Lecture 2: Basics of Quantum Mechanics

Scribed by: Vitaly Feldman

Department of Mathematics, MIT

September 9, 2003

In this lecture we will cover the basics of Quantum Mechanics which are required to understand the process of quantum computation. To simplify the discussion we assume that all the Hilbert spaces mentioned below are finite-dimensional. The process of quantum computation can be abstracted via the following four postulates.

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

Qubits were the example of such system that we saw in the previous lecture. In its physical realization as a polarization of a photon we have two basis vectors: $|\updownarrow\rangle$ and $|\leftrightarrow\rangle$ representing the vertical and the horizontal polarizations respectively. In this basis vector polarized at angle θ can be expressed as $\cos\theta\,|\leftrightarrow\rangle - \sin\theta\,|\updownarrow\rangle$.

An important property of a quantum system is that multiplying a quantum state by a unit complex factor ($e^{i\theta}$) yields the same complex state. Therefore $e^{i\theta}\,|\updownarrow\rangle$ and $|\updownarrow\rangle$ represent essentially the same state.

Notation 1. *State χ is denoted by $|\chi\rangle$ (often called a* ket*) is a column vector, e.g.,*

$$\begin{pmatrix} 1/2 \\ i\sqrt{3}/2 \\ 0 \end{pmatrix}$$

$|\chi\rangle^\dagger = \langle\chi|$ (often called a bra*) denotes a conjugate transpose of $|\chi\rangle$. In the previous example we would get $(1/2, -i\sqrt{3}/2, 0)$. It is easy to verify that $\langle\chi|\chi\rangle = 1$ and $\langle x|y\rangle \leq 1$.*

Postulate 2. *Evolution of a closed quantum system is described by a unitary transformation. If $|\psi\rangle$ is the state at time t, and $|\psi'\rangle$ is the state at time t', then $|\psi'\rangle = U\,|\psi\rangle$ for some* unitary *operator U which depends only on t and t'.*

Definition 1. *A* unitary *operator is a linear operator that takes unit vectors to unit vectors.*

For every ψ, $\langle\psi|\,U^\dagger U\,|\psi\rangle = 1$ and therefore $U^\dagger U = I$. Here by $A^\dagger$ we denote the *adjoint* operator of A, that is, the operator that satisfies $(\langle x|\,A^\dagger)^\dagger = A\,|x\rangle$ for every x.

Definition 2. *A* Hermitian *operator is an operator that satisfies $A^\dagger = A$.*

Commonly used operators on qubits are *Pauli* matrices $I, \sigma_x, \sigma_y, \sigma_z$ and Hadamard transform H described as follows.

$$\begin{aligned}
\sigma_x &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} && \text{Maps: } |0\rangle \to |1\rangle\,;\ |1\rangle \to |0\rangle\,;\ \tfrac{|0\rangle+|1\rangle}{\sqrt{2}} \circlearrowleft \\
\sigma_y &= \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} && \text{Maps: } |0\rangle \to i\,|1\rangle\,;\ |1\rangle \to -i\,|0\rangle \\
\sigma_z &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} && \text{Maps: } |0\rangle \to |0\rangle\,;\ |1\rangle \to -\,|1\rangle \\
H &= \tfrac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} && \text{Maps: } |0\rangle \to \tfrac{|0\rangle+|1\rangle}{\sqrt{2}};\ |1\rangle \to \tfrac{|0\rangle-|1\rangle}{\sqrt{2}}
\end{aligned}$$

Postulate 2 stems from Srödinger equation for physical systems, namely

$$i\hbar \frac{d\,|\psi\rangle}{dt} = H\,|\psi\rangle$$

where H is a fixed Hermitian operator known as the Hamiltonian of a closed system.

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

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

and the state of the system after the measurement is

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

The measurement operators satisfy the completeness equation,

$$\sum_m {M_m}^\dagger M_m = I \ .$$

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

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

We will mostly see the following types of measurements. Suppose $|v_1\rangle, |v_2\rangle, \ldots, |v_d\rangle$ form an orthonormal basis. Then $\{M_i = |v_i\rangle\langle v_i|\}$ is a quantum measurement. From state $|\psi\rangle$ in this measurement we will obtain

$$\frac{|v_i\rangle\,\langle v_i|\psi\rangle}{|\langle v_i|\psi\rangle|} \text{ with probability } |\langle v_i|\psi\rangle|^2 \ .$$

Definition 3. *A* projector *is a Hermitian matrix with eigenvalues 0 and 1. The subspace with eigenvalue 1 is the subspace associated with this operator.*

Suppose $S_1, S_2, \ldots, S_k$ are orthogonal subspaces that span the state space. Then $\{P_i\}$ is a quantum measurement where P_i is the projector onto S_i. We can write

$$|\psi\rangle = \alpha_1 |\psi_1\rangle + \alpha_2 |\psi_2\rangle + \cdots + \alpha_k |\psi_k\rangle \ ,$$

where $|\psi_i\rangle \in S_i$. Then this measurement takes $|\psi\rangle$ to $|\psi_i\rangle$ with probability $|\alpha_i|^2$.

Postulate 4. *The state space of a composite quantum system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state $|\psi_i\rangle$, then the joint state of the total system is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$.*

Definition 4. *Let S_1 and S_2 be Hilbert spaces with bases $|e_1\rangle, \ldots, |e_k\rangle$ and $|f_1\rangle, \ldots, |f_l\rangle$ respectively. Then a* tensor product *of S_1 and S_2 denoted $S_1 \otimes S_2$ is a kl-dimensional space consisting of all the linear combinations of all the possible pairs of original bases elements, that is, of $\{|e_i\rangle \otimes |f_j\rangle\}_{i \le k, j \le l}$ ($|v\rangle \otimes |w\rangle$ is often contracted to $|v\rangle |w\rangle$ or $|vw\rangle$).*

In a more concrete matrix representation the tensor product of two vectors is the *Kronecker product* of vectors. For example,

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{3}{5} \\ \frac{4}{5} \end{pmatrix} = \begin{pmatrix} \frac{3}{5\sqrt{2}} \\ \frac{4}{5\sqrt{2}} \\ \frac{-3}{5\sqrt{2}} \\ \frac{-4}{5\sqrt{2}} \end{pmatrix}$$

The tensor product satisfies the property that the product of two unit vectors is a unit vector. This is to verify as follows. Let $|v_1\rangle = \sum a_i |e_i\rangle$ and $|v_2\rangle = \sum b_j |f_j\rangle$ be two unit vectors. Then

$$|v_1\rangle \otimes |v_2\rangle = \sum a_i |e_i\rangle \otimes \sum b_j |f_j\rangle = \sum a_i b_j |e_i\rangle |f_j\rangle \ .$$

Therefore,

$$||v_1\rangle \otimes |v_2\rangle|^2 = \sum |a_i b_j|^2 = \sum |a_i|^2 \sum |b_j|^2 = ||v_1\rangle|^2 \, ||v_2\rangle|^2$$

Another important property of the tensor product space is that it contains vectors which are not tensor product themselves. For example, it can be easily verified that the vector

$$\frac{1}{\sqrt{2}}(|e_1\rangle |f_2\rangle - |e_2\rangle |f_1\rangle)$$

is not a tensor product itself. Such vectors are called “entangled”.

1 Classical Computation Models

1.1 Turing Machines

Though not quite the first, *Turing Machines* are perhaps one of the most important models of classical computation. A Turing Machine (or TM) has an infinite read/write tape, of which all but a finite portion is blank. There is some finite *alphabet* of symbols that can be written on the tape, in addition to the blank symbol □. There is a "head" which does the computation. The head (or *control*) can be in a finite set of states. The program is a set of rules of the form

state	symbol	state	symbol	movement
1	'0'	2	'1'	R
1	'1'	2	'1'	R
1	'□'	2	'1'	L
...				

Turing proved that Turing Machines are universal: that is, that there is a universal Turing Machine program that is able to simulate any other Turing Machine's behavior on any input, when that other TM is written appropriately as part of the input to the universal TM.

There is a special state called "HALT". Each TM computes a function f where $f(s)$ for some string s is whatever is left on the tape when the state reaches "HALT," where initially, the machine is in its initial state, the input s is on the tape, and the head is on the leftmost symbol of the input. Actually, this is not quite a function in all cases, because not every TM will ever reach "HALT" on every input, so for some values of s, it may be that $f(s)$ is undefined. When we say that a TM computes a particular function f, this means that it computes that f on every input, and thus it must halt on every input.

1.2 The Halting Problem

This is the famous Halting Problem: given a TM M and an input j, does M halt on j? This is an uncomputable (also called *undecidable*) function.

If the Halting Problem were computable, there would be a TM M which on input i halts if and only if TM number i does *not* halt on input i. But then M would be TM number j for some j, and M could neither halt nor not halt on input j, which is a contradiction. Thus, the halting problem is undecidable.

1.3 The Wires and Gates model

Here, the idea is that computation devices are circuits consisting of a finite number of *gates* joined up by finitely many *wires,* which carry 0s and 1s on them. Each gate is a function of one or two bits. The three basic gates are the NOT gate, the AND gate, and the OR gate, whose operation is simple.

x	y	NOT(x)	AND(x,y)	OR(x,y)
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

A theorem: any boolean function can be composed out of AND, OR, and NOT gates. In fact, these three can be computed using just a NAND gate, which gives the opposite output from an AND gate.

An *acyclic* circuit is a directed graph with gates on the notes, where NOT gates have one input and AND and OR gates have two inputs; each gate may have many outputs. Certain edges may have no gate at their beginning; these are the *input* wires. Others may have no gate at their end; these are the *output* wires. In order to compute the output from the input, we need only propagate values along the wires, computing each gate as we get to it, and using that output on every wire out of the gate. The values that end up on the output wires make up the output of the circuit. (Alternatively, we can place special "input" and "output" gates at the beginning of input wires and the end of output wires, respectively.)

A function f is computable by circuits if there exists a *uniform* family of circuits C_n such that C_n maps x to $f(x)$ for all strings x of length n.

Note: there is a family of circuits such that C_n computes the following function: $f(0^n) = 0$ if TM number n halts on input n, or 1 if TM number n does not halt on input n, and $f(x)$ for $x \neq 0^n$ is 0. However, this family is not *uniform.*

C_n is uniform if the description of C_n can be output by a TM on input n. C_n is polynomially uniform if C_n can be output by a TM on input n, in $\leq p(n)$ steps for some polynomial p.

1.4 Circuits can simulate TMs

In order to show this, we conside the *history* of a TM computation. For each step, we can write down a configuration: the contents of the tape, where the head is, and which state the head is in.

For each space on the tape, we have a wire or two to determine what symbol, if any, belongs there, and a wire to say whether or not the head is there, and some other wires to determine which state the head is in, if it is at this space. Given the wires for a particular space, and for the spaces to the left and right, a circuit can compute what belongs in that space on the tape at the next step, going out on wires similar to the ones that came in. Thus, by making enough of these circuits for individual tape spaces at individual times, we can compute what the Turing Machine computes.. almost.

The problem is the infinite tape. We can make this circuit/tape arbitrarily broad, and arbitrarily deep, but how deep does it need to be, and how broad? Certainly if the TM in question takes time $\leq p(n)$ for all inputs of size n, then there is a uniform family of circuits C_n computing the same function with $|C_n| \leq \alpha p(n)^2$, where we simply make the circuit as broad as it is deep, with dimensions $p(n)$ for each.

1.5 Reversible computation

A result of Landower, mentioned in the first lecture: erasing a bit takes energy. Some gates are reversible (NOT) while others are not (AND). The FANOUT gate (a gate that replicates its input twice, to represent branches in the wires) is also reversible.

We can, however, do computation reversibly, using Toffoli gates. The essential Toffoli gate is the controlled NOT gate, which takes 3 inputs and has 3 outputs. If the first two bits are 11, the output switches the third bit and leaves the first two alone. Otherwise, the output is the same as the input. This gate is universal: we can compute a NOT gate by using two wires with 1s on them with the wire we wish to put a NOT gate on. We can compute an AND gate by computing the Toffoli gate on $(x, y, 0)$, which will output $(x, y, x \wedge y)$. (We can also do a FANOUT gate: $(1, x, 0)$ goes to $(1, x, x)$. Of course, doing this leads to having junk wires, which is a bit problematic. However, Bennett [1973] showed that any circuit computation can be made reversible *without* junk wires if the input is kept around.

The idea of Bennett's argument is that we can do the following:

INPUT	00000000000000000000	0000000000
INPUT	OUTPUT JUNK	0000000000
INPUT	OUTPUT JUNK	OUTPUT
INPUT	00000000000000000000	OUTPUT

The idea here is that we compute the first transition by computing the function using Toffoli gates, which produces some junk. Then, we can use the zeros that are untouched to copy the output, and finally, we can just compute the first part backwards on the INPUT-OUTPUT-JUNK, which gives us the INPUT and the 0s again, thus getting only INPUT, OUTPUT and 0s. Actually, we don't have to keep the input around in the middle; if it gets corrupted in the first step, this will be undone in the third.

2 Quantum Gates

This idea of gates is fairly similar to what we can do in quantum computers. A quantum gate is a linear map (matrix) on a vector space. Single qubit gates are 2×2 matrices, 2-qubit gates are 4×4 matrices, and 3-qubit gates are 8×8 matrices. It is open whether we need any larger gates than 3-qubit gates. In any case, they are very difficult to construct.

Here are the 1-qubit gates we have.

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$
These are all 1-qubit gates. Here is a 2-qubit gate:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

so that

$$\begin{array}{ccc} 00 & \rightarrow & 00 \\ 01 & \rightarrow & 01 \\ 10 & \rightarrow & 11 \\ 11 & \rightarrow & 10 \end{array}$$

What happens if we change the basis and look at the CNOT gate? Let $|+\rangle = 1/\sqrt{2}(|0\rangle + |1\rangle), |-\rangle = 1/\sqrt{2}(|0\rangle - |1\rangle)$.

$$|+\rangle|+\rangle = 1/2(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \overset{CNOT}{\longrightarrow} |+\rangle|+\rangle$$

$$|+\rangle|-\rangle = 1/2(|00\rangle - |01\rangle + |10\rangle - |11\rangle) \overset{CNOT}{\longrightarrow} 1/2(|00\rangle - |01\rangle - |10\rangle + |11\rangle) = |-\rangle|-\rangle.$$

Since $CNOT$ is symmetric, $CNOT(|-\rangle|-\rangle) = |+\rangle|-\rangle$ and $CNOT(|-\rangle|+\rangle) = |-\rangle|+\rangle$.

This is quite interesting: CNOT remains a controlled not bit, but now, it is the *second* bit that determines whether or not we flip the first!

(Note: H is the matrix that changes the $|0\rangle, |1\rangle$ basis to the $|+\rangle, |-\rangle$.)

$$\begin{array}{ccccc} H & -- & \cdot & -- & H \\ & & | & & \\ H & -- & \oplus & -- & H \end{array}$$

yeilds a controlled not switched the other way:

$$\begin{array}{ccc} -- & \oplus & -- \\ & | & \\ -- & \cdot & -- \end{array}$$

Lecture 5: Quantum Circuits and a Simple Quantum Algorithm

Scribed by: Dion Harmon

Department of Mathematics, MIT

September 18, 2003

1 Administrativa

- On homework 1, the book uses σ_1, σ_2, and σ_3 for σ_x, σ_y, and σ_z.
- We'll go over the homework next lecture.

2 Review from last lecture

2.1 Classical circuits [§3.1.2 pp 129–135]

Classical circuits take input bits (and possibly some work bits) on lines and are supposed to use a network of various logic gates to deliver output bits to other lines. Circuits can compute any function that a Turing machine can compute. It can be proved that if a Turing machine can compute a function G in $f(n)$ steps for input of length n there is a uniform family of circuits (see below) computing G such that the circuit with n input bits has $\leq cf(n)^2$ where c depends on the complexity of the Turing machine.

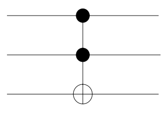
A uniform family of circuits is a set of circuits with one circuit for each number of input bits. Some Turing machine must be able to construct these circuits in the following manner: on input n, the machine outputs a description of the circuit for n input bits. These are the only sorts of circuits we can realistically work with.

2.2 Reversible circuits and computation [§3.2.5 pp 153–161]

Reversible circuits are circuits which do not lose information. This implies more than just keeping the input bits around during computation: information conservation is a local rather than global statement. Given the output from any gate in the reversible circuit, we must be able to uniquely determine the bits on the gate's input lines. This implies global conservation: given the output from the circuit, we can unambiguously determine the input. Reversible computations are defined similarly (see textbook). A Toffoli gate is a controlled controlled flip:

$$\begin{pmatrix} 1 &&&&&&& \\ & 1 &&&&&& \\ && 1 &&&&& \\ &&& 1 &&&& \\ &&&& 1 &&& \\ &&&&& 1 && \\ &&&&&&& 1 \\ &&&&&& 1 & \end{pmatrix}$$

or



or

IN	OUT
000	000
001	001
010	010
011	011
100	100
101	101
110	111
111	110

This gate is universal: with ancillary input zeros, it can be used to compute any function.

3 Quantum programs and quantum circuits [Chapter 4, particularly §4.6 pp 202–204]

Quantum programs are usually specified in terms of circuits (though other models do exist). The model used in the book (and in class) is described in §4.6 of the textbook on page 202. The major points are summarized below:

- **Classical resources:** For efficiency and simplicity, we can use classical computations when we want (but not directly on the quantum state space). We show later in the lecture that this is not needed, but it can make describing problems much simpler.
- **Suitable state space:** The quantum portion of our state space is a 2^n-dimensional Hilbert space which is the tensor product of n quantum bits. Product states of the form $|x_0\rangle \otimes |x_1\rangle \otimes \cdots \otimes |x_n\rangle$ with $x_i \in \{0, 1\}$ are computational basis states of the computer.
- **Ability to prepare states in the computational basis:** We assume we can set our quantum state space to any state of the computational basis in n steps.
- **Ability to perform quantum gates:** We can apply quantum gates to any set of our qubits of fixed cardinality. Usually we say three or fewer qubits, but we could get away with only two. Note that all of the quantum gates "operate" on the entire state space, but
- **Ability to perform measurements in the computational basis:** We can measure the quantum state vector in computational basis.

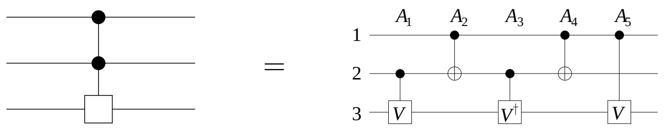
A uniform family of quantum circuits is a set of circuits $\{C_n\}$ such that a *classical* Turing machine can produce a description of C_n on input n in time polynomial in the length of n.

4 Arbitrary quantum gates from controlled gates (and not gates)

We now show that we can build up arbitrary quantum gates (on any number of bits) by using operators which operate on at most two bits at a time.

4.1 A doubly controlled gate from singly controlled gate [§4.3]

We wish to build a three qubit controlled controlled gate from some combination of singly controlled gates. The controlled controlled gate does nothing unless bits one and two are both one, when it applies the unitary transform U to bit three. We claim that



does the trick where $V^2 = U$. It is straight forward to check, but we can do the matrix multiplication for rigor. The matrix for the controlled controlled gate is

$$\begin{pmatrix} I & & & \\ & I & & \\ & & I & \\ & & & U \end{pmatrix}$$

where the identity matrices are all 2×2. The two-bit gates are labeled A_1 through A_5. Their actions are straight forward to determine, and their matrices can be read off by examination or a little scratch work. They are:

$$A_1 = \begin{pmatrix} I & & & \\ & V & & \\ & & I & \\ & & & V \end{pmatrix} \qquad A_2 = \begin{pmatrix} I & & & \\ & I & & \\ & & & I \\ & & I & \end{pmatrix} = A_4$$

$$A_3 = \begin{pmatrix} I & & & \\ & V^\dagger & & \\ & & I & \\ & & & V^\dagger \end{pmatrix} \qquad A_5 = \begin{pmatrix} I & & & \\ & I & & \\ & & V & \\ & & & V \end{pmatrix}$$

Carrying out the multiplications sequentially yields the following.

$$A_2A_1 = \begin{pmatrix} I & & & \\ & V & & \\ & & & V \\ & & I & \end{pmatrix} \qquad A_3A_2A_1 = \begin{pmatrix} I & & & \\ & I & & \\ & & & V \\ & & V^\dagger & \end{pmatrix}$$

$$A_4A_3A_2A_1 = \begin{pmatrix} I & & & \\ & I & & \\ & & V^\dagger & \\ & & & V \end{pmatrix} \qquad A_5A_4A_3A_2A_1 = \begin{pmatrix} I & & & \\ & I & & \\ & & I & \\ & & & V^2 \end{pmatrix}$$

We can always take the square root of a unitary matrix (and the root will also be unitary), so picking an appropriate V is always possible.

If U is the bit flip, the above construction makes a Toffoli gate. We can use this to compute any computable function with our quantum computer (though we would then lose the hypothesized time benefits of the quantum computer).

4.2 More complicated gates [§4.5.1 pp 189–191]

Consider an arbitrary $2^n \times 2^n$ unitary matrix U representing a quantum gate on n qubits. We can zero out the ith element of the first column by left multiplying by

$$V_1 = \begin{pmatrix} \alpha^\dagger & 0 & \cdots & \beta^\dagger & \cdots \\ 0 & 1 & \cdots & & \\ \vdots & & \ddots & & \\ \beta & & \cdots & -\alpha & \cdots \\ \vdots & & & \vdots & \ddots \end{pmatrix}.$$

The row with β as the first element is the ith row and

$$\begin{aligned} \alpha &= \frac{u_{11}}{\sqrt{|u_{11}|^2 + |u_{i1}|^2}}, \\ \beta &= \frac{u_{i1}}{\sqrt{|u_{11}|^2 + |u_{i1}|^2}}. \end{aligned}$$

It is easy to check that ${V_1}^\dagger V_1 = I$ so $V_1 U$ is also unitary. We can continue using similar V's to zero out the rest of the first column at which point the matrix will be of the form

$$V_{2^n-1} \cdots V_1 U = \begin{pmatrix} 1 & 0 & 0 & \cdots \\ 0 & u'_{22} & u'_{23} & \cdots \\ 0 & u'_{32} & \ddots & \\ \vdots & \vdots & & \end{pmatrix}$$

The first row is in this form as the matrix is unitary. Now we can continue with matrices of the form

$$V_1 = \begin{pmatrix} 1 & 0 & \cdots & & & \\ 0 & \alpha^\dagger & 0 & \cdots & \beta^\dagger & \cdots \\ 0 & 0 & 1 & \cdots & & \\ \vdots & \vdots & & \ddots & & \\ 0 & \beta & & \cdots & -\alpha & \cdots \\ \vdots & & & & \vdots & \ddots \end{pmatrix}$$

to zero out the next layer of the matrix. Continuing in this form we will get the identity matrix. The V matrices gates are permutations with some not gates and finally a controlled gate where all but one of the bits are control bits, and the other bit can be operated on arbitrarily.

5 Deutsch-Jozsa algorithm [§1.4.4 pp 34–36]

Problem We are given an oracle for a black box function $f : \{0,1\}^n \rightarrow \{0,1\}$. The function f is guaranteed to be either constant or balanced and we wish to decide which. (A balanced function is 0 on exactly half of all possible inputs and hence 1 on the other half.)

Classical random algorithm Deciding if f is balanced or constant is possible in polynomial time with bounded error provided we have an oracle for f. Pick m random n bit inputs (uniformly). If f on all the inputs is the same, decide that the function is constant. If we get both 0 and 1 on some of the inputs, decide the function is balanced. If the function is balanced, the probability that we get the same output on all is 2^{-m+1} for $m \geq 2$. (We need it to be the same as the *first* output, which can be either 0 or 1). Thus, to get an error probability of less than (say) 1/3, we need only pick $m > 3$ and the error probability decreases exponentially in the number of random inputs we pick.

5.1 Quantum algorithm

We can compute the answer with certainty using quantum computation. Input is of the form $|0\rangle^{\otimes n} |1\rangle$, and we assume that we have a black box quantum gate, B_b on $n+1$ bits satisfying

$$B_b |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle .$$

where x is an n-bit binary number.

In the following description, we make use of the Hadamard gate

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} .$$

Let $|x\rangle$ represent the tensor product of $|0\rangle$ and $|1\rangle$ vectors in the order of the binary representation of x and $x \cdot z$ be the dot product of bit representations of x and z. Note that

$$H^{\otimes n} |x\rangle = \sum_{z=0}^{2^n-1} (-1)^{x \cdot z} |z\rangle$$

We begin:

Apply $H^{\otimes(n+1)}$: $$\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) .$$

Apply B_b: $$\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle (-1)^{f(x)} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) .$$

Apply $H^{\otimes n} \otimes I_{n+1}$: $$\frac{1}{2^n} \sum_{x=0}^{2^n-1} \sum_{z=0}^{2^n-1} (-1)^{f(x)+x \cdot z} |z\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) .$$

Now measure all bits in the $|0\rangle$, $|1\rangle$ basis. There are two possibilities. Exchange the order of summations.

- If $f(x)$ is constant, the dot product for $z = 0$ is uniformly 0 so the summation over x with $z = 0$ produces a factor of 2^n. Since the state vector is normalized, it must be $(-1)^{f(0^n)} |0\rangle^{\otimes n} (|0\rangle - |1\rangle)/\sqrt{2}$. Thus we will measure $|0\rangle^{\otimes n}$ for the first n bits with certainty if f is constant.
- If $f(x)$ is balanced, the x summation over $|0\rangle^{\otimes n}$ is negative on half the x and positive on the other half and so cancels out: there is no probability of measuring $|0\rangle^{\otimes n}$.

Lecture 11: Applications of Grover's Search Algorithm

Scribed by: Yuan-Chung Cheng

Department of Chemistry, MIT

October 9, 2003

In this lecture we will cover several applications of Grover's search algorithm, and show that Grover's search algorithm is optimal.

Grover's search algorithm: $\exists$ function $\mathbf{O}$ such that

$$\begin{array}{rcll}\mathbf{O}|x\rangle & = & -|x\rangle & \textit{if } x \in \mathbf{T}, \\ \mathbf{O}|x\rangle & = & |x\rangle & \textit{otherwise.}\end{array}$$

We want to find elements in the target set $\mathbf{T}$. If there are N $|x\rangle$'s and M targets, after $c \cdot \sqrt{\frac{N}{M}}$ iterations of $\mathbf{G}$, most amplitude will be in the target set. The following operator $\mathbf{G}$ is performed in each iteration:

$$\mathbf{G} = \mathbf{H}^{\otimes n}(2|0\rangle\langle 0| - \mathbf{I})\mathbf{H}^{\otimes n} \cdot \mathbf{O},$$

and the initial state is prepared in $\frac{1}{\sqrt{2^n}} \sum_x |x\rangle$.

In addition to perform the search of a target in a set of elements, the Grover's search algorithm has other possible applications. Here we will demonstrate how to use Grover's search algorithm to speed up classical algorithms, and do target counting.

Speed up classical algorithms

NP problems are those for which a solution is easy to check, but not necessarily easy to find. Examples of NP problems are:

Travel Salesman Problem (TSP): Having n cities and $\left(\begin{array}{c} n \\ 2 \end{array}\right)$ distances between them, we want to find a tour of the cities of length at most D.

3-SAT: Given a boolean formula of totally n variables in conjunctive normal form with at most 3 variables in each clause, ex. $(x_1 \vee x_3 \vee x_7) \wedge (x_1 \vee \bar{x_5} \vee x_9) \wedge (x_2 \vee \bar{x_3} \vee x_{11}) \ldots$, we want to know if there exist a set of $\{x_i, i = 1 \ldots n\}$ such that the whole formula is satisfied.

Using the 3-SAT problem as an example, classically we have to search exhaustively and try every set of values, therefor the algorithm takes 2^n time. Here we will show that the problem can be done using the Grover's search algorithm in $2^{n/2}$ time.

A better way to start is to find a maximal set of disjoint clauses in the formula. We can rename the variables and write the formula as

$$(x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6) \wedge \ldots \wedge (x_{m-2} \vee x_{m-1} \vee x_m) \wedge \ldots (x_i \vee x_{n-1} \vee x_n),$$

where $1 < m < n$, and $1 < i < m$. In this form, we have a disjoint clauses set (from x_1 to x_m), and the rest of the formula are 2-SAT. Since we know polynomial time algorithm for 2-SAT problems, we can easily solve the left-over part. Notice that we need to try only 7 values for each clause,and there are at most $\frac{n}{3}$ clauses. The time is $7^{n/3} \approx 2^{.93n}$. If we apply Grover's search algorithm, we can do with time in $O(2^{.93n/2})$. This is almost close to the best classical algorithm $O(2^{.43n})$. This demonstrates that applying Grover's algorithm to a relatively simple classical algorithm can gain substantial speed up (however, not exponential speed up). It is possible that the Grover's algorithm and be used on best classical algorithm can gain speed up that can not be done classically.

Approximate counting: How many solutions are there?

Grover's algorithm can be combined with the quantum phase estimation algorithm to approximately count the number of targets in the set, i.e. the value of M. Classically, suppose you need to sample A values to get $M \pm \varepsilon M$. The expected number in target set is $A \cdot \frac{M}{N}$, standard deviation is $\sqrt{A \cdot \frac{M}{N}}$. The estimate of M, M_{est}, is the number of targets you found times N/A.

$$\begin{aligned} M_{est} &= \tfrac{N}{A} \cdot number\, found \\ &= \tfrac{N}{A} A \tfrac{M}{N} \pm \tfrac{N}{A}\sqrt{\tfrac{AM}{N}} \\ &= M \pm \sqrt{\tfrac{NM}{A}} \\ &= M\left(1 \pm \sqrt{\tfrac{N}{AM}}\right) \end{aligned}$$

Therefore, $M(1+\varepsilon) = M(1 \pm \sqrt{\frac{N}{AM}})$. To get M within $\pm\varepsilon$ range, we need to sample A times, where $A \approx \frac{N}{M\varepsilon^2}$. The efficient of the estimation is $\sim 1/\varepsilon^2$.

We can use Grover's algorithm to improve the performance. Recall that in the Grover's search algorithm, if we define

$$\begin{aligned} |\alpha\rangle &= \tfrac{1}{\sqrt{N-M}} \sum_{x \notin \mathbf{T}} |x\rangle \\ |\beta\rangle &= \tfrac{1}{\sqrt{M}} \sum_{x \in \mathbf{T}} |x\rangle, \end{aligned}$$

what Grover's algorithm does in each iteration is to rotate the state vector in the $|\alpha\rangle, |\beta\rangle$ basis counter-clockwise for θ angle, where $\frac{\theta}{2} = \arcsin(\sqrt{\frac{M}{N}})$. In the $|\alpha\rangle, |\beta\rangle$ basis, we can write the $\mathbf{G}$ operator as a rotation matrix (for $N \gg M$):

$$\mathbf{G} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

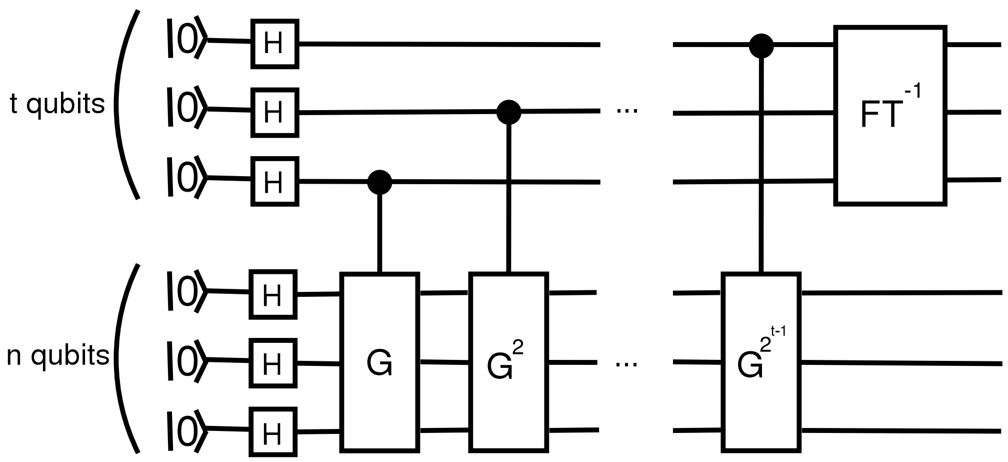


Figure 1: Circuit for quantum counting.

The eigenvalues of the matrix are $e^{\pm i\theta}$. Therefore, we can use phase estimation algorithm to obtain the phase factor θ, and compute the number of targets in the set. Figure 1. shows the circuit for approximate quantum counting to t qubits accuracy on a quantum computer.

We now analyze the efficiency of the quantum counting algorithm. Suppose we get θ to t bits accuracy, i.e. $\theta_{est} = \theta \pm \frac{1}{2^t}$. We need 2^t calls to the function. By the definition of θ, we have

$$\sin^2 \frac{\theta}{2} = \frac{M}{N}.$$

To estimate the error, we take the derivative on both side to obtain the relationship between the change of θ, $\delta\theta$, and the change of M, δM:

$$2 \sin \frac{\theta}{2} \cos \frac{\theta}{2} \cdot \delta\theta = \frac{\delta M}{N}.$$

Using $N \gg M$, and $\theta \ll 1$, we obtain

$$\delta M = 2N\delta\theta \sqrt{\frac{M}{N}} \cos \frac{\theta}{2} \leq 2\delta\theta\sqrt{MN}.$$

We define $\varepsilon M = \delta M \approx 2\delta\theta\sqrt{MN}$, so $\delta\theta = \sqrt{\frac{M}{N}} \cdot \frac{2}{\varepsilon}$. Therefore, in terms of ε, the number of calls needed to get to the accuracy of ε is $\sim \sqrt{\frac{M}{N}} \cdot \frac{2}{\varepsilon}$. This is the square-root of what we need in the classical algorithm.

Grover's search algorithm is optimal

Suppose we have a new quantum algorithm that can find state $|x\rangle$. We define $\mathbf{O}_x$ as the oracle of finding state $|x\rangle$:

$$\mathbf{O}_x = \begin{cases} |x\rangle & \rightarrow & -|x\rangle \\ |y\rangle & \rightarrow & |y\rangle \quad if\, y \neq x \end{cases}$$

Let $|\psi\rangle$ be the starting state of the algorithm. One can write any algorithm of finding state $|x\rangle$ as

$$\sqrt{1-\varepsilon}|x\rangle + \sqrt{\varepsilon}|g\rangle \approx \mathbf{U}_n\mathbf{O}_x \ldots \mathbf{U}_3\mathbf{O}_x\mathbf{U}_2\mathbf{O}_x\mathbf{U}_1\mathbf{O}_x|\psi\rangle,$$

where $|g\rangle$ is composed of states $\perp$ to $|x\rangle$. For a successful algorithm, the outcome of the operations should be very close to the target state, that is, ε has to be very small.

Let's also define

$$\begin{aligned} |\psi_k^x\rangle &= \mathbf{U}_k\mathbf{O}_x\mathbf{U}_{k-1}\mathbf{O}_x \ldots \mathbf{U}_1\mathbf{O}_x|\psi\rangle, \\ |\psi_k\rangle &= \mathbf{U}_k\mathbf{U}_{k-1} \ldots \mathbf{U}_1|\psi\rangle. \end{aligned}$$

Notice that $|\psi_k^x\rangle$ and $|\psi_k\rangle$ differ by the oracle operator $\mathbf{O}_x$ that separates the target state from other states. From this, we can define the Euclidean distance between these two states after k operations:

$$D_k = \sum_x \| |\psi_k^x\rangle - |\psi_k\rangle \|^2. \tag{1}$$

This distance can serve as a measurement of how much this algorithm can made to distinguish $|x\rangle$ and $|y\rangle$ states. We will show that $D_k \leq O(k^2)$. To solve Grover's problem, one needs at least $D_{least} \approx O(N)$. Therefore, at least $k \sim \sqrt{N}$ is necessary, which is what Grover's search algorithm does.

Consider the Euclidean distance of the $k+1$ states:

$$\begin{aligned} D_{k+1} &= \sum_x \|\mathbf{U}_{k+1}\mathbf{O}_x|\psi_k^x\rangle - \mathbf{U}_{k+1}|\psi_k\rangle\|^2 \\ &= \sum_x \|\mathbf{O}_x|\psi_k^x\rangle - |\psi_k\rangle\|^2 \\ &= \sum_x \|\mathbf{O}_x(|\psi_k^x\rangle - |\psi_k\rangle) + (\mathbf{O}_x - \mathbf{I})|\psi_k\rangle\|^2, \end{aligned}$$

where we have applied the property that the unitary operator $\mathbf{U}_{k+1}$ preserves the norm. Use the inequality $\|b+c\|^2 \leq \| b \|^2 + 2 \| b \| \cdot \| c \| + \| c \|^2$, we obtain

$$\begin{aligned} D_{k+1} &\leq \sum_x \left[\|\mathbf{O}_x(|\psi_k^x\rangle - |\psi_k\rangle)\|^2 + 2\|\mathbf{O}_x(|\psi_k^x\rangle - |\psi_k\rangle)\| \cdot \|(\mathbf{O}_x - \mathbf{I})|\psi_k\rangle\| + \|(\mathbf{O}_x - \mathbf{I})|\psi_k\rangle\|^2 \right] \\ &\leq \sum_x \||\psi_k^x\rangle - |\psi_k\rangle\|^2 + \sum_x 2 \cdot \||\psi_k^x\rangle - |\psi_k\rangle\| \cdot \|(\mathbf{O}_x - \mathbf{I})|\psi_k\rangle\| + \sum_x \|(\mathbf{O}_x - \mathbf{I})|\psi_k\rangle\|^2 \\ &\leq D_k + 4 \cdot \sum_x \||\psi_k^x\rangle - |\psi_k\rangle\| \cdot \|\langle x|\psi_k\rangle \cdot |x\rangle\| + 4, \end{aligned} \tag{2}$$

where we have used the property that the unitary operator $\mathbf{O}_x$ preserves the norm and $\mathbf{O}_x - \mathbf{I} = -2|x\rangle\langle x|$. Now apply the Cauchy's inequality,

$$\sum_x \||\psi_k^x\rangle - |\psi_k\rangle\| \cdot \|\langle x|\psi_k\rangle \cdot |x\rangle\| \leq \sum_x \sqrt{\||\psi_k^x\rangle - |\psi_k\rangle\|^2} \cdot \sqrt{\|\langle x|\psi_k\rangle \cdot |x\rangle\|^2},$$

in Eq. (2), we obtain the recursion relation

$$D_{k+1} \leq D_k + 4\sqrt{D_k} + 4.$$

Assume $D_k \leq 4k^2$, we get

$$D_{k+1} \leq (4k^2 + 8k + 4) = 4(k+1)^2.$$

Hence we have shown that the best you can do to distinguish the target states and other states using j operations is $D_j \leq 4 \cdot j^2$. Note that the least distance you need to distinguish target states in totally N states is $D_{least} \approx O(N)$. Therefore, at least $k \sim \sqrt{N}$ is necessary to solve the Grover's problem. This shows that you can not do better than Grover's algorithm.

18.435 Lecture 13
October 16th, 2003

Scribed by: Eric Fellheimer

This lectured started with details about the homework 3:

Typo in Nielsen and Chuang: If you pick random x such that gcd(x, N) = 1, x < N and N is the product of m distinct primes raised to positive integral powers, and r is the order of x mod N, then the probability that r is even and $x^{r/2} \neq -1 \bmod N \geq 1-\frac{1}{2^{m-1}}$. The book erroneously has the power of 2 as m opposed to m -1.

In exercise 5.20 : The book states at the bottom of the problem that a certain sum has value $\sqrt{\frac{N}{R}}$ when l is a multiple of N/r. The answer should actually be N/r when l is a multiple of N/r.

Also, there will be a test on Thursday, October 23rd
-Open books
-Open notes
-in class
-covers through Grover's algorithm, teleportation, and superdense coding

From last lecture:
We know that quantum circuits can simulate Quantum Turing Machines (QTM) with polynomial overhead.

Now we will look in the reverse direction: implementing a Turing machine to simulate a quantum circuit.

We will need to show that we can approximate any gate with a finite set of gates.

<u>Thm</u>: CNOT gates and one-qubit gates are universal for quantum computation

Proof:

We already know gates of the form $\begin{bmatrix} a & b & & \\ g & d & & \\ & & 1 & \\ & & & 1 \end{bmatrix}\begin{bmatrix} a & & b & \\ & 1 & & \\ g & & d & \\ & & & 1 \end{bmatrix}\begin{bmatrix} a & & & b \\ & 1 & & \\ & & 1 & \\ g & & & d \end{bmatrix}$ are sufficient, where $\begin{bmatrix} a & b \\ g & d \end{bmatrix}$ is a unitary matrix.

We know use the fact that:

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & & 1 \\ & & 1 & \end{bmatrix}\begin{bmatrix} a & & b & \\ & 1 & & \\ g & & d & \\ & & & 1 \end{bmatrix}\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & & 1 \\ & & 1 & \end{bmatrix} = \begin{bmatrix} a & & & b \\ & 1 & & \\ & & 1 & \\ g & & & d \end{bmatrix}$$

This reduces the proof to only finding the first 2 of the 3 matrices above. The first 2, however, can be considered single-qubit operations. So if we can construct arbitrary single qubit operations, our proof is complete. We now look at forming controlled T^2 gates with

$$T = \begin{bmatrix} e^{i\Phi_1} & \\ & e^{-i\Phi_1} \end{bmatrix} \text{ or } T = \begin{bmatrix} \cos(q) & -\sin(q) \\ \sin(q) & \cos(q) \end{bmatrix}$$

We now know:

$$\begin{bmatrix} e^{i\Phi_1} & \\ & e^{-i\Phi_1} \end{bmatrix}\begin{bmatrix} \cos(q) & -\sin(q) \\ \sin(q) & \cos(q) \end{bmatrix}\begin{bmatrix} e^{i\Phi_2} & \\ & e^{-i\Phi 2} \end{bmatrix}$$

give arbitrary determinant 1, unitary 2X2 matrices. Thus, our proof is complete.

We know suppose Alice and Bob share stat (1/2) (|0000> + |0101> + |1011> + |1110>) where Alice owns the first 2 qubits.
They can use this state to teleport Alice's 2 qubits to Bob. To do this, Alice must send Bob 4 classical bits.

Quantum linear optics as a means for computation

- suppose you have a probabilistic method of applying CNOT gates and you know when it has worked
- you can measure in the Bell basis
- you can de single qubit operations

We argue that this strange set of requirements actually allows universal computation

We want

$$s_{1}^{'} \otimes s_{2}^{'} \quad CNOT \quad s_{1}^{-1} \otimes s_{1}^{-2} |a,b\rangle = CNOT|a,b\rangle$$

We now want to know that for each a,b {X, Y, Z, I} there exists a', b' such that

$$s_{a'} \otimes s_{b'} \quad CNOT \quad s_{a} \otimes s_{b} = CNOT$$

Knowing that the Pauli matrices are self inverses, we get:

$$s_{a'} \otimes s_{b'} \quad = CNOT \quad s_{a} \otimes s_{b} \quad CNOT$$

$$CNOT \ s_{x}(1)\ CNOT = s_{x}(1) \otimes s_{x}(2)$$

$$CNOT \ s_{x}(2)\ CNOT = s_{x}(2)$$

$$CNOT \ s_{z}(1)\ CNOT = s_{z}(1)$$

Thus, we have:

$$CNOT \ s_{y}(1)\ CNOT = -i\, CNOT s_{z}(1) s_{x}(1) CNOT$$

$$CNOT \ s_{y}(1)\ CNOT = -i\, CNOT s_{z}(1)\ CNOT \ CNOT s_{x}(1) CNOT$$

$$CNOT \ s_{y}(1)\ CNOT = -i\, s_{z}(1)\ s_{x}(1) s_{x}(2)$$

$$CNOT \ s_{y}(1)\ CNOT = \ s_{y}(1)\ s_{x}(2)$$

We have shown that we can teleport through controlled not gates to use quantum linear optics as a means of quantum computation.

Adiabatic Quantum Computation

Physical systems have Hamiltonians H such that $\langle \Psi \mid H \mid \Psi \rangle$ = E is the energy of the system.

H is a Hermitian operator.

The wave function satisfies the Schrödinger Equation:

$$i\hbar \frac{d|\ \Psi\rangle}{dt} = H|\ \Psi\rangle$$

<u>Thm</u>: If you change the Hamiltonian sufficiently slowly, and start in the ground state, you remain in the ground state.

Here, "sufficiently slow" means T is proportional to 1/|g|^2, where g is the gap between first and second energy eigenvalues.

If we start in state H_{init} and end in H_{final}, H_{init} / H_{final} are sums of Hamiltonians involving no more than a few qubits.

Finally, there is a theorem which states that using this setup can be equated to using quantum circuits.

Lecture 14: Cluster States

Scribed by: Ilia Mirkin

Department of Mathematics, MIT

October 21, 2003

A cluster state is a highly entangled rectangular array of qubits. We measure qubits one at a time. The wiring diagram tells us which basis to measure each qubit in, and which order to measure them in. A wiring diagram is represented by connecting up the dots (which represent qubits) with lines. A junction of two separate lines represents a gate. These gates do not have to be unitary, but if done right, are.

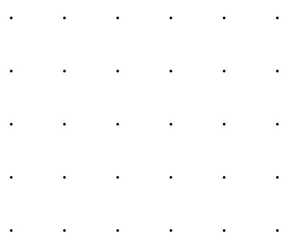


Figure 1: Unconnected cluster states

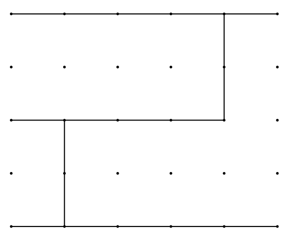


Figure 2: Wiring diagram for two gates

Measurement on a wiring diagram is done by first measuring all the qubits that are not in the wiring diagram (i.e. unconnected) in the σ_z basis. Once those qubits are measured, we measure the qubits in the circuit from left to right in the specified basis.

Cluster state given by eigenvalue equations. The neighborhood of a qubit are the up, down, left, and right qubits.

$$K^{(a)} = \sigma_x^{(a)} \otimes \bigotimes_b \sigma_z^{(b)} | b \in \text{neighborhood}(a) \tag{1}$$

The claim is that $K^{(a)}$ commutes with $K^{(b)}$ when $a \neq b$. To show that this is true, we can look at the following cases:

$$\text{neighborhood}(a) \cap \text{neighborhood}(b) = \emptyset \tag{2}$$

When this is true, then there is absolutely no overlap between $K^{(a)}$ and $K^{(b)}$ and thus the two commute.

$$\text{neighborhood}(a) \not\ni b \tag{3}$$

This means that neighborhoods overlap, but that the qubit b is not in the neighborhood of a in an arrangement such as

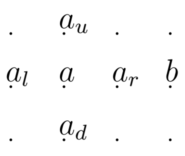


Figure 3: Neighborhoods overlap

$$\begin{aligned}
K^{(a)} &= \sigma_x^{(a)} \otimes \sigma_z^{(a_r)} \otimes \cdots \quad (4)\\
K^{(b)} &= \sigma_x^{(b)} \otimes \sigma_z^{(a_r)} \otimes \cdots \quad (5)\\
K^{(a)}K^{(b)} &= \sigma_x^{(a)} \otimes \sigma_z^{(a_r)} \otimes \cdots \otimes \sigma_x^{(b)} \otimes \sigma_z^{(a_r)} \otimes \cdots \quad (6)
\end{aligned}$$

And in the third case, a and b are adjacent to each other.

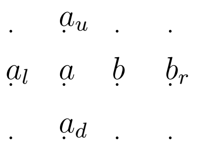


Figure 4: a and b are adjacent

$$K^{(a)}K^{(b)} = \sigma_x^{(a)} \otimes \sigma_z^{(b)} \otimes \cdots \otimes \sigma_x^{(a)}\sigma_z^{(b)} \cdots \tag{7}$$

In all three cases $K^{(a)}$ and $K^{(b)}$ both commute, so the claim holds. This means that $K^{(a)}$ are all simultaneously diagonalizable. Any simultaneous eigenvector of $K^{(a)}, a \in C$ (cluster) is a cluster state. Each $K^{(a)}$ has eigenvalue ± 1, making for 2^n vectors of eigenvalues $\{K_a\}$.

$\left|\phi_{\{\kappa_a\}}\right\rangle_C$ is a cluster state with eigenvalue κ_a on qubit a, $\{\kappa_a\} = \{\pm 1\}$. Thus $\left\langle \phi_{\{\kappa_a\}} | \phi_{\{\kappa_a'\}} \right\rangle_C = 0$ if $\{\kappa_a\} \neq \{\kappa_a'\}$. For example,

$$\begin{aligned} \kappa_b &= +1 && (8)\\ \kappa_a &= -1 && (9)\\ \left\langle \phi_{\{\kappa_a\}} | \phi_{\{\kappa_a'\}} \right\rangle_C = -\left\langle \phi_{\{\kappa_a\}} | K_b | \phi_{\{\kappa_a'\}} \right\rangle_C &= -\left\langle \phi_{\{\kappa_a\}} | \phi_{\{\kappa_a'\}} \right\rangle_C = 0 && (10) \end{aligned}$$

If $\{\kappa_a\} = \{\kappa_a'\}$ except for $\kappa_b = -\kappa_b'$, then $\sigma_z^{(b)} \left|\phi_{\{\kappa_a\}}\right\rangle_C = \left|\phi_{\{\kappa_a'\}}\right\rangle_C$

$$\begin{aligned} K_a \sigma_z^{(b)} \left|\phi_{\{\kappa_a\}}\right\rangle_C &= (-1)^{\delta_{ab}} \sigma_z^{(b)} K_a \left|\phi_{\{\kappa_a\}}\right\rangle_C && (11)\\ &= (-1)^{\delta_{ab}} \sigma_z^{(b)} \kappa_a \left|\phi_{\{\kappa_a\}}\right\rangle_C && (12)\\ &= \kappa_a' \sigma_z^{(b)} \left|\phi_{\{\kappa_a\}}\right\rangle_C && (13) \end{aligned}$$

Cluster state for $\forall_a \kappa_a = 1$, start in state $|\psi\rangle_C = \bigotimes_a |+\rangle_a$, where $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. We then apply S_{ab} to all neighbors a, b.

$$\begin{aligned} S_{ab} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} && (14)\\ &= \frac{1}{2}\left(I + \sigma_z^{(a)} + \sigma_z^{(b)} - \sigma_z^{(a)} \otimes \sigma_z^{(b)}\right) && (15) \end{aligned}$$

Here are a few examples of gates that can be made using wiring diagrams:

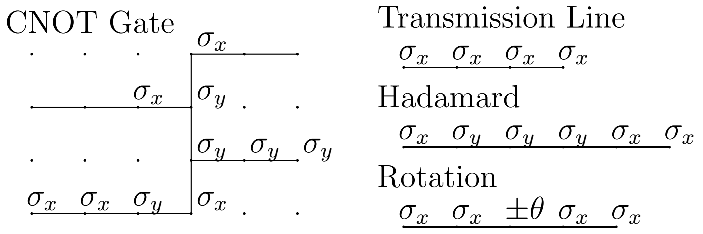

CNOT Gate
σ_x
σ_x
σ_y
σ_y σ_y σ_y
σ_x σ_x σ_y
σ_x
Transmission Line
σ_x σ_x σ_x σ_x
Hadamard
σ_x σ_y σ_y σ_y σ_x σ_x
Rotation
σ_x σ_x $\pm\theta$ σ_x σ_x

Figure 5: CNOT Gate, Transmission Line, Hadamard, Rotation

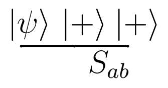


Figure 6: Transmission line

We also know that S_{ab} commutes with $S_{a'b'}$. In the $|0\rangle$, $|1\rangle$ basis, S_{ab} can be represented by a diagonal matrix, which means that they have to commute. $K^{(a)} \bigotimes_{a,b} S_{ab} |+\rangle^{\otimes n}$ is an eigenvector of $K^{(a)}$.

Demonstration of a transmission line effect:

$$\begin{aligned} S_{ab} |+\rangle |+\rangle &= S_{ab} \frac{1}{2} (|00\rangle + |01\rangle + |10\rangle + |11\rangle) && (16) \\ &= \frac{1}{2} (|00\rangle + |01\rangle + |10\rangle - |11\rangle) && (17) \\ &= \frac{1}{\sqrt{2}} (|+\rangle |0\rangle + |-\rangle |1\rangle) && (18) \end{aligned}$$

With this, we apply S_{ab} and measure both a and b in the $|+\rangle$, $|-\rangle$ basis. This is equivalent to measuring in the $\langle ++| S_{ab}$, $\langle +-| S_{ab}$, $\langle -+| S_{ab}$, $\langle --| S_{ab}$ basis, which is also equivalent to measuring in the $\frac{1}{\sqrt{2}} (\langle 0+| + \langle 1-|)$ basis. In this way we get the teleportation effect on the original $|\psi\rangle$.

Lecture 16: Quantum Error Correction

Scribed by: Dah-Yoh Lim

October 30, 2003

1 Introduction

Today we are going to look at how one can do error correction in the quantum world. In the PreShannon days, simple repetition codes were used: to transmit bit 0, it is first encoded into a string of zeros, say 0000. Similarly, 1 gets encoded into 1111. One can prove that if you want to reduce the error rate to $1/n$, and the channel flips bits with probability $1/\epsilon$, then you need roughly $\frac{\log n}{\log(1/\epsilon)}$ repetitions.

2 Quantum Analog to Repetition Code

The analog to the repetition code is to say encode $|0\rangle$ as $|000\rangle$ and $|1\rangle$ as $|111\rangle$. Note that this is not the same as copying/cloning the bit, since in the quantum world we know that cloning is not possible.

So for instance, under this encoding E the EPR state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ would be encoded as $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$.

2.1 Bit Errors

Suppose there is a bit error σ_X at the third qubit.

$$\sigma_X^{(3)} E(|0\rangle) = \sigma_X^{(3)}(|000\rangle) = |001\rangle.$$

Similarly,

$$\sigma_X^{(3)} E(|1\rangle) = \sigma_X^{(3)}(|111\rangle) = |110\rangle.$$

To correct bit errors, simply project onto the subspaces $\{|000\rangle, |111\rangle\}$, $\{|001\rangle, |110\rangle\}$, $\{|010\rangle, |100\rangle\}$, $\{|100\rangle, |011\rangle\}$.

2.2 Phase Errors

Suppose there is a phase error σ_Z.

$$\sigma_Z^{(j)} E(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) = \sigma_Z^{(j)} \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle) = E(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)),$$

for $j = 1, 2, 3$. Therefore, if we use this code, single bit errors can be corrected but phase errors will be come 3 times more likely!

Recall that

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

This takes bit errors to phase errors. If we apply H to the 3-qubit protection code, we get:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \rightarrow \frac{1}{2\sqrt{2}}(|000\rangle + ... + |111\rangle),$$

and

$$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \rightarrow \frac{1}{2\sqrt{2}}(|000\rangle - |001\rangle + ... - |111\rangle),$$

with a negative sign iff the string contains an odd number of 1s.

Now consider the following 3-qubit phase error correcting code:

$$|0\rangle \rightarrow \frac{1}{2}(|000\rangle + |011\rangle + |101\rangle + |110\rangle),$$

and

$$|1\rangle \rightarrow \frac{1}{2}(|011\rangle + |100\rangle + |010\rangle + |001\rangle).$$

Now if there is a phase error on the first qubit:

$$\begin{aligned} \sigma_Z^{(1)} E(|0\rangle) &= \frac{\sigma_Z^{(1)}}{2}(|000\rangle + |011\rangle + |101\rangle + |110\rangle) \\ &= \frac{1}{2}(|000\rangle + |011\rangle - |101\rangle - |110\rangle). \end{aligned}$$

Next, note that $\sigma_Z^{(i)} E(|0\rangle)$ and $\sigma_Z^{(i)} E(|1\rangle)$ are orthogonal, for $i = 1, 2, 3$. Therefore, to correct phase errors we can project onto the four subspaces $\{\sigma_Z^{(i)} E(|0\rangle), \sigma_Z^{(i)} E(|1\rangle)\}$ and $\{E(|0\rangle), E(|1\rangle)\}$. Now we have a code that corrects phase error but not bit errors.

2.3 Bit and Phase Errors

If we concatenate the codes that corrected bit and phase errors respectively, then we can get a code that corrects both errors. Consider:

$$E(|0\rangle) = \frac{1}{2}(|000000000\rangle + |000111111\rangle + |111000111\rangle + |111111000\rangle)$$

$$E(|1\rangle) = \frac{1}{2}(|111111111\rangle + |111000000\rangle + |000111000\rangle + |000000111\rangle).$$

It is easy to see that this corrects bit errors. Note that the error correcting procedure does not collapse the superposition, so it can be applied to superpositions as well. There is a continuum of $\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$ that one can apply to the 1st qubit.

In general, phase errors can be expressed as $\begin{bmatrix} 1 & 0 \\ 0 & e^{2i\theta} \end{bmatrix} = e^{i\theta} \begin{bmatrix} e^{-i\theta} & 0 \\ 0 & e^{i\theta} \end{bmatrix} \equiv R_\theta.$

$$\begin{aligned}
&R_\theta(\frac{1}{2}(|000\rangle + |011\rangle + |101\rangle + |110\rangle)) \\
&\quad = \frac{1}{2}(e^{-i\theta}|000\rangle + e^{-i\theta}|011\rangle + e^{i\theta}|101\rangle + e^{i\theta}|110\rangle) \\
&\quad = \frac{1}{2}\frac{e^{i\theta}+e^{-i\theta}}{2}(|000\rangle + |011\rangle + |101\rangle + |110\rangle) + \frac{1}{2}\frac{-e^{i\theta}+e^{-i\theta}}{2}(|000\rangle + |011\rangle - |101\rangle - |110\rangle) \\
&\quad = \cos\theta E(|0\rangle) - i\sin\theta\sigma_Z^{(1)}E(|0\rangle),
\end{aligned}$$

i.e. phase error on the 1st qubit.

With our 9-qubit code, let's say we apply $\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$ to some qubit. Since

$$\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} = \frac{\alpha+\delta}{2}I + \frac{\alpha-\delta}{2}\sigma_Z + \frac{\beta+\gamma}{2}\sigma_X + i\frac{\beta-\gamma}{2}\sigma_Y,$$

we can separate its operation on $\eta E|0\rangle + \mu E|1\rangle$ as follows:

$$\begin{aligned}
&\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}(\eta E|0\rangle + \mu E|1\rangle) \\
&\qquad = \frac{\alpha+\delta}{2}(\eta E|0\rangle + \mu E|1\rangle) + \frac{\alpha-\delta}{2}\sigma_Z(\eta E|0\rangle + \mu E|1\rangle) \\
&\qquad\quad + \frac{\beta+\gamma}{2}\sigma_X(\eta E|0\rangle + \mu E|1\rangle) + i\frac{\beta-\gamma}{2}\sigma_Y(\eta E|0\rangle + \mu E|1\rangle).
\end{aligned}$$

However, we know that projection onto $|\phi\rangle$ is equivalent to applying the projection matrix $|\phi\rangle\langle\phi|$; therefore, we see that the code corrects phase errors too.

3 7 bit Hamming Code

The Hamming code encodes 4 bits into 7 bits. The 2^4 codewords are:

S_0	S_1
0000000	1111111
1110100	1011000
0111010	0101100
0011101	0010110
1001110	0001011
0100111	1000101
1010011	1100010
1101001	0110001

Recall that a linear code is a code where the sum of two codewords (mod 2) is another codeword. The Hamming code is a linear code, i.e. one can chose 4 basis elements generated by $G_C = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$, where the rowspace of this matrix gives all the codewords. The parity check matrix is $H_C = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$, which is exactly the generator matrix for the dual code (the set of vectors orthogonal to every vector in the code C).

3.1 Quantum Hamming Code

The quantum analog of the Hamming Code is as follows:

$$|0\rangle \rightarrow \frac{1}{2^{3/2}} \sum_{v \in H_C} |v\rangle; \qquad |1\rangle \rightarrow \frac{1}{2^{3/2}} \sum_{v \in H_C} |v + e\rangle,$$

where e is the string of all 1s.

Note that this corrects σ_X on any qubit (because of the properties of the Hamming code). Also, if we apply the Hadamard transformation to the quantum Hamming code, we can correct phase errors as well:

$$\begin{aligned}
H^{\otimes 7} E|0\rangle &= \frac{1}{2^{3/2}} \sum_{v \in H_C} H^{\otimes 7} |v\rangle \\
&= \frac{1}{2^{3/2}} \frac{1}{2^{7/2}} \sum_{x=0}^{2^7-1} \sum_{v \in H_C} (-1)^{x \cdot v} |x\rangle \\
&= \frac{1}{2^{7/2}} 2^3 \sum_{x \in G_C} |x\rangle \\
&= \frac{1}{\sqrt{2}} (E|0\rangle + E|1\rangle). \\
H^{\otimes 7} E|1\rangle &= \frac{1}{2^{3/2}} \sum_{v \in H_C} H^{\otimes 7} |v + e\rangle \\
&= \frac{1}{2^{3/2}} \frac{1}{2^{7/2}} \sum_{x=0}^{2^7-1} \sum_{v \in H_C} (-1)^{x \cdot (v+e)} |x\rangle \\
&= \frac{1}{2^{7/2}} 2^3 \sum_{x \in G_C} (-1)^{x \cdot e} |x\rangle \\
&= \frac{1}{\sqrt{2}} (E|0\rangle - E|1\rangle) \\
&= EH|1\rangle.
\end{aligned}$$

So, the σ_X and σ_Z errors are "independent", and therefore this code can correct σ_X error in any qubit and σ_Z error in any other qubit.

Lecture 19: How to Build Your Own Quantum Computer

Guest Lecturer: Isaac Chuang
Scribed by: Fen Zhao

Department of Mathematics, MIT

November 13, 2003

1 The DiVicenzo Criteria

The DiVicenzo Criteria list four things required for quantum computing: robust qubits, a universal gate set, a fiducial input state, and projective measurements.

1.1 Robust Qubits

A quantum qubit is based on 2 level quantum systems. One must also remember that a tensor product Hilbert space is needed; for example a harmonic oscillator is not a tensor product space and therefore makes a bad quantum computer. In general, a two level atom would make a good quantum computer.

One must also have a long coherence time. This is characterizes how the environment interacts with your ideal qubit system. The imperfect qubit system will have many states besides the two you are interested in. One can think of decoherence as the effects of all the interactions outside your idea set of interactions.

$$|\Psi(t)\rangle = e^{-iHt}|\Psi(0)\rangle$$

$$H = \begin{bmatrix} \mathbf{a} & \mathbf{b} & g & \dots \\ \mathbf{c} & \mathbf{d} & h & \\ e & f & i & \\ \vdots & & & \ddots \end{bmatrix}$$

In the Hamiltonian above, the elements in bold represent the your ideal set of interactions, and everything else is the non-ideal part that causes decoherence.

There are many sources of decoherence. Gravity causes decoherence if one states weights more thant the other. There may be stray long range fields, typically associated with charge. There can be leakage into larger Hilbert spaces; a two state atom may have higher energy levels that the state can move to. However, there does exists true finite spaces in nature, such as spin.

There are two measurements of decoherence, T_1 and T_2. T_1 is called the “longitudinal coherence time,” or the “spin lattice time,” or the “spontaneous emission time,” or the“amplitude damping.” It measures the loss of energy from the system. One can do an experiment to determine T_1. First

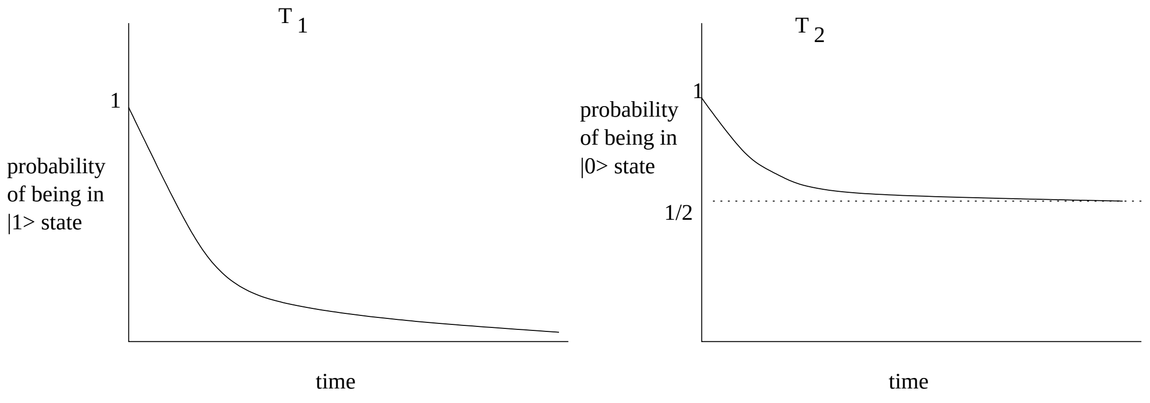


Figure 1: expected results for experiments for T_1 and T_2

initialize the qubit to the ground state $|0\rangle$. Then apply $X = |0\rangle\langle 1| + |1\rangle\langle 0|$, and wait for time t and measure the probability of being in the $|1\rangle$ state. We expect an exponential decay e^{-t/T_i}.

T_2 is called the "transverse coherence time," or the "spin-spin relaxation time," or the "phase coherence time," or the "elastic scattering time," or the "phase damping." One can do an experiment to determine T_2. First initialize the qubit to the ground state $|0\rangle$. Then apply the Hadamard transform H to get the state to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$, wait for time t, apply H again, and measure the probability of being in the $|0\rangle$ state. We expect that the measurement goes to $1/2$ after a long time because after a long time, most likely something popped the state into either $|0\rangle$ or $|1\rangle$ state, which after the H transform sends the state to $\frac{|0\rangle\pm|1\rangle}{\sqrt{2}}$.

In general $T_1 > T_2$.

1.2 Universal Gate Sets

There are many different universal gate sets. We have seen that CNOT and single qubit gates form a universal gate set. CNOT, the Hadamard gate, and $\pi/8$ gate is another universal gate set.

In practice experimentalists use controlled Hamiltonians that they turn on and off for certain time intervals.

$$H = H_{sys} + P_1(t)X + P_2(t)Y...$$

For example, for an NMR setup, we may have

$$H = \frac{\pi\hbar}{2} J\sigma_z \otimes \sigma_z + P_1(t) I \otimes \sigma_x + P_2(t) I \otimes \sigma_y...$$

In the real world, there is no such thing as time dependent Hamiltonians. So how do we perform a sequence of operations? It is a just an approximation; fundemantally we will always have decoherence and will need fault tolerance.

What happens is that our classical controls are actually quantum systems, and we must take into action the back action of the control system on our system. $P_1(t)X$ is just an approximation; in reality, we have a Jaynes-Cumming type interaction Hamiltonian:

$$H = \hbar\omega N + \delta Z + g(a^\dagger \sigma_- + a_+^\sigma)$$

where $\sigma_\mp = \frac{X\pm Y}{2}$. In less abstract terms, there is decoherence that results after a photon interacts with a qubit because the photon will carry away information about the state of the qubit.

2 Implementations

In general, the challenge of quantum computing lies in the fact that quantum systems have short lifetimes, and that we need to control it externally.

System	T (sec)
NMR	10^2 to 10^8
Ion Trap	10^{-3}
Dots	10^{-6}
Microwave Cavity	10^0
Optical Cavity	10^{-5}

Table 1: Some relaxation times for different implementations

2.1 Cavity QED

The Hamiltonian is described by atom, photon, and atom-photon interactions. The qubit is the single photon. Gates for $\pi/8$ and Hadamard are implemented by beam splitter and the like. Turchette managed to achieve a control Z gate.

2.2 Ion Trap

The Hamiltonian is described by spin, photon-phonon (vibrational mode) atom-photon-phonon interactions. The qubit is the atom (spin) and the phonon. The Deutsch-Joza algorithm has been implemented with ion traps. The system has a lifetime of $O(1-100)$ (order of) milliseconds. Pulsed lasers are the universal gate set. The challenge is to get a stable $O(10)$ Hz laser and cooling the ions to absolute zero.

2.3 NMR

The Hamiltonian is described by spins, spin-spin, and external control photon interactions. The spins interact with chemical bonds. The gates are implemented by radio frequency magnetic field pulses. Factoring with 6 qubits has been accomplished. In terms of robustness, ^1H, ^{13}C, ^9F, ^{14}N has coherence times of $O(1)$ seconds.

2.4 All Silicon Quantum Computer

This implementation implants individual atoms into silicon surrounding and controls the atom electronically. It has a coherence of 100 ms at $T = 9.2K$. This implementation is scalable and uses current techniques of classical computer engineering.

3 Quantum Cryptography

Currently two companies make quantum cryptography systems. The purpose of quantum cryptography is to make it more likely to detect an eavesdropper. They are based on the fact that an eavesdropper measuring a quantum system transmitted will collapse the system.

We can relate various techniques of quantum computing to features of it's implementation. Data compression is related to cooling. Error correction is related to control and T_1 and T_2. Noisy coding is related to precision of measurements. Cryptography is related to entanglement and non-locality.

Lecture 23: Fault-Tolerant Quantum Computation

Scribed by: Jonathan Hodges

Department of Nuclear Engineering, MIT

December 4, 2003

1 Introduction

Before von Neumann proposed classical fault-tolerance in the 1940's, it was assumed that a computational device comprised of more than 10^6 components could not perform a computation wihout encountering a fatal hardware error. Von Neumann proved that one could indeed make the computation work, as long as a certain degree of overhead was tolerable. Thus follows the classical fault-tolerance theorem:

Theorem 1 (Classical Fault-Tolerance). *A computation of length n using perfect computational components can be executed reliably (i.e. with probability $1 - \frac{1}{n^\alpha}$ for polynomial α) using $\mathcal{O}(n \log n)$ steps, provided the components work with probability 1 - ϵ of the time and that the faults encountered are independent.*

We can sketch von Neumann's proof of Fault-Tolerance as follows: Given classical AND,NOT, and OR gates let us encode a 0 into many 0's for *c log n* times, where c is some constant.

$$0 \rightarrow 0000000 \tag{1}$$

Now take two identical copies of "0", call them a and b, and put them through the AND gate. Ideally one should get 1111111. Instead, suppose the strings received are 1110101 on a and 0111111 on b. If one performs a bit-wise AND on each successive bit of the bit strings a and b, the result is 0110101. Taking "triples" of bits of this resulting addition, one performs a majority vote. Thus, if one bit has an error probablity of p , two bits in a triple being erroneous occurs with probabiltiy $3p^2$. As long as p is small, one can perform operations with fault-tolerance. The same type of proof can be shown for NOT and OR gates, thus giving universality.

In short, if one has compenents of a computer whose fidelity are high enough, and adding additional compents is relatively easy, then the computation is indeed plausible. As it turns out the critics were too critical. Your desktop computer does not even use a software error-correction for doing computations, as the ϵ for our silicon-based hardware has become increasingly small.

2 Using classical techniques for quantum computation

Classically, four methods exist for dealing with fault-tolerant computing, but only one of these will prove feasible. Consistency checks, like the parity of a bit string, work classically, but in the quantum world are simply not powerful enough. Checkpoints require stoping the computation at a specific point, checking the result, then starting the computation again. The probablistic nature of quantum mechanics and the no-cloning theorem make this technique useless for QC. Classically, one might make many copies of the computation to perform a massive redundancy scheme; however, this errs like consistency checks, as it is not "powerful enough" for quantum computations. Thus, one is left with error correcting codes, which have previously been shown portable from the classical to the quantum domain.

3 Quantum Fault Tolerance

In order to take an errorless quantum computation to a fault-tolerant computation, one first encodes each qubit into a quantum error correcting code. Every gate in the circuit should then be replaced by a fault tolerant version of it. Finally, insert an error correction step after every gate. Above we argued that fault-tolerance in classical computations need only AND, NOT, OR gates. For universal quantum computation only the CNOT and single-qubit gates are required, but formulating fault-tolerant operations becomes easier with a finite gate set. CNOT, Hadamard, σ_x, σ_z, T, Toffoli, and $\frac{\pi}{8}$ will be proven useful.

Theorem 2 (Kitaev-Solovay Theorem). *Given a set of gates on SU(2) (or SU(k) generally) that generates a dense set in SU(2), then any gate $U \in SU(2)$ can be approximated to ϵ using $\mathcal{O}(log^c \frac{1}{\epsilon})$ gates where $1 \leq c \leq 2$. See Appendix 3 of Nielsen and Chuang for more details.*

3.1 Fault Tolerance of σ_x

In order to show that a σ_x gate can be done with fault tolerance, let us encode 1 qubit into k qubits using a CSS Code (the Steane Code).

$$|0\rangle \longrightarrow \frac{1}{\sqrt{|C_2|}} \sum_{x \in C_2} |x\rangle \tag{2}$$

$$|1\rangle \longrightarrow \frac{1}{\sqrt{|C_2|}} \sum_{x \in C_2} |x + v\rangle \tag{3}$$

Since dim(C_1) is k, dim(C_2) is k-1, this code will encode a single qubit, and satisfies the inequality $0 \subseteq C_2 \subseteq C_1$. The codewords v are those not overlapping the two classical codes; $v \in C_1 - C_2$. Since the encoded $|0\rangle$ and $|1\rangle$ are orthogonal, a σ_x should just interchange the two encodings. These two states are separated by v, which amounts to peforming a σ_x on each individual qubit. Now suppose an error is made in performing σ_x on one of the qubits, where the errors on each qubit are uncorrelated. Then this code will be able to correct for these errors, resulting in a quantum error correcting code and operation that performs σ_x with fault-tolerance.

3.2 Fault Tolerance of σ_z

By using the Steane code above, the equivalent of σ_z on an unencoded qubit is to apply σ_z to those individual qubits with a 1 in the codeword w where $w \in C_2^{\perp} - C_1^{\perp}$. This results in a state $|a\rangle$ acquiring a phase of $(-1)^{a.w}$. Under such a transformation the code words become

$$\frac{1}{\sqrt{|C_2|}} \sum_{x \in C_2} |x\rangle \longrightarrow \frac{1}{\sqrt{|C_2|}} \sum_{x \in C_2} (-1)^{x.w} |x\rangle = \frac{1}{\sqrt{|C_2|}} \sum_{x \in C_2} |x\rangle \tag{4}$$

$$\frac{1}{\sqrt{|C_2|}}\sum_{x\in C_2}|x+v\rangle \longrightarrow \frac{1}{\sqrt{|C_2|}}\sum_{x\in C_2}(-1)^{x.w}(-1)^{v.w}|x+v\rangle = -\frac{1}{\sqrt{|C_2|}}\sum_{x\in C_2}|x+v\rangle \tag{5}$$

since at least 1 vector in C_1 gives $v.w = 1$.

3.3 Fault Tolerance of Hadamard Gate

The fault tolerance of the Hadamard gate under this CSS encoding can be seen under the additional constraint $C_1 = C_2^{\perp}$. If the function $E(x)$ represents the act of encoding the bit, the action of a Hadamard on an encoding qubit must follow the transformations:

$$E(|0\rangle) \longrightarrow \frac{1}{\sqrt{2}}(E(|0\rangle) + E(|1\rangle)) \tag{6}$$

$$E(|1\rangle) \longrightarrow \frac{1}{\sqrt{2}}(E(|0\rangle) - E(|1\rangle)) \tag{7}$$

A Hadamard transformation of each individual qubit, $H^{\otimes k}$, applied to $E(|0\rangle)$ will give precisely the correct encoded transformation.

$$\begin{aligned} H\frac{1}{\sqrt{|C_2|}}\sum_{x\in C_2}|x\rangle &= \frac{1}{2^{\frac{k}{2}}\sqrt{|C_2|}}\sum_{y,x\in C_2}(-1)^{x.y}|x\rangle \\ &= \frac{1}{\sqrt{|C_2^{\perp}|}}\sum_{y\in C_1}|y\rangle \\ &= E(\frac{|0\rangle+|1\rangle}{\sqrt{2}}) \end{aligned}$$

The last line follows because $E(|0\rangle)$ is composed of all codeword in C_2 and $E(|1\rangle)$ is everything in C_1, but not in C_2 by definition. A Hadamard transformation on $E(|1\rangle)$, simply adds in the phase factor of $(-1)^{y.v}$, which obviously follows from above. This phase factor will be unity if $y \in C_2$, but -1 if $y \in (C_1 - C_2)$, thus appropriately adding a phase to the states in the code that are $E(|1\rangle)$.

3.4 Fault Tolerance of CNOT Gate

The σ_x, σ_z, and H gates can all be performed on a single encoded qubit with fault-tolerance because these gates are always applied to single qubits. Likewise, given two single-qubit encoded states, one can perform CNOT operations between the k^{th} qubit of one set, with the k^{th} qubit of the other. Thus there are at most two qubits interacting for a single gate, making errors independent among the sets of qubits, and thus correctible with the CSS Code. This can be shown as follows:

$$\begin{aligned} U_{CNOT}^{\otimes k}\frac{1}{|C_2|}\sum_{x\in C_2}|x+v_a\rangle\otimes\sum_{y\in C_2}|y+v_b\rangle &= \frac{1}{|C_2|}\sum_{x\in C_2}|x\rangle\otimes\sum_{y\in C_2}|x+y+v_a+v_b\rangle \\ &= \frac{1}{|C_2|}\sum_{x\in C_2}|x\rangle\otimes\sum_{y\in C_2}|y+v_a+v_b\rangle \end{aligned}$$

If $v_a = v_b$, then $v_a + v_b = 0$ in binary addition and the vector is unchanged. Otherwise, the resulting state will be a string of 1's added to all states $|y\rangle$, which is just the encoding $E(|1\rangle)$. If an error occurs in any of the two-qubit CNOT operations, this will result in v_a or v_b not being all 0's or all 1's, and the CSS code will correct the the appropriate state.

4 Error Correction With Fault-Tolerant Precision

Both classical and quantum error correction schemes require encoding information into a code, computing the syndrome of the code after errors may have occurred, then applying a syndrome-dependent correction step to the coding to recover the information. A simple means of checking the syndrome would be to find the parity of a subset of qubits in a code. One could thus perform a series of CNOT gates where a single target qubit will be flipped depending on the states of the controlled qubits. Measurement of this ancilla qubit would unveil the syndrome, but not with fault-tolerant precision.

This can easily be seen under a Hadamard transformation, $H^{\otimes k+1}$, which reverses the direction of the CNOT gates and gives the dual CSS code in the $\mathcal{H}^{\perp}$ space. ($\mathcal{H}$ is the parity check matrix of the code C_1.) Due to the reversed CNOT, if any of the gates have an error, the error will propagate forward in time due to the back-action of the CNOT. The stringent requirement of each error not affecting more than a single qubit (or pair in the FT CNOT construction) is not fulfilled.

Using the idea of single failure points between qubits, as seen in the FT CNOT construction, we start our parity check register on k qubits in the state:

$$|\psi\rangle = \frac{1}{2^{k-1}} \sum_{s \in \mathbb{Z}_2^k} |s\rangle \tag{8}$$

Measurement of $|\psi\rangle$ in the canonical basis will result in either an odd parity bit string, indicating a syndrome of 1, or an even parity string for a 0 syndrome. The state $|\psi\rangle$ can be created by applying the Hadamard transformation to the "cat" state.

$$H^{\otimes k} |\psi\rangle = \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle) \tag{9}$$

(The state $|\mathbf{x}\rangle$ represents a k length string of $\mathbf{x}$'s.)

Now suppose a maximally entangled state can be created and verified by performing a few CNOT gates between the bits of the cat state and an ancilla. Fault tolerance is not an issue here; one only wants to know if the state is maximally entangled. Measuring $|0\rangle$ on the ancilla for a reasonable number of test qubits ensures that the state is in some superposition of the states $|\mathbf{0}\rangle$ and $|\mathbf{1}\rangle$. The Hadamard transform of this state:

$$H^{\otimes k}(\alpha\,|\mathbf{0}\rangle + \beta\,|\mathbf{1}\rangle) = (\frac{\alpha+\beta}{\sqrt{2}})\frac{1}{2^{k-1}} \sum_{s \in even} |s\rangle + (\frac{\alpha-\beta}{\sqrt{2}})\frac{1}{2^{k-1}} \sum_{s \in odd} |s\rangle \tag{10}$$

Thus if $\alpha = \beta$, the state is all zeros and no backaction will occur. The all ones state simply adds the ones vector to the qubits. Thus, fault-tolerant measurements can be obtained.