. Mesaurement-based (or "one-way") quantum computation is a uniquely quantum architecture for computation, with no classical analoge; it is perhaps closer to suggesting how to build a physical quantum computer. It is universal in the usual sense.

Essentially, we prepare a quantum system in a large entangled state, a graph state or cluster state; we shall cover the former first since the equivalence to quantum circuits is clearer. For these, we prepare a state structured to mimic a particular quantum circuit - the graph has vertices corresponding to single-qubit gates, and edges to the circuit links between them. Two-qubit gates (i.e. CNOT and the like) are modelled by two vertices with an edge between them. Then we perform some sequence of single-qubit measurements on this large entangled state.

More specifically, for a graph, we define the corresponding graph state as follows: for every vertex, prepare a qubit in the $|+\rangle$ state; then along every edge, apply a controlled-Z gate $C_Z = |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10| - |11\rangle\langle 11|$. This gives a maximally entangled state - the state is an equally-weighted superposition of every computational basis state, with a phase factor of -1 for each pair of 1s in a state which are connected in the graph. E.g. if the graph is a triangle (K_3) the state is $\frac{1}{\sqrt{8}}(|000\rangle + |001\rangle + |010\rangle + |100\rangle - |011\rangle - |110\rangle - |101\rangle - |111\rangle)$.

We measure the qubits one by one to carry out the computation. We shall use two types of measurement: computational basis measurements $M_0 = |0\rangle\langle 0|, M_1 = |1\rangle\langle 1|$, and phase measurements, paramaterised by angle θ: $M_0 = |v_0^\theta\rangle\langle v_0^\theta|, M_1 = |v_1^\theta\rangle\langle v_1^\theta|$, using the orthonormal pair of states $|v_0^\theta\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\theta}|1\rangle), |v_1^\theta\rangle = \frac{1}{\sqrt{2}}(|0\rangle - e^{i\theta}|1\rangle)$. (For $\theta = 0$ these are $|+\rangle, |-\rangle$).

The key idea behind this approach is that we can use entanglement and measurements to perform unitary qubit operations. Consider: given a qubit in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ prepare an ancilla in state $|+\rangle$ and apply a C_z operation; $C_z|\psi\rangle|+\rangle = \alpha|0\rangle|+\rangle + \beta|1\rangle|-\rangle$. We can rearrange this state as $\frac{1}{\sqrt{2}}(|v_0^\theta\rangle(\alpha|+\rangle + \beta e^{-i\theta}|-\rangle) + |v_1^\theta\rangle(\alpha|+\rangle - \beta e^{-i\theta}|-\rangle))$. Then make a measurement with angle θ on the first qubit; with probability $\frac{1}{2}$ we read the result 0, and the state of the second qubit becomes $\alpha|+\rangle + \beta e^{-i\theta}|-\rangle = U(\theta)|\psi\rangle$ where $U(\theta)$ denotes the unitary transformation $|+\rangle\langle 0| + e^{-i\theta}|-\rangle\langle 1|$ (which we shall shortly show forms part of a universal gate set). If the result is 1, the state of the secont qubit becomes $\alpha|+\rangle - \beta e^{-i\theta}|-\rangle = XU(\theta)|\psi\rangle$; we view the X as an "error". So the overall effect of our measurement, with result r, is $X^r U(\theta)$. We can iterate this procedure to perform a sequence of unitaries: if we have a graph of three nodes in a line (i.e. $|+\rangle$ states entangled using C_Zs), but the leftmost qubit's state has somehow become $|\psi\rangle$, then if we measure the first qubit at angle θ_1 obtaining result r_1 and the second at θ_2 obtaining r_2, this implements the operation $X^{r_2}U(\theta_2)X^{r_1}U(\theta_1)$. Therefore, by performing such a sequence and then measuring the final qubit in the computational basis, we can perform any single-qubit quantum computation up to X "errors" (it is simple to show that $U(\theta)$s can generate any single-qubit unitary. Requiring the input state to be $|+\rangle$ rather than $|0\rangle$ does not change the computational power at all, since we only need a Hadamard gate (which we have by $U(0)$) to change between the two. (But note that in many quantum algorithms we would start by Hadamarding all our qubits into $|+\rangle$ states anyway).

Clearly, we can trivially simulate a C_Z gate; we simply include an edge between our two parallel qubit "path"s. These are enough to form a universal gate set; a H is given by a $U(0)$, a ϕ phase gate is given by $U(0)U(-\phi)$, and a CNOT gate can be given by a C_Z where the "controlled" qubit has $U(0)$s on either side of it. Therefore, up to random X errors, we can implement any quantum computation using measurements on an appropriate graph state

- and note that at this stage, there is nothing preventing us from performing all measurements simultaneously. If each "phase" measurement gives the result 0, there are no X errors and we perform precisely the computation we wanted; however, the probability of this drops exponentially with the size of the circuit.

To deal with errors, we "push" them to the end of the computation: to push an X error through a $U(\theta)$, $U(\theta)X = ZU(-\theta)$ (up to an irrelevant global phase). To push a Z error through, $U(\theta)Z = XU(\theta)$. We push errors through CZ gates using $C_Z(Z \otimes I) = (Z \otimes I)C_Z, C_Z(X \otimes I) = (X \otimes Z)C_Z$ (and their vertically-flipped versions); see the second example sheet for more on this. Once we have "pushed" the errors to the end of the computation we can deal with them in ordinary, classical fashion (given that we know what they are): a Z-error at this point has no effect (it only changes the phase), while an X-error flips the relevant output bit. Sine $X^2 = Z^2 = I, ZX = -XZ$, the only ultimate error combinations we need to consider are I, X, Z, XZ.

Note that we now need to adapt our measurements based on previous results, in order to successfully "push" the errors. E.g. suppose we want to perform the comutation $U(\theta_3)U(\theta_2)U(\theta_1)|+\rangle$; we first prepare a 4-qubit graph state, with the four in a line. We measure the first (leftmost) qubit at angle θ_1, getting result r_1; we have thus performed $X^{r_1}U(\theta_1)$. So we make the second measurement (on the second qubit) at an angle of $(-1)^{r_1}\theta_2$, obtaining result R_2; we have therefore performed $X^{r_2}U((-1)^{r_1}\theta_2)X^{r_1}U(\theta_1) = X^{r_2}Z^{r_1}U(\theta_2)U(\theta_1)$. We similarly need to adapt the third measurement; this time we also need to push a Z-error through. We measure at $(-1)^{r_2}\theta_3$, and thus have performed $X^{r_3}U((-1)^{r_2}\theta_3)X^{r_2}Z^{r_1}U(\theta_2)U(\theta_1) = X^{r_3}Z^{r_2}U(\theta_3)Z^{r_1}U(\theta_2)U(\theta_1) = X^{r_1+r_3}Z^{r_3}U(\theta_3)U(\theta_2)U(\theta_1)$. Then we measure the final qubit in the computational basis, obtaining the result $i' = i \oplus r_1 \oplus r_3$, where i is the result we want; we can obtain this by $i' \oplus r_1 \oplus r_3$.

Therefore, we can replicate the results of any quantum circuit with certainty, by a sequence of single-qubit measurements on an appropriate graph state. Measurement can be done left-to-right on the graph state, pushing forward the errors and adapting measurements as we go. Any circuit with poly(n) gates can be simulated using a graph state with poly(n) qubits and poly(n) classical processing to deal with the errors. Since we need to adapt measurements, it also takes poly(n) timesteps (for most graphs, some clever analysis will likely find it is not necessary to perform the measurements strictly left-to-right, but we will not be able to do better than poly(n) even with such optimization). A curiousity is that since the "result" measurement is never adaptive (it's always performed in the computational basis), we can in fact perform this first, and have our "answer" i' at the start (This does not let us avoid performing the computation; i' gives us no information about i until we know what the "errors" r_j are).

A brief note on cluster states: these are the graph states corresponding to 2D lattices. We can use these for "more general" measurement-based quantum computation: measuring a qubit in the Z-direction disconnects it from the graph state (introducing a Z-error on all connected qubits if the result is 1, but this can be dealt with as usual). Thus we can "etch" a graph state into the cluster by choosing which qubits we wish to use for computation, and then meausring all others in the Z-direction. Note that to separate two horizontal qubit lines we need to leave a gap between them (since we can only disconnect qubits, not individual edges); the vertical links (which should C_Z gates) are therefore longer than we should be. However, if we make a phase measurement with $\theta = 0$ on

an intermediate qubit, up to a Z-error this is equivalent to removing it; we also use this technique to implement "empty wires" in the circuit. Therefore, we can use a cluster state to replicate any graph state for a planar graph (one with no crossed edges); this is equivalent to any quantum circuit in which the C_Z gates only act between neighbouring qubits. We have seen before that such circuits are universal; we can therefore simulate any polynomially sized quantum circuit on n qubits by a measurement-based quantum computation on a square cluster state containing poly(n) qubits.

Mixed States and Open-system Dynamics

So far we have considered a quantum computer as a closed system under our perfect control. In practice our gates will be imperfectly constructed and qubits will undergo some interactions with the environment, introducing errors. We would like to be able to model these processes probabilisticly; we shall therefore use the quantum theory of mixed states, which obviously allows us to handle probabilistic errors in the circuit, and somewhat surprisingly, also allows us to treat interaction with the environment.

Suppose we do not know the precise quantum state; rather, we know the system is in one of the states $|\psi_i\rangle$ with respective probabilities p_i. Observe that if we perform a measurement characterised by operators M_k, the probability of obtaining the result k is $\sum_i p_i \text{prob}(k \mid \psi_i) = \sum_i p |M_k|\psi_i\rangle|^2 = \sum_i p_i \langle\psi_i|M_k^\dagger M_k|\psi_i\rangle = \sum_{i,j} p_i \langle\psi_i|M_k^\dagger|j\rangle\langle j|M_k|\psi_i\rangle = \sum_{i,j} p_i \langle j|M_k|\psi_i\rangle\langle\psi_i|M_k^\dagger|j\rangle = \sum_j \langle j|M_k(\sum_i p_i|\psi_i\rangle\langle\psi_i|)M_k^\dagger|j\rangle$; the probability of obtaining a given measurement result depends only on the density operator $\rho = \sum_i p_i|\psi_i\rangle\langle\psi_i|$. Therefore, if two different ensembles of quantum states give the same density operator, they will give the same probabilities for all measurement results and so be experimentally indistinguishable - e.g. an equal mixture of $|0\rangle, |1\rangle$ gives the same mixed state $\rho = \frac{I}{2}$ as an equal mixture of $|+\rangle, |-\rangle$, or any other pair of orthogonal states; thus these are indistinguishable. This is a deep property; on some levels it is what allows causality to be preserved even given quantum phenomena.

We can express a density operator using the computational basis as a density matrix, e.g. $\rho = \frac{2}{3}|0\rangle\langle 0| + \frac{1}{3}|+\rangle\langle +| = \begin{pmatrix} \frac{5}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} \end{pmatrix}$. Note that the diagonal elements here sum to 1, as will always be the case; see later.

It is useful to use the the trace $\text{tr}(A) = \sum_i \langle i|A|i\rangle$ (we can use any orthonormal basis $\{|i\rangle\}$); for A written as a matrix this is simply the sum of the diagonal elements. The trace has cyclic symmetry $\text{tr}(AB) = \text{tr}(BA)$.

It is easy to see that all density operators ρ are 1. Hermitian $\rho = \rho^\dagger$ 2. Positive operators $\langle\phi|\rho|\phi\rangle \geq 0 \forall \phi$ (sometimes this is taken to imply 1., since the inequality only makes sense if the quantity on the left is always real) 3. $\text{tr}(\rho) = 1$; we can here use the slight abuse of notation that $\text{tr}(|\psi\rangle\langle\phi|) = \text{tr}(\langle\phi|\psi\rangle) = \langle\phi|\psi\rangle$. Furthermore, any operator satisfying these properties can be viewed as the density operator for some mixed state (e.g. a mixture of the eigenstates of ρ with probabilities equal to the corresponding eigenvalues, since we have that said eigenvalues are ≥ 0 (since ρ is positive) and sum to 1 (since $\text{tr}\rho = 1$)).

We can in fact reformulate all of quantum theory in terms of density operators rather than vectors: the state of a physical system is represented by a density operator (a trace 1 positive operator) ρ on a Hilbert space, the evolu-

tion of a closed system is given by a unitary operator U and takes the form $\rho \to U\rho U^\dagger$ (the reader may observe this corresponds to $|\psi\rangle \to U|\psi\rangle$). When a measurement (characterised by operators M_k) is performed on the state, the result k is obtained with probability $\mathrm{tr}(M_k \rho M_k^\dagger) = \mathrm{tr}(M_k^\dagger M_k \rho)$, and in this case the state becomes $\frac{M_k \rho M_k^\dagger}{\mathrm{tr}(M_k \rho M_k^\dagger)}$; observe this means that if we only care about the outcome probabilities and not the post-measurement states, we only need the "POVM elements" $M_k^\dagger M_k$ to characterise the measurements.

If the system is in a definite quantum state $|\psi\rangle$ we say it is a pure state; ρ is then a projector $|\psi\rangle\langle\psi|$. Note that in this formalism the irritating global phase factors simply do not appear; if $|\psi'\rangle = e^{i\phi}|\psi\rangle$ then $|\psi'\rangle\langle\psi'| = |\psi\rangle\langle\psi|$. This is good, since such phases have no physical relevance.

The density operator of a mixture is simply the mixture of the corresponding density operators. Unfortunately, expressing superpositions is more difficult than in the vector formalism. Density operators for independent systems combine via the tensor product as usual.

An advantage of this formalism is that we can define a state corresponding to a part of an entangled system, e.g. one of the qubits for $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Given a general state ρ_{AB} of two systems, the state of system A is given by the reduced density operator $\rho_A = \mathrm{tr}_B(\rho_{AB}) = \sum_i (I \otimes \langle i|)\rho_{AB}(I \otimes |i\rangle)$, the partial trace of ρ_{AB} over B. This can be used to obtain the outcome probabilities for any measurement on system A (provided there is no further interaction between A and B): the result of $M_k \otimes I$ on ρ_{AB} is the same as that of M_k on ρ_A. For example, for the pure entangled state $|\psi\rangle = \frac{1}{\sqrt{3}}|0\rangle|0\rangle + \sqrt{\frac{2}{3}}|1\rangle|1\rangle$, the reduced density operators of the systems A, B are given by $\rho_A = \rho_B = \frac{1}{3}|0\rangle\langle 0| + \frac{2}{3}|1\rangle\langle 1|$. Note that even though the system is in a pure state, e.g. the system A considered alone behaves as though it were in a probabilistic mixture of states. Also note that $\rho_{AB} \neq \rho_A \otimes \rho_B$ - but this difference only becomes apparrent if the systems interact again.

We can represent the density operators for a single qubit geometrically as the Bloch Sphere: if we write a general normalised single-qubit pure state as $|\psi\rangle = e^{i\alpha}(\cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}e^{i\phi}|1\rangle)$ (it is convenient to use $\frac{\theta}{2}$ rather than θ), the density matrix ρ can be expressed as $\frac{1}{2}(I + \boldsymbol{r}\cdot\boldsymbol{\sigma})$, where $\boldsymbol{r}$ is the unit vector $(\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$ (i.e. the point (θ, ϕ) on the unit sphere) and σ_i are the Pauli matrices. Thus pure states correspond to points on the surface of the unit sphere (e.g. $|0\rangle\langle 0|$ is the "North" pole $z = 1$, $|1\rangle\langle 1|$ the "south" pole, and $|+\rangle\langle +|$ the point $x = 1$. Mixed states lie inside the sphere, with their $\boldsymbol{r}$ vector being the mixture of the $\boldsymbol{r}$ vectors of their components. All single-qubit density operators have $\rho = \frac{I + \boldsymbol{r}\cdot\boldsymbol{\sigma}}{2}$ for some real vector $\boldsymbol{r}, |\boldsymbol{r}| \leq 1$.

We can write any single-qubit unitary as $U \propto e^{-i\theta|v\rangle\langle v|}$; this is a rotation of the Bloch sphere by angle θ about the axis through the point $|v\rangle\langle v$. Orthogonal states correspond to opposite points on the surface of the Bloch sphere.

Unfortunately, this approach does not generalise to higher dimensional systems; in fact, there is no "nice" characterisation of general trace 1 positive operators.

Density operators allow us to represent more general or complicated quantum dynamics, essential for modelling possible errors in quantum computers. For example, our gates may have some probabilistic error in their construction e.g. a phase gate R_α might actually be $R_{\alpha'}$ for some α' with some probability

distribution close to α. Also, the qubits may undergo some (unitary) interaction with their environment, which may cause entanglement, meaning the states of the qubits considered alone become mixed:

Consider a system in state ρ_S and an environment, initially in a state $|0\rangle\langle 0|$ (for simplicity) uncorrelated with it (this is crucial), so $\rho_{SE} = \rho_S \otimes |0\rangle\langle 0|$. If these interact unitarily the state becomes $\rho'_{SE} = U(\rho_S \otimes |0\rangle\langle 0|)U^\dagger$; the final state of the system is then $\rho'_S = \mathrm{tr}_E(\rho'_{SE}) = \sum_k E_k \rho_S E_k^\dagger$ where $E_k = (I \otimes \langle k|)U(I \otimes |0\rangle$.

More generally, the unitary evolution of a system in state ρ with an initially uncorrelated environment can always be represented by a map of the form $\rho \to D(\rho) = \sum_k E_k \rho E_k^\dagger$. This D is sometimes caled a superoperator; the operators E_k are Kraus operators. Since the trace of ρ must be preserved, the $\{E_k\}$ must satisfy the completeness relation $\sum_k E_k^\dagger E_k = I$. The converse is also true: any map $D(\rho)$ with Kraus operators satisfying the completeness relation can be implemented by unitary interaction with an initially uncoupled environment; we need to define $U|i\rangle|0\rangle = \sum_k (E_k|i\rangle)|k\rangle \forall i$. This does not define U completely, but the completeness relation ensures this U maps orthogonal states to orthogonal states; this means we can complete U to form an operator which is unitary on the whole space.

Maps expressible as such $D(\rho) = \sum_k E_k \rho E_k^\dagger, \sum_k E_k^\dagger E_k = 1$ are actually the most general maps with some entirely reasonable properties: linearity ($D(p_1\rho_1 + p_2\rho_2) = p_1 D(\rho_1) + p_2 D(\rho_2)$, which the behaviour would be very weird without, complete positivity (D is positive even when acting as part of a larger system, i.e. $I \otimes D(\rho)$ is positive when ρ is) (e.g. matrix transposition is positive but not completely positive - but if we put a transposition as part of a larger quantum system, the combined action will not give a valid density matrix as a result), and trace preservation (so that $D(\rho)$ has trace 1, given that ρ does).

Some examples: if there is just one Kraus operator it must be unitary, and this reduces to the usual case. For a probabilistic mixture of unitaries U_k with probabilities p_k, set $E_k = \sqrt{p_k} U_k$, thus $D(\rho) = \sum_i p_i U_i \rho U_i^\dagger$. Decoherence: if $E_k = |k\rangle\langle k|$ for some orthonormal basis, we say it decoheres the state into that basis: it collapses any quantum superposition to a "classical" mixture of basis states. Finally, a useful test case for general results is $E_k = |\psi\rangle\langle k|$ for any orthonormal basis $|k\rangle$, which "throws away" our state and maps any state to $|\psi\rangle$.

Note that the dynamics E_k are equivalent to performing the measurement $M_k = E_k$ and then ignoring the result: $\rho \to \sum_k \mathrm{prob}(k)\rho_k = \sum_k \mathrm{tr}(E_k \rho E_k^\dagger) \frac{E_k \rho E_k^\dagger}{\mathrm{tr}(E_k \rho E_k^\dagger)} = \sum_k E_k \rho E_k^\dagger$. We can model an imperfect measurement (in which we lose some of the result) by associating some subset of Kraus operators $\{E_{rk}\}$ with each possible result r, so $\mathrm{prob}(r) = \mathrm{tr}(\sum_k E_{rk} \rho E_{rk}^\dagger), \rho'_r = \frac{\sum_k E_{rk} \rho E_{rk}^\dagger}{\mathrm{prob}(r)}$. As with general quantum dynamics, we can also derive this form of a general measurement from some basic requirements. Thus, any completely general open system dynamics, including any number of measurements with results $r_1, \ldots, r_n$, can be described by a set of Kraus operators $\{E_{r_1 r_2 \ldots r_N k}\}$ with $\mathrm{prob}(r_1, \ldots, r_N) = \mathrm{tr}(\sum_k E_{r_1,\ldots,r_N k} \rho E_{r_1,\ldots,r_N k}^\dagger), \rho'_{r_1 \ldots r_N} = \frac{\sum_k E_{r_1,\ldots,r_N k} \rho E_{r_1,\ldots,r_N k}^\dagger}{\mathrm{prob}(r_1,\ldots,r_N)}$. (Interestingly, any general quantum measurement can be modelled by introducing an ancilla, applying a joint unitary evolution and then performing an ordinary, projective measurement on the ancilla).

Quantum Error Correction

The ability to control errors is not only a practical concern; some problems believed to not be in P seem to have efficient solutions on analogue computers (where variables may be continuous parameters), but any remotely realistic model of errors requires one to effectively discretise the parameters, effectively reducing to an ordinary Turing machine. It is therefore a concern to see whether we can correct for errors in a quantum computer without reducing its power to that of a classical computer. (Fortunately we will show that we can).

First, consider we are storing a classical bit for a while, or transmitting it through some channel, such that it has probability ϵ of becoming flipped. To increase the chance of error recovery we can store 3 copies of this bit, and then take a "majority vote" as the output at the end of the storage period. Then the probability of an error becomes $\epsilon' = 3\epsilon^2 - 2\epsilon^3$, which is $< \epsilon$ for ϵ small (in fact for $\epsilon < \frac{1}{2}$. This is a simple error correcting code.

We can use the same code to protect against quantum bit flip errors; consider $\rho \to D_X(\rho) = (1-\epsilon)\rho + \epsilon X\rho X$. However, we cannot create three copies of a general state $|\psi\rangle$, by the no-cloning theorem; by linearity, if we encode the basis states by $|0\rangle \to |0_X\rangle = |0\rangle|0\rangle|0\rangle, |1\rangle \to |1_X\rangle = |1\rangle|1\rangle|1\rangle$ then we must have $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \to |\psi_X\rangle = \alpha|0\rangle|0\rangle|0\rangle + \beta|1\rangle|1\rangle|1\rangle$, which is not (generally) equal to $|\psi\rangle|\psi\rangle|\psi\rangle$. However, this encoding can still protect arbitrary states against bit flip errors: we cannot (as in the classical case) simply measure each of the three qubits in the computational basis and take a majority vote, as then our output would be $|0\rangle$ or $|1\rangle$, and not $|\psi\rangle$. So instead, we measure which qubit has been flipped without learning anything about $|\psi\rangle$; we can do this by the error detection measurement $M_0 = |000\rangle\langle 000| + |111\rangle\langle 111|, M_1 = |100\rangle\langle 100| + |011\rangle\langle 0111|, M_2 = |010\rangle\langle 010| + |101\rangle\langle 101|, M_3 = |001\rangle\langle 001| + |110\rangle\langle 110|$. (this works because e.g. $X \otimes I \otimes I|\psi_X\rangle = \alpha|100\rangle + \beta|011\rangle$ etc.) The result of this is called the error syndrome. If no qubits have been flipped, we'll obtain the result $k = 0$ with certainty; if qubit n alone has been flipped we will obtain result $k = n$ with certainty. (We can also track exactly what happens in the case where more than one bit has been flipped, but this is of less interest). In any case, the post-measurement state will be unchanged from pre-measurement.

Given the error syndrome, if we assume there has been at most one bit flip, we can correct the error and recover the inital encoded state: if $k = 0$ we do nothing, if $k \in \{1, 2, 3\}$ we apply an X gate to the kth qubit, this reversing any single bit flip (if more than one qubit has been flipped, our resulting state will be $X \otimes X \otimes X|\psi_X\rangle$ rather than $|\psi_X\rangle$). Finally, we decode to the original state using the reverse of the encoding circuit (viz: CNOT gate controlled by qubit 1 acting on qubit 2, CNOT gate controlled by qubit 1 acting on qubit 3; qubits 2 and 3 outputting $|0\rangle$). The process as a whole gives precisely $|\psi\rangle$ when there are 0 or 1 bit-flips, thus reducing the error probability from ϵ to $\epsilon' = 3\epsilon^2 - 2\epsilon^3$ (as in the classical case).

We shall refer to the process of error detection and recovery as error correction; although it is most clearly described as a measurement followed by some operation depending on it, we can implement it by a unitary quantum circuit. (viz: add two ancillas in state $|0\rangle$. Perform CNOTS controlled on qubits 1 and 2 on the first ancilla, and CNOTS controlled on qubits 2 and 3 on the second ancilla; the two ancillas can now be seen as bits representing "does 1 differ from 2?" and "does 2 differ from 3?". So we perform a CCNOT on qubit 2 controlled

by the two ancillas, a CCNOT on qubit 1 controlled by the two ancillas with Xs surrounding the second qubit control, and a CCNOT on qubit 3 controlled by the two ancillas with Xs surrounding the first qubit control).

By repeatedly performing error correction during the storage time T we can achieve arbitrarily small error probabilities: the error probability in time $\frac{T}{N}$ will be approximately $\frac{\epsilon}{N}$; performing error correction after each $\frac{T}{N}$ will reduce this to approximateli $3(\frac{\epsilon}{N})^2$. Adding the error probabilities for each of the N intervals, the total error probability is approximately $\frac{3\epsilon^2}{N}$, which can be made arbitrarily small by increasing N. Of course, we are (for now) assuming our error correction circuits are perfect, errors only occur during storage, not computation. Also note that we use a fresh set of $|000\rangle$ ancillas each time, which are "carrying away entropy" from the system; in this course we obey the laws of thermodynamics.

We can use a similar scheme to protect against random Z errors $\rho \to D(\rho) = (1-\epsilon)\rho + \epsilon Z\rho Z$: we simply replace the computational basis states $|0\rangle, |1\rangle$ in the encoding and error-correction scheme by $|+\rangle, |-\rangle$ respectively, since a phase flip or Z gate acting on a qubit changes $|+\rangle$ to $|-\rangle$ and vice versa.

Suppose we have not just one of bit or phase flips to worry about, but the possibility of either or both; consider the depolarising map $\rho \to D(\rho) = (1-\epsilon)\rho + \frac{\epsilon}{3}(X\rho X + Y\rho Y + Z\rho Z)$. This corresponds to a uniform shrinking of the Bloch sphere, and is equivalent to with some probability (not in fact the same ϵ) throwing away ρ and replacing it with the maximally mixed state $\frac{I}{2}$. Note that $Y = iZX$; hence on a single qubit we can consider this map as applying either I, X, Z or ZX.

We can protect against such errors by combining (the lecturer says "concatenating", but I don't believe that's the correct term) the two previous codes to generate Shor's 9-qubit code: encode the input with the phase-flip code and then the bit-flip code. E.g. $|0\rangle \to |0_Z\rangle = |{+}{+}{+}\rangle \to |0_{ZX}\rangle = |+_X\rangle|+_X\rangle|+_X\rangle = \frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)$; similarly $|1_{ZX}\rangle = \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)$. We correct errors by correcting any single bit-flip error on each triplet 1-3, 4-6, 7-9 as for the bit-flip code, and then correcting any single phase flip error: applying Z to one of the qubits in the kth triplet flips it between $|+_X\rangle$ and $|-_X\rangle$. So we can detect the result k by applying an encoded phase-flip measurement e.g. $M_1 = |-_X {+}_X{+}_X\rangle\langle -_X {+}_X{+}_X| + |+_X -_X -_X\rangle\langle +_X -_X -_X|$. Note that M_0, M_1, M_2, M_3 do not cover the whole space, we have some spare dimensions; we need to include $M_4 = I - (M_0 + M_1 + M_2 + M_3)$, which will be a measurement result that simply tells us things have gone very wrong. While conceptually clear this set of measurements might be hard to apply in practice, but there is a simpler set which will replicate the results; see the example sheet.

By protecting against a single X, Y or Z error, Shor's code protects against an arbitrary single qubit error: suppose an error has occured on one of the qubits, given by a Kraus operator E_k. As I, X, Y, zZ form an operator basis for a single qubit, we can expand $E_k = aI + bX + cY + dZ$. But each of these components gives a different error syndrome, so the syndrome measurement will collapse the effect of the error to I with probability $|a|^2$, X with probability $|b|^2$ etc. Then in the recovery phase we correct the detected error, and thus recover the original state perfectly.

If the error probability on a single qubit is ϵ, the error probability of the output will be $\epsilon' \leq \binom{9}{2}\epsilon^2 = 36\epsilon^2$. Also note that in general, any error correcting

code correcting a set of errors S will also correct any error whose Kraus operators are linear combinations of the elements of S, e.g. a code to protect against Z will protect against $E_k = \alpha I + \beta Z$.

In general, a code is given by a subspace $H_C \subset H$ of a Hilbert space into which the initial states are encoded. It corrects errors $D(\rho)$ if there is an allowed dynamics $R(\rho)$ such that $R(D(\rho_C)) = \rho_C \forall \rho_C \in H_C \otimes H_C^\star$. For schemes where correction is a syndrome measurement M_k and corresponding recovery U_k we have $R(\rho) = \sum_k U_k M_k \rho M_k^\dagger U_k^\dagger$, but there is no requirement for recovery to be like this.

Some other codes useful for protecting against single-qubit errors are the 7-qubit Steane code, which is useful in practice since gates are easy to implement directly on the encoded state: $|0\rangle = \frac{1}{\sqrt{8}}(|0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle + |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle), |1\rangle = (X \otimes X \otimes X \otimes X \otimes X \otimes X \otimes X)|0\rangle$, and the 5-qubit code, which is the shortest possible for correcting arbitrary single-qubit errors (it's necessary that each of the 3 possible errors applied to each of 5 possible qubits of the codeword for $|0\rangle$, plus the codeword itself, give distinct points in the space and that all those arrived at from $|1\rangle$ by the same operations are different; thus the space must be of dimension at least $2 \times (1+15) = 32$, which it is. But the same calculation means that for a 4-qubit code the 4-qubit space would have to be of dimension at least $2 \times (1+12) = 26$, which it is not). It has $|0_5\rangle = \frac{1}{\sqrt{8}}(|00000\rangle + |10010\rangle + |01001\rangle + |10100\rangle + |01010\rangle + |00101\rangle - |00110\rangle - |11000\rangle - |00011\rangle - |10001\rangle - |01100\rangle - |01111\rangle - |11110\rangle - |11101\rangle - |10111\rangle - |11011\rangle), |1_5\rangle = (X \otimes X \otimes X \otimes X \otimes X)|0_5\rangle$; it isn't practically very useful.

Now that we have methods for error-resistant storage or transmission of information, what about errors in computation itself? Suppose that each basic gate or time-step of storage in the circuit has a probability ϵ of introducing an X, Y or Z error (or, for e.g. C_Z gates, possibly some constant number of errors, i.e. one for each output qubit). As before, correcting for this simple error model also corrects more general quantum dynamics, where the correct unitary operator U for each gate is replaced by some $D(\rho)$ in some sense "close" to $U\rho U^\dagger$. Note that gates which act on more than 1 qubit (e.g. C_Z) may introduce correlated errors on these qubits.

To protect against these errors, we encode each qubit using an error correcting code (as we shall see later, we usually want to use Steane's 7-qubit code), and perform gates and measurements directly on the encoded state. We also perform an error correction (syndrome detection and recovery) after each time step; thus for the circuit where we measure $H|0\rangle$ we would prepare an encoded $|0\rangle$, error-correct, perform an encoded H, error-correct and then perform an encoded measurement. However, this alone is not enough to ensure a small error probability: our error-correcting code ensures that so long as there is only one error on an encoded qubit, the logical state will be unaffected. But a single error within a procedure may propagate to many errors in the output, e.g. if we have an X error before the control of a CNOT gate, this gives us two X errors - one on each output line. We must avoid this; in particular, we cannot decode the encoded state, perform the computation and then reencode, as a single error in the computation would result in too many errors in the final state.

For error correction to work, we require that a single error inside a procedure generates at most one error on each encoded qubit at the output (i.e. one in

each "block" of outputs corresponding to a single qubit); we also require that if the procedure is a measurement it outputs the correct result. Such a procedure is called fault-tolerant.

Suppose that all the procedures in our circuit are fault-tolerant ("FT") (it is not at all obvious at this stage taht this is possible). So we prepare an encoded $|0\rangle$ FTly, perform an FT error correction, apply an FT encoded H, another round of FT error correction and finally an FT encoded measurement. Now, we can "pass through" an error: if there was e.g. an error in the first FT error correction, this will result in only one error on the output qubits from it, which is equivalent to an error at the very start of the FT encoded H - so if there are no further errors in the H itself, we will have at most one error on the output to the H, since the H is FT.

This circuit could therefore only fail if there are two or more errors before the end of the first error correction, or after the start of the final error correction, or between the start of one error correction and the end of the next. Each basic operation in the original circuit (preparation, gate, or measurement) corresponds to a block of procedures [in which two errors will ruin us]; the probability of two or more errors in each block of procedures is $\epsilon' \leq c\epsilon^2$ for some constant c. If the maximum number of elementary components in a block (which we hope will exist, and be a constant, independent of the circuit size) is k, then $c = \frac{1}{2}k(k+1)$; we gain an advantage from error correction if $\epsilon < \frac{1}{c}$. A reasonable value might be $k = 10^2$, giving $c = 10^4$. Then given a circuit composed of n basic operations, the probability of an error in the encoded fault-tolerant version will be $\leq n\epsilon' \leq nc\epsilon^2$. These arguments generalise to arbitrary circuits, with each basic gate corresponding to a block of FT error correction on the input, an FT encoded gate and FT error correction on the output.

Thus, we may reduce the probability of error in a circuit, provided that the error per component $\epsilon <$ some threshold $\epsilon_{\text{th}} = \frac{1}{c}$; we aslo need that the required procedures for universal quantum computation can be made fault-tolerant; that is a universal set of gates, computational basis measurements, preparation of $|0\rangle$ and error correction:

1. Gates: the simplest gates to apply fault-tolerantly are those which can be achieved simply by applying an operation on each encoded qubit in parallel; these are called transversal gates. E.g. for many codes, a FT encoded X gate can be done simply by applying an X to each encoded qubit; in this situation, one error will clearly only affect one output. Our reason for liking the Steane code is that many standard gates have such transversal (and therefore fault tolerant) implementations, including $H, X, Y, Z, C_X, R_{\frac{\pi}{2}}$; however, these gates do not form a universal set; in fact any circuit made of them can be efficiently simulated classically (the Gottesman-Knill Theorem, which we sadly lack time to cover in this course; essentially, we can push tensor products of X, Y, Z, I states through these gates easily, similar to what we did in the measurement-based quantum computation case).

To complete a universal set, we need to also have the gate $R_{\frac{\pi}{4}}$ (also sometimes called $R_{\frac{\pi}{8}}$, since it is a rotation of the Bloch sphere by $\frac{\pi}{8}$; $\begin{pmatrix} e^{-\frac{i\pi}{8}} & \\ & e^{\frac{i\pi}{8}} \end{pmatrix} = \begin{pmatrix} 1 & \\ & e^{\frac{i\pi}{4}} \end{pmatrix}$). This can be implemented fault-tolerantly in a similar way to implementing $U(\theta)$ in measurement-based quantum computation; we prepare an

ancilla using the method below, entangle it with the state, measure the original qubit and apply a unitary correction - but we lack the time to cover this in detail.

2. Preparation. This is easy; while creating a $|0\rangle$ can be hard because a single error near the start can corrupt many qubits, since we know what we are trying to create we can prepare several copies, test them, and discard those which are not correct.

3. Measurement; in the Steane code, computational basis measurements can be done transversally. Syndrome diagnosis requires the use of $|0\rangle$ or $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ ancillas, but these can be prepared and tested using the above method; we then apply transversal C_X gate and measure each ancilla qubit.

Error-correcting codes and fault-tolerant procedures therefore allow us to reduce the error probability in a circuit of n basic components to $nc\epsilon^2$. However, for n large this may still be close to 1. But, to correct errors in arbitrarily large circuits we can simply concatenate our codes; at each new level of encoding we encode using qubits in the previous level (So e.g. if we were using the bit-flip code $|0\rangle = |000\rangle$ etc., $|0_{X^k}\rangle = |0_{X^{k-1}}\rangle|0_{X^{k-1}}\rangle|0_{X^{k-1}}\rangle$). Similarly, we build a level k FT gate out of a number of level $k-1$ FT gates and FT error correction (built out of level $k-1$ FT components), and likewise measurements and preparations. Each level of encoding improves (provided $\epsilon < \frac{1}{c}$) the error probability in the same way as before; we go from ϵ to $c\epsilon^2$ to $c(c\epsilon^2)^2$ and so on, so the error probability on a level k component is $\epsilon_k \leq c\epsilon_{k-1}^2 \leq \cdots \leq \frac{(c\epsilon)^{2^k}}{c}$. Thus we can achieve the total error probability $n\epsilon_k$ to be $\leq \delta$ by taking $k \geq \log \frac{\log \frac{n}{c\delta}}{\log \frac{1}{c\epsilon}} = O(\log \log n)$. The overhead in each layer of encoding is at most a constant factor d, thus the total overhead is $d^k = \text{poly}(\log n)$ - very efficient.

We thus have the threshold theorem for quantum computation: a quantum circuit of n gates can be simulated with arbitrarily low probability of error δ using $O(n\text{poly}(\log \frac{n}{\delta}))$ imperfect components, provided the probability of error on each component is $\epsilon <$ some constant threshold ϵ_{th}.

The precise value of this threshold depends on the details of the error model, code used and procedures; values around $\epsilon_{\text{th}} = 10^{-4}$ can be proven for reasonable schemes, and simulations suggest values an order of magnitude larger work. This is not within current manufacturing capability, but does not seem an infeasible level of tolerance. While the scaling of the overhead is good in a computer-science sense, however, for reasonable sized computations it seems to amount to around 10^3 physical qubits and gates per logical qubit. (There are tradeoffs to be made; we can achieve a proven value of $\epsilon_{\text{th}} \approx 10^{-3}$ and simulations showing values up to 10^{-2} working, but then have an overhead factor of 10^6 or more for moderate computations).

Successful error correction seems to require fresh ancillas (preparing a large number of ancillas at the start of the computation isn't enough, since we cannot store them without accumulating errors; also this assumption seems like it might be absolutely necessary on entropic grounds) and parallel computation (since otherwise for each timestep on n qubits we introduce $O(n)$ errors while correcting only $O(1)$. Also note that all our proofs rely on some "reasonable" assumptions about errors, such as errors on different components being independent; it remains to be seen whether this will hold in practice. However, it must be emphasised that this situation is much better than that with analogue

computation, which fails when we apply any error model, realistic or not. It *seems* there is no fundamental physical reason to prevent quantum computers being realised.

Physical Implementations

Building a physical quantum computer which can outperform a classical computer (e.g. for factoring) appears some way off. There is a tension between two requirements; we need the qubits to be well isolated, to minimise errors and decoherence from interactions with the environment, but we also need to be able to manipulate them directly to be able to prepare, apply gates and measure. E.g. photons are easy to isolate, but hard to interact with at all; spins of particles are the opposite.

Trying to be positive, the lecturer points out that there are many side benefits to attempting to construct a quantum computer; precise control over quantum systems (e.g. photons, ions, etc.) is something of general application, and quantum computation provides a good benchmark for how much control we have over physical system. (The lecturer even strretced to claim that there are foundational implications; quantum computation provides a clean, precise test that shows that quantum behaviour really is ocurring. Therefore, if we can show that some problem not in P can be solved with a quantum computer, this demonstrates that the basis of physics really is quantum rather than classical. But the Bell inequalities do this a lot more clearly and simply).

DiVincenzo suggested some requirements to say we have a scalable quantum computer: 1. A scalable physical system, with well-defined qubits, 2. The ability to initialize the qubits to a simple initial state, 3. Decoherence times (this informally means the time for serious environmental disturbance; there are many ways to define it, but the principle is clear) much longer than individual gate times, 4. The ability to implement a universal gate set, and 5. The ability to measure individual qubits. Fault tolerance requires in addition to these: low errors, largely uncorrelated errors, the ability to perform parallel operations, and the ability to initialize fresh qubits during the computation.

The initial approach was NMR; the qubits are nuclear spins on a molecule in solution in a strong magnetic field, and gates are applied by transverse electromagnetic pulses. This has achieved some proof-of-concept success, e.g. running Shor's algorithm with 7 qubits, but does not scale (since it is necessary to find a molecule where each nucleus has a different energy level so that we can address each qubit individually, and large molecules tend to be highly symmetric); it is mostly defunct as a modern approach. There are also some foundational concerns; the states which are used are highly mixed and typically unentangled, so questions have been raised as to whether this is "true" quantum computation; nevertheless, it does seem capable of running quantum algorithms with no known classical equivalent.

Ion traps are a somewhat attractive approach; qubits are electron energy levels in ions, confined in an electric field. Interactions are done rather elegantly by coupling these energy levels to the collective motion of the set of ions. 8 entangled ions have been achieved, and the process scales to a certain extent in that it is easy to use multiple traps. However, interaction and in particular parallel operations are difficult with a larger computer, and the whole process

is generally quite slow.

As a more theoretical than practical approach, Knill, Laflame and Milburn proved it is possible to do universal fault-tolerant quantum computation using only linear optics; the qubits are photon polarisations/positions, interactions are done via non-deterministic measurements. It is hard to have two photons interact directly; the demonstrated gate acts correctly only when two measurements return the results we want, which happens $\frac{1}{4}$ of the time (but we do know whether the measurement has worked correctly). Thus there is a huge number of photons required per logical qubit; also the need for good sources and detectors for individual photons may be somewhat problematic. However, this method does have advantages; low decoherence, scalability, and the ease of building graph states.

There are various schemes for solid state implementations, and this is where most of the money is currently being invested; generally the qubits are quanta of spin, charge or magnetic flux inside a semiconductor. These are scalable, compact approaches, and we are used to working with semiconductors, but creating even 1 or 2 qubits is difficult; the systems are also generally quite noisy and require very precise fabrication.

One approach favoured by the lecturer is the use of atoms trapped in optical lattices, formed by counter-propagating laser beams. The qubits are internal atomic states; shifting the beams can cause atoms to collide and become entangled. This allows us to generate very large cluster states, but individual qubits are hard to measure; it might be useful for simulation of many-body systems.

There is also the adiabatic approach; a many qubit system is prepared in the ground state of a simple Hamiltonian, which in then slowly changed such that the system remains in the ground state. The final Hamiltonian encodes the problem; its ground state encodes the answer, which we can then measure. The principles are valid, but there are proofs that in any nontrivial quantum computation the first excited state must necessarily fall very close to the ground state, making keeping the system in its ground state difficult. A company, "D-Wave", have claimed a working 28-bit quantum computer using this approach, but have yet to demonstrate any actual quantum computation.

This concludes the course; the lecturer recommends Nielson and Chuang's textbook for further information.