Lecture 1: Overview of quantum information

January 10, 2006

References

Most of the material in these lecture notes is discussed in greater detail in the following two books, which I recommend you study if you are interested in quantum computation.

1. M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
2. A. Kitaev, A. Shen, and M. Vyalyi. *Classical and Quantum Computation*, volume 47 of *Graduate Studies in Mathematics*. American Mathematical Society, 2002.

Quantum Information

For the remainder of this lecture we will take a first look at quantum information, a concept upon which quantum computation is based.

A probabilistic model

It is helpful to start classically, with a model that will probably seem completely simple to everyone. Imagine that we have some physical device, called X, that has some finite, non-empty set Σ of possible *states*[1]. For example, we might have $\Sigma = \{0, 1\}$, in which case we would think of X as representing a bit. For the following discussion let us restrict ourselves to this example (but keep in mind that everything can easily be generalized to sets other than $\{0, 1\}$).

Suppose that we do not necessarily have complete information about the state of X, but instead represent our knowledge of its state by assigning probabilities to the different states. For example, we might have

$$\Pr[\text{state of } \mathsf{X} \text{ is } 0] = 1/4,$$

$$\Pr[\text{state of } \mathsf{X} \text{ is } 1] = 3/4.$$

Mathematically we can represent this type of knowledge about the state of X with a *probability vector*, which is a column vector whose entries are all nonnegative real numbers that sum to 1. In the case at hand, the associated probability vector is

$$v = \begin{pmatrix} 1/4 \\ 3/4 \end{pmatrix}.$$

[1] Shortly we will change our terminology and use the term *classical states* to refer to elements of Σ, because the term *state* will be used in a different context. Nevertheless, for the time being we will stick with the term *state* when referring to elements of the set Σ.

The understanding is that the entries of v are indexed by Σ, and when we write such a vector in the above form we are using the most natural way of ordering the elements of Σ:

$$v = \begin{pmatrix} 1/4 \\ 3/4 \end{pmatrix} \begin{array}{l} \leftarrow \text{entry indexed by } 0 \\ \leftarrow \text{entry indexed by } 1 \end{array}$$

We may write $v[0]$ and $v[1]$ to refer to the entries of v when necessary.

What happens when you look at X? Of course you will not see a probability vector v. Instead you will see some element of Σ. If our representation of the state of X by a probability vector v is in some way meaningful, you may as well imagine that the state you saw was determined randomly according to the probabilities associated with the various states. Notice that by looking at the state of X you effectively change the description of your knowledge of its state. Continuing with the example above, if you look and see that the state is 0, the description of your knowledge changes from v to a new probability vector w:

$$v = \begin{pmatrix} 1/4 \\ 3/4 \end{pmatrix} \quad \longrightarrow \quad w = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

You know that the state is 0, and the vector w represents this knowledge. If you saw that the state was 1 instead of 0, the vector would become

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

instead.

What sorts of operations can you imagine performing on X? There are not very many deterministic operations: you could initialize X to either 0 or 1, you could perform a NOT operation to X, or you could do nothing to X (which can still be considered an operation even though it has no effect). You could also perform an operation involving randomness—for instance perform a NOT operation with probability 1/100, and otherwise do nothing. I claim that any *physically meaningful* operation can be represented by a matrix, with the effect of the operation being determined by matrix-vector multiplication. For instance, these four matrices

$$\mathrm{INIT}_0 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \quad \mathrm{INIT}_1 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad \mathrm{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \text{and} \quad I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

represent the deterministic operators mentioned above. For example, if our knowledge of the state of X is represented by

$$v = \begin{pmatrix} 1/4 \\ 3/4 \end{pmatrix}$$

and we perform a NOT operation on X, the new probability vector that results is

$$w = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1/4 \\ 3/4 \end{pmatrix} = \begin{pmatrix} 3/4 \\ 1/4 \end{pmatrix}.$$

The probabilistic operation mentioned above is represented by the matrix

$$\begin{pmatrix} \frac{99}{100} & \frac{1}{100} \\ \frac{1}{100} & \frac{99}{100} \end{pmatrix}.$$

All of these matrices have the property that (i) all entries are nonnegative real numbers, and (ii) the entries in each column sum to 1. In other words, every column is a probability vector. Such matrices have a name: they are called *stochastic matrices*. In the simple model we are discussing, physically meaningful operations are described by stochastic matrices. It works the other way as well; any stochastic matrix describes some physically meaningful operation.

As mentioned before, this entire picture is easily generalized to the case where Σ is not necessarily $\{0, 1\}$. In general the dimension of the vectors and matrices will be equal to the size of Σ.

Quantum bits (qubits)

The framework of quantum information works in a similar way to the simple probabilistic model we just saw, but with some key differences. Let us again imagine that we have a physical device called X. As before we imagine that there is some set Σ of possible states of X, and we will again consider for now just the simple case $\Sigma = \{0, 1\}$. At this point, to avoid confusion let us now refer to elements of Σ as *classical states* rather than just *states*. Intuitively you can think of a classical state that you as a human can look at, touch, and recognize without ambiguity. The device X will represent the quantum analogue of a bit, which we call a *qubit*.

We will still represent our knowledge[2] of X with column vectors indexed by Σ, but this time they will not be probability vectors. Instead of representing probability distributions, the vectors represent what we call a *superposition* or just a *state* (by which we mean a *quantum state*). For example, here are a few vectors representing superpositions:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}, \quad \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} \frac{3}{5} \\ \frac{4i}{5} \end{pmatrix}.$$

Notice that the entries in these vectors are not probabilities: they are not necessarily nonnegative (in fact they are not even necessarily real numbers), and they do not necessarily sum to 1. We call these numbers *amplitudes* instead of probabilities. The condition that replaces the probabilities summing to 1 in a probability vector is this: vectors representing superpositions have Euclidean length equal to 1. In the simple case at hand where $\Sigma = \{0, 1\}$, this means that any vector representing a superposition has the form

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

[2]We discussed briefly in the lecture whether or not the column vectors represent knowledge in the same sense as the probabilistic model or something more "actual". My choice of the word "knowledge" is really only intended to stress the similarity with the probabilistic model; and although the question makes for an interesting philosophical discussion, I don't intend that this course will go in that direction. As soon as possible we will be treating everything mathematically and just thinking of these things as vectors.

for $\alpha, \beta \in \mathbb{C}$ satisfying $|\alpha|^2 + |\beta|^2 = 1$.

Similar to the probabilistic case, if you look at the qubit X you will not see a superposition. Instead, you will see either 0 or 1 just like before. The probability associated with the two possible outcomes is given by the absolute value squared of the associated amplitude—so if the superposition of X is represented by the vector

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

and you look at X, you will see 0 with probability $|\alpha|^2$ and 1 with probability $|\beta|^2$. This is why we have the condition $|\alpha|^2 + |\beta|^2 = 1$, because the probabilities have to sum to 1 for the model to make sense. The same rules apply as for the probabilistic case for determining the superposition of X after you look at it: the superposition becomes

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

depending on whether you see 0 or 1, respectively.

So far the model does not seem qualitatively different from the probabilistic model, but that changes a lot when the possible operations that can be performed are considered. Again the possible operations are represented by matrices; but now instead of being stochastic matrices, the matrices that represent valid physical operations correspond to *unitary* matrices. A matrix is unitary if and only if it preserves the Euclidean norm. Fortunately there is a very simple condition to check this: a matrix U is unitary if and only if

$$U^\dagger U = I,$$

where $U^\dagger$ is the conjugate transpose of U (meaning that you take the transpose of U and then take the complex conjugate of each of the entries). For example, these are unitary matrices:

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}, \quad I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \text{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad R_\theta = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

(for any real number θ in the case of R_θ). For example, if X is in a superposition described by

$$v = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

and the operation corresponding to the matrix H (called the Hadamard transform) is performed, the superposition becomes

$$Hv = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}.$$

If you measured X at this point you would see outcome 0 or 1, each with probability 1/2. If you didn't measure and instead applied the Hadamard transform again, the superposition would become

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

To recapitulate, these are the two things you can do to a qubit:

1. **Perform a measurement.** If the superposition of the qubit is

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

and a measurement is performed, the outcome is 0 or 1, with probabilities $|\alpha|^2$ and $|\beta|^2$, respectively. The superposition of the qubit becomes

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

depending on whether the measurement outcome was 0 or 1.

2. **Perform a unitary operation.** For any unitary matrix U, the operation described by U transforms any superposition v into the superposition Uv.

Later on in the course we will see that there are somewhat more general operations and measurements that can be performed, but this simple model will turn out to be sufficient for discussing quite a lot about quantum computing.

Example 1. Suppose your friend has a qubit that he knows is in one of the two superpositions

$$v_0 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \quad \text{or} \quad v_1 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix},$$

but he isn't sure which. How can you help him determine which one it is?

Measuring right away will not help—you would see a random bit in either case. Instead, you should perform the Hadamard transform and then measure. Performing the Hadamard transform changes the superpositions as follows:

$$Hv_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad Hv_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Now if you measure, you will see 0 (with certainty, meaning probability 1) if the original superposition was v_0 and you will see 1 (with certainty) if the original superposition was v_1.

Lecture 2: Overview of quantum information (continued)

January 12, 2006

In the previous lecture we started discussing the basics of quantum information, beginning with the example of single qubit systems. In this lecture we will continue this discussion. In particular, we will discuss multiple qubit systems and a more convenient notation for describing superpositions.

Multiple qubits

In order to talk about what happens when we have multiple qubits, it will be helpful to briefly return to the probabilistic model from before. Suppose that X and Y are devices that implement bits. Then there are 4 possible states of the pair (X, Y), namely 00, 01, 10, and 11. Thus, our set of states Σ corresponding to this pair is now $\{00, 01, 10, 11\}$. In the probabilistic model we represent our knowledge of the state of the pair (X, Y) with a 4 dimensional probability vector. For example we could have the following probability vector:

$$\begin{pmatrix} \frac{1}{8} \\ \frac{1}{2} \\ 0 \\ \frac{3}{8} \end{pmatrix} \begin{array}{ll} \leftarrow & \text{probability associated with state } 00 \\ \leftarrow & \text{probability associated with state } 01 \\ \leftarrow & \text{probability associated with state } 10 \\ \leftarrow & \text{probability associated with state } 11 \end{array}$$

(The vector indices are labeled by the states in the order given by binary notation.) Operations again correspond to stochastic matrices, but this time the matrices are 4×4 matrices. For example, the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

is stochastic. It happens to correspond to the operation where you do nothing if the first bit is 0, but if the first bit is 1 then replace the second bit with a random bit.

The quantum variant works in an analogous way. If we have two qubits (X, Y), then a superposition of these two qubits is a 4 dimensional vector with Euclidean length equal to 1. For example:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{i}{2} \\ -\frac{1}{2} \end{pmatrix}$$

Measurements work the same way as before, except that the outcome will be two bits. For example, measuring the previous superposition gives results as follows:

$$\begin{aligned}
&00 \quad \text{with probability } \left|\tfrac{1}{\sqrt{2}}\right|^2 = \tfrac{1}{2}\\
&01 \quad \text{with probability } |0|^2 = 0\\
&10 \quad \text{with probability } \left|\tfrac{i}{2}\right|^2 = \tfrac{1}{4}\\
&11 \quad \text{with probability } \left|-\tfrac{1}{2}\right|^2 = \tfrac{1}{4}
\end{aligned}$$

We will see next lecture how it works when you just measure one qubit. Unitary operations also work the same way as before, but this time are 4×4 matrices. For example, here is a 2 qubit unitary operation called the controlled-NOT:

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

The same pattern is used for 3 qubits, 4 qubits, etc. The dimension of the vectors and matrices grows exponentially: 8 dimensional vectors for 3 qubits, 16 dimensional vectors for 4 qubits, etc.

By the way, there is no reason why you cannot consider the model for any other choice of Σ, instead of Σ corresponding to all possible strings of a given length. Typically, however, we will focus on the case where $\Sigma = \{0,1\}^n$ for some positive integer n.

Tensor products

Returning again briefly to the probabilistic model, let us suppose that as before X and Y are devices implementing bits, and the two devices are completely uncorrelated with one another—let us say that the probability vector corresponding to X is

$$v = \begin{pmatrix} \frac{2}{3} \\ \frac{1}{3} \end{pmatrix}$$

and the probability vector corresponding to Y is

$$w = \begin{pmatrix} \frac{1}{4} \\ \frac{3}{4} \end{pmatrix}.$$

Then the 4 dimensional probability vector corresponding to the pair (X, Y) is easily determined by multiplying the corresponding probabilities. In particular, the resulting vector is

$$v \otimes w = \begin{pmatrix} \frac{1}{6} \\ \frac{1}{2} \\ \frac{1}{12} \\ \frac{1}{4} \end{pmatrix}.$$

The operation $\otimes$ is called the *Kronecker product* or the *tensor product*. (It is most common in quantum computing to use the term *tensor product* to refer to this operation.) In general, for any two matrices

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,l} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ b_{k,1} & b_{k,2} & \cdots & b_{k,l} \end{pmatrix}$$

we define $A \otimes B$ to be the $nk \times ml$ matrix

$$A \otimes B = \begin{pmatrix} a_{1,1}B & a_{1,2}B & \cdots & a_{1,m}B \\ a_{2,1}B & a_{2,2}B & \cdots & a_{2,m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}B & a_{n,2}B & \cdots & a_{n,m}B \end{pmatrix}.$$

The definition works for vectors by thinking of them as matrices with only one column.

The tensor product satisfies many nice properties. For example, it is an associative operation; $(A \otimes B) \otimes C = A \otimes (B \otimes C)$ for any choice of matrices A, B and C. Thus, it makes sense to talk about products such as $A \otimes B \otimes C \otimes \cdots \otimes Z$ without including parentheses, because it doesn't matter in which order the products are evaluated. Next, we have

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$$

for any choice of matrices A, B, C and D (assuming the sizes of the matrices are such that the products AC and BD make sense). The distributive law holds for tensor products;

$$A \otimes (B + C) = A \otimes B + A \otimes C \quad \text{and} \quad (A + B) \otimes C = A \otimes C + B \otimes C.$$

Also, for matrices A and B and any scalar α, we have

$$(\alpha A) \otimes B = A \otimes (\alpha B) = \alpha(A \otimes B).$$

In other words, scalars "float freely" through the tensor product. A word of warning, however, is that the tensor product is not commutative; in general it may be the case that $A \otimes B \neq B \otimes A$.

Not every probability vector v representing a distribution of (X, Y) can be written as a tensor product. For example,

$$v = \begin{pmatrix} 1/2 \\ 0 \\ 0 \\ 1/2 \end{pmatrix}$$

cannot be written as a tensor product. In this distribution we have

$$\Pr[\text{state of } (\mathsf{X}, \mathsf{Y}) \text{ is } 00] = \Pr[\text{state of } (\mathsf{X}, \mathsf{Y}) \text{ is } 11] = \frac{1}{2}.$$

We say that X and Y are *correlated* in this case. The only way a probability vector can be written as a tensor product is when the associated systems are *uncorrelated* (or independent).

As you might have guessed, we do exactly the same thing in the quantum case as in the classical, probabilistic case. If X and Y are qubits having associated superpositions

$$v = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad \text{and} \quad w = \begin{pmatrix} \gamma \\ \delta \end{pmatrix},$$

then the superposition of the pair (X, Y) is

$$v \otimes w = \begin{pmatrix} \alpha\gamma \\ \alpha\delta \\ \beta\gamma \\ \beta\delta \end{pmatrix}.$$

The superposition

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

is an example of a superposition that cannot be written as a tensor product. In the quantum case, this type of correlation between X and Y is special and we call it *entanglement*. We will talk about entanglement a lot during the course.

We use tensor products for independent (or uncorrelated) operations as well. For example, suppose that we have two single-qubit unitary transformations such as

$$U = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \text{NOT} \quad \text{and} \quad V = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = H,$$

and we perform transformation U on X and V on Y. Then the effect of these two independent operations on any superposition of the pair (X, Y) is determined by the matrix

$$U \otimes V = \begin{pmatrix} 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \end{pmatrix}.$$

This matrix describes the effect of the two independent unitary operations even on entangled states. For example, applying U to X and V to Y when the pair (X, Y) is in the entangled superposition

from above result in the superposition

$$\begin{pmatrix} 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}.$$

Dirac Notation

Because the dimension of vectors representing superpositions and matrices representing unitary transformations grows exponentially in the number of qubits, it quickly becomes difficult to write these things down with the notation we have been using. One way to avoid this problem is to use the *Dirac notation*, named after its inventor Paul Dirac (who made many important contributions to quantum mechanics and mathematical physics). The notation is simple but very convenient.

In the Dirac notation, column vectors are represented by "kets", such as

$$|0\rangle \stackrel{\text{def}}{=} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad |1\rangle \stackrel{\text{def}}{=} \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

We use symbols such as $|\phi\rangle$ and $|\psi\rangle$ to represent arbitrary vectors (even when the symbols ϕ and ψ have been assigned no meaning by themselves). Any vector indexed by the set $\{0, 1\}$ can be represented by a linear combination of $|0\rangle$ and $|1\rangle$, because $\{|0\rangle, |1\rangle\}$ is a basis for this space of vectors. For instance, we would write

$$\frac{1}{\sqrt{2}} |0\rangle + \frac{i}{\sqrt{2}} |1\rangle$$

to represent the vector

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{pmatrix}.$$

Juxtaposition of kets implicitly refers to the tensor product:

$$|\psi\rangle |\phi\rangle \stackrel{\text{def}}{=} |\psi\rangle \otimes |\phi\rangle.$$

For spaces indexed by $\{00, 01, 10, 11\}$ we define

$$|00\rangle \stackrel{\text{def}}{=} |0\rangle |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |01\rangle \stackrel{\text{def}}{=} |0\rangle |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad \text{etc.}$$

The pattern continues in this way for any number of bits. For example, $|1010\rangle$ is a 16 dimensional vector with a 1 in the position indexed by 1010 in binary (which is the eleventh entry because we start with 0000). The vector

$$\frac{1}{\sqrt{2}}|000000\rangle + \frac{1}{\sqrt{2}}|111111\rangle$$

would be written

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \vdots \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} \Bigg\} \text{62 zeroes}$$

in the usual vector notation. An arbitrary vector with entries indexed by $\{0,1\}^n$, which perhaps refers to a superposition of n qubits, can again be written as a linear combination of the elements in the basis

$$\{|x\rangle \,:\, x \in \{0,1\}^n\},$$

for instance as

$$|\phi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle .$$

Example 2. Let us suppose that we have two qubits X and Y in the superposition

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}.$$

Using the Dirac notation we write this superposition as

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle .$$

Suppose now that we perform a Hadamard transform to the first qubit and do nothing to the second qubit. We can determine the effect of these operations on the above superposition of (X,Y) by computing

$$H \otimes I = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

and multiplying:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{pmatrix},$$

which is written

$$\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle$$

in the Dirac notation.

However, we can perform this computation directly and more easily by not converting back and forth between notations, and instead just sticking with the Dirac notation. Let us start by noting that

$$H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \quad \text{and} \quad H|1\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

The starting superposition is

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

which is equivalent to

$$\frac{1}{\sqrt{2}}|0\rangle|0\rangle + \frac{1}{\sqrt{2}}|1\rangle|1\rangle.$$

The superposition after performing a Hadamard transform on the first qubit and doing nothing (performing the identity operation) to the second qubit is

$$\frac{1}{\sqrt{2}}(H|0\rangle)|0\rangle + \frac{1}{\sqrt{2}}(H|1\rangle)|1\rangle.$$

Substituting for $H|0\rangle$ and $H|1\rangle$ and using the distributive law, we get

$$\begin{aligned} &\frac{1}{\sqrt{2}}(H|0\rangle)|0\rangle + \frac{1}{\sqrt{2}}(H|1\rangle)|1\rangle \\ &\qquad = \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)|0\rangle + \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)|1\rangle \\ &\qquad = \frac{1}{2}|0\rangle|0\rangle + \frac{1}{2}|1\rangle|0\rangle + \frac{1}{2}|0\rangle|1\rangle - \frac{1}{2}|1\rangle|1\rangle \\ &\qquad = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle. \end{aligned}$$

The two computations of course agree. The second method is much easier (once you know the basics of how it works), particularly for larger numbers of qubits.

For every ket $|\psi\rangle$ there is a corresponding object $\langle\psi|$, called a "bra". You may think that this is a strange name for a mathematical object, but the names "bra" and "ket" are derived from the the

fact that when you put a bra and a ket together, you get a "bracket". For this to make sense you need to know what a bra is—for any vector $|\psi\rangle$ we define

$$\langle\psi| = (|\psi\rangle)^\dagger,$$

which is the conjugate transpose of $|\psi\rangle$. In other words, $\langle\psi|$ is the row vector you get by transposing $|\psi\rangle$ and taking the conjugate of each of its entries. For instance:

$$|\psi\rangle = \begin{pmatrix} \frac{1+i}{2} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \quad \Rightarrow \quad \langle\psi| = \begin{pmatrix} \frac{1-i}{2} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

Now, when you juxtapose a bra and a ket, the implicit operation is matrix multiplication (thinking of the vectors as matrices with only one row or one column). A row vector times a column vector results in a scalar, and this scalar will be the *inner product* (or *bracket*) of the vectors involved. For instance, if

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad \text{and} \quad |\phi\rangle = \begin{pmatrix} \gamma \\ \delta \end{pmatrix}$$

then

$$\langle\psi|\phi\rangle \stackrel{\text{def}}{=} \langle\psi|\,|\phi\rangle = \begin{pmatrix} \overline{\alpha} & \overline{\beta} \end{pmatrix} \begin{pmatrix} \gamma \\ \delta \end{pmatrix} = \overline{\alpha}\gamma + \overline{\beta}\delta.$$

When you have an expression such as

$$|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x\,|x\rangle$$

it is easy to express $\langle\psi|$ using similar notation; it is

$$\langle\psi| = \sum_{x\in\{0,1\}^n} \overline{\alpha_x}\,\langle x|\,.$$

When you juxtapose a ket and a bra in the opposite order, such as

$$|\psi\rangle\,\langle\phi|\,,$$

you do not get a scalar—a column vector times a row vector gives you a matrix. It is easy to determine the action of this matrix on another vector. For instance,

$$|\psi\rangle\,\langle\phi|\,|\gamma\rangle = |\psi\rangle\,\langle\phi|\gamma\rangle = \langle\phi|\gamma\rangle\,|\psi\rangle\,.$$

Later on when we wish to speak at a higher level of abstraction about computational problems, algorithms, etc., we may refer to $|x\rangle$ where x is some arbitrary mathematical object (such as a matrix, a graph, or a list of numbers). In this case the interpretation is that we are implicitly referring to the encoding of x with respect to some agreed upon encoding scheme.

Lecture 3: Superdense coding, quantum circuits, and partial measurements

January 24, 2006

Superdense Coding

Imagine a situation where two people (named Alice and Bob) are in different parts of the world. Alice has two bits: a and b. She would like to communicate these two bits to Bob by sending him just a single qubit. It turns out that there is no way they can can accomplish this task without additional resources. This is not obvious, but it is true—Alice cannot encode two classical bits into a single qubit in any way that would give Bob more than just one bit of information about the pair (a, b).

However, let us imaging that a long time ago, before Alice even knew what a and b are, that the two of them prepared two qubits A and B in the superposition

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle .$$

Alice took the qubit A and Bob took the qubit B. We say that Alice and Bob *share an e-bit* or *share an EPR pair* in this situation. It is natural to view entanglement as a resource as we will see; and when Alice and Bob each have a qubit and the two of them are in the state above, it is natural to imagine that Alice and Bob share one unit (i.e., one entangled bit, or e-bit for short) of entanglement.

Given the additional resource of a shared e-bit of entanglement, Alice will be able to transmit both a and b to Bob by sending just one qubit. Here is how:

Superdense coding protocol

1. If $a = 1$, Alice applies the unitary transformation

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

   to the qubit A. (If $a = 0$ she does not.)

2. If $b = 1$, Alice applies the unitary transformation

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

   to the qubit A. (If $b = 0$ she does not.)

3. Alice sends the qubit A to Bob. (This is the only qubit that is sent during the protocol.)

4. Bob applies a controlled-NOT operation to the pair (A, B), where A is the control and B is the target. The corresponding unitary matrix is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

5. Bob applies a Hadamard transform to A.
6. Bob measures both qubits A and B. The output will be (a, b) with certainty.

To see that the protocol works correctly, we simply compute the state of (A, B) after the steps involving transformations:

ab	state after step 1	state after step 2	state after step 4	state after step 5
00	$\frac{1}{\sqrt{2}}\|00\rangle + \frac{1}{\sqrt{2}}\|11\rangle$	$\frac{1}{\sqrt{2}}\|00\rangle + \frac{1}{\sqrt{2}}\|11\rangle$	$\left(\frac{1}{\sqrt{2}}\|0\rangle + \frac{1}{\sqrt{2}}\|1\rangle\right)\|0\rangle$	$\|00\rangle$
01	$\frac{1}{\sqrt{2}}\|00\rangle + \frac{1}{\sqrt{2}}\|11\rangle$	$\frac{1}{\sqrt{2}}\|10\rangle + \frac{1}{\sqrt{2}}\|01\rangle$	$\left(\frac{1}{\sqrt{2}}\|1\rangle + \frac{1}{\sqrt{2}}\|0\rangle\right)\|1\rangle$	$\|01\rangle$
10	$\frac{1}{\sqrt{2}}\|00\rangle - \frac{1}{\sqrt{2}}\|11\rangle$	$\frac{1}{\sqrt{2}}\|00\rangle - \frac{1}{\sqrt{2}}\|11\rangle$	$\left(\frac{1}{\sqrt{2}}\|0\rangle - \frac{1}{\sqrt{2}}\|1\rangle\right)\|0\rangle$	$\|10\rangle$
11	$\frac{1}{\sqrt{2}}\|00\rangle - \frac{1}{\sqrt{2}}\|11\rangle$	$\frac{1}{\sqrt{2}}\|10\rangle - \frac{1}{\sqrt{2}}\|01\rangle$	$\left(\frac{1}{\sqrt{2}}\|1\rangle - \frac{1}{\sqrt{2}}\|0\rangle\right)\|1\rangle$	$-\|11\rangle$

When Bob measures at the end of the protocol, it is clear that he sees ab as required.

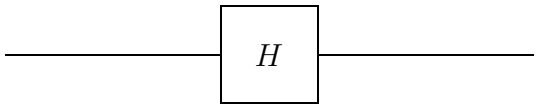
Quantum circuits

The previous discussion of superdense coding used a fairly inefficient way to describe the protocol; basically just plain English along with a few equations. We will need a more efficient way to describe sequences of quantum operations and measurements than this, particularly for when the complexity of the algorithms and protocols has increased. The most common way of doing this is to use *quantum circuit diagrams*, which is the way we will use. The basic idea is as follows:

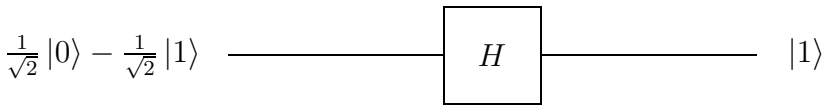
- Time goes from left to right.
- Horizontal lines represent qubits.
- Operations and measurements are represented by various symbols.

It is easiest to describe the model more specifically by using examples.

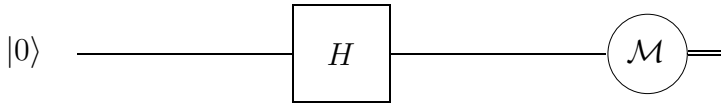
Example 3. The following diagram represents a Hadamard transform applied to a single qubit:



If the input is $|\psi\rangle$, the output is $H|\psi\rangle$. Sometimes when we want to explain what happens for a particular input, we label the inputs and outputs with superpositions, such as:

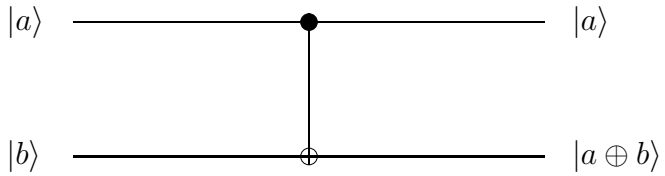


Example 4. Measurements are indicated by circles (or ovals) with the letter $\mathcal{M}$ inside. For example:

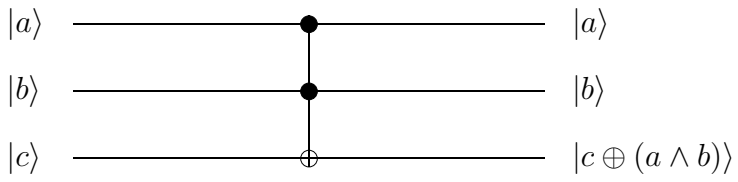


The result of the measurement is a classical value, and sometimes (as in the above diagram) we draw double lines to indicate classical bits. In this case the outcome is a uniformly distributed bit.

Example 5. Multiple-qubit gates are generally represented by rectangles, or have their own special representation. For instance, this is a *controlled-not* operation:

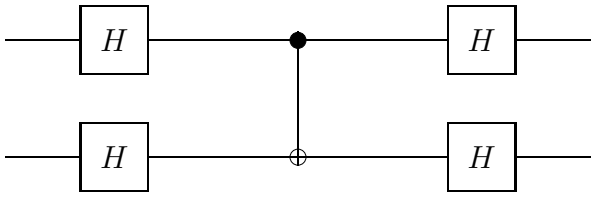


Here the action is indicated for classical inputs (meaning $a, b \in \{0, 1\}$). Along similar lines, here is a *controlled-controlled-not* operation, better known as a *Toffoli gate*:

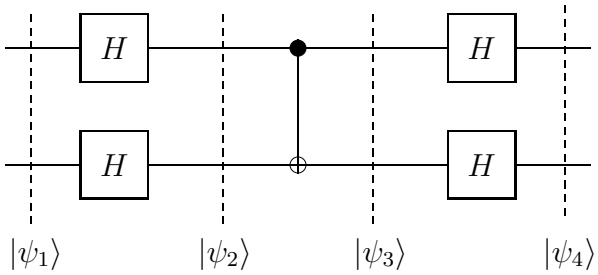


Here the action is described for each choice of $a, b, c \in \{0, 1\}$.

Example 6. Usually we have multiple operations, such as in this circuit:



It is not immediately obvious what this circuit does, so let us consider the superposition that would be obtained at various times for a selection of inputs:



Suppose first that $|\psi_1\rangle = |00\rangle$. Then

$$|\psi_2\rangle = \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle,$$

$$|\psi_3\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|11\rangle + \frac{1}{2}|10\rangle = \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)$$

$$|\psi_4\rangle = |00\rangle.$$

Next suppose that $|\psi_1\rangle = |01\rangle$. Then

$$|\psi_2\rangle = \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle,$$

$$|\psi_3\rangle = \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|11\rangle - \frac{1}{2}|10\rangle = \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)$$

$$|\psi_4\rangle = |11\rangle.$$

Next suppose that $|\psi_1\rangle = |10\rangle$. Then

$$|\psi_2\rangle = \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle - \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle,$$

$$|\psi_3\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle - \frac{1}{2}|11\rangle - \frac{1}{2}|10\rangle = \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)$$

$$|\psi_4\rangle = |10\rangle.$$

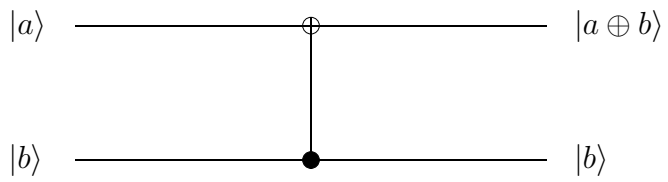
Finally, suppose that $|\psi_1\rangle = |11\rangle$. Then

$$|\psi_2\rangle = \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle - \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle,$$

$$|\psi_3\rangle = \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle - \frac{1}{2}|11\rangle + \frac{1}{2}|10\rangle = \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)$$

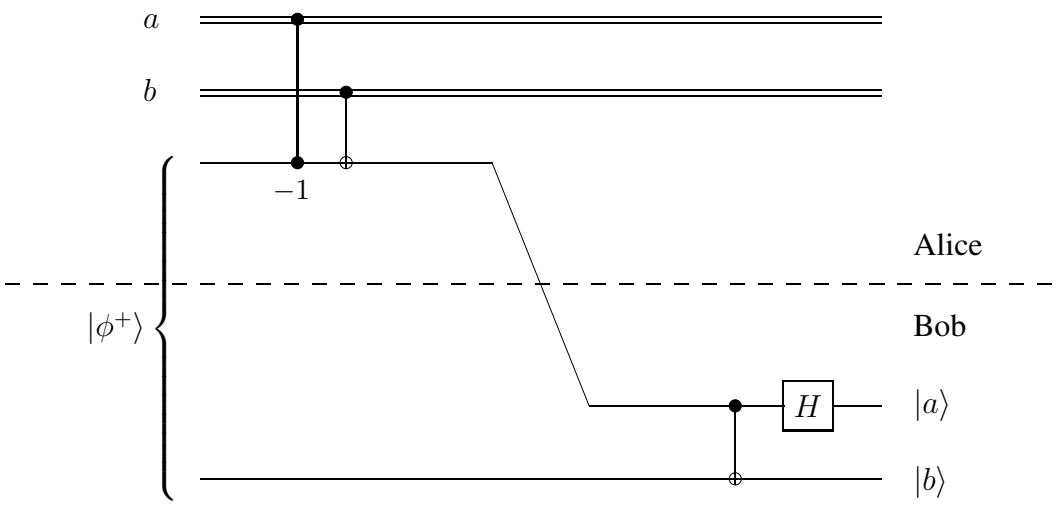
$$|\psi_4\rangle = |01\rangle.$$

It turns out that the circuit is equivalent to this gate:



This may seem counter to your intuition because the roles of control and target effectively switched in the controlled-not gate because of the Hadamard transforms. Don't let that bother you—instead take it as an example of how your intuition can be wrong.

Example 7. A quantum circuit diagram for superdense coding.



In this picture the first gate represents a controlled-σ_z gate, whose corresponding unitary matrix is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

It may seem strange that the first two gates are acting on one classical bit and one qubit. Generally we will only do this when the classical bit is a control bit for some operation. All that this means is that if the (classical) control bit is 1 then perform the corresponding operation on the qubit (σ_z or σ_x in the present case), otherwise do nothing.

Partial measurements

Suppose we have a system consisting of two or more qubits and we only measure one of them. For example, suppose the qubits are X and Y, and these qubits are in the state

$$|\psi\rangle = \frac{1}{2}|00\rangle - \frac{i}{2}|10\rangle + \frac{1}{\sqrt{2}}|11\rangle .$$

We know what the distribution of measurement outcomes would be if we measured both qubits:

$$\Pr[\text{outcome is 00}] = \frac{1}{4} \qquad \Pr[\text{outcome is 01}] = 0$$

$$\Pr[\text{outcome is 10}] = \frac{1}{4} \qquad \Pr[\text{outcome is 11}] = \frac{1}{2}.$$

The probability that a measurement of the first qubit, for instance, results in outcome 0 should therefore be $1/4+0 = 1/4$ and the probability of outcome 1 should be $1/4+1/2 = 3/4$. Intuitively, this should be the case regardless of whether or not the second qubit was actually measured. Indeed this is the case.

However, what is different between the case where the second qubit is measured and the case where it is not is the superposition of the system after the measurement (or measurements). If both qubits are measured, the superposition will be one of $|00\rangle$, $|01\rangle$, $|10\rangle$, or $|11\rangle$, depending on the measurement outcome. If only the first qubit is measured, however, this may not be the case.

I'll explain how you determine the state of the system after the measurement of the first qubit for the example above, and the method for a general case should be clear. We begin by considering the possible outcome 0 for the measurement. We consider the state

$$|\psi\rangle = \frac{1}{2}|00\rangle - \frac{i}{2}|10\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

and cross off all of the terms in the sum that are inconsistent with measuring a 0 for the first qubit:

$$|\psi\rangle = \frac{1}{2}|00\rangle - \frac{i}{2}\cancel{|10\rangle} + \frac{1}{\sqrt{2}}\cancel{|11\rangle}$$

which leaves

$$\frac{1}{2}|00\rangle .$$

The probability of measuring 0 is the norm-squared of this vector,

$$\Pr[\text{measurement outcome is 0}] = \left\| \frac{1}{2}|00\rangle \right\|^2 = \frac{1}{4},$$

and the state of the two qubits after the measurement conditioned on the measurement outcome being 0 is the vector itself *renormalized*:

$$\frac{\frac{1}{2}|00\rangle}{\left\|\frac{1}{2}|00\rangle\right\|} = |00\rangle .$$

The possible measurement outcome 1 is handled similarly. We cross off the terms inconsistent with measuring a 1:

$$\frac{1}{2}\cancel{|00\rangle} - \frac{i}{2}|10\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

which leaves

$$-\frac{i}{2}|10\rangle + \frac{1}{\sqrt{2}}|11\rangle .$$

The probability of measuring a 1 is then the norm-squared of this vector,

$$\Pr[\text{measurement outcome is 1}] = \left\| -\frac{i}{2}|10\rangle + \frac{1}{\sqrt{2}}|11\rangle \right\|^2 = \left| -\frac{i}{2} \right|^2 + \left| \frac{1}{\sqrt{2}} \right|^2 = \frac{3}{4},$$

and conditioned on measuring a 1 the state of the two qubits becomes

$$\frac{-\frac{i}{2}|10\rangle + \frac{1}{\sqrt{2}}|11\rangle}{\left\| -\frac{i}{2}|10\rangle + \frac{1}{\sqrt{2}}|11\rangle \right\|} = -\frac{i}{\sqrt{3}}|10\rangle + \sqrt{\frac{2}{3}}|11\rangle .$$

A completely equivalent way of performing this calculation is to begin by writing

$$|\psi\rangle = |0\rangle|\phi_0\rangle + |1\rangle|\phi_1\rangle$$

for some choice of $|\phi_0\rangle$ and $|\phi_1\rangle$. (There will always be a unique choice for each that works.) In this case we have

$$|\psi\rangle = |0\rangle\left(\frac{1}{2}|0\rangle\right) + |1\rangle\left(-\frac{i}{2}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right),$$

so $|\phi_0\rangle = \frac{1}{2}|0\rangle$ and $|\phi_1\rangle = -\frac{i}{2}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. The probabilities for the two measurement outcomes are as follows:

$$\Pr[\text{outcome is 0}] = \||\phi_0\rangle\|^2 = \left|\frac{1}{2}\right|^2 = \frac{1}{4}$$

$$\Pr[\text{outcome is 1}] = \||\phi_1\rangle\|^2 = \left|-\frac{i}{2}\right|^2 + \left|\frac{1}{\sqrt{2}}\right|^2 = \frac{3}{4}.$$

Conditioned on the measurement outcome, the state of the two qubits is as follows:

$$\text{measurement outcome is 0} \;\Rightarrow\; \text{state becomes } \frac{1}{\||\phi_0\rangle\|}|0\rangle|\phi_0\rangle = |00\rangle$$

$$\text{measurement outcome is 1} \;\Rightarrow\; \text{state becomes } \frac{1}{\||\phi_1\rangle\|}|1\rangle|\phi_1\rangle = \frac{-i}{\sqrt{3}}|10\rangle + \sqrt{\frac{2}{3}}|11\rangle .$$

The same method works for more than 2 qubits as well.

Example 8. Suppose 3 qubits are in the superposition

$$|\psi\rangle = \frac{1}{2}|000\rangle + \frac{1}{2}|100\rangle + \frac{1}{2}|101\rangle - \frac{1}{2}|111\rangle$$

and the **third** qubit is measured. What are the probabilities of the two possible measurement outcomes and what are the resulting superpositions of the three qubits for each case?

To determine the answer, we write

$$|\psi\rangle = \left(\frac{1}{2}|00\rangle + \frac{1}{2}|10\rangle\right)|0\rangle + \left(\frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle\right)|1\rangle .$$

The probability that the measurement outcome is 0 is

$$\left\| \frac{1}{2}|00\rangle + \frac{1}{2}|10\rangle \right\|^2 = \frac{1}{2}$$

and in this case the resulting superposition is

$$\sqrt{2}\left(\frac{1}{2}|00\rangle + \frac{1}{2}|10\rangle\right)|0\rangle = \frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|100\rangle .$$

The probability that the measurement outcome is 1 is

$$\left\| \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle \right\|^2 = \frac{1}{2}$$

and in this case the resulting superposition is

$$\sqrt{2}\left(\frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle\right)|1\rangle = \frac{1}{\sqrt{2}}|101\rangle - \frac{1}{\sqrt{2}}|111\rangle .$$

Lecture 4: Quantum Teleportation; Deutsch's Algorithm

January 26, 2006

Quantum teleportation

Suppose Alice has a qubit that she wants to send to Bob. Let us say that the state of the qubit is $\alpha|0\rangle + \beta|1\rangle$. How many classical bits would be required to accomplish this task?

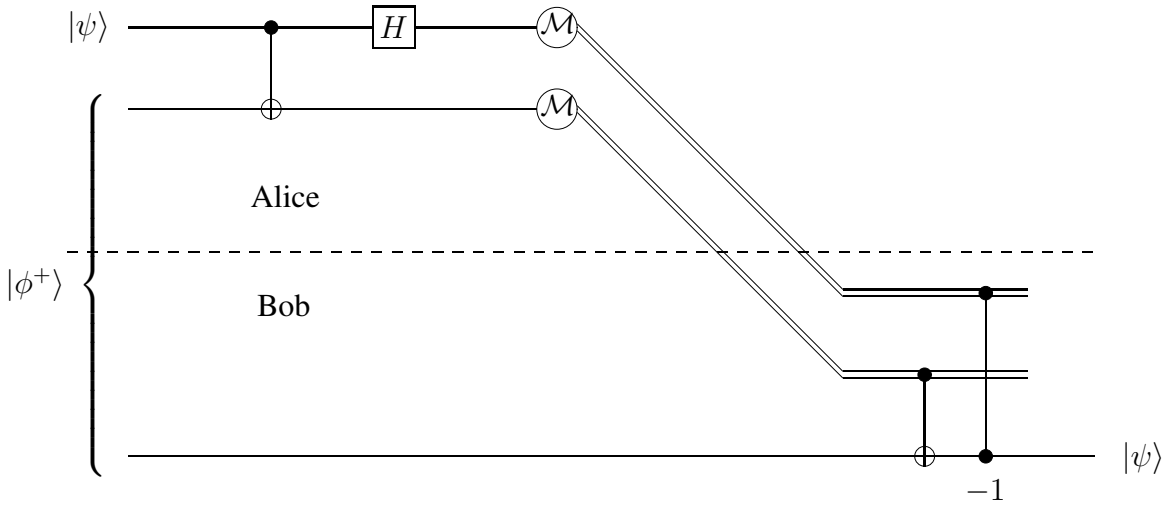
There are two reasonable answers to the question. The first answer is that the number of classical bits depends on the desired precision: α and β are arbitrary complex numbers (with $|\alpha|^2 + |\beta|^2 = 1$), but Alice could send approximations of α and β to Bob.

The second answer, which in some sense is the better answer, is that no number of classical bits of communication will suffice. Alice may not know α and β, and there is no way for her to perform measurements on her qubit that would reveal these numbers. Thus, she could not communicate this information to Bob because she could not even determine it for herself. There is also the possibility that Alice's qubit is entangled with one or more other qubits (in which case of course we would not be able to describe the state of the qubit as $\alpha|0\rangle + \beta|1\rangle$). Classical communication from Alice to Bob would not be able to somehow transmit this entanglement.

However, if we give Alice and Bob the additional resource of sharing an e-bit, just as in superdense coding, then it becomes possible for Alice to transmit a qubit to Bob using classical communication by means of *quantum teleportation*. Specifically, two bits of classical information will be needed to perform this task.

Quantum teleportation protocol

Here is the protocol described in terms of a quantum circuit diagram.



Let us assume that $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$. The starting state is

$$(\alpha\,|0\rangle + \beta\,|1\rangle)\left(\frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle\right) = \frac{1}{\sqrt{2}}\left(\alpha\,|000\rangle + \alpha\,|011\rangle + \beta\,|100\rangle + \beta\,|111\rangle\right).$$

First the CNOT gate is applied, which transforms the state to

$$\frac{1}{\sqrt{2}}\left(\alpha\,|000\rangle + \alpha\,|011\rangle + \beta\,|110\rangle + \beta\,|101\rangle\right).$$

Next, the Hadamard transform is applied, which transforms the state to

$$\begin{aligned}&\frac{1}{2}\left(\alpha\,|000\rangle + \alpha\,|100\rangle + \alpha\,|011\rangle + \alpha\,|111\rangle + \beta\,|010\rangle - \beta\,|110\rangle + \beta\,|001\rangle - \beta\,|101\rangle\right)\\ &= \frac{1}{2}\,|00\rangle\,(\alpha\,|0\rangle + \beta\,|1\rangle) + \frac{1}{2}\,|01\rangle\,(\alpha\,|1\rangle + \beta\,|0\rangle) + \frac{1}{2}\,|10\rangle\,(\alpha\,|0\rangle - \beta\,|1\rangle) + \frac{1}{2}\,|11\rangle\,(\alpha\,|1\rangle - \beta\,|0\rangle).\end{aligned}$$

Note that there have been no measurements or communication at this point. Bob's qubit *seems* to depend on α and β, but this is not really so. There would be no way for Bob to transform and measure his qubit alone at this point so that he would learn anything about α and β.

The state above has been expressed in a convenient way to determine the distribution of measurement outcomes and the resulting state of Bob's qubit after the measurements.

Case 1: Alice measures 00. This happens with probability

$$\left\|\frac{1}{2}(\alpha\,|0\rangle + \beta\,|1\rangle)\right\|^2 = \frac{1}{4}.$$

Conditioned on this outcome, the state of the three qubits becomes

$$|00\rangle\,(\alpha\,|0\rangle + \beta\,|1\rangle).$$

Alice transmits the classical bits 00 to Bob. Because both bits are zero, he does not perform either of the two possible operations, and so his qubit remains in the state $\alpha\,|0\rangle + \beta\,|1\rangle$ at the end of the protocol.

Case 2: Alice measures 01. This happens with probability

$$\left\|\frac{1}{2}(\alpha\,|1\rangle + \beta\,|0\rangle)\right\|^2 = \frac{1}{4}.$$

Conditioned on this outcome, the state of the three qubits becomes

$$|01\rangle\,(\alpha\,|1\rangle + \beta\,|0\rangle).$$

Alice transmits the classical bits 01 to Bob. Because the first transmitted bit is 0 and the second is 1, Bob performs a NOT operation on his qubit. Thus, the state of his qubit becomes $\alpha\,|0\rangle + \beta\,|1\rangle$.

Case 3: Alice measures 10. This happens with probability

$$\left\| \frac{1}{2}(\alpha \left|0\right\rangle - \beta \left|1\right\rangle) \right\|^2 = \frac{1}{4}.$$

Conditioned on this outcome, the state of the three qubits becomes

$$\left|10\right\rangle (\alpha \left|0\right\rangle - \beta \left|1\right\rangle).$$

Alice transmits the classical bits 10 to Bob. Because the first transmitted bit is 1 and the second is 0, Bob performs a σ_z operation on his qubit. Thus, the state of his qubit becomes $\alpha \left|0\right\rangle + \beta \left|1\right\rangle$.

Case 4: Alice measures 11. This happens with probability

$$\left\| \frac{1}{2}(\alpha \left|1\right\rangle - \beta \left|0\right\rangle) \right\|^2 = \frac{1}{4}.$$

Conditioned on this outcome, the state of the three qubits becomes

$$\left|11\right\rangle (\alpha \left|1\right\rangle - \beta \left|0\right\rangle).$$

Alice transmits the classical bits 11 to Bob. Because both transmitted bits are 1, Bob first performs a NOT operation on his qubit, transforming it to $\alpha \left|0\right\rangle - \beta \left|1\right\rangle$, and then performs a σ_z gate to it, transforming it to the state $\alpha \left|0\right\rangle + \beta \left|1\right\rangle$.

Thus, we see that in all four cases, Bob's qubit is in the state $\alpha \left|0\right\rangle + \beta \left|1\right\rangle$ at the end of the protocol.

Something stronger is actually true. If Alice's initial qubit was entangled with other qubits, this entanglement will be preserved. In other words, teleportation works like a perfect quantum channel—it is exactly as if Alice had physically sent her qubit to Bob.

Deutsch's Algorithm

We have seen two interesting protocols that can be performed using quantum information: superdense coding and teleportation. Next, we will discuss various quantum algorithms, starting with a very simple one: Deutsch's Algorithm. This is a very different setting in which some advantage is gained by using quantum information over classical information.

Suppose that we have a device that computes some function $f : \{0, 1\} \to \{0, 1\}$. It is useful for the purposes of the present investigation to think of this device as a *black box*. This means that we cannot look inside the device to see how it works—the only way to gain information about the function f computed by the device is to give some input $a \in \{0, 1\}$ and allow the device to output $f(a) \in \{0, 1\}$. There are four possible functions from $\{0, 1\}$ to $\{0, 1\}$; let us call them f_0, f_1, f_2, and f_3.

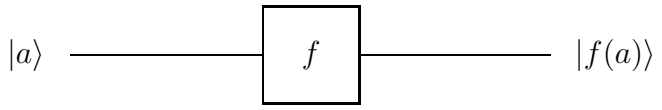
		f_0	f_1	f_2	f_3
Input	0	0	0	1	1
	1	0	1	0	1

Suppose that we are interested in determining which of the two possible alternatives holds:

1. f is **constant**, or
2. f is **balanced** (meaning each output appears the same number of times).

In particular, f_0 and f_3 are constant and f_1 and f_2 are balanced. Obviously two evaluations of the function are necessary and sufficient to answer the question. If only one evaluation is permitted, the function could still be either constant or balanced regardless of the input and output obtained.

Now let us consider the same question in the context of quantum information. We need to change the question slightly, however, in order for it to fit into the model of quantum information that we are considering. The change is that we need the black box to conform to a valid quantum operation. More specifically, we must insist that the action of the device corresponds to a unitary transformation. It is therefore **not** sufficient to consider the black box to be a one-qubit gate acting as follows:

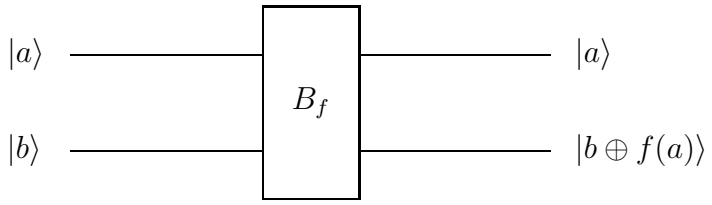


For example, if $f = f_0$ then this gate would correspond to the matrix

$$\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix},$$

which is not unitary.

Instead, for any function $f : \{0,1\} \to \{0,1\}$ we define a 2-qubit quantum gate B_f as follows:



It can be verified that the corresponding matrix is unitary for any function f. For example, if $f = f_2$ from the table on the previous page, then $f(0) = 1$ and $f(1) = 0$, so the corresponding matrix is

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

For any function f, the matrix corresponding to B_f will always be a *permutation matrix*, meaning that all of the entries are 0 or 1 and every row and every column has exactly one 1 in it. Permutation matrices are always unitary.

In general, if

$$f : \{0,1\}^n \to \{0,1\}^m$$

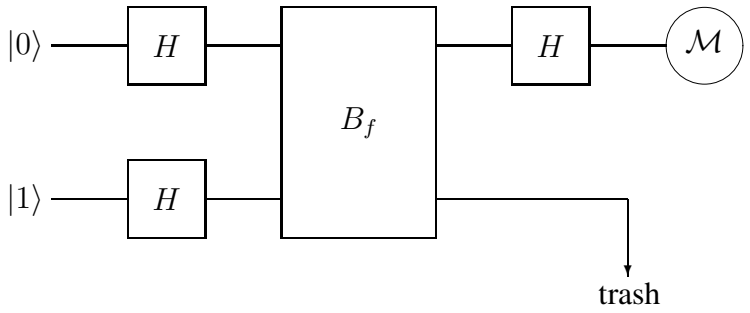
is any function (for any positive integers n and m), the associated quantum transformation B_f will be defined by

$$B_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$$

(where $\oplus$ denotes the bitwise exclusive OR). The associated matrix will always be a permutation matrix, and is therefore unitary.

Getting back to the problem at hand, let us suppose that we have access to the transformation B_f for some $f : \{0,1\} \to \{0,1\}$, and the goal is the same as before: determine whether f is constant or balanced. Note that classically having access to B_f is no more helpful than access to f (both as black boxes)—two evaluations of B_f are necessary and sufficient to answer the question.

Using a quantum algorithm, however, it is only necessary to use one application of B_f to solve the problem. Here is a quantum circuit diagram explaining the procedure:



This procedure is known as *Deutsch's Algorithm*, and the problem of determining whether a one-bit function is constant or balanced is sometimes called *Deutsch's Problem*.

The output of the measurement is a single bit, and the interpretation is that the value 0 indicates that the function was constant and 1 indicates balanced. The qubit labeled trash is inconsequential at the end of the procedure. It is labeled trash only to highlight the (seemingly unusual) fact that it is not necessary to measure or do anything to it in order to determine the answer.

Let us analyze the algorithm to determine that it works correctly. Rather than taking the four cases separately, let us try to be more sophisticated and deal with the four cases all at the same time. The initial state is $|0\rangle |1\rangle$, and so the state after the first two Hadamard transforms is

$$\left(\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle\right) \left(\frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle\right).$$

We can partially expand this state as

$$\frac{1}{2} |0\rangle (|0\rangle - |1\rangle) + \frac{1}{2} |1\rangle (|0\rangle - |1\rangle).$$

Performing the B_f operation transforms this state to

$$\begin{aligned}
&\frac{1}{2}|0\rangle\left(|0\oplus f(0)\rangle - |1\oplus f(0)\rangle\right) + \frac{1}{2}|1\rangle\left(|0\oplus f(1)\rangle - |1\oplus f(1)\rangle\right) \\
&\qquad = \frac{1}{2}(-1)^{f(0)}|0\rangle\left(|0\rangle - |1\rangle\right) + \frac{1}{2}(-1)^{f(1)}|1\rangle\left(|0\rangle - |1\rangle\right) \\
&\qquad = \left(\frac{1}{\sqrt{2}}(-1)^{f(0)}|0\rangle + \frac{1}{\sqrt{2}}(-1)^{f(1)}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right).
\end{aligned}$$

Here we have used the fact that

$$|0\oplus a\rangle - |1\oplus a\rangle = (-1)^a\left(|0\rangle - |1\rangle\right)$$

for $a \in \{0,1\}$.

Notice that something seemingly strange has happened. The B_f transformation has apparently not changed the state of the second qubit; it has remained in the state

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

There has been an important effect, however, which is that the factors $(-1)^{f(0)}$ and $(-1)^{f(1)}$ have appeared in the state of the first qubit. This phenomenon is sometimes known as "phase kick-back". It is a commonly used trick in quantum algorithms.

At this point the second qubit is thrown in the trash, which should not bother us because we know that its state is completely independent from the state of the first qubit; we know that its state is $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$, regardless of the choice of f. This leaves the first qubit in the state

$$\frac{1}{\sqrt{2}}(-1)^{f(0)}|0\rangle + \frac{1}{\sqrt{2}}(-1)^{f(1)}|1\rangle,$$

which we can write as

$$(-1)^{f(0)}\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}(-1)^{f(0)\oplus f(1)}|1\rangle\right).$$

The final Hadamard transform takes this state to

$$(-1)^{f(0)}|f(0)\oplus f(1)\rangle.$$

Here we have used the observation that

$$H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}(-1)^a|1\rangle\right) = |a\rangle$$

for $a \in \{0,1\}$, which is again easily verified by considering the cases $a = 0$ and $a = 1$. The measurement therefore results in the value $f(0)\oplus f(1)$ with certainty. This value is 0 if f is constant and 1 if f is balanced.

Lecture 5: A simple searching algorithm; the Deutsch-Jozsa algorithm

January 31, 2006

In the previous lecture we discussed Deutsch's Algorithm, which gives a simple example of how quantum algorithms can give some advantages over classical algorithms in certain restricted settings. We will start this lecture by discussing another simple example, and then move on to the Deutsch-Jozsa Algorithm, which generalizes Deutsch's Algorithm to functions from n bits to 1 bit.

A simple searching problem

Suppose that we are given a function of the form $f : \{0,1\}^2 \to \{0,1\}$, but this time we are *promised* that it is one of these four functions:

f_{00}

input	output
00	1
01	0
10	0
11	0

f_{01}

input	output
00	0
01	1
10	0
11	0

f_{10}

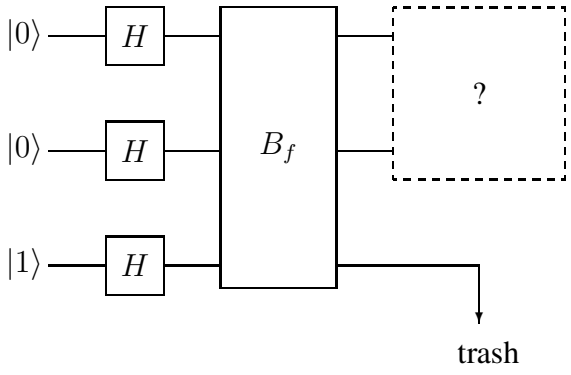
input	output
00	0
01	0
10	1
11	0

f_{11}

input	output
00	0
01	0
10	0
11	1

The goal is to determine which one it is. In other words, the function takes value 1 on one input and value 0 on all others, and the goal is to find the input on which the function takes value 1. Thus, we can think of this problem as representing a simple searching problem. As before, we assume that access to the function is restricted to evaluations of the transformation B_f. Now is a good time to mention some terminology: we say that an evaluation of B_f on some input (quantum or classical) is a *query*.

Classically, three queries are necessary and sufficient to solve the problem. In the quantum case, one query will be sufficient to solve the problem. Here is a circuit diagram describing part of the procedure. (The missing part will be filled in later.)



Let us analyze the state of the above circuit at various instants to try and determine what (if anything) would work for the omitted part of the circuit. The state after the Hadamard transforms can be written

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

After the transformation B_f is performed, I claim that the state is

$$\left(\frac{1}{2}(-1)^{f(00)}|00\rangle + \frac{1}{2}(-1)^{f(01)}|01\rangle + \frac{1}{2}(-1)^{f(10)}|10\rangle + \frac{1}{2}(-1)^{f(11)}|11\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right).$$

The reasoning is the same as in the analysis of Deutsch's Algorithm—we are again using the *phase kickback* effect.

The last qubit is independent of the first two, so when it goes in the trash we are left with one of these four possibilities for the state of the first two qubits:

$$\begin{aligned}
f = f_{00} &\Rightarrow |\phi_{00}\rangle = -\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle \\
f = f_{01} &\Rightarrow |\phi_{01}\rangle = \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle \\
f = f_{10} &\Rightarrow |\phi_{10}\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle - \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle \\
f = f_{11} &\Rightarrow |\phi_{11}\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle
\end{aligned}$$

There is something special about these four states: they form an *orthonormal set*. This means that they are unit vectors and are *pairwise orthogonal*:

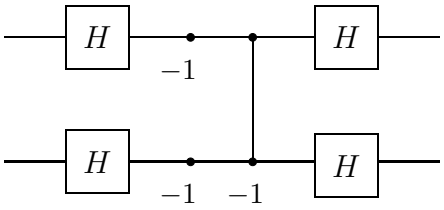
$$\langle\phi_{ab}|\phi_{cd}\rangle = \begin{cases} 1 & \text{if } a = c \text{ and } b = d \\ 0 & \text{otherwise.} \end{cases}$$

Whenever you have an orthonormal set like this, it is always possible to build a quantum circuit that exactly distinguishes the states. In fact, the corresponding unitary transformation is easy to obtain: you let the vectors form the columns of a matrix and then take the conjugate transform. In this case we want this matrix:

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

You can check that $U|\phi_{ab}\rangle = |ab\rangle$ for all four choices of $a, b \in \{0, 1\}$.

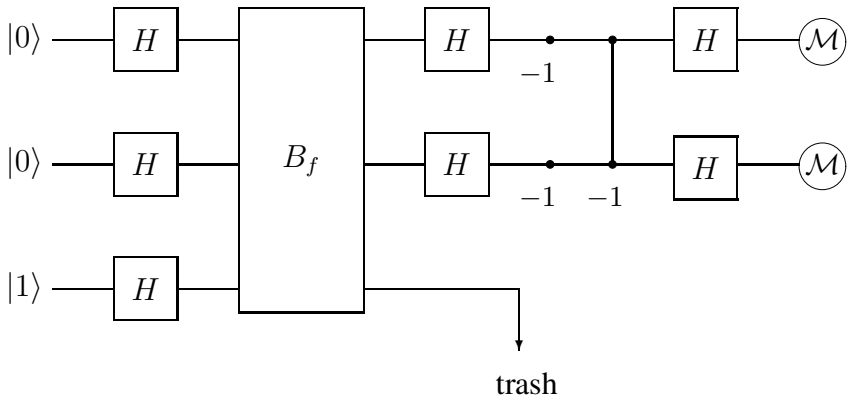
In case you are interested, it is possible to build a small circuit that uses gates that we have already seen to perform the unitary transformation U:



The gate consisting of a single circle labeled -1 corresponds to the matrix

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

which is also called a *phase flip* or σ_z gate. The complete circuit looks like this:



The Deutsch-Jozsa Algorithm

Our next algorithm is a generalization of Deutsch's Algorithm called the Deutsch-Jozsa Algorithm. Although the computational problem being solved is still artificial and perhaps not particularly impressive, we are moving toward more interesting problems.

This time we assume that we are given a function $f : \{0,1\}^n \to \{0,1\}$, where n is some arbitrary positive integer, and we are promised that one of two possibilities holds:

1. f is **constant**. In other words, either $f(x) = 0$ for all $x \in \{0,1\}^n$ or $f(x) = 1$ for all $x \in \{0,1\}^n$.
2. f is **balanced**. This means that the number of inputs $x \in \{0,1\}^n$ for which the function takes values 0 and 1 are the same:

$$|\{x \in \{0,1\}^n : f(x) = 0\}| = |\{x \in \{0,1\}^n : f(x) = 1\}| = 2^{n-1}.$$

The goal is to determine which of the two possibilities holds.

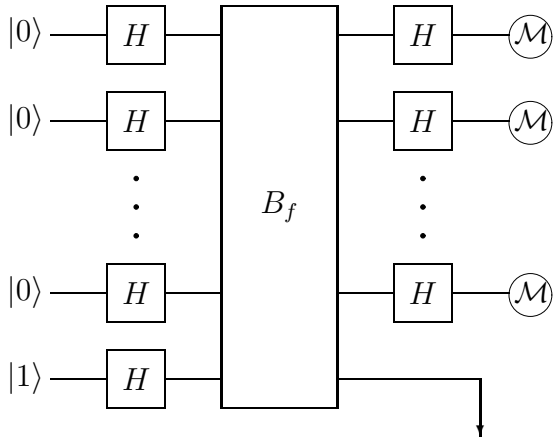
As for the previous two algorithms, we assume that access to the function f is restricted to queries to a device corresponding to the transformation B_f defined similarly to before:

$$B_f |x\rangle |b\rangle = |x\rangle |b \oplus f(x)\rangle$$

for all $x \in \{0,1\}^n$ and $b \in \{0,1\}$.

It turns out that classically this problem is pretty easy given a small number of queries if we allow randomness and accept that there may be a small probability of error. Specifically, we can randomly choose say k inputs $x_1, \ldots, x_k \in \{0,1\}^n$, evaluate $f(x_i)$ for $i = 1, \ldots, k$, and answer "constant" if $f(x_1) = \cdots = f(x_k)$ and "balanced" otherwise. If the function really was constant this method will be correct every time, and if the function was balanced, the algorithm will be wrong (and answer "constant") with probability $2^{-(k-1)}$. Taking $k = 11$, say, we get that the probability of error is smaller than $1/1000$. However, if you demand that the algorithm is correct every time, then $2^{n-1} + 1$ queries are needed in the worst case.

In the quantum case, 1 query will be sufficient to determine with certainty whether the function is constant or balanced. Here is the algorithm, which is called the *Deutsch-Jozsa Algorithm*:



There are n bits resulting from the measurements. If all n measurement results are 0, we conclude that the function was constant. Otherwise, if at least one of the measurement outcomes is 1, we conclude that the function was balanced.

Before we analyze the algorithm, it will be helpful to think more about Hadamard transforms. We have already observed that for $a \in \{0,1\}$ we have

$$H |a\rangle = \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} (-1)^a |1\rangle ,$$

which we can also write as

$$H |a\rangle = \frac{1}{\sqrt{2}} \sum_{b \in \{0,1\}} (-1)^{ab} |b\rangle .$$

If we instead had two qubits, starting in state $|x\rangle$ for $x = x_1x_2 \in \{0,1\}^2$, and applied Hadamard transforms to both, we would obtain

$$\begin{aligned}(H \otimes H)|x\rangle &= \left(\frac{1}{\sqrt{2}} \sum_{y_1 \in \{0,1\}} (-1)^{x_1 y_1} |y_1\rangle\right) \left(\frac{1}{\sqrt{2}} \sum_{y_2 \in \{0,1\}} (-1)^{x_2 y_2} |y_2\rangle\right) \\ &= \frac{1}{2} \sum_{y \in \{0,1\}^2} (-1)^{x_1 y_1 + x_2 y_2} |y\rangle .\end{aligned}$$

The pattern generalizes to any number of qubits. Writing $H^{\otimes n}$ to mean $H \otimes \cdots \otimes H$ (n times), we have

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x_1 y_1 + \cdots + x_n y_n} |y\rangle$$

for every $x \in \{0,1\}^n$. We also use the shorthand

$$x \cdot y = \sum_{i=1}^{n} x_i y_i \pmod 2$$

so that we may write

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle .$$

The fact that $x \cdot y$ is defined modulo 2 is irrelevant for the previous expression (because $x \cdot y$ appears as an exponent of -1), but it will be convenient later to use this shorthand again and have the quantity defined modulo 2.

Using these formulas, the Deutsch-Jozsa Algorithm becomes easier to analyze. The state after the first layer of Hadamard transforms is performed is

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right).$$

When the B_f transformation is performed, the state will change to

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right).$$

Once again we are seeing the *phase kick-back* effect. Now, the last qubit is discarded and n Hadamard transforms are applied. The resulting state is

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} \left(\frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle\right),$$

which simplifies to

$$\sum_{y\in\{0,1\}^n} \left(\frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)+x\cdot y} \right) |y\rangle \,.$$

Now, all that we really care about is the probability that the measurements all give outcome 0. The amplitude associated with the classical state $|0^n\rangle$ is

$$\frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}.$$

Thus, the probability that the measurements all give outcome 0 is

$$\left| \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} \right|^2 = \begin{cases} 1 & \text{if } f \text{ is constant} \\ 0 & \text{if } f \text{ is balanced,} \end{cases}$$

so the algorithm works as claimed.

Lecture 6: Simon's algorithm

February 2, 2006

The reason why we looked at the particular "black-box" problems from the previous couple of lectures is because they give examples in which quantum algorithms give some advantage over classical algorithms. These problems seem rather unnatural, and it is difficult to imagine any realistic setting in which one would need or want to solve these problems. But perhaps more importantly, so far they have not demonstrated any significant advantage of quantum algorithms over probabilistic algorithms.

The next problem we will study will still be in the category of completely artificial problems, but now the advantage of quantum over probabilistic is quite significant: a linear number of queries will be needed by the quantum algorithm whereas an exponential number will be needed for any classical probabilistic algorithm. Also, although the problem is artificial, it is surprisingly close to a very natural and important real-world problem as we will see.

Simon's problem

The input to the problem is a function of the form

$$f : \{0,1\}^n \to \{0,1\}^n$$

for a positive integer n. Access to this function is restricted to queries to the black-box transformation B_f as before. Similar to the Deutsch-Jozsa problem, the function f is *promised* to obey a certain property, although the property is slightly more complicated than before. The property is that there exists a string $s \in \{0,1\}^n$ such that

$$[f(x) = f(y)] \Leftrightarrow [x \oplus y \in \{0^n, s\}]$$

for all $x, y \in \{0,1\}^n$. The goal of the problem is to find the string s.

For example, if $n = 3$, then the following function is an example of a function that satisfies the required property:

x	$f(x)$
000	101
001	010
010	000
011	110
100	000
101	110
110	101
111	010

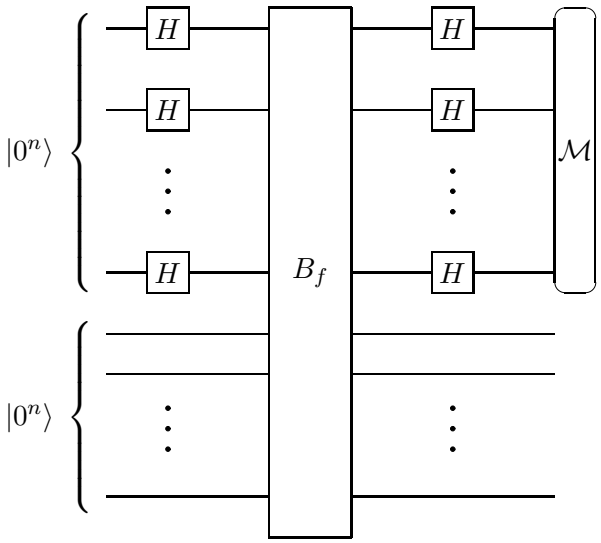
Specifically, the string s is 110. Every output of f occurs twice, and the two input strings corresponding to any one given output have bitwise XOR equal to $s = 110$.

Note that the possibility that $s = 0^n$ is not ruled out—in this case the function f is simply required to be a one-to-one function.

Intuitively this is a very hard problem classically, even if one uses randomness and accepts a small probability of error. We will not go through the proof of this fact, because (like many lower-bound proofs) it is harder than one might expect and I prefer that our focus remain on the quantum algorithmic aspects of this problem. However, the intuition is reasonably simple: if you want to solve the problem classically you need to find two different inputs x and y for which $f(x) = f(y)$. There is not necessarily any structure in the function f that would help you to find two such inputs—without any additional constraints on f you are as likely to find two such inputs by randomly guessing them as you would using any other method. But you would need to guess $\Omega(\sqrt{2^n})$ different inputs before being likely to find a pair on which f takes the same output.

Simon's algorithm

A quantum algorithm (called Simon's Algorithm) for solving this task consists of iterating the following quantum circuit and doing some classical post-processing:



Note that the B_f gate has a different (but very similar) form from before because the input and output of the function f are n-bit strings:

$$B_f |x\rangle |y\rangle = |x\rangle |f(x) \oplus y\rangle$$

where $\oplus$ denotes the bitwise XOR.

Before describing the classical post-processing, let us try to determine some properties of the measurement outcome in the above circuit. The circuit begins in state $|0^n\rangle |0^n\rangle$, and Hadamard

transforms are performed on the first n qubits. This produces the state

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \, |0^n\rangle \, .$$

The state after the B_f transformation is performed is

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \, |f(x)\rangle \, .$$

This time there is no phase kick-back as there was in the previous algorithms we studied. Finally, n Hadamard transforms are applied, which results in state

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} \, |y\rangle \, |f(x)\rangle \, .$$

Here, we have used the formula

$$H^{\otimes n} \, |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} \, |y\rangle$$

to obtain this expression.

Now, we are interested in the probability with which each string results from the measurement. Let us first consider the special case where $s = 0^n$, which means that f is a one-to-one function. We can write the state from above as

$$\sum_{y \in \{0,1\}^n} |y\rangle \left(\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} \, |f(x)\rangle \right) ,$$

so the probability that the measurement results in each string y is

$$\left\| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} \, |f(x)\rangle \right\|^2 = \frac{1}{2^n}.$$

One way to see that the previous equation is true is to note that

$$\left\| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} \, |f(x)\rangle \right\|^2 = \left\| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} \, |x\rangle \right\|^2$$

because the two vectors only differ in the ordering of their entries (as f is one-to-one). The value of the right-hand-side is more easily seen to be 2^{-n}. Thus, the outcome is simply a uniformly distributed n bit string when $s = 0^n$.

The second, more interesting, case is where $s \neq 0^n$. The analysis from before still works to imply that the probability to measure any given string y is

$$\left\| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} \, |f(x)\rangle \right\|^2 .$$

Let $A = \mathrm{range}(f)$. If $z \in A$, there must exist 2 distinct strings $x_z, x_z' \in \{0,1\}^n$ such that

$$f(x_z) = f(x_z') = z,$$

and moreover it is necessary that $x_z \oplus x_z' = s$ (which is equivalent to $x_z' = x_z \oplus s$). Now,

$$\begin{aligned}
\left\| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} \, |f(x)\rangle \right\|^2 &= \left\| \frac{1}{2^n} \sum_{z \in A} \left((-1)^{x_z \cdot y} + (-1)^{x_z' \cdot y} \right) |z\rangle \right\|^2 \\
&= \left\| \frac{1}{2^n} \sum_{z \in A} \left((-1)^{x_z \cdot y} + (-1)^{(x_z \oplus s) \cdot y} \right) |z\rangle \right\|^2 \\
&= \left\| \frac{1}{2^n} \sum_{z \in A} (-1)^{x_z \cdot y} \left(1 + (-1)^{s \cdot y} \right) |z\rangle \right\|^2 \\
&= \begin{cases} 2^{-(n-1)} & \text{if } s \cdot y = 0 \\ \\ 0 & \text{if } s \cdot y = 1. \end{cases}
\end{aligned}$$

In this calculation we have used the fact that

$$(x_z \oplus s) \cdot y = (x_z \cdot y) \oplus (s \cdot y)$$

which you can verify for yourself.

So, the measurement always results in some string y that satisfies $s \cdot y = 0$, and the distribution is uniform over all of the strings that satisfy this constraint. Is this enough information to determine s? The answer is yes, as we will discuss next.

Classical post-processing

When we run the circuit above, there are two cases: in the special case where $s = 0^n$, the measurement results in each string $y \in \{0,1\}^n$ with probability

$$p_y = \frac{1}{2^n};$$

otherwise, in the case that $s \neq 0^n$ the probability to obtain each string y is

$$p_y = \begin{cases} 2^{-(n-1)} & \text{if } s \cdot y = 0 \\ \\ 0 & \text{if } s \cdot y = 1. \end{cases}$$

Thus, in both cases the measurement results in some string y that satisfies $s \cdot y = 0$, and the distribution is uniform over all of the strings that satisfy this constraint.

Is this enough information to determine s? The answer is yes, provided that you repeat the process several times and accept a small probability of failure. Specifically, if the above process is run $n-1$ times, you will get $n-1$ strings $y_1, \ldots, y_{n-1}$ such that

$$\begin{aligned} y_1 \cdot s &= 0 \\ y_2 \cdot s &= 0 \\ &\vdots \\ y_{n-1} \cdot s &= 0. \end{aligned}$$

This is a system if $n-1$ linear equations in n unknowns (the bits of s), and the goal is to solve to obtain s. All of the operations are modulo 2 operations, so it is not exactly the type of system of linear equations you were used to in linear algebra, but the system can be efficiently solved using similar methods. You will only get a unique non-zero solution s if you are lucky and $y_1, \ldots, y_{n-1}$ are linearly independent. The probability that $y_1, \ldots, y_{n-1}$ are linearly independent is at least

$$\prod_{k=1}^{\infty} \left(1 - \frac{1}{2^k}\right) = 0.288788\cdots > \frac{1}{4}.$$

If you have linear independence, solve the system to get a candidate solution $s' \neq 0^n$, and test that $f(0^n) = f(s')$. If $f(0^n) = f(s')$, you know that $s = s'$ and the problem has been solved. If $f(0^n) \neq f(s')$, it must be the case that $s = 0^n$ (because if this were not so, the unique nonzero solution to the linear equations would have been s). Either way, once you have linear independence, you can solve the problem.

Now, repeat the entire process $4m$ times. The probability of not finding a linearly independent set during one of the iterations is less than

$$\left(1 - \frac{1}{4}\right)^{4m} < e^{-m}.$$

For example, if $m = 10$ this probability is less than $1/20000$.

The upshot is that for any constant $\varepsilon > 0$, Simon's algorithm can solve the problem we are considering with error probability at most ε using $O(n)$ queries to the black-box.

Lecture 7: Arithmetic/number-theoretic problems; reversible computation

February 9, 2006

Arithmetic/number-theoretic problems

Before we move on to Shor's Algorithm, we need to understand more about the topic with which it is concerned: computational number theory.

Let us begin with an example—suppose we are given two integers x and y, represented as strings of bits using binary notation[3], and our goal is to determine $x + y$. A specification of this problem is as follows:

Integer Addition

Input: $x, y \in \mathbb{Z}$.

Output: $x + y$.

One can formalize the notion of an *efficient* algorithm for a given problem in various ways by describing precise models of computation. For the purposes of this discussion let us just think in terms of Boolean circuits and identify an algorithm with a collection of acyclic circuits (one for each possible input length) that produce the correct outputs for all possible input strings. The circuits should be composed of AND, OR, and NOT gates, along with FANOUT operations (which we often would not consider as gates at all), and in order to be considered efficient the total number of gates in each circuit should be *polynomial* in the number of input bits.

The number of bits needed to write down the number x in binary notation is

$$\lg(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } x = 0 \\ \lfloor \log_2 |x| \rfloor + 2 & \text{if } x \neq 0, \end{cases}$$

assuming we keep a sign bit for each non-zero integer, and so the number of input bits of the above problem is $\lg(x) + \lg(y)$. Now, if you consider the usual method for adding numbers that you probably learned in elementary school or before, but using base 2 instead of base 10, you could turn that method into a construction of Boolean circuits of size $O(\lg x + \lg y)$ for solving this problem. This is in fact *linear* in the input size, and so this algorithm will be considered to be very efficient.

By the way, I should mention that it is typical to place additional constraints on collections of Boolean circuits in order for the circuits to be identified with efficient algorithms. In particular, the circuits should satisfy certain *uniformity constraints*, which means that not only do they have polynomial size, but moreover there should exist an efficient construction of the circuits themselves. This issue can safely be ignored in this class, however.

[3] We will take it as implicit from now on that integers are represented in binary notation in computational problems.

Summing up, we would simply say that the Boolean circuit complexity, or "bit complexity" of computing $x + y$ for integers x and y is $O(\lg x + \lg y)$. Or, even more simply, the bit complexity of adding two n bit integers is $O(n)$.

Here are some other arithmetic and number theoretic problems:

Integer Multiplication

Input: $x, y \in \mathbb{Z}$.

Output: xy.

The "elementary school" multiplication algorithm establishes that the bit complexity of multiplying x and y is $O((\lg x)(\lg y))$, or more simply that the bit complexity of multiplying two n bit numbers is $O(n^2)$. In fact, there are asymptotically better methods, such as the Schönhage-Strassen Algorithm that has bit complexity $O(n \log n \log\log n)$. (Large constant factors in the running time make this particular method less efficient than other methods until n becomes quite large—several thousand bits perhaps.)

Integer Division

Input: Integers x and $y \neq 0$.

Output: Integers a and b, with $0 \leq b < |y|$ such that $x = ay + b$.

The cost is roughly the same as integer multiplication—the "elementary school" division method gives an algorithm with bit complexity $O((\lg x)(\lg y))$. (In fact it is actually somewhat better: $O((\lg a)(\lg y))$.)

Greatest Common Divisor

Input: Nonnegative integers x and y.

Output: $\gcd(x, y)$.

Euclid's Algorithm, which is over 2,000 years old, computes $\gcd(x, y)$ in $O((\lg x)(\lg y))$ bit operations.

Modular Integer Addition

Input: A positive integer N and integers $x, y \in \mathbb{Z}_N \stackrel{\text{def}}{=} \{0, \ldots, N-1\}$.

Output: $x + y \pmod N$.

The bit complexity of this problem is $O(\lg N)$.

Modular Integer Multiplication

Input: A positive integer N and integers $x, y \in \mathbb{Z}_N$.

Output: $xy \pmod N$.

Here the bit complexity is $O((\lg N)^2)$ by the elementary school algorithm, and again asymptoti-

cally faster methods exist.

Modular Inverse

Input: A positive integer N and an integer $x \in \mathbb{Z}_N^* \stackrel{\text{def}}{=} \{a \in \mathbb{Z}_N \,:\, \gcd(a, N) = 1\}$.

Output: $y \in \mathbb{Z}_N$ such that $xy \equiv 1 \pmod{N}$.

The bit complexity is $O((\lg N)^2)$.

Modular Exponentiation

Input: A positive integer N, an integer $x \in \{0, \ldots, N-1\}$, and any integer k.

Output: $x^k \pmod{N}$.

Using the method of repeated squaring, the bit complexity is $O((\lg k)(\lg N)^2)$.

Primality Testing

Input: A positive integer N.

Output: "prime" if N is a prime number, "not prime" otherwise.

Randomized algorithms having bit complexity $O((\lg N)^3)$ that allow a probability $1/4$, say, of making an error have been known for many years. In a breakthrough a few years ago, Agrawal, Kayal and Saxena gave a deterministic prime testing algorithm that gives (the last I heard) a $O((\lg N)^6)$ deterministic algorithm for testing primality.

Integer Factoring

Input: A positive integer N.

Output: a prime factorization $N = p_1^{a_1} \cdots p_k^{a_k}$ of N.

No classical algorithm is known to give a polynomial bit complexity for factoring—the best known algorithm asymptotically gives a bit complexity of

$$2^{O\left((\log N)^{1/3}(\log\log N)^{2/3}\right)}.$$

We will see that when we turn to quantum algorithms, Shor's algorithm will solve the Integer Factoring problem using $O((\lg N)^3)$ operations, giving a remarkable speed-up over known classical algorithms for this task.

Reversible computation

In order to implement Shor's algorithm, it is necessary to efficiently perform some of the arithmetic operations described above (modular exponentiation, in particular) with a quantum computer. Intuitively this seems like it should not be a problem, given that there is an efficient way to implement these operations classically. But does an efficient classical implementation necessarily imply the

existence of an efficient quantum implementation? The fact that classical algorithms can use non-unitary operations such as ANDs and ORs, while quantum computers are restricted to unitary operations, makes this question non-trivial to answer.

Before we address the question of whether efficient classical algorithms for arithmetic problems (or any other type of computational problem) imply efficient quantum algorithms for these problems, we need to briefly discuss the notion of quantum circuit complexity. What does it mean to have an efficient quantum algorithm for some task?

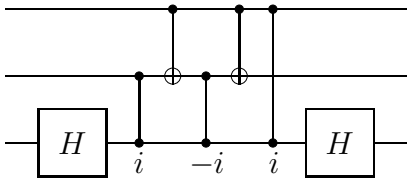
Our answer to this question will be similar to the classical case. In order to completely specify a quantum algorithm, it is necessary to give a construction that produces a quantum circuit solving the problem at hand for any given input size. Also, as in the classical case, one assumes the quantum circuit is composed entirely of quantum gates acting on some small number of qubits—in analogy to the classical case, one may require that all gates in the quantum circuit act on just one or two qubits. Again, to be considered efficient, one usually requires that the number of gates is polynomially related to the number of input qubits.

There is, however, a crucial difference between the classical and quantum cases. Whereas there are a finite number of classical one and two bit operations, and it is well known that AND and NOT operations (for instance) are sufficient to generate any such operation, there are infinitely many one and two qubit unitary operations. Although there are interesting technical aspects of this issue, such as the degree to which a finite number of quantum gates can approximate arbitrary gates, we will essentially sweep the issue under the rug by saying that arbitrary one and two qubit gates are allowed in our circuits.

By the way, it will be very helpful to use Toffoli gates to construct quantum circuits for various tasks. Recall that a Toffoli gate performs the transformation

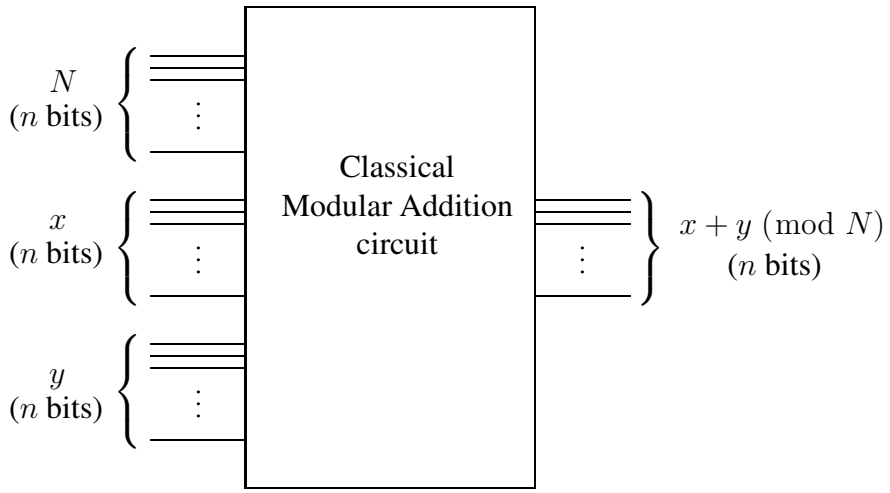
$$T|a\rangle|b\rangle|c\rangle = |a\rangle|b\rangle|c \oplus (a \wedge b)\rangle$$

for $a, b, c \in \{0, 1\}$. A Toffoli gate can be decomposed into one- and two-qubit quantum gates as follows:



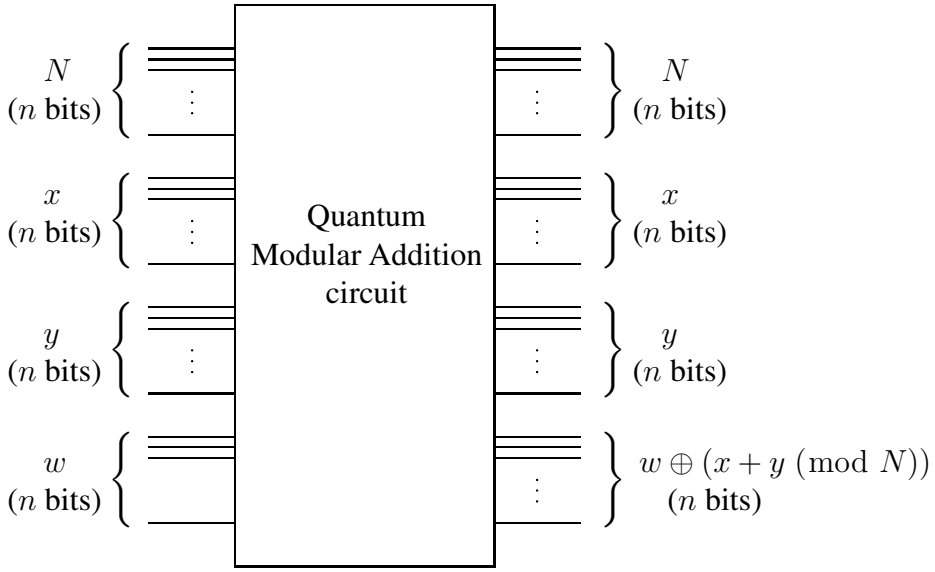
A careful check of the 8 standard basis states shows that this circuit implements a Toffoli gate as claimed.

Now let us return to the question of whether an arbitrary efficient classical algorithm can be converted to an efficient quantum algorithm. Suppose that we have some classical Boolean circuit. For the sake of example, suppose it performs modular addition as suggested in the following diagram.



Obviously such a circuit is non-unitary—the number of input and output qubits do not even agree. If we are to somehow transform this circuit into a valid quantum circuit, it will have to correspond to a unitary transformation.

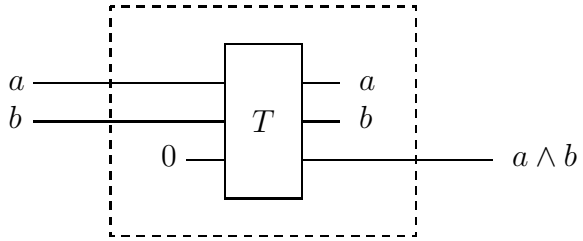
Based on our previous discussions concerning black-box transformations, we might hope to implement this operation unitarily as follows:



This is a special type of unitary transformation, because it always maps classical states to classical states, therefore giving rise to a permutation matrix. We will show how to implement such transformations using only gates that map classical states to classical states, so none of what we are discussing right now needs to be thought of as inherently quantum. Usually when we are talking about quantum gates and circuits that always map classical states to classical states, we refer to them as *reversible* gates or circuits. In other words, a reversible gate is a special type of quantum gate that maps classical states to classical states, or equivalently gives rise to a permutation matrix.

Now let us see how we can transform the original classical circuit into such a circuit, obeying the constraint that only reversible gates are used inside the circuit. We do not want to have to reinvent the wheel, so to speak, so we can begin by trying to simply replace the gates of the classical circuit with reversible gates. The Toffoli gates will be handy for doing this.

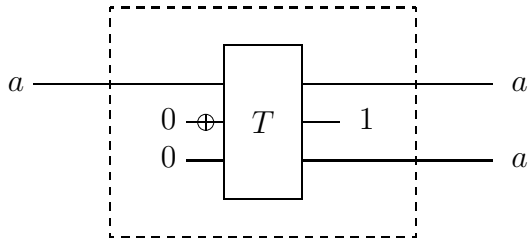
AND gates. We can simulate an AND gate using a Toffoli gate as follows:



NOT gates. These are already unitary, so nothing needs to be done.

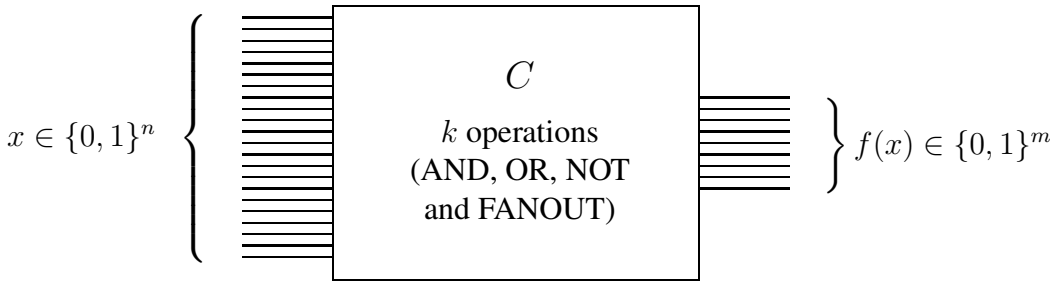
OR gates. These can be replaced by an AND gate and three NOT gates using DeMorgan's Laws.

FANOUT. We often do not even think of this as an operation, but we could have fanout in a classical Boolean circuit, and this needs to be explicitly implemented in a quantum circuit. Note that this does not mean replicating the quantum state of a qubit—it means replicating the classical state. Here is how this operation can be simulated using a Toffoli gate:

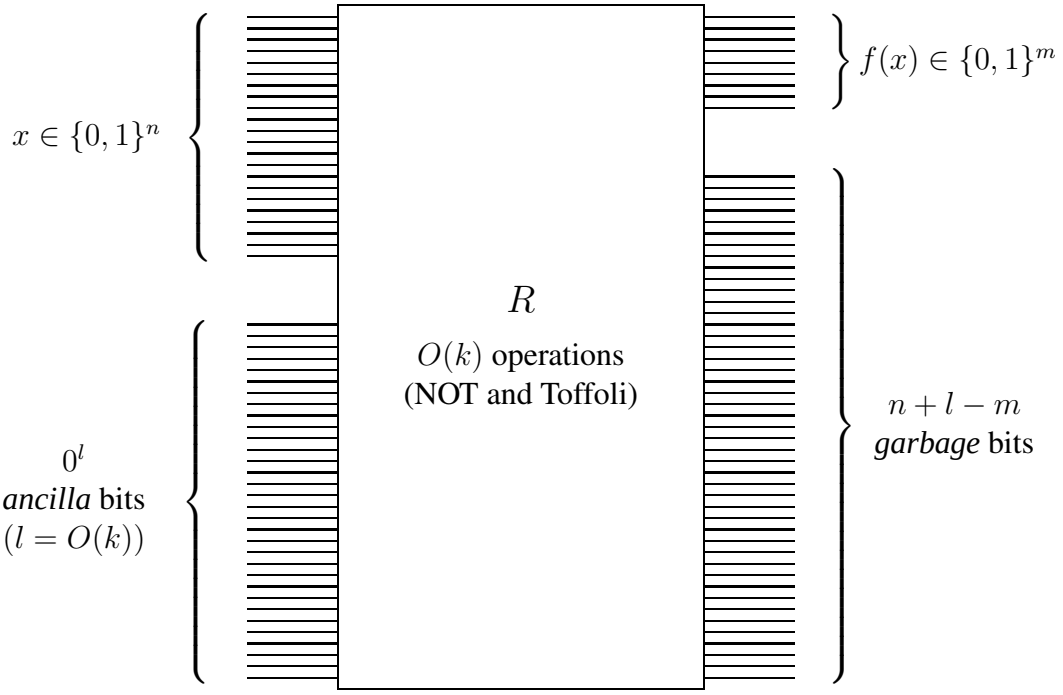


Of course it would have been just as easy to input a 1 into the second input of the Toffoli gate rather than performing a NOT operation on a 0. I've only written it like this because it will be convenient later to have all of the auxiliary inputs start in the 0 state.

Now, we can simply go through the original circuit and replace each of the AND, OR, NOT, and FANOUT operations with reversible gates as just described. If we started with a general circuit C computing a function $f : \{0,1\}^n \to \{0,1\}^m$ as follows:



Then we would end up with a reversible circuit R as follows.



The auxiliary bits needed for the AND and FANOUT simulations in this circuit are sometimes called *ancilla* bits. The bits labeled *garbage* are the extra bits that resulted from the simulations (that were left inside the dashed boxes in the above pictures).

The circuit above is not quite what we want, because the garbage bits are undesirable—we have no control over them, and they would cause problems in many quantum computations. However, we almost have what we are after. By taking advantage of the fact that NOT gates and Toffoli gates are their own inverses, we can simply take a "mirror image" of R to get R^{-1}, which simply undoes the computation of R. Combining R, R^{-1}, and a collection of controlled-NOT gates as described in Figure 1 essentially gives us what we were looking for. The only difference between this circuit and our initial aim is the existence of the ancilla bits. (It should be noted that commonly when people use the term "ancilla", they often expect that such bits are returned to their initial 0 state as in the circuit in Figure 1.) These bits will not have any adverse effects, and often we do not even explicitly mention them.

So, in summary, if you have a classical Boolean circuit C for computing some function

$$f : \{0,1\}^n \to \{0,1\}^m,$$

then the procedure above will transform a description of this circuit into a description of a quantum circuit S_C composed only of Toffoli gates, NOT gates, and controlled-NOT gates that satisfies

$$S_C \left|x\right\rangle \left|y\right\rangle \left|0^l\right\rangle = \left|x\right\rangle \left|y \oplus f(x)\right\rangle \left|0^l\right\rangle .$$

The number of ancilla qubits l and the total number of gates in S_C is linear in the number of operations in the original circuit C. The presence of the ancilla qubits will have no effect on

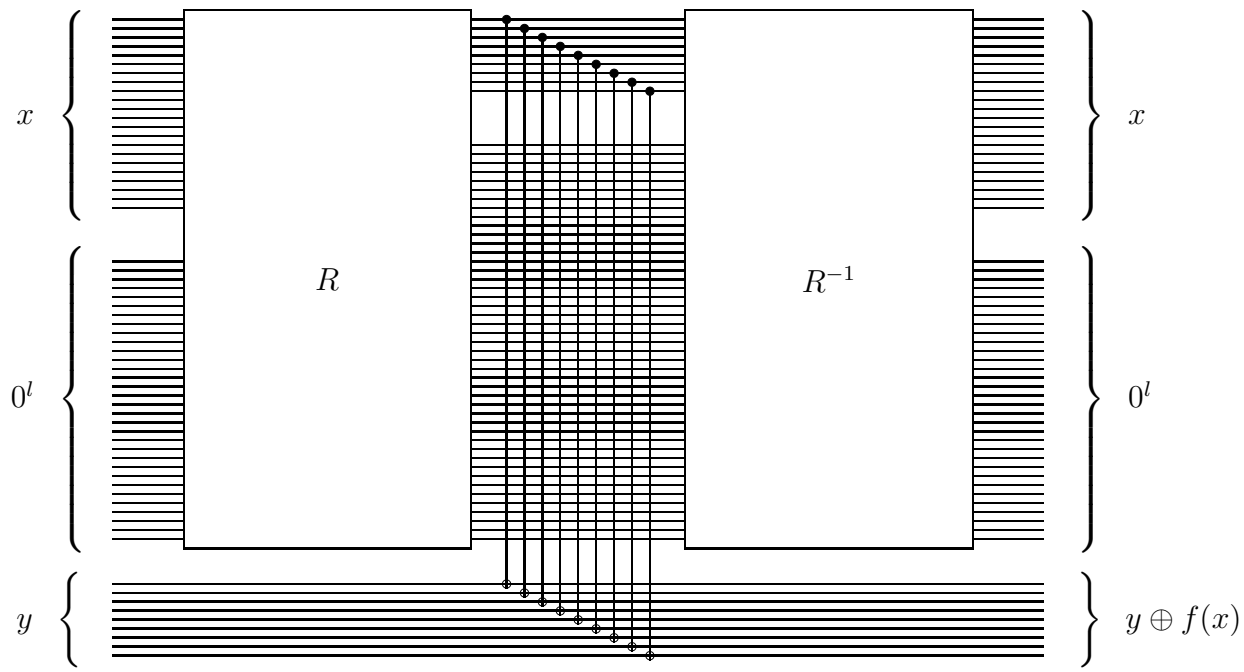


Figure 1: The final reversible circuit S_C implementing the transformation we are after.

any algorithm that uses the circuit S_C, and in fact they could be re-used to perform some other transformation after S_C is performed. For this reason, we will usually say that S_C performs the transformation

$$S_C \left|x\right\rangle \left|y\right\rangle = \left|x\right\rangle \left|y \oplus f(x)\right\rangle ,$$

leaving it implicit that some number of ancilla qubits may have been used to implement S_C.

Reversible implementation of invertible functions

Finally, suppose that you have a function of the form

$$f : \{0,1\}^n \to \{0,1\}^n$$

that itself is invertible (one-to-one and onto), and suppose that f is efficiently computed by some Boolean circuit C. We already know that it is possible to build a reversible circuit S_C that performs the transformation

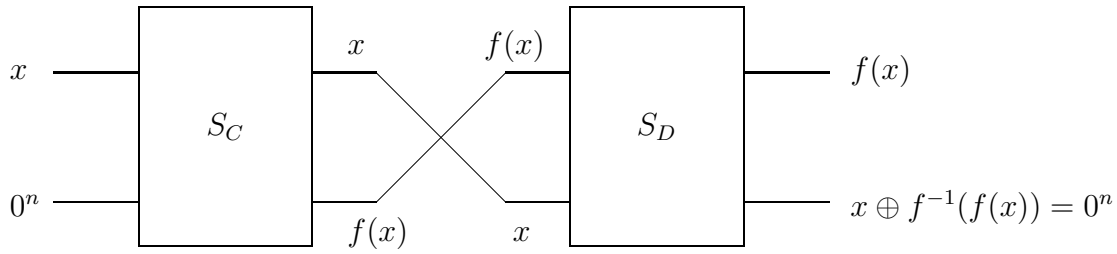
$$S_C \left|x\right\rangle \left|y\right\rangle = \left|x\right\rangle \left|y \oplus f(x)\right\rangle$$

for all $x, y \in \{0,1\}^n$ (possibly using some number of ancilla bits). Is it possible to go further and build a reversible circuit Q, possibly using some ancilla bits, that efficiently performs the transformation

$$Q \left|x\right\rangle = \left|f(x)\right\rangle$$

for all $x \in \{0,1\}^n$? Because f is assumed to be invertible, the transformation to be performed by Q is indeed reversible, so in principle it is possible to build Q. The question, however, is about efficiency.

Without any further assumptions it is not known how to construct Q given only C. However, if in addition we have a Boolean circuit D that efficiently computes f^{-1}, then Q can be constructed that efficiently performs the required transformation as follows:



(In order to simplify the picture, each "wire" represents n bits and the ancilla bits needed by the circuits S_C and S_D have not been shown explicitly.)

Lecture 8: Phase estimation

February 14, 2006

Last time we discussed reversible computation. We established that any classical Boolean circuit can be converted to a reversible (and therefore unitary) circuit that efficiently implements the function computed by the original circuit. Specifically, if the original Boolean circuit computes the function

$$f : \{0,1\}^n \to \{0,1\}^m,$$

then the reversible circuit efficiently implements the transformation

$$B_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$$

for all $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, possibly using some ancilla qubits that we do not mention explicitly. Moreover, if

$$f : \{0,1\}^n \to \{0,1\}^n$$

is invertible and we have Boolean circuits for efficiently computing both f and f^{-1}, then we can construct an efficient reversible circuit that performs the transformation

$$P_f |x\rangle = |f(x)\rangle$$

for all $x \in \{0,1\}^n$.

The next step toward Shor's Algorithm for factoring integers efficiently on a quantum computer will be to subject a particular reversible circuit based on one of the arithmetic problems we saw previously to a process known as *phase estimation*. This was not historically the way that Shor's algorithm was first described, but it is a nice way to understand how it works.

The process of phase estimation can be described in a more general context than what we need. So, although we will eventually apply it to a particular reversible transformation based on an arithmetic function, we will speak in greater generality when discussing phase estimation.

The phase estimation problem

Suppose that we have a description of some quantum circuit Q acting on n qubits. Associated with Q is some $2^n \times 2^n$ unitary matrix U. Obviously, if n is reasonably large such as $n = 1,000$, it would be impossible to write down an explicit description of U because it is too large—all of the computers in the world could only store a tiny fraction of its entries. Even if you just wanted to compute a single entry of U from the description of Q, you might be faced with a computationally difficult task.

Because U is unitary, we know from linear algebra that it has a *complete, orthonormal collection of eigenvectors*

$$|\psi_1\rangle, \ldots, |\psi_N\rangle$$

(where $N = 2^n$), and associated eigenvalues having the form

$$e^{2\pi i\theta_1}, \ldots, e^{2\pi i\theta_N}$$

where $\theta_1, \ldots, \theta_N \in [0, 1)$. This means that

$$U |\psi_j\rangle = e^{2\pi i\theta_j} |\psi_j\rangle$$

for each $j \in \{1, \ldots, N\}$, and furthermore that

$$\langle\psi_j|\psi_k\rangle = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k \end{cases}$$

for all $j, k \in \{1, \ldots, N\}$. The reason why each of the eigenvalues has the form $e^{2\pi i\theta}$, which is equivalent to saying that these eigenvalues are on the complex unit circle, is that U is unitary and therefore preserves Euclidean length.

The problem that we will focus on may be stated as follows:

Phase Estimation Problem

Input: A quantum circuit Q that performs a unitary operation U, along with a quantum state $|\psi\rangle$ that is promised to be an eigenvector of U:

$$U |\psi\rangle = e^{2\pi i\theta} |\psi\rangle .$$

Output: An approximation to $\theta \in [0, 1)$.

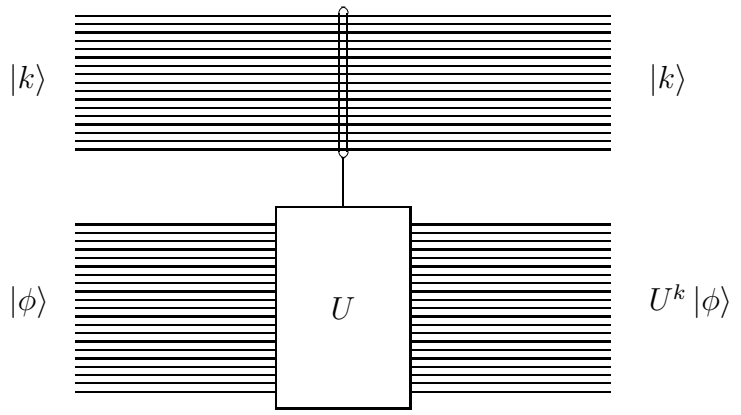
This problem is somewhat informally stated, because no specific requirements have been placed on the precision to which θ must be approximated. It will turn out that for an arbitrary circuit Q and eigenvector $|\psi\rangle$, the number θ can efficiently be approximated by the procedure that we will describe, but only to low precision (to a logarithmic number of bits in the circuit size). However, for certain choices of U it will be possible to achieve much higher precision, and when we apply our method to factoring this is the case in which we will be interested.

The phase estimation procedure

In order to describe the quantum procedure for phase estimation, let us introduce some notation. Suppose that U is a unitary transformation acting on n qubits, and suppose m is any positive integer. Then we let $\Lambda_m(U)$ denote the unique unitary transformation on $m+n$ qubits that satisfies

$$\Lambda_m(U) |k\rangle |\phi\rangle = |k\rangle \left(U^k |\phi\rangle\right)$$

for all choices of $k \in \{0, \ldots, 2^m - 1\}$ and an arbitrary n-qubit vector $|\phi\rangle$. In words, the first m qubits specify the number of times that U is to be applied to the remaining n qubits. We sometimes denote this transformation as suggested in the following quantum circuit diagram:

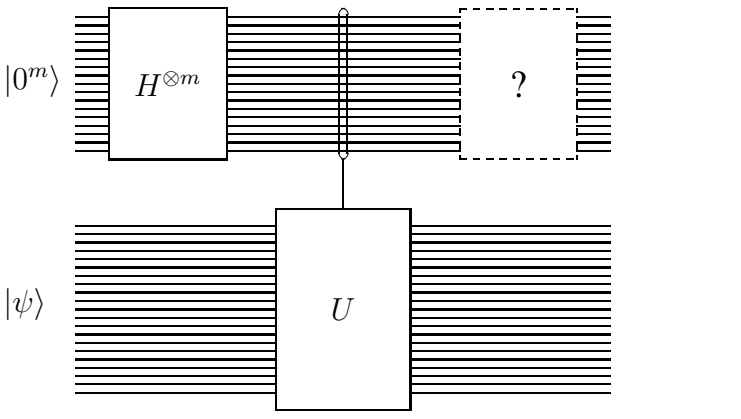


Note that it is possible that n and m differ significantly.

Now, if someone gives you a quantum circuit Q that performs the unitary operation U, there is no guarantee that you will be able to build a quantum circuit that efficiently implements $\Lambda_m(U)$ for your choice of m. For example, if $m \approx n$, you would probably require that U has some special properties to allow an efficient implementation. This is because the number of times k that U may effectively need to be performed can be exponential in m. If $m = O(\log n)$, however, you can always construct an efficient implementation of $\Lambda_m(U)$ (assuming you have a circuit Q that efficiently implements U).

We will not go into the details of how one would construct $\Lambda_m(U)$ for small values of m given a circuit Q implementing U, because our interest will be focused on a particular choice of U where it is easy to implement $\Lambda_m(U)$, even for $m = \Theta(n)$. Specifically, the transformation U will correspond to modular multiplication by a fixed number a, and so $\Lambda_m(U)$ will correspond to modular exponentiation, which we have already shown is efficiently implementable. However, for now let us just forget about these specifics and suppose that we have a quantum circuit implementing $\Lambda_m(U)$ for some particular choice of m (which may be large or small).

Now, consider the following quantum circuit diagram:



Note that the input to the second collection of qubits is the state $|\psi\rangle$, which is promised to be an eigenvector of U. We will try to fill in the missing part momentarily, but for now let us just

consider the state after the $\Lambda_m(U)$ operation is performed. The initial state is $|0^m\rangle |\psi\rangle$, and after the Hadamard transforms are performed the state becomes

$$\frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} |k\rangle |\psi\rangle .$$

Then the $\Lambda_m(U)$ transformation is performed, which transforms the state to

$$\frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} |k\rangle \left(U^k |\psi\rangle\right) .$$

Now, let us use the fact that $|\psi\rangle$ is an eigenvector of U to simplify this expression. Specifically we assume that

$$U |\psi\rangle = e^{2\pi i\theta} |\psi\rangle$$

for some real number $\theta \in [0, 1)$, which is the value we are trying to approximate. Applying U^k to $|\psi\rangle$ is equivalent to applying U to $|\psi\rangle$ a total of k times, so

$$U^k |\psi\rangle = \left(e^{2\pi i\theta}\right)^k |\psi\rangle = e^{2\pi ik\theta} |\psi\rangle .$$

Thus, we can rewrite the state of the circuit after the $\Lambda_m(U)$ gate has been performed as

$$\frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} |k\rangle \left(e^{2\pi ik\theta} |\psi\rangle\right) = \frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} e^{2\pi ik\theta} |k\rangle |\psi\rangle .$$

Notice that the same "phase kickback" effect has happened as for some of the algorithms we saw previously. The first m qubits and the last n qubits are uncorrelated at this point, given that they are in a tensor product state:

$$\frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} e^{2\pi ik\theta} |k\rangle |\psi\rangle = \left(\frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} e^{2\pi ik\theta} |k\rangle\right) |\psi\rangle .$$

So, if we discard the last n qubits, we are left with the state

$$\frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} e^{2\pi ik\theta} |k\rangle .$$

A simple case: $\theta = j/2^m$

Now, recall that our goal is to approximate θ. Suppose for the moment that θ happens to have a special form:

$$\theta = \frac{j}{2^m}$$

for some integer $j \in \{0, \ldots, 2^m - 1\}$. In general we cannot assume that θ has this form, but it is helpful to consider this case first. Then the above state can be written

$$\frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} e^{2\pi i \frac{jk}{2^m}} |k\rangle = \frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} \omega^{jk} |k\rangle$$

for $\omega = e^{2\pi i/2^m}$.

Several lectures ago we discussed the problem where a set of states is known, an unknown state from that collection is given to you, and your goal is to determine which of the possible states it is. It is possible to solve the problem perfectly if the set of states is orthonormal. This is the situation we have here. Let us define

$$|\phi_j\rangle = \frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} \omega^{jk} |k\rangle$$

for each $j \in \{0, \ldots, 2^m - 1\}$. We know that the state of the first m qubits of our circuit is one of the states $\{|\phi_j\rangle : j = 0, \ldots, 2^m - 1\}$ and the goal is to determine which one. Once we know j, we know θ as well (because we are still considering the special case $\theta = j/2^m$).

We have

$$\langle \phi_j | \phi_{j'} \rangle = \frac{1}{2^m} \sum_{k=0}^{2^m-1} \omega^{k(j'-j)} = \frac{1}{2^m} \sum_{k=0}^{2^m-1} \left(\omega^{j'-j}\right)^k.$$

Using the formula

$$\sum_{k=0}^{n-1} x^k = \frac{x^n - 1}{x - 1}$$

for $x \neq 1$ (and $\sum_{k=0}^{n-1} 1^k = n$ of course), along with the observation that $\omega^{2^m l} = 1$ for any integer l, we obtain

$$\langle \phi_j | \phi_{j'} \rangle = \begin{cases} 1 & \text{if } j = j' \\ 0 & \text{if } j \neq j'. \end{cases}$$

Thus, the set $\{|\phi_0\rangle, \ldots, |\phi_{2^m-1}\rangle\}$ is indeed orthonormal.

There is therefore a unitary transformation F that satisfies

$$F |j\rangle = |\phi_j\rangle$$

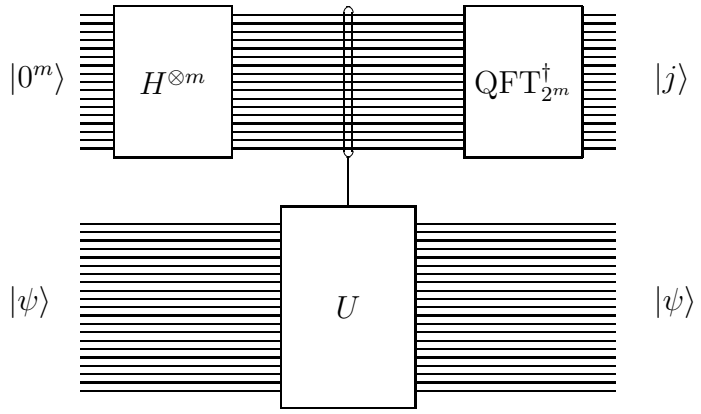
for $j = 0, \ldots, 2^m - 1$. We can describe this matrix explicitly by allowing the vectors $|\phi_j\rangle$ to determine the columns of F:

$$F = \frac{1}{\sqrt{2^m}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{2^m-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(2^m-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{2^m-1} & \omega^{2(2^m-1)} & \cdots & \omega^{(2^m-1)^2} \end{pmatrix}$$

This matrix defines a linear transformation that is of tremendous importance in many areas of science: the *discrete Fourier transform*. When we refer to this transformation in the context of quantum computing we call it the *quantum Fourier transform*, and for that reason it is also sometimes denoted QFT_{2^m}. For convenience, let us write it explicitly:

$$\mathrm{QFT}_{2^m} |j\rangle = \frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} e^{2\pi i j k/2^m} |k\rangle .$$

Plugging the inverse of this transformation into our circuit from before, we obtain:



Thus, measuring the first m qubits and dividing by 2^m tells us precisely the value θ.

Keep in mind, however, that this picture assumes that

$$U |\psi\rangle = e^{2\pi i j/2^m} |\psi\rangle .$$

We still have to worry about the more general case that θ does not have the form $j/2^m$ for some integer j. We will use exactly the same circuit for the general case, but the analysis will become more complicated (and the answer will only be an approximation to θ). In fact we have two major issues to address at this point:

1. What can be said about the measurement outcome in the case that θ does not have the form $j/2^m$ for some integer j, and
2. Can the quantum Fourier transform be implemented efficiently?

We will need the remainder of this lecture and much of the next to address these issues.

General values of θ

The analysis from above showed that, for an arbitrary value of θ, the state of the above circuit immediately before the inverse of the QFT is applied is

$$\frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} e^{2\pi i k\theta} |k\rangle |\psi\rangle .$$

We know that the second collection of qubits is uncorrelated with the first m qubits, so we are free to disregard these qubits and consider just the first m qubits. Applying the transformation $\mathrm{QFT}_{2^m}^{\dagger}$ to these qubits results in the state

$$\frac{1}{2^m} \sum_{k=0}^{2^m-1} \sum_{j=0}^{2^m-1} e^{2\pi i(k\theta - kj/2^m)} |j\rangle = \sum_{j=0}^{2^m-1} \left(\frac{1}{2^m} \sum_{k=0}^{2^m-1} e^{2\pi ik(\theta - j/2^m)} \right) |j\rangle .$$

The probability that the measurement results in outcome j is therefore

$$p_j = \left| \frac{1}{2^m} \sum_{k=0}^{2^m-1} e^{2\pi ik(\theta - j/2^m)} \right|^2$$

for each $j \in \{0, \ldots, 2^m - 1\}$.

We have already dealt with the case that $\theta = j/2^m$ for some choice of $j \in \{0, \ldots, 2^m - 1\}$, so let us assume that this is not the case: assume

$$e^{2\pi i(\theta - j/2^m)} \neq 1$$

for every integer j. Using the same formula for the sum of a geometric series from last time,

$$\sum_{k=0}^{n-1} x^k = \frac{x^n - 1}{x - 1}$$

for $x \neq 1$, we may then simplify:

$$p_j = \frac{1}{2^{2m}} \left| \frac{e^{2\pi i(2^m\theta - j)} - 1}{e^{2\pi i(\theta - j/2^m)} - 1} \right|^2$$

Our goal will be to show that the probability p_j is large for values of j that satisfy $j/2^m \approx \theta$ and small otherwise. We will do this in the next lecture.

Lecture 9: Phase estimation (continued); the quantum Fourier transform

February 16, 2006

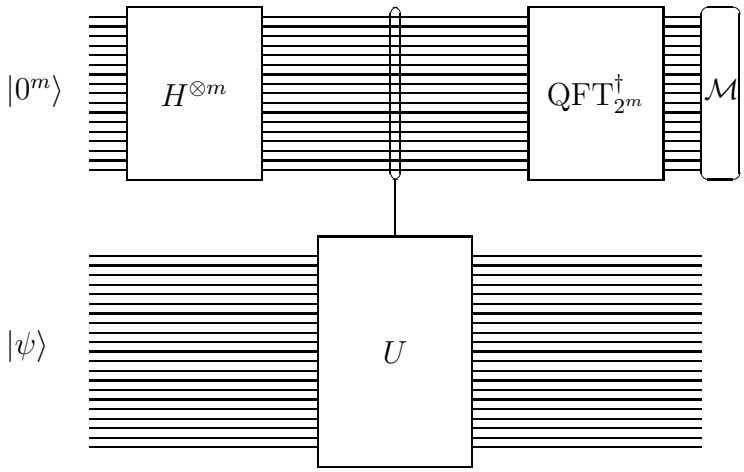
Phase estimation (continued)

In the previous lecture we discussed phase estimation. Recall that the set-up was as follows. We have a quantum circuit for performing the transformation $\Lambda_m(U)$, defined by

$$\Lambda_m(U) \left|k\right\rangle \left|\phi\right\rangle = \left|k\right\rangle U^k \left|\phi\right\rangle,$$

for some unitary transformation U and positive integer m, along with a quantum state $|\psi\rangle$ that is an eigenvector of U. The eigenvalue associated with $|\psi\rangle$ is $e^{2\pi i\theta}$ for $\theta \in [0,1)$, and the goal is to approximate θ.

We had devised the following procedure, which works perfectly when $\theta = j/2^m$ for some integer $j \in \{0,\ldots,2^m-1\}$:



Specifically, in the case $\theta = j/2^m$, the measurement results in outcome j with probability 1.

We were in the process of analyzing the (more typical) case when θ does not have the form $j/2^m$ for some integer j. We had determined that the probability associated with each possible outcome $j \in \{0,\ldots,2^m-1\}$ of the measurement was

$$p_j = \left| \frac{1}{2^m} \sum_{k=0}^{2^m-1} e^{2\pi i k(\theta - j/2^m)} \right|^2.$$

Because we already dealt with the case that $\theta = j/2^m$ for some choice of $j \in \{0,\ldots,2^m-1\}$, we may now assume this is not the case, and so

$$e^{2\pi i(\theta - j/2^m)} \neq 1$$

for every integer j. Using the same formula for the sum of a geometric series from last time,

$$\sum_{k=0}^{n-1} x^k = \frac{x^n - 1}{x - 1}$$

for $x \neq 1$, we may then simplify:

$$p_j = \frac{1}{2^{2m}} \left| \frac{e^{2\pi i(2^m\theta - j)} - 1}{e^{2\pi i(\theta - j/2^m)} - 1} \right|^2$$

Let us first consider the probability of obtaining the **best possible** j, meaning that

$$e^{2\pi i\theta} = e^{2\pi i(j/2^m + \varepsilon)}$$

for some real number ε with $|\varepsilon| \leq 2^{-(m+1)}$. This is equivalent to saying

$$\theta = \frac{j}{2^m} + \varepsilon \pmod 1$$

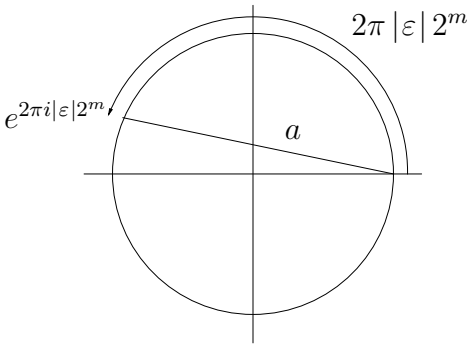
for $|\varepsilon| \leq 2^{-(m+1)}$, where "equality mod 1" means that the fractional parts of the two sides of the equation agree. Assuming that j satisfies this equation we may prove a lower bound on p_j as follows. Let

$$a = \left| e^{2\pi i(2^m\theta - j)} - 1 \right| = \left| e^{2\pi i\varepsilon 2^m} - 1 \right|,$$
$$b = \left| e^{2\pi i(\theta - j/2^m)} - 1 \right| = \left| e^{2\pi i\varepsilon} - 1 \right|,$$

so that

$$p_j = \frac{1}{2^{2m}} \frac{a^2}{b^2}.$$

To get a lower bound on p_j we need a lower bound on a and an upper bound on b. To get a lower bound on a, consider the following picture:



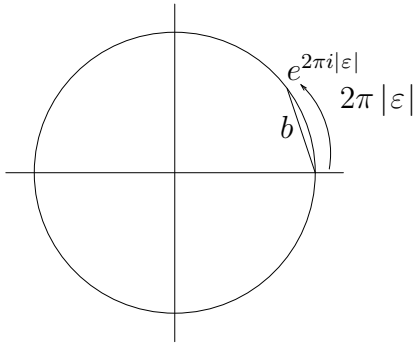
The ratio of the minor arc length to the chord length is at most $\pi/2$, so

$$\frac{2\pi\left|\varepsilon\right|2^m}{a} \leq \frac{\pi}{2},$$

which implies

$$a \geq 4\left|\varepsilon\right|2^m.$$

Along similar lines, we may consider b along with the fact that the ratio of arc length to chord length is at least 1:



We obtain

$$\frac{2\pi\left|\varepsilon\right|}{b} \geq 1$$

so

$$b \leq 2\pi\left|\varepsilon\right|.$$

Putting the two bounds together, we obtain

$$p_j \geq \frac{1}{2^{2m}}\frac{16\left|\varepsilon\right|^2 2^{2m}}{4\pi^2\left|\varepsilon\right|^2} = \frac{4}{\pi^2} > 0.4.$$

Although you might not think that 40% is very good, in fact it is amazing in a way—this is the probability that every single one of the bits you measure is correct, so that your approximation to θ is good to m bits of precision.

We can use basically the same methods to put upper bounds on the probability of obtaining inaccurate results. Suppose now that for a given value of j we have

$$e^{2\pi i\theta} = e^{2\pi i(j/2^m+\varepsilon)}$$

for some real number ε with $\frac{\alpha}{2^m} \leq |\varepsilon| < 1/2$. Here α is an arbitrary positive number that we can choose later. As before we have

$$p_j = \frac{1}{2^{2m}}\frac{a^2}{b^2}$$

for

$$a = \left|e^{2\pi i\varepsilon 2^m} - 1\right|,$$
$$b = \left|e^{2\pi i\varepsilon} - 1\right|.$$

This time we will simply use the fact that $a \leq 2$. The bound $b \geq 4\,|\varepsilon|$ follows by similar reasoning to the bound on a from before. Now we have

$$p_j \leq \frac{4}{2^{2m}(4\,|\varepsilon|)^2} = \frac{1}{4\alpha^2}.$$

This implies that highly inaccurate results are very unlikely. For example, if we consider $\alpha = 1$, meaning that our assumption is only that $|\varepsilon| \geq 2^{-m}$, the probability of obtaining the corresponding value of j is at most $1/4$. For worse approximations, implying a larger bound on $|\varepsilon|$, the probability of obtaining the corresponding value of j quickly becomes very small.

So, what should you do if you want better than a $4/\pi^2$ probability of obtaining an approximation of θ that is good to, say, k bits of precision? One way to do this is to set $m = k+2$, say, run the phase estimation procedure several times, and to look for the most commonly appearing outcome. At least one outcome, which is accurate to $k+2$ bits of precision, occurs with probability at least $4/\pi^2$. Outcomes with fewer than k bits of precision are much less likely as argued above. If you now take the most commonly occurring outcome and round it to k bits of precision, the probability of correctness approaches 1 exponentially fast in the number of times the procedure is repeated. Notice also that you do not need multiple copies of the state $|\psi\rangle$ to perform this process, because the state $|\psi\rangle$ remains on the lower collection of qubits each time the procedure is performed and can simply be fed into the next iteration.

Efficient implementation of the quantum Fourier transform

Now let us consider how the quantum Fourier transform may be implemented by quantum circuits. Recall that

$$\mathrm{QFT}_{2^m}\,|j\rangle = \frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} e^{2\pi i j k/2^m}\,|k\rangle\,.$$

Let us generalize some notation we used last time and let

$$\omega_N = e^{2\pi i/N}$$

for any positive integer N. Let us also define a unitary mapping $\widetilde{\mathrm{QFT}}_{2^m}$ to be the same as QFT_{2^m} except with the output qubits in reverse order. Specifically, if an integer $k \in \{0, \ldots, 2^m - 1\}$ is written in binary notation as $k_{m-1}k_{m-2}\cdots k_0$ then we define

$$\widetilde{\mathrm{QFT}}_{2^m}\,|j_{m-1}j_{m-2}\cdots j_0\rangle = \frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} \omega_{2^m}^{jk}\,|k_0k_1\cdots k_{m-1}\rangle\,.$$

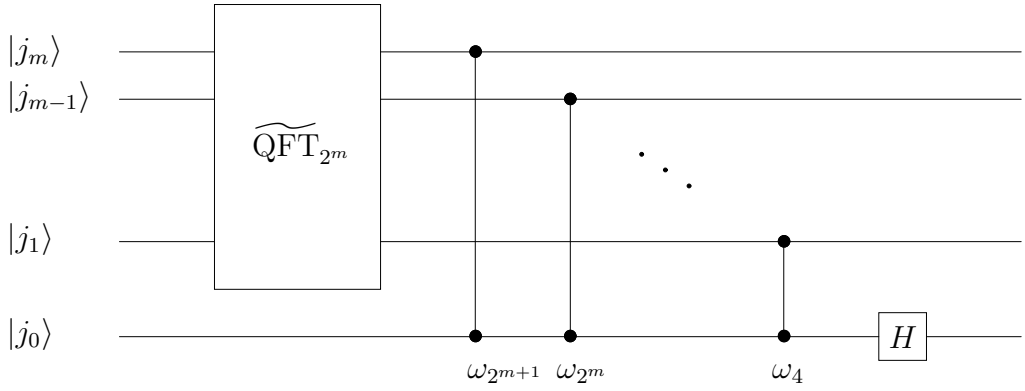
Certainly if we can come up with an efficient implementation of $\widetilde{\mathrm{QFT}}_{2^m}$, then an efficient implementation of QFT_{2^m} follows—just reverse the order of the output qubits after performing $\widetilde{\mathrm{QFT}}_{2^m}$. The reason why we consider $\widetilde{\mathrm{QFT}}_{2^m}$ rather than QFT_{2^m} is simply for convenience.

Our description of quantum circuits for performing $\widetilde{\mathrm{QFT}}_{2^m}$ for any given value of m is essentially recursive. Let us start with the base case, which is $m = 1$. The transformation $\widetilde{\mathrm{QFT}}_2$ is just

a fancy name for a Hadamard transform:

$$\widetilde{\mathrm{QFT}}_2 \left|j\right\rangle = \frac{1}{\sqrt{2}} \sum_{k=0}^{1} \omega_2^{jk} \left|k\right\rangle = \frac{1}{\sqrt{2}} \left|0\right\rangle + \frac{1}{\sqrt{2}} (-1)^j \left|1\right\rangle = H \left|j\right\rangle .$$

For general $m \geq 2$, the following circuit computes $\widetilde{\mathrm{QFT}}_{2^{m+1}}$:



Of course the diagram assumes you know how to implement the transformation $\widetilde{\mathrm{QFT}}_{2^m}$, but using the fact that $\widetilde{\mathrm{QFT}}_2$ is the same as a Hadamard transform we can easily unwind the recursion if we want an explicit description of a circuit.

Now let us show that the circuit works correctly. It suffices as usual to show that it works correctly on classical states. We wish to show that

$$\widetilde{\mathrm{QFT}}_{2^{m+1}} \left|j_m \, j_{m-1} \, \cdots \, j_0\right\rangle = \frac{1}{\sqrt{2^{m+1}}} \sum_{k=0}^{2^{m+1}-1} \omega_{2^{m+1}}^{jk} \left|k_0 \, k_1 \, \cdots \, k_m\right\rangle$$

for each $j \in \{0, \ldots, 2^{m+1} - 1\}$.

Let us write

$$\begin{aligned} j' &= j_m \, j_{m-1} \, \cdots \, j_1 = \lfloor j/2 \rfloor, \\ k' &= k_{m-1} \, k_{m-2} \, \cdots \, k_0 = k - k_m 2^m. \end{aligned}$$

The initial state $\left|j\right\rangle$ may therefore be written $\left|j'\right\rangle \left|j_0\right\rangle$, and the operation $\widetilde{\mathrm{QFT}}_{2^m}$ maps this state to

$$\frac{1}{\sqrt{2^m}} \sum_{k'=0}^{2^m-1} \omega_{2^m}^{j'k'} \left|k'_0 \, k'_1 \, \cdots \, k'_{m-1}\right\rangle \left|j_0\right\rangle .$$

The controlled phase-shifts then transform this state to

$$\frac{1}{\sqrt{2^m}} \sum_{k'=0}^{2^m-1} \omega_{2^m}^{j'k'} \omega_{2^{m+1}}^{j_0 k'_0} \omega_{2^m}^{j_0 k'_1} \cdots \omega_4^{j_0 k'_{m-1}} \left|k'_0 \, k'_1 \, \cdots \, k'_{m-1}\right\rangle \left|j_0\right\rangle .$$

Using the fact that $\omega_N = \omega_{rN}^r$ for any choice of positive integers N and r, we may simplify the above expression and conclude that the state of the circuit after the controlled phase shifts is

$$\frac{1}{\sqrt{2^m}} \sum_{k'=0}^{2^m-1} \omega_{2^{m+1}}^{2j'k'+j_0k'_0+j_0(2k'_1)+\cdots j_0(2^{m-1}k'_{m-1})} |k'_0\, k'_1\, \cdots\, k'_{m-1}\rangle\, |j_0\rangle$$
$$= \frac{1}{\sqrt{2^m}} \sum_{k'=0}^{2^m-1} \omega_{2^{m+1}}^{jk'} |k'_0\, k'_1\, \cdots\, k'_{m-1}\rangle\, |j_0\rangle\,.$$

Finally, the Hadamard transform maps this state to

$$\frac{1}{\sqrt{2^{m+1}}} \sum_{k'=0}^{2^m-1} \sum_{k_m=0}^{1} \omega_{2^{m+1}}^{jk'} (-1)^{k_m j_0} |k'_0\, k'_1\, \cdots\, k'_{m-1}\rangle\, |k_m\rangle\,.$$

Notice that

$$(-1)^{k_m j_0} = (-1)^{k_m j} = \omega_{2^{m+1}}^{j(2^m k_m)},$$

which implies that the final state is

$$\frac{1}{\sqrt{2^{m+1}}} \sum_{k'=0}^{2^m-1} \sum_{k_m=0}^{1} \omega_{2^{m+1}}^{jk'+j(2^m k_m)} |k'_0\, k'_1\, \cdots\, k'_{m-1}\rangle\, |k_m\rangle = \frac{1}{\sqrt{2^{m+1}}} \sum_{k=0}^{2^{m+1}-1} \omega_{2^{m+1}}^{jk} |k_0\, k_1\, \cdots\, k_m\rangle$$

as required.

How many gates are required in the above circuit? Letting $g(m)$ denote the number of gates needed to perform $\widetilde{QFT}_{2^m}$, we have the following recurrence:

$$g(1) = 1$$
$$g(m+1) = g(m) + (m+1).$$

The solution to this recurrence is

$$g(m) = \sum_{j=1}^{m} j = \binom{m+1}{2}.$$

Thus, we need only $O(m^2)$ gates to compute the quantum Fourier transform on m qubits. In fact there are better bounds known that are based on fast multiplication methods. However, these constructions are much more complicated and would probably not be practical (assuming we had a quantum computer) until m is quite large.

Lecture 10: Order finding

February 28, 2006

The Order Finding problem

Now that we have discussed the phase estimation technique, it is time to apply it to an interesting computational problem. The problem is called Order Finding, and as we will see in the next couple of lectures it is only one step away from integer factoring. In fact, this will be the quantum part of Shor's factoring algorithm—the rest is completely classical after this.

Some additional notation will be helpful for discussing the Order Finding problem. If a, b, and N are integers with $N \geq 1$ we write

$$a \equiv b \pmod{N}$$

to mean $N \mid (a - b)$ (shorthand for N divides $a - b$). As I mentioned before, we let $\mathbb{Z}_N$ denote the set $\mathbb{Z}_N = \{0, \ldots, N-1\}$. When we also associate with $\mathbb{Z}_N$ the operations of addition and multiplication modulo N, $\mathbb{Z}_N$ forms a ring. If in addition N is prime, then $\mathbb{Z}_N$ forms a field. We write $\mathbb{Z}_N^*$ to denote the following set:

$$\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N \,:\, \gcd(a, N) = 1\}.$$

The number of elements in $\mathbb{Z}_N^*$ determines a function called the *Euler φ-function*: $\varphi(N) = |\mathbb{Z}_N|$. When we associate with the set $\mathbb{Z}_N^*$ the operation of multiplication modulo N, it forms a group. This implies that for any element $a \in \mathbb{Z}_N^*$, there exists a unique element $b \in \mathbb{Z}_N^*$ that satisfies

$$ab \equiv 1 \pmod{N}.$$

We generally write $a^{-1} \pmod{N}$ (or just a^{-1} when N is understood) to denote this element b.

Now, we can define the *order* of a given element $a \in \mathbb{Z}_N^*$, which is what the order-finding problem concerns. For $a \in \mathbb{Z}_N^*$, the *order of a in $\mathbb{Z}_N^*$* (or the *order of a modulo N*) is the smallest positive integer r such that

$$a^r \equiv 1 \pmod{N}.$$

The fact that every $a \in \mathbb{Z}_N^*$ indeed has a well-defined order follows from Euler's Theorem, which states that

$$a^{\varphi(N)} \equiv 1 \pmod{N}$$

for any integers a and $N \geq 2$ with $\gcd(a, N) = 1$. It turns out that the order of any element $a \in \mathbb{Z}_N^*$ is always a divisor of $\phi(N)$.

For example, let $a = 4$ and $N = 35$. Because $\gcd(4, 35) = 1$ we have $4 \in \mathbb{Z}_{35}^*$. Computing powers of 4 modulo 35 gives:

$$4^1 \equiv 4, \quad 4^2 \equiv 16, \quad 4^3 \equiv 29, \quad 4^4 \equiv 11, \quad 4^5 \equiv 9, \quad 4^6 \equiv 1$$

(where all congruences are of course modulo 35). Thus, the order of 4 modulo 35 is 6.

Now that we know how the order of a given element $a \in \mathbb{Z}_N^*$ is defined, we can state the order finding problem. It is as you would probably guess:

Order Finding

Input: A positive integer $N \geq 2$ and an element $a \in \mathbb{Z}_N^*$.

Output: The order of a in $\mathbb{Z}_N^*$.

Classically this problem is hard (at least as far as we know). Certainly the obvious approach of computing powers of a modulo N until 1 is obtained can take time exponential in $\lg N$.

Solving order finding using phase estimation

Our main goal for the remainder of the lecture and part of the next will be to show that the order finding problem can be solved using the method of phase estimation.

Assume $N \geq 2$ is given and let n be the number of bits needed to encode elements of $\mathbb{Z}_N$ in binary (so $n = \lfloor \log_2(N-1) \rfloor + 1$). Given any $a \in \mathbb{Z}_N^*$, define an n-qubit transformation M_a as

$$M_a |x\rangle = |ax \pmod N\rangle .$$

for every $x \in \mathbb{Z}_N$. This is not a complete specification of M_a because it does not specify $M_a |x\rangle$ for $N \leq x < 2^n$, but we will not care about its action on such states. For the sake of choosing a well-defined transformation, we may say for simplicity that $M_a |x\rangle = |x\rangle$ for $N \leq x < 2^n$. The transformation M_a is reversible, and therefore unitary. This follows from the fact that it maps classical states to classical states, along with the observation that it has an inverse: $M_{a^{-1}} M_a = I$, where a^{-1} is the inverse of a modulo N.

Let us consider subjecting the transformation M_a to phase estimation. It will turn out that in doing this we will be able to determine the order of a modulo N. One can efficiently implement a reversible circuit for performing M_a given that the functions $f(x) = ax \pmod N$ and $g(x) = a^{-1}x \pmod N$ are efficiently computable by Boolean circuits. If we wish to subject this transformation to phase estimation, however, recall that we will need to have an efficient implementation of $\Lambda_m(M_a)$ for m (roughly) corresponding to the number of bits of precision we need. In fact, this transformation, which can be expressed as

$$\Lambda_m(M_a) |k\rangle |x\rangle = |k\rangle |a^k x \pmod N\rangle ,$$

can also be implemented efficiently. Specifically, based on the modular exponentiation algorithm I mentioned several lectures ago, it can be implemented using $O(mn^2)$ gates. We will need to wait and see how precise our procedure needs to be to determine the order of a, but we may keep in the back of our minds that $m = O(n)$ will be sufficient.

What are the eigenvectors and eigenvalues of M_a? It turns out that there are lots, but we will only need to consider a subset of them. Letting r be the order of a in $\mathbb{Z}_N^*$, we see that the following vector is an eigenvector of M_a:

$$|\psi_0\rangle = \frac{1}{\sqrt{r}} \left(|1\rangle + |a\rangle + |a^2\rangle + \cdots + |a^{r-1}\rangle \right) .$$

(From now on the operations inside kets are implicitly assumed to be modulo N.) To see that it is an eigenvector, just compute:

$$\begin{aligned} M_a |\psi_0\rangle &= \frac{1}{\sqrt{r}} \left(|a\rangle + |a^2\rangle + |a^3\rangle + \cdots + |a^r\rangle \right) \\ &= \frac{1}{\sqrt{r}} \left(|a\rangle + |a^2\rangle + \cdots + |a^{r-1}\rangle + |1\rangle \right) \\ &= |\psi_0\rangle . \end{aligned}$$

So, the associated eigenvalue is 1. Here is another:

$$|\psi_1\rangle = \frac{1}{\sqrt{r}} \left(|1\rangle + \omega_r^{-1} |a\rangle + \omega_r^{-2} |a^2\rangle + \cdots + \omega_r^{-(r-1)} |a^{r-1}\rangle \right)$$

where as before we define $\omega_r = e^{2\pi i/r}$. We have

$$\begin{aligned} M_a |\psi_1\rangle &= \frac{1}{\sqrt{r}} \left(|a\rangle + \omega_r^{-1} |a^2\rangle + \omega_r^{-2} |a^3\rangle + \cdots + \omega_r^{-(r-1)} |a^r\rangle \right) \\ &= \frac{\omega_r}{\sqrt{r}} \left(\omega_r^{-1} |a\rangle + \omega_r^{-2} |a^2\rangle + \omega_r^{-3} |a^3\rangle + \cdots + \omega_r^{-r} |a^r\rangle \right) \\ &= \omega_r |\psi_1\rangle . \end{aligned}$$

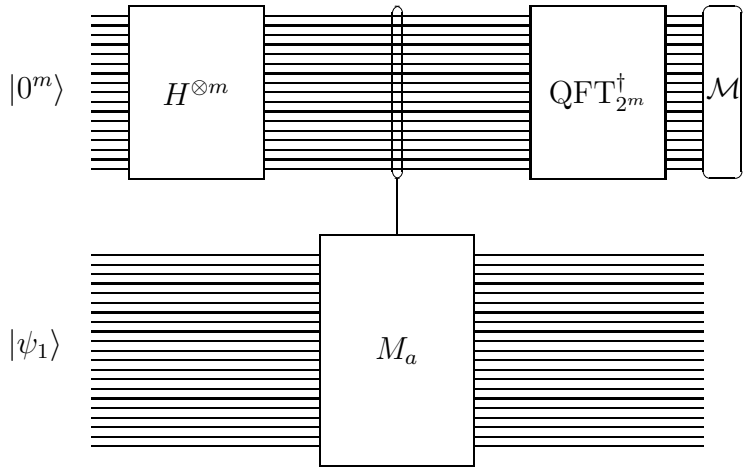
In general, for

$$|\psi_j\rangle = \frac{1}{\sqrt{r}} \left(|1\rangle + \omega_r^{-j} |a\rangle + \omega_r^{-2j} |a^2\rangle + \cdots + \omega_r^{-j(r-1)} |a^{r-1}\rangle \right)$$

we have

$$M_a |\psi_j\rangle = \omega_r^j |\psi_j\rangle .$$

At this point we don't know how to get our hands on any of these eigenvectors, but let's imagine, for the sake of understanding how phase estimation could help us with our problem, that we have a copy of the state $|\psi_1\rangle$. Consider the phase estimation procedure for this eigenvector:



The eigenvalue associated with $|\psi_1\rangle$ is $\omega_r = e^{2\pi i(1/r)}$. Assuming we were to perform the procedure several times and take the most commonly appearing result, we will have an approximation $j/2^m$ that, with very high probability, is within distance $1/2^{m+1}$ to $1/r$. I mentioned this before, but it is worth noting again that we only need one copy of the eigenvector $|\psi_1\rangle$ even if we want to run the phase estimation procedure several times, because the eigenvector is unaffected by the phase estimation procedure.

Remember that our goal is to find r. If we have some integer $j \in \{0, \ldots, 2^m - 1\}$ for which

$$\frac{j}{2^m} \approx \frac{1}{r},$$

there is an obvious strategy that (hopefully) will find r: you just type $j/2^m$ into your calculator and press the button that looks like this:

$$\boxed{1/x}$$

In other words, whether it is with a calculator or (more likely) with an ordinary classical computer, compute the reciprocal of $j/2^m$. Although r must be an integer, you probably would not get an integer when you compute $2^m/j$ because $j/2^m$ is only an approximation to r. The natural thing to do is to round off to the nearest integer, so your guess for r would be

$$\left\lfloor \frac{2^m}{j} + \frac{1}{2} \right\rfloor.$$

Now, if your approximation $j/2^m$ to $1/r$ does not have sufficiently many bits of precision, you cannot be sure your answer is correct. So how accurately do we need to approximate $1/r$ to be correct? Intuitively, you need enough precision to discriminate between estimates for $1/(r-1)$, $1/r$, $1/(r+1)$, and so on, so a good guess is that the approximation from the phase estimation procedure should be within $1/(2r(r+1))$ of the correct value—because this is half the smallest distance between $1/r$ and $1/s$ for some integer $s \neq r$. Of course we do not know what r is, but we do know that $r < N$. So let us guess that it is sufficient that our approximation satisfies

$$\frac{j}{2^m} = \frac{1}{r} - \varepsilon$$

for ε satisfying

$$|\varepsilon| \leq \frac{1}{2N^2}.$$

Indeed this accuracy is sufficient. We can be sure that rounding $2^m/j$ to the nearest integer will give r if

$$\left| \frac{2^m}{j} - r \right| < \frac{1}{2}.$$

Assuming that $|\varepsilon| \leq 1/(2N^2)$ implies

$$\left| \frac{2^m}{j} - r \right| = \left| \frac{1}{\frac{1}{r} - \varepsilon} - r \right| = \left| \frac{r^2\varepsilon}{1 + r\varepsilon} \right| \leq \frac{\frac{r^2}{2N^2}}{1 - \frac{r}{2N^2}} = \frac{r^2}{2N^2 - r} \leq \frac{(N-1)^2}{2N^2 - N} < \frac{1}{2}.$$

Therefore, if we take $m = 2n$ in the phase estimation procedure, the resulting accuracy will be sufficient to find r.

We still have a big problem to contend with, however, which is that we do not know how to get our hands on $|\psi_1\rangle$. Obtaining this vector is probably no easier than finding r, so we will have to change our approach slightly. The solution will be to run the phase estimation procedure on the state $|1\rangle$ rather than on an eigenvector. It follows from the observation that

$$\frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |\psi_k\rangle = \frac{1}{r} \sum_{k=0}^{r-1} \sum_{l=0}^{r-1} \omega_r^{-kl} |a^l\rangle = |a^0\rangle = |1\rangle$$

that running the phase estimation procedure on the state $|1\rangle$ is equivalent to running the procedure on an eigenvector $|\psi_k\rangle$ for $k \in \{0, 1, \ldots, r-1\}$ chosen uniformly at random. It is not obvious that this is so—it depends on the fact that the phase estimation procedure leaves eigenvectors unchanged, and that these eigenvectors form an orthonormal set. Specifically, if we were to run the phase estimation procedure on the state

$$|1\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |\psi_k\rangle ,$$

the state immediately before the measurement would have the form

$$\frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |\phi_k\rangle |\psi_k\rangle$$

where each $|\phi_k\rangle$ is the state of the first m qubits that you would get by running the phase estimation procedure just on $|\psi_k\rangle$. Because the states $|\psi_0\rangle, \ldots, |\psi_{r-1}\rangle$ are orthonormal, the probability to obtain some value j from the measurement is just the average over $k \in \{0, \ldots, r-1\}$ chosen uniformly to have measured that value starting with the eigenvector $|\psi_k\rangle$.

So, when we run the phase estimation procedure on $|1\rangle$, we may as well imagine that we instead ran the phase estimation procedure on an eigenvector $|\psi_k\rangle$ for $k \in \{0, \ldots, r-1\}$ chosen uniformly at random. This will be almost as good as having $|\psi_1\rangle$, as we will discuss in the next lecture.

Lecture 11: Order finding (continued); reducing factoring to order finding

March 2, 2006

Solving order finding using phase estimation (continued)

We will begin this lecture by finishing the discussion of how the order finding problem can be solved via phase estimation.

Assume that we are given $a \in \mathbb{Z}_N^*$, and our goal is to calculate the order of a in $\mathbb{Z}_N^*$, which we suppose is r. Recall that we defined a unitary operation M_a so that

$$M_a |x\rangle = |ax\rangle$$

for every $x \in \mathbb{Z}$. Then for

$$|\psi_k\rangle = \frac{1}{\sqrt{r}} \left(|1\rangle + \omega^{-k} |a\rangle + \omega^{-2k} |a^2\rangle + \cdots + \omega^{-k(r-1)} |a^{r-1}\rangle \right)$$

we have $M_a |\psi_k\rangle = \omega^k |\psi_k\rangle$.

Last time we considered what happens when we perform phase estimation on the state $|\psi_1\rangle$. The eigenvalue associated with $|\psi_1\rangle$ is $\omega = e^{2\pi i (1/r)}$. Assuming we were to perform the phase estimation procedure several times and take the most commonly appearing result, with very high probability we will have an approximation $j/2^m$ that is close to $1/r$. Our guess for r would therefore be $2^m/j$, rounded off to the nearest integer. We determined that if $m = 2n$ and the phase estimation procedure succeeds in finding a best m bit approximation $j/2^m$ to $1/r$, then rounding $2^m/j$ to the nearest integer will indeed give r.

Unfortunately we do not know how to get our hands on $|\psi_1\rangle$. In fact it seems like one would need to know r before preparing it, but if we know r there is no need to bother. There is no known way to prepare $|\psi_1\rangle$ without knowing r, so a different approach is required. Instead, we considered running the phase estimation procedure on the state

$$|1\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |\psi_k\rangle .$$

We argued that this would give a result that is indistinguishable from running the phase estimation procedure on an eigenvector $|\psi_k\rangle$ for a random value $k \in \{0, \ldots, r-1\}$ chosen uniformly. In other words, the phase estimation procedure would give an approximation

$$\frac{j}{2^m} \approx \frac{k}{r}$$

for $k \in \{0, \ldots, r-1\}$ chosen uniformly. The question we must now address is: if you have enough precision, can you now find r from such an approximation? It is important for us to keep in mind that k will not be known explicitly when we attempt to find r.

The answer is that you cannot be guaranteed to find r given just one sample, but if you repeat the process several times (each time for a different k) you can find r with high probability. The way to do it is to use the *continued fraction algorithm*. I described how this algorithm works in lecture but I will not include a description in these notes. If you are interested in learning more about the algorithm, I would be happy to give you some good references. We can sum up what the continued fraction algorithm accomplishes in the following fact:

Fact. Given a real number $\alpha \in (0, 1)$ and $N \geq 2$, there is at most one fraction

$$\frac{x}{y}$$

with $0 \leq x, y < N$, $y \neq 0$, $\gcd(x, y) = 1$ and

$$\left| \alpha - \frac{x}{y} \right| < \frac{1}{2N^2}.$$

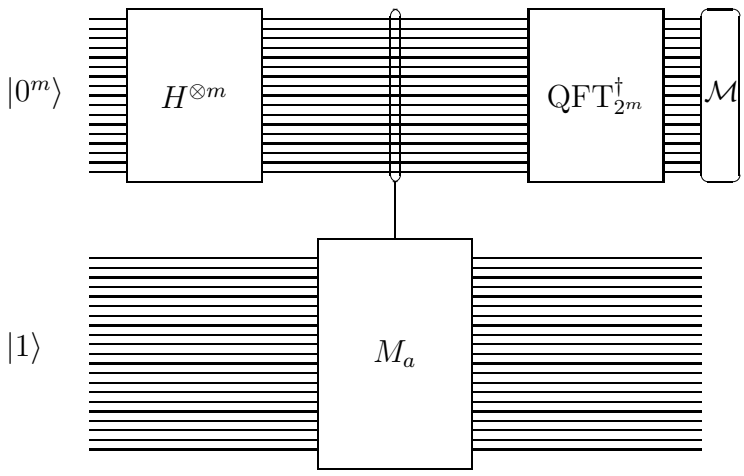
Given α and N, the continued fraction will find x and y (if they exist) in $O((\lg N)^3)$ bit operations.

So, by again setting $m = 2n$, applying phase estimation, and then applying the continued fraction algorithm to the result, we can find x and y such that

$$\frac{x}{y} = \frac{k}{r}.$$

In other words, we can find k/r **in lowest terms**. This may fail to find r if $\gcd(k, r) > 1$, but we know that at least y divides r. Repeating several times (each time for a different k) and taking the least common multiple of the resulting y values gives r with high probability.

So, in the end, the algorithm is as follows. First, choose $m = 2n$ and run the phase estimation procedure:



(It can be run sequentially several times to improve accuracy.) Plug the measurement outcome $j/2^m$ along with N into the continued fraction algorithm, which gives a fraction x/y. Repeat the entire process several times, taking the least common multiple of the y values. The result will be r with high probability. The entire process is known as Shor's algorithm for order finding.

Reduction of factoring to order finding

I've mentioned several times that we can factor integers efficiently using quantum computers, and that this is the problem solved by Shor's algorithm. Really, it turns out that "Shor's algorithm" is more of a technique than a specific algorithm, and it can be used to solve an interesting collection of problems. (Order finding is one of the simplest to describe.)

Now let us see that the integer factoring problem can be efficiently solved given an algorithm for order-finding. In other words, factoring *reduces* to order-finding. To be specific, it is a randomized polynomial-time Turing reduction, which is probably different from (but similar to) the types of reductions you might have seen in CPSC 313 and 413. The idea, however, is essentially the same—if you have an efficient algorithm for order finding, then the reduction gives you an efficient algorithm for factoring.

Integer Factorization

Input: A positive integer $N \geq 2$.

Output: A prime factorization $N = p_1^{k_1} \cdots p_m^{k_m}$.

Let us note a few facts, which we will not prove or discuss in any detail. First, if N is a prime number or a prime power (i.e., p^k for some prime p and integer $k \geq 1$), then there are efficient classical algorithms for solving the integer factorization problem in these cases. So, we can imagine that we first run such an algorithm on the input N; if it succeeds then we are done, otherwise we continue on under the assumption that N has at least two distinct prime factors (i.e., $m \geq 2$).

Next, it is enough to have an algorithm that takes a composite N as input and just finds two integers $u, v \geq 2$ such that $N = uv$. If we have such an algorithm, we can run it recursively (interleaved with the classical algorithm for prime powers) to find a complete prime factorization of N.

Finally, if our goal is to find integers $u, v \geq 2$ such that $N = uv$ for N an **even** composite number, then we really don't need to work very hard; let $u = 2$ and $v = N/2$.

Now, consider the process described in Figure 2. Why does it work? The basic idea is as follows. Suppose that the random choice of a is in $\mathbb{Z}_N^*$ (which is very likely), and that the order r of a is even. Then

$$a^r \equiv 1 \pmod{N}$$

so

$$N \mid a^r - 1.$$

Because $a^r - 1 = (a^{r/2} + 1)(a^{r/2} - 1)$ we have

$$N \mid (a^{r/2} + 1)(a^{r/2} - 1).$$

It cannot happen that $N \mid (a^{r/2} - 1)$, because this would mean that r was not the order of a after all. If we are lucky and it is not the case that $N \mid (a^{r/2} + 1)$, then we know the algorithm will work. This is because the factors of N are then necessarily split between $(a^{r/2} + 1)$ and $(a^{r/2} - 1)$, so computing $\gcd(a^{r/2} - 1, N)$ will reveal a nontrivial factor of N.

Input: an odd, composite integer N that is not a prime power.

Repeat
    Randomly choose $a \in \{2, \ldots, N-1\}$.
    Compute $d = \gcd(a, N)$.
    If $d \geq 2$ then /* We've been incredibly lucky. */
        Return $u = d$ and $v = N/d$.
    Else /* Now we know $a \in \mathbb{Z}_N^*$. */
        Let r be the order of a in $\mathbb{Z}_N^*$. /* Requires the order finding algorithm. */
        If r is even then
            Compute $x = a^{r/2} - 1 \pmod{N}$.
            Compute $d = \gcd(x, N)$.
            If $d \geq 2$ then
                Return $u = d$ and $v = N/d$. /* Answer is found. */
Until answer is found (or you get tired).

Figure 2: Reduction of factoring to order finding

Each iteration of the loop therefore fails to give an answer if either r is odd or r is even but N divides $a^{r/2} + 1$. The probability that neither of these events occur is at least $1/2$. (In fact, it is at least $1 - 2^{m-1}$ for m the number of distinct primes dividing N. This is why the assumption that N is not a prime power was important. Also, the analysis of this fact requires that N is odd.)

Lecture 12: Grover's Algorithm

March 7, 2006

We have completed our study of Shor's factoring algorithm. The basic technique behind Shor's algorithm, which we described in terms of phase estimation, can also be used to solve some other number-theoretic and group-theoretic problems as well (such as computing discrete logarithms and calculating orders of abelian groups). This family of problems represents one of the main reasons why there is such great interest in building quantum computers, as the associated quantum algorithms give an exponential advantage over the best known classical algorithms for these problems.

We will now study a different quantum algorithmic technique. It does not give as impressive an advantage over classical algorithms as we have for Shor's algorithm, but the technique is applicable to a much broader class of problems. The simplest general problem to which this technique can be applied is *search*; and although there are other applications of the technique, this will be the focus of our investigation. The technique was first considered by Lov Grover, and is known as Grover's Algorithm in the context of search.

Unstructured search

We will be returning to the black-box context that was used to describe Deutsch's algorithm, the Deutsch-Jozsa algorithm, and Simon's algorithm. (The fact that Grover's algorithm works in this very general context represents one of its strengths, because in general there will not need to be any promises on the black box.) Suppose that we have a function

$$f : \{0,1\}^n \to \{0,1\}$$

that is implemented by a reversible transformation B_f in the usual way:

$$B_f |x\rangle |a\rangle = |x\rangle |a \oplus f(x)\rangle$$

for all $x \in \{0,1\}^n$ and $a \in \{0,1\}$ The problem of *search* is simply to find a string $x \in \{0,1\}^n$ such that $f(x) = 1$, or to conclude that no such x exists if f is identically 0).

It is important to note that this searching problem is completely *unstructured*. There are no promises on the function f, so it is not possible to use binary search or any other fast searching method to efficiently solve the problem classically.

What is the best *classical* algorithm for solving the above search problem? As we have noted before, it is more complicated than one might initially think to rigorously prove lower bounds on algorithms. In this particular case it is not too hard, but because our focus is on the quantum algorithm for the problem we will only discuss informally the issue of classical complexity.

It is not hard to see that a deterministic algorithm would need to make 2^n queries to the black-box in the worst case (to distinguish the case where f is identically 0 from any of the cases where there is a single x for which $f(x) = 1$, for instance).

Probabilistically, a best strategy for an algorithm that makes k queries is to simply choose k distinct values of x and to query the black-box at these k values. In the case that there is a single value of x for which $f(x) = 1$, and we require that our algorithm succeeds in finding this x with probability at least $1 - \varepsilon$, then we must have $1 - \frac{k}{2^n} \leq \varepsilon$. This implies $k \geq (1 - \varepsilon)2^n$, so for constant error we therefore need $k = \Omega(2^N)$ queries to solve the problem.

In contrast, Grovers algorithm will solve the problem using $O(\sqrt{2^n})$ queries.

Description of Grover's algorithm

Grover's algorithm is remarkably simple to describe. We will need to use the following two unitary transformations on n qubits:

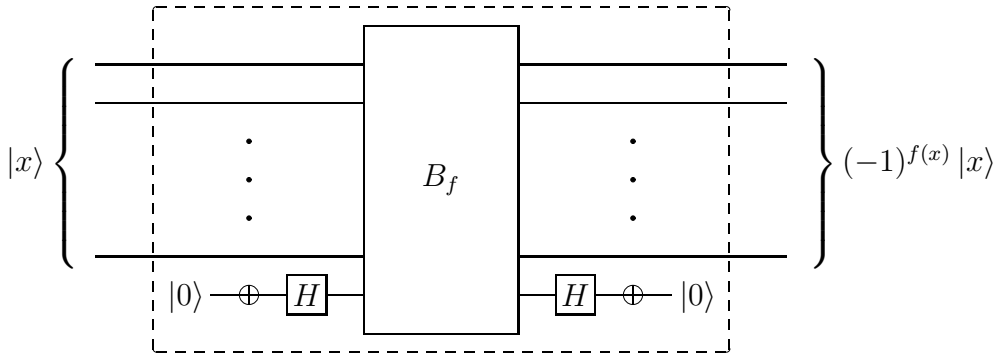
$$Z_f |x\rangle = (-1)^{f(x)} |x\rangle$$

and

$$Z_0 |x\rangle = \begin{cases} -|x\rangle & \text{if } x = 0^n \\ |x\rangle & \text{if } x \neq 0^n \end{cases}$$

for each $x \in \{0,1\}^n$.

Given the black-box transformation B_f, we can implement Z_f using a single ancillary qubit using the phase kick-back phenomenon:



Just one query to B_f is therefore required to implement Z_f.

The Z_0 transformation can be implemented in precisely the same way, except replacing the B_f transformation by a reversible circuit (possibly using additional ancillary qubits) for computing the transformation

$$|x\rangle |a\rangle \mapsto |x\rangle |a \oplus (\neg x_1 \wedge \cdots \wedge \neg x_n)\rangle .$$

Such a circuit can be implemented using a classical Boolean circuit for computing $\neg x_1 \wedge \cdots \wedge \neg x_n$, along with the method we studied in Lecture 7 for constructing reversible circuits. Obviously no queries to B_f are required to do this: Z_0 has nothing to do with f.

Now we are ready to describe the quantum part of the algorithm. Similar to Shor's algorithm and Simon's algorithm, we can view that there will be some classical post-processing after this algorithm is run, possibly several times.

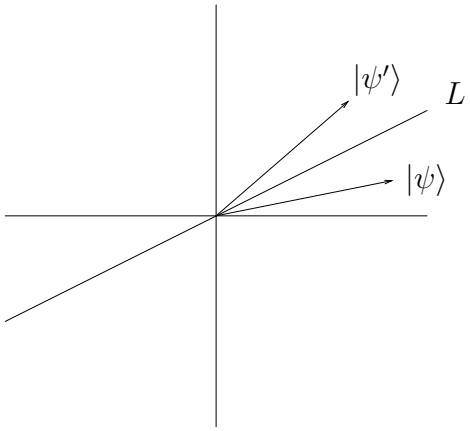
Grover's Algorithm

1. Let X be an n-qubit quantum register (i.e., a collection of n qubits to which we assign the name X). Let the starting state of X be $|0^n\rangle$ and perform $H^{\otimes n}$ on X.
2. Apply to the register X the transformation
$$G = -H^{\otimes n} Z_0 H^{\otimes n} Z_f$$
k times (where k will be specified later).
3. Measure X and output the result.

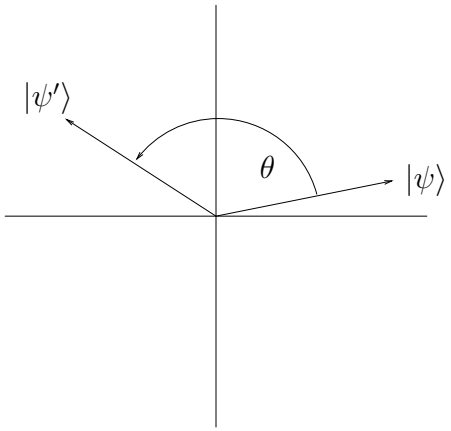
Analysis of Grover's algorithm

As you would imagine, analyzing Grover's algorithm is more difficult than describing it. When analyzing it, it will help to think about *reflections* and *rotations* in the plane.

Here is a *reflection* of a vector $|\psi\rangle$ about the line L:

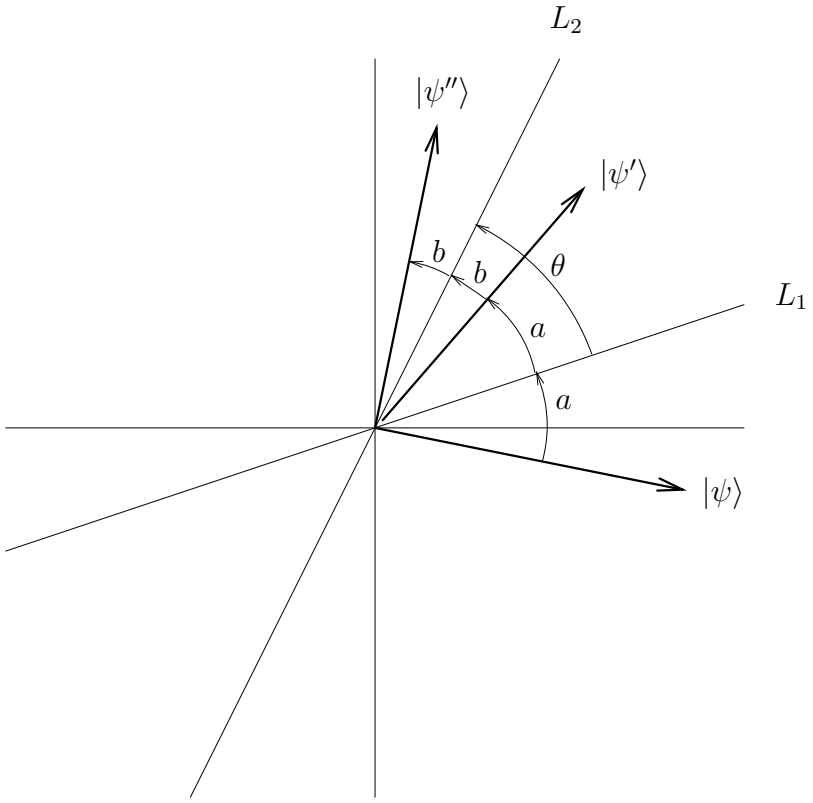


Rotations are about the origin, by some angle:



The effect of a given reflection and rotation might be the same for a particular vector $|\psi\rangle$, but they are always different transformations (e.g., reflections have determinant -1 while rotations have determinant 1).

Now, suppose we have two lines L_1 and L_2, and the angle between them is θ. Then for any choice of $|\psi\rangle$ if you first reflect $|\psi\rangle$ about L_1 and then reflect about L_2, the effect will be the same as rotating by an angle 2θ:



Why do we care about rotations and reflections? The answer is that

$$H^{\otimes n} Z_0 H^{\otimes n} \quad \text{and} \quad -Z_f$$

may be viewed as reflections, so

$$G = -H^{\otimes n} Z_0 H^{\otimes n} Z_f$$

is a rotation by a particular angle. This rotation will happen in the plane defined by two vectors that we will see shortly.

Define sets of strings A and B as follows:

$$A = \{x \in \{0,1\}^n \,:\, f(x) = 1\}$$
$$B = \{x \in \{0,1\}^n \,:\, f(x) = 0\}\,.$$

We will think if the set A as the set of "good" strings $x \in \{0,1\}^n$; the goal of the algorithm is to find one of these strings. The set B contains all of the "bad"strings $x \in \{0,1\}^n$ that do not satisfy the search criterion. Let

$$a = |A| \quad \text{and} \quad b = |B| .$$

The case where either a or b is zero will be handled as an easy special case, so assume for now that $a \neq 0$ and $b \neq 0$. Although the analysis will be completely general, you might imagine that an interesting case of the problem is where a is very small (perhaps $a = 1$) and therefore b is large (close to $N = 2^n$).

Next, define

$$|A\rangle = \frac{1}{\sqrt{a}} \sum_{x \in A} |x\rangle$$

$$|B\rangle = \frac{1}{\sqrt{b}} \sum_{x \in B} |x\rangle .$$

Notice that $|A\rangle$ and $|B\rangle$ are orthogonal unit vectors. What we will show is that the rotation mentioned above will occur in the space spanned by $|A\rangle$ and $|B\rangle$. This will simplify the analysis greatly because immediately before and after each application of G in the algorithm, the state of X will have the form $\alpha |A\rangle + \beta |B\rangle$ for α and β that will happen to be real numbers.

The state of the register X immediately after step 1 in the algorithm is given by

$$|h\rangle \stackrel{\text{def}}{=} H^{\otimes n} |0^n\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle .$$

We can write

$$|h\rangle = \sqrt{\frac{a}{N}} |A\rangle + \sqrt{\frac{b}{N}} |B\rangle ,$$

so the state of X before step 2 is a vector in the subspace spanned by $\{|A\rangle , |B\rangle\}$.

Next let us see how G affects states in the span of $|A\rangle$ and $|B\rangle$. In order to do this it will help to express Z_0 as follows:

$$Z_0 = I - 2 \, |0^n\rangle \langle 0^n| .$$

(If you forgot what vectors that look like $\langle\psi|$ mean, please refer back to the first couple of lectures of the course.) This means that

$$H^{\otimes n} Z_0 H^{\otimes n} = H^{\otimes n} (I - 2 \, |0^n\rangle \langle 0^n|) H^{\otimes n} = I - 2 \, |h\rangle \langle h| .$$

Here we are using two facts: (i) that $H^\dagger = H$, and (ii) that if $|\psi\rangle = U |\phi\rangle$ then $\langle\psi| = \langle\phi| U^\dagger$.

Now the action of G on elements in the space spanned by $\{|A\rangle , |B\rangle\}$ can be determined by

examining its effect on $|A\rangle$ and $|B\rangle$:

$$\begin{aligned}
G|A\rangle &= -H^{\otimes n} Z_0 H^{\otimes n} Z_f |A\rangle \\
&= (I - 2|h\rangle\langle h|)(-Z_f)|A\rangle \\
&= (I - 2|h\rangle\langle h|)|A\rangle \\
&= |A\rangle - 2\langle h|A\rangle |h\rangle \\
&= |A\rangle - 2\sqrt{\frac{a}{N}}\left(\sqrt{\frac{a}{N}}|A\rangle + \sqrt{\frac{b}{N}}|B\rangle\right) \\
&= \left(1 - \frac{2a}{N}\right)|A\rangle - \frac{2\sqrt{ab}}{N}|B\rangle
\end{aligned}$$

and

$$\begin{aligned}
G|B\rangle &= -H^{\otimes n} Z_0 H^{\otimes n} Z_f |B\rangle \\
&= -(I - 2|h\rangle\langle h|)Z_f|B\rangle \\
&= -(I - 2|h\rangle\langle h|)|B\rangle \\
&= -|B\rangle + 2\langle h|B\rangle |h\rangle \\
&= -|B\rangle + 2\sqrt{\frac{b}{N}}\left(\sqrt{\frac{a}{N}}|A\rangle + \sqrt{\frac{b}{N}}|B\rangle\right) \\
&= \frac{2\sqrt{ab}}{N}|A\rangle - \left(1 - \frac{2b}{N}\right)|B\rangle .
\end{aligned}$$

Indeed, we have that G maps the subspace spanned by $\{|A\rangle, |B\rangle\}$ to itself.

We will determine in the next lecture that this action is in fact a rotation by some angle as we have speculated, and determine exactly what the angle is. Once this is done, the analysis of the algorithm will be fairly simple.

Lecture 13: Grover's Algorithm (continued)

March 9, 2006

In the previous lecture we stated Grover's Algorithm and began analyzing it. Today we will complete this analysis and discuss how the algorithm can be used to solve various searching problems. For convenience, here is the algorithm again:

Grover's Algorithm

1. Let X be an n-qubit quantum register with initial state $|0^n\rangle$. Perform $H^{\otimes n}$ on X.
2. Apply to the register X the transformation
$$G = -H^{\otimes n} Z_0 H^{\otimes n} Z_f$$
k times (where k will be specified later).
3. Measure X and output the result.

The operations Z_f and Z_0 are defined as

$$Z_f |x\rangle = (-1)^{f(x)} |x\rangle ,$$

where $f : \{0,1\}^n \to \{0,1\}$ is the function being considered and for which we have a black box B_f, and

$$Z_0 |x\rangle = \begin{cases} -|x\rangle & \text{if } x = 0^n \\ |x\rangle & \text{if } x \neq 0^n. \end{cases}$$

The algorithm requires k queries to the black box B_f for whatever value of k we choose in step 2.

For the analysis, we had defined sets A and B as follows:

$$A = \{x \in \{0,1\}^n \,:\, f(x) = 1\} ,$$
$$B = \{x \in \{0,1\}^n \,:\, f(x) = 0\} ,$$

and let $a = |A|$ and $b = |B|$. The case where either $a = 0$ or $b = 0$ turns out to be an easy special case, so assumed for the time being that $a \neq 0$ and $b \neq 0$. We also defined states

$$|A\rangle = \frac{1}{\sqrt{a}} \sum_{x \in A} |x\rangle \quad \text{and} \quad |B\rangle = \frac{1}{\sqrt{b}} \sum_{x \in B} |x\rangle ,$$

which are both unit vectors and are orthogonal to one another. Because the state of register X immediately after step 1 in the algorithm is given by

$$|h\rangle \stackrel{\text{def}}{=} H^{\otimes n} |0^n\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle ,$$

we can write

$$|h\rangle = \sqrt{\frac{a}{N}}\,|A\rangle + \sqrt{\frac{b}{N}}\,|B\rangle\,.$$

Thus, the state of X before step 2 is a vector in the subspace spanned by $\{|A\rangle\,, |B\rangle\}$. The last thing we did was to determine the effect of the operation G on $|A\rangle$ and $|B\rangle$:

$$G\,|A\rangle = \left(1 - \frac{2a}{N}\right)|A\rangle - \frac{2\sqrt{ab}}{N}\,|B\rangle$$
$$G\,|B\rangle = \frac{2\sqrt{ab}}{N}\,|A\rangle - \left(1 - \frac{2b}{N}\right)|B\rangle\,.$$

This implies that the state of X after each application of G will remain in the subspace spanned by $|A\rangle$ and $|B\rangle$.

Now, we can express the action of G on the space spanned by $\{|A\rangle\,, |B\rangle\}$ as a matrix:

$$M = \begin{bmatrix} -\left(1 - \frac{2b}{N}\right) & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & \left(1 - \frac{2a}{N}\right) \end{bmatrix} = \begin{bmatrix} \frac{b-a}{N} & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & \frac{b-a}{N} \end{bmatrix}.$$

Here, the first row and column correspond to $|B\rangle$ and the second to $|A\rangle$. Notice that

$$\begin{bmatrix} \sqrt{\frac{b}{N}} & -\sqrt{\frac{a}{N}} \\ \sqrt{\frac{a}{N}} & \sqrt{\frac{b}{N}} \end{bmatrix}^2 = \begin{bmatrix} \frac{b-a}{N} & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & \frac{b-a}{N} \end{bmatrix} = M,$$

which shows that M is a rotation as we originally hypothesized. Specifically, if $\theta \in (0, \pi/2)$ is the angle that satisfies

$$\sin\theta = \sqrt{\frac{a}{N}} \quad \text{and} \quad \cos\theta = \sqrt{\frac{b}{N}}$$

then

$$R_{2\theta} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^2 = \begin{bmatrix} \sqrt{\frac{b}{N}} & -\sqrt{\frac{a}{N}} \\ \sqrt{\frac{a}{N}} & \sqrt{\frac{b}{N}} \end{bmatrix}^2 = M.$$

In other words, G causes a rotation by an angle 2θ in the space spanned by $\{|A\rangle\,, |B\rangle\}$ for

$$\theta = \sin^{-1}\sqrt{\frac{a}{N}}.$$

As stated above, the register X is in the state

$$|h\rangle = \sqrt{\frac{b}{N}}\,|B\rangle + \sqrt{\frac{a}{N}}\,|A\rangle = \cos\theta\,|B\rangle + \sin\theta\,|A\rangle$$

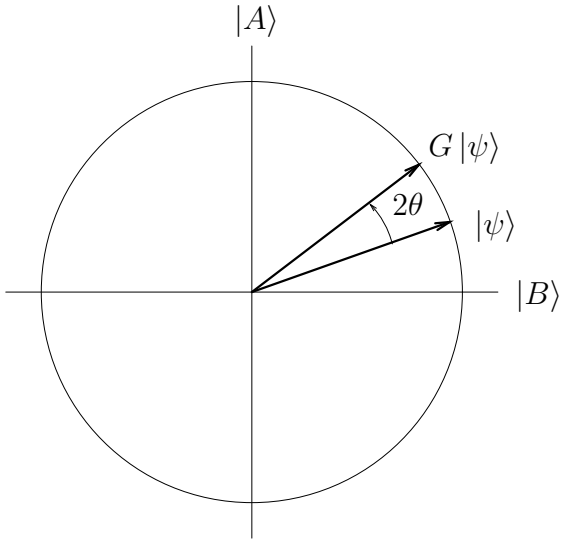


Figure 3: One iteration of G induces a rotation by an angle 2θ.

immediately after step 1 is performed. After k iterations of G, the state will be

$$\cos((2k+1)\theta)\,|B\rangle + \sin((2k+1)\theta)\,|A\rangle\,.$$

The goal is to measure some element $x \in A$, so we would like the state of X to be as close to $|A\rangle$ as possible. If we want

$$\sin((2k+1)\theta) \approx 1$$

then

$$(2k+1)\theta \approx \frac{\pi}{2}$$

will suffice, so we should choose

$$k \approx \frac{\pi}{4\theta} - \frac{1}{2}.$$

Of course k must be an integer, which is why we can only hope to approximate this quantity.

Suppose $a = 1$. Then

$$\theta = \sin^{-1}\sqrt{\frac{1}{N}} \approx \frac{1}{\sqrt{N}}$$

so

$$k = \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor$$

is a reasonable choice for our algorithm. Remember that k is also the number of queries to B_f needed by the algorithm, so we are apparently on the right track to proving that the algorithm needs $O(\sqrt{N})$ queries. Although the case $a = 1$ only represents a special case, it is clearly an interesting special case. With the above choice of k, the probability of finding the single x such that $f(x) = 1$ is

$$\sin^2\left(\left(2\left\lfloor \pi\sqrt{N}/4 \right\rfloor + 1\right)\sin^{-1}\left(1/\sqrt{N}\right)\right).$$

The limit of this probability is 1 as N goes to infinity, and it is always at least $1/2$. Here are some sample values:

N	success prob.
2	.5
4	1.0
8	.9453125
16	.9613190
32	.9991823
64	.9965857
128	.9956199
256	.9999470
512	.9994480
1024	.9994612
2048	.9999968
4096	.9999453

So, even in the worst case, repeating the algorithm some small constant number of times and evaluating f at the output each time will find the unique x such that $f(x) = 1$ with very high probability.

In the general case when we do not know that $a = 1$, the situation is more challenging. For instance, if $a = 4$ but we still choose

$$k = \left\lfloor \pi\sqrt{N}/4 \right\rfloor,$$

the success probability goes to 0 in the limit of large N. This is because we are effectively rotating twice as far as we should. There are different strategies for dealing with this problem. One possibility is simply to choose a *random* value

$$k \in \left\{1, \ldots, \sqrt{N} + 1\right\}.$$

Still operating under the assumption that $a \geq 1$, the algorithm will fail to find an x with $f(x) = 1$ with probability at most $3/4$. Repeating some constant number of times until a "good" x is found quickly decreases error.

A somewhat better strategy is to apply the method above for successively larger values. Specifically, instead of choosing a random k in the range $1, \ldots, \sqrt{N} + 1$, do the following:

1. Set $m = 1$.
2. Choose $k \in \{1, \ldots, m + 1\}$ uniformly and run Grover's algorithm for this choice of k. If the algorithm finds an x such that $f(x) = 1$, then output x and halt.
3. If $m > \sqrt{N}$ then "fail". Else, set $m = \lfloor (8/7)m \rfloor$ and goto step 2.

This will succeed in finding $x \in A$ with probability at least 1/4 after

$$O(\sqrt{N/a})$$

queries. By repeating step 2 a constant number of times during each iteration of the loop, the error decreases exponentially. (The number 8/7 just happens to be a number that works for the analysis, which we will not discuss. The point is that m increases exponentially, but not too fast to cause the algorithm to fail.)

Finally, we still need to deal with the special cases where $a = 0$ or $b = 0$. Obviously if $a = 0$, the output x of the algorithm will never satisfy $f(x) = 1$ because there are no such values of x. It is easy to check directly that Grover's Algorithm will output a choice of $x \in \{0, 1\}^n$ that is uniformly distributed in this case. Supposing that we do not know a, if we run Grover's Algorithm as described above and always measure a value of x for which $f(x) = 0$, we can reasonably conclude with high confidence that $a = 0$.

It is also the case when $b = 0$ that Grover's Algorithm will output a uniformly distributed choice of x. Of course in this case $f(x) = 1$ for all x, so the problem is trivially solved.

This represents the end of our discussion of Grover's algorithm, and of quantum algorithms in general. The next topics coming up in the course are quantum error correction and quantum cryptography, with a short discussion of a more general mathematical description of quantum information than the one we have been using coming first.

Lecture 14: Quantum information revisited

March 14, 2006

So far, this course has focused almost entirely on quantum algorithms. The next topics to be discussed will be of a somewhat different flavor, starting with quantum error correction and moving on to quantum cryptography. Before discussing these topics, however, it will be helpful to say more about the mathematics of quantum information in general.

In the beginning of the course I described the model of quantum information that we have used up to this point. Specifically, the model describes states of qubits as unit vectors, and the allowed operations come from a fairly restricted set (unitary operations and measurements of a simple type). This description was sufficient for discussing quantum algorithms, so it has not been necessary to extend it until now.

To describe the more general model of quantum information, some notation will be helpful. First, for any finite, nonempty set Σ, let $\mathbb{C}(\Sigma)$ denote the vector space of all column vectors indexed by Σ. Just as before, elements of such spaces are denoted by kets, e.g., $|\psi\rangle \in \mathbb{C}(\Sigma)$. It will be typical and often helpful to assign specific names to spaces of the form $\mathbb{C}(\Sigma)$, and scripted letters such as $\mathcal{A}$, $\mathcal{B}$, $\mathcal{X}$, $\mathcal{Y}$, etc., are generally used. For example, we might write $\mathcal{X} = \mathbb{C}(\{0,1\}^n)$ to indicate that the space $\mathcal{X}$ is indexed by the set $\{0,1\}^n$, and simply use the symbol $\mathcal{X}$ to refer to that space from that point on. Although it will seldom be necessary, we may also write $\mathbb{C}(\Sigma)^\dagger$ or $\mathcal{X}^\dagger$ (for example) to refer to the space of all row vectors (or bra vectors) $\langle\psi|$, for $|\psi\rangle \in \mathbb{C}(\Sigma)$ or $|\psi\rangle \in \mathcal{X}$. You will also see these things written with a star $*$ instead of a dagger $\dagger$ sometimes, as in $\mathcal{X}^*$.

Also similar to before, when we consider a particular quantum system we assume that it has some finite set Σ of associated *classical states*. We will typically use the term *register* from now on to refer to abstract physical devices (such as qubits or collections of qubits). Associated with any register having classical state set Σ is the vector space $\mathbb{C}(\Sigma)$. It is sometimes helpful, but certainly not necessary, to use the same letter (in different fonts) to refer to a register and its associated space. For example, register X may have associated space $\mathcal{X}$.

Density matrices

So far nothing is new except for a little bit of notation. In order to describe what really is new, it is helpful to start with a special case. Suppose that in the "old" representation, a register X having classical state set Σ is in a quantum state $|\psi\rangle \in \mathcal{X}$ for $\mathcal{X} = \mathbb{C}(\Sigma)$. The "new" way of representing this state will be:

$$|\psi\rangle\langle\psi|.$$

We have seen objects like this before (in the analysis of Grover's algorithm. It is effectively a matrix. For example, suppose $\Sigma = \{0, 1\}$ and $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$. Then

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad \text{and} \quad \langle\psi| = \begin{pmatrix} \overline{\alpha} & \overline{\beta} \end{pmatrix}$$

so

$$|\psi\rangle \langle\psi| = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \begin{pmatrix} \overline{\alpha} & \overline{\beta} \end{pmatrix} = \begin{pmatrix} \alpha\overline{\alpha} & \alpha\overline{\beta} \\ \overline{\alpha}\beta & \beta\overline{\beta} \end{pmatrix} = \begin{pmatrix} |\alpha|^2 & \alpha\overline{\beta} \\ \overline{\alpha}\beta & |\beta|^2 \end{pmatrix}.$$

This is called a *density matrix* or *density operator*. (The term *operator* usually just means a linear map from a space to itself.) Not all density operators have the special form $|\psi\rangle \langle\psi|$ for some unit vector $|\psi\rangle$, as we will see. In this special case, the density matrix describes what is called a *pure state*. All of the quantum states we have considered so far in the course have actually been of this special kind of state.

Suppose we have a probability distribution $(p_1, \ldots, p_k)$, for some positive integer k, as well as unit vectors $|\psi_1\rangle, \ldots, |\psi_k\rangle \in \mathcal{X}$, where $\mathcal{X} = \mathbb{C}(\Sigma)$ is the space corresponding to some register X. Somebody randomly chooses $j \in \{1, \ldots, k\}$ according to the probability distribution $(p_1, \ldots, p_k)$, prepares the register X in the state $|\psi_j\rangle$ for the chosen j, and then hands you X without telling you the value of j. The collection $\{(p_1, |\psi_1\rangle), \ldots, (p_k, |\psi_k\rangle)\}$ that describes the different possible states $|\psi_j\rangle$ along with their associated probabilities is called a *mixture*. How do you represent your knowledge of the state of X? In the "old" representation, there is no convenient way to do this beyond specifying the mixture. In the "new" representation, the density matrix corresponding to the above mixture is:

$$\sum_{j=1}^{k} p_k |\psi_j\rangle \langle\psi_j|.$$

In other words, it is meaningful to take a weighted average of the pure states $|\psi_j\rangle \langle\psi_j|$.

Example 9. Suppose Alice has a qubit A. She flips a fair coin: if the result is HEADS she prepares A in the state $|0\rangle$, and if the result is TAILS she prepares A in the state $|1\rangle$. She gives Bob the qubit without revealing the result of the coin-flip. Bob's knowledge of the qubit is described by the density matrix

$$\frac{1}{2} |0\rangle \langle 0| + \frac{1}{2} |1\rangle \langle 1| = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}.$$

Example 10. Suppose Alice has a qubit A as in the previous example. As before she flips a fair coin, but now if the result is HEADS she prepares A in the state $|+\rangle = \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$, and if the result is TAILS she prepares A in the state $|-\rangle = \frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle$. Bob's knowledge of the qubit is now described by the density matrix

$$\frac{1}{2} |+\rangle \langle +| + \frac{1}{2} |-\rangle \langle -| = \frac{1}{2} \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} + \frac{1}{2} \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}.$$

Same as before!

The previous two examples demonstrate that different mixtures can yield precisely the same density matrix. This is not an accident, but rather is one of the strengths of the density matrix formalism. A given density matrix in essence represents a perfect description of the state of a quantum system—two different mixtures can be distinguished (in a statistical sense) if and only if they yield different density matrices.

Typically, lower case Greek letters such as ρ, ξ, σ, and τ are used to denote density matrices. The state of a system that corresponds to a given density matrix is called a ***mixed state***.

Some important facts about density matrices:

1. **The trace of a density matrix is 1.**

   (The trace of a matrix is the sum of its diagonal entries.)

   In fact, the diagonal entries describe precisely the probability distribution that would result if the system in question were measured (with respect to the restricted type of measurement we have so far considered).

   The fact that the trace is a linear function together with the useful formula $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$ (which holds for any choice of matrices A and B for which the product AB is square) makes the above fact easy to verify for any given mixture:

$$\mathrm{Tr}\left(\sum_{j=1}^{k} p_k \left|\psi_j\right\rangle \left\langle\psi_j\right|\right) = \sum_{j=1}^{k} p_k \,\mathrm{Tr}\left(\left|\psi_j\right\rangle \left\langle\psi_j\right|\right) = \sum_{j=1}^{k} p_k \,\mathrm{Tr}\left(\left\langle\psi_j\right| \left|\psi_j\right\rangle\right) = \sum_{j=1}^{k} p_k \left\langle\psi_j|\psi_j\right\rangle = 1.$$

2. **Every density matrix is *positive semidefinite*.**

   In general, a square matrix A is positive semidefinite if $\langle\psi|A|\psi\rangle$ is a nonnegative real number for every vector $|\psi\rangle$. An equivalent definition is that A is positive semidefinite if (i) $A = A^\dagger$ (i.e., A is Hermitian), and (ii) all eigenvalues of A are nonnegative real numbers.

   Again this condition is easy to check for mixtures: if

$$\rho = \sum_{j=1}^{k} p_k \left|\psi_j\right\rangle \left\langle\psi_j\right|$$

   and $|\phi\rangle$ is any vector, we have

$$\langle\phi|\rho|\phi\rangle = \sum_{j=1}^{k} p_k \left\langle\phi|\psi_j\right\rangle \left\langle\psi_j|\phi\right\rangle = \sum_{j=1}^{k} p_k \left|\left\langle\phi|\psi_j\right\rangle\right|^2 \geq 0.$$

You can interpret the above facts as the *defining* properties of density matrices: by definition, a density matrix is any matrix that is positive semidefinite and has trace equal to 1. Given such a matrix, it is possible to find a mixture that yields the given density matrix by letting $p_1, \ldots, p_k$ be the nonzero eigenvalues and $|\psi_1\rangle, \ldots, |\psi_k\rangle$ a collection of corresponding eigenvectors.

Operations on density matrices

In our "old" description of quantum information, a unitary operation U applied to a state $|\psi\rangle$ resulted in the state $U|\psi\rangle$. In the density matrix description, a unitary operation U applied to a pure state $|\psi\rangle\langle\psi|$ results in the density matrix

$$U|\psi\rangle\langle\psi|U^\dagger.$$

This is consistent with the observation that $(U|\psi\rangle)^\dagger = \langle\psi|U^\dagger$. More generally, applying U to the mixed state ρ results in state $U\rho U^\dagger$. You can easily check that the two required conditions of density matrices are necessarily met by this new matrix.

We can, however, consider a much more general set of possible operations than just unitary operations. Any operation Φ that can be written as

$$\Phi(\rho) = \sum_{j=1}^{k} A_j \rho A_j^\dagger$$

for some collection of matrices $A_1, \ldots, A_k$ satisfying

$$\sum_{j=1}^{k} A_j^\dagger A_j = I$$

represents an operation that can (in an idealized sense) be physically implemented. Such operations are called *admissible operations*. (There are several other names that are used as well, such as *completely positive trace preserving* operations and other variations on these words.)

If ρ is a matrix and Φ is admissible, then $\Phi(\rho)$ is also a density matrix. In fact a somewhat stronger property holds, which is that if Φ is applied to just part of a larger system whose state is described by some density matrix, then the resulting state will also be described by a density matrix.

Example 11. Suppose we have a single qubit X. Consider the operation that corresponds to measuring the qubit and forgetting the result. An admissible operation that describes this process is given by

$$\begin{aligned} A_0 &= |0\rangle\langle 0|, \\ A_1 &= |1\rangle\langle 1|, \\ \Phi(\rho) &= \sum_{j=0}^{1} A_j \rho A_j^\dagger = |0\rangle\langle 0|\rho|0\rangle\langle 0| + |1\rangle\langle 1|\rho|1\rangle\langle 1|. \end{aligned}$$

First let us check that this is a valid admissible operation:

$$\sum_{j=0}^{1} A_j^\dagger A_j = |0\rangle\langle 0|0\rangle\langle 0| + |1\rangle\langle 1|1\rangle\langle 1| = |0\rangle\langle 0| + |1\rangle\langle 1| = I.$$

It satisfies the required property, so indeed it is admissible.

What does it do to the state $\alpha|0\rangle + \beta|1\rangle$, for example? First we need to represent $|\psi\rangle$ as a density matrix: $\rho = |\psi\rangle\langle\psi|$. Now

$$\begin{aligned}\Phi(\rho) &= \Phi(|\psi\rangle\langle\psi|)\\ &= |0\rangle\langle 0|\psi\rangle\langle\psi|0\rangle\langle 0| + |1\rangle\langle 1|\psi\rangle\langle\psi|1\rangle\langle 1|\\ &= |\langle 0|\psi\rangle|^2 |0\rangle\langle 0| + |\langle 1|\psi\rangle|^2 |1\rangle\langle 1|\\ &= |\alpha|^2 |0\rangle\langle 0| + |\beta|^2 |1\rangle\langle 1|.\end{aligned}$$

In terms of matrices:

$$\begin{pmatrix} |\alpha|^2 & \alpha\overline{\beta} \\ \overline{\alpha}\beta & |\beta|^2 \end{pmatrix} \xrightarrow{\Phi} \begin{pmatrix} |\alpha|^2 & 0 \\ 0 & |\beta|^2 \end{pmatrix}.$$

In general, off diagonal entries get zeroed out.

In general, admissible operations do not need to preserve the sizes of quantum systems. For example, one might consider the situation in which one of a collection of qubits is lost or somehow destroyed. More notation will help to discuss this issue in greater specificity.

Given two spaces $\mathcal{X}$ and $\mathcal{Y}$, we let

$$\mathrm{L}(\mathcal{X}, \mathcal{Y})$$

denote the set of all linear mappings from $\mathcal{X}$ to $\mathcal{Y}$. The shorthand $\mathrm{L}(\mathcal{X})$ is used to mean $\mathrm{L}(\mathcal{X}, \mathcal{X})$. Also let

$$\mathrm{D}(\mathcal{X})$$

denote the set of all density matrices on $\mathcal{X}$ (so that $\mathrm{D}(\mathcal{X}) \subset \mathrm{L}(\mathcal{X})$). For example, if X is a quantum register with classical state set Σ and $\mathcal{X} = \mathbb{C}(\Sigma)$, then any mixed state of the register X is represented by some element of $\mathrm{D}(\mathcal{X})$.

Now, if we have a collection of mappings (or matrices) $A_1, \ldots, A_k \in \mathrm{L}(\mathcal{X}, \mathcal{Y})$ that satisfy

$$\sum_{j=1}^{k} A_j^\dagger A_j = I$$

(the identity matrix in $\mathrm{L}(\mathcal{X})$) then the admissible operation Φ defined by

$$\Phi(\rho) = \sum_{j=1}^{k} A_j \rho A_j^\dagger$$

maps elements of $\mathrm{D}(\mathcal{X})$ to elements of $\mathrm{D}(\mathcal{Y})$.

Example 12. Let us suppose that we have two qubits: X and Y. We will consider the admissible operation that corresponds to discarding the second qubit. Thus, once the operation has been performed, we will be left with a single qubit X. The vector space corresponding to X will be $\mathcal{X}$ and the space corresponding to Y will be $\mathcal{Y}$.

Now, if Φ is to describe the operation of discarding the second qubit, then we must have that $\Phi(\rho) \in \mathrm{D}(\mathcal{X})$ whenever $\rho \in \mathrm{D}(\mathcal{X} \otimes \mathcal{Y})$. This means that if

$$\Phi(\rho) = \sum_{j=1}^{k} A_j \rho A_j^\dagger$$

for some choice of matrices $A_1, \ldots, A_k$, then these matrices must come from the set $\mathrm{L}(\mathcal{X} \otimes \mathcal{Y}, \mathcal{X})$. This means that they must be 2×4 matrices.

Let

$$A_0 = I \otimes \langle 0| \,, \quad A_1 = I \otimes \langle 1| \,,$$

where I denotes the identity operator on the first qubit, and define Φ by

$$\Phi(\rho) = \sum_{j=0}^{1} A_j \rho A_j^\dagger = A_0 \rho A_0^\dagger + A_1 \rho A_1^\dagger$$

for all ρ. Writing A_0 and A_1 in ordinary matrix notation gives

$$A_0 = I \otimes \langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

$$A_1 = I \otimes \langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

To see that Φ is admissible, we compute:

$$\begin{aligned} A_0^\dagger A + A_1^\dagger A_1 &= (I \otimes |0\rangle)(I \otimes \langle 0|) + (I \otimes |1\rangle)(I \otimes \langle 1|) \\ &= I \otimes |0\rangle\langle 0| + I \otimes |1\rangle\langle 1| \\ &= I \otimes (|0\rangle\langle 0| + |1\rangle\langle 1|) \\ &= I \otimes I \\ &= I_{\mathcal{X} \otimes \mathcal{Y}}. \end{aligned}$$

(The subscript on the identity operator in the last line is just representing what space the identity is acting on.)

Let us now consider the effect of this operation on a couple of states. First, suppose that (X, Y) is in the state $\xi \otimes \sigma$ for two 2×2 density matrices ξ and σ. Just as for vectors in the "old" description of quantum information, we view states of the form $\xi \otimes \sigma$ as representing completely uncorrelated qubits. What do you expect the state to be if you discard the second qubit? Now let us check:

$$\begin{aligned} \Phi(\xi \otimes \sigma) &= A_0(\xi \otimes \sigma)A_0^\dagger + A_1(\xi \otimes \sigma)A_1^\dagger \\ &= (I \otimes \langle 0|)(\xi \otimes \sigma)(I \otimes |0\rangle) + (I \otimes \langle 1|)(\xi \otimes \sigma)(I \otimes |1\rangle) \\ &= \xi \otimes \langle 0|\sigma|0\rangle + \xi \otimes \langle 1|\sigma|1\rangle \\ &= \left(\langle 0|\sigma|0\rangle + \langle 1|\sigma|1\rangle\right) \, \xi \\ &= \mathrm{Tr}(\sigma) \, \xi \\ &= \xi. \end{aligned}$$

Indeed this is what you presumably expected.

In the next lecture we'll see what happens when this operation is applied to entangled qubits.

The previous example can be generalized to describe the operation that corresponds to discarding part of a system. Suppose X and Y are registers with corresponding spaces $\mathcal{X}$ and $\mathcal{Y}$. A mixed state of these two registers is represented by some element of $\mathrm{D}(\mathcal{X} \otimes \mathcal{Y})$. If, say, the register Y is discarded, we will be left with some mixed state of X that is represented by some element of $\mathrm{D}(\mathcal{X})$. Specifically, if $\rho \in \mathrm{D}(\mathcal{X} \otimes \mathcal{Y})$ is a density matrix representing the state of (X, Y) and Y is discarded, the resulting state of X is denoted $\mathrm{Tr}_{\mathcal{Y}}(\rho)$. We say that this state is the *reduced state* of X, and we call the admissible operation $\mathrm{Tr}_{\mathcal{Y}}$ the *partial trace*. (It is called the partial trace because when extended by linearity to arbitrary matrices it satisfies $\mathrm{Tr}_{\mathcal{Y}}(X \otimes Y) = \mathrm{Tr}(Y)X$ for all $X \in \mathrm{L}(\mathcal{X})$ and $Y \in \mathrm{L}(\mathcal{Y})$.) We also refer to the action corresponding to this operation as *tracing out* the space $\mathcal{Y}$. To express $\mathrm{Tr}_{\mathcal{Y}}$ in the form of an admissible operation, let Σ denote the set of classical states of Y. Then

$$\mathrm{Tr}_{\mathcal{Y}}(\rho) = \sum_{a \in \Sigma} (I \otimes \langle a|)\rho(I \otimes |a\rangle).$$

The partial trace is a particularly important admissible operation that we will continue discussing in the next lecture.

Lecture 15: Quantum information revisited (continued)

March 16, 2006

More on the partial trace

In the previous lecture we were discussing an important admissible operation called the partial trace. We will continue to discuss this operation in the first part of this lecture.

Recall that if we have registers X and Y with corresponding spaces $\mathcal{X}$ and $\mathcal{Y}$, then the admissible operation that describes the action of throwing away Y and leaving X alone is the partial trace (over the space $\mathcal{Y}$). Assuming that $\mathcal{Y} = \mathbb{C}(\Gamma)$ for some arbitrary classical state set Γ, we can express the partial trace over $\mathcal{Y}$ as follows:

$$\mathrm{Tr}_{\mathcal{Y}}\, \rho = \sum_{a \in \Gamma} (I_{\mathcal{X}} \otimes \langle a|)\rho(I_{\mathcal{X}} \otimes |a\rangle).$$

You could replace the vectors $\{|a\rangle \,:\, a \in \Gamma\}$ (and their corresponding bra vectors) by any other orthonormal basis of $\mathcal{Y}$ without changing the operation. Another way of expressing the partial trace, which makes this fact apparent, is $\mathrm{Tr}_{\mathcal{Y}} = I_{\mathrm{L}(\mathcal{X})} \otimes \mathrm{Tr}$ (where $I_{\mathrm{L}(\mathcal{X})}$ refers to the identity operator on $\mathrm{L}(\mathcal{X})$, which is the admissible operation that corresponds to doing nothing at all).

One can define $\mathrm{Tr}_{\mathcal{X}}$, the partial trace over the space $\mathcal{X}$, similarly to $\mathrm{Tr}_{\mathcal{Y}}$. More generally, we could imagine any number of registers and consider the operation that corresponds to discarding some sub-collection of them.

Let us now go back to the simple case of two qubits, and consider what happens when the partial trace is applied to an entangled state. Suppose that the pair of qubits (X, Y) is in the state

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle\,.$$

The corresponding density matrix is therefore $|\phi^+\rangle\langle\phi^+|$. The effect of discarding the second qubit is computed as follows. By the definition of the partial trace, we have

$$\mathrm{Tr}_{\mathcal{Y}}\, |\phi^+\rangle\langle\phi^+| = (I \otimes \langle 0|)\,|\phi^+\rangle\langle\phi^+|\,(I \otimes |0\rangle) + (I \otimes \langle 1|)\,|\phi^+\rangle\langle\phi^+|\,(I \otimes |1\rangle).$$

In order to simplify this, note that

$$(I \otimes \langle 0|)\,|\phi^+\rangle = (I \otimes \langle 0|)\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}\,|0\rangle\,,$$

$$(I \otimes \langle 1|)\,|\phi^+\rangle = (I \otimes \langle 1|)\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}\,|1\rangle\,,$$

and so

$$\mathrm{Tr}_{\mathcal{Y}}\, |\phi^+\rangle\langle\phi^+| = \frac{1}{2}\,|0\rangle\langle 0| + \frac{1}{2}\,|1\rangle\langle 1|\,.$$

By the way, we have seen this density matrix a couple of times now—it is sometimes called the *totally mixed state*, and essentially represents complete ignorance about the qubit in question. Try repeating this sort of calculation for initial states $|\phi^-\rangle$, $|\psi^+\rangle$, and $|\psi^-\rangle$. You will find that in each of these cases the resulting state is also the totally mixed state.

Of course it is not necessary to view the partial trace as representing the *destruction* of one or more quantum registers—we can use the partial trace when we simply wish to consider some part of a larger system in isolation. For instance, if Alice and Bob share some state, and we want to consider the outcomes of any measurements Bob can perform on his part of the shared system alone, then we would consider the state that results from tracing out Alice's part of the system. When this is done, the resulting state is called the *reduced state* of Bob's registers.

Example 13. Alice and Bob each have one qubit, and the two qubits together are known to be in one of the four Bell states:

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle, \quad |\psi^+\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle,$$
$$|\phi^-\rangle = \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle, \quad |\psi^-\rangle = \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle.$$

Let $\mathcal{A}$ and $\mathcal{B}$ be the spaces corresponding to Alice's and Bob's qubit, respectively. Bob wants to determine which of the four Bell states they share without Alice's help. It is impossible for him to gain any information whatsoever about which of the four states they have, however, because

$$\mathrm{Tr}_{\mathcal{A}}|\phi^+\rangle\langle\phi^+| = \mathrm{Tr}_{\mathcal{A}}|\phi^-\rangle\langle\phi^-| = \mathrm{Tr}_{\mathcal{A}}|\psi^+\rangle\langle\psi^+| = \mathrm{Tr}_{\mathcal{A}}|\psi^-\rangle\langle\psi^-| = \frac{1}{2}I.$$

In all four cases, Bob's reduced state is the same, so there will be no difference in the distribution of outcomes Bob would get using any measurement whatsoever.

On the other hand, if Alice and Bob shared one of two states for which Bob's reduced state was *different* for the two states, then he could gain some information about which state they share by making some well-chosen measurement.

Example 14. Alice and Bob again each have one qubit. This time they share one of the two states

$$|\psi_1\rangle = \frac{3}{5}|00\rangle + \frac{4}{5}|11\rangle \quad \text{or} \quad |\psi_2\rangle = \frac{4}{5}|00\rangle - \frac{3}{5}|11\rangle.$$

The corresponding reduced states for Bob are

$$\rho_1 = \mathrm{Tr}_{\mathcal{A}}|\psi_1\rangle\langle\psi_1| = \frac{9}{25}|0\rangle\langle 0| + \frac{16}{25}|1\rangle\langle 1| \quad \text{and} \quad \rho_2 = \mathrm{Tr}_{\mathcal{A}}|\psi_2\rangle\langle\psi_2| = \frac{16}{25}|0\rangle\langle 0| + \frac{9}{25}|1\rangle\langle 1|.$$

These density matrices are different, so there is some measurement that Bob could make that would give him some information about which state they share. If Bob measures with respect to the standard basis, he could guess that they shared $|\psi_1\rangle$ if the measurement result was 1 and guess $|\psi_2\rangle$ if the measurement result was 0. He would be right 64% of the time, implying a gain of partial information about which state it was.

Purifications

Suppose that X is a register with corresponding space $\mathcal{X} = \mathbb{C}(\Sigma)$, and let $\rho \in \mathrm{D}(\mathcal{X})$ be a mixed state of X. Sometimes it is helpful to view ρ as being the reduced state of some pure state of the pair (X, Y), where Y is some other register. It is always possible to do this so long as the space $\mathcal{Y}$ corresponding to the register Y has dimension at least that of $\mathcal{X}$. (Really, as long as the dimension of $\mathcal{Y}$ is as large as the rank of ρ, this will be possible.) One way to see this is to consider a *spectral decomposition* of ρ:

$$\rho = \sum_{i=1}^{n} p_i \left|\psi_i\right\rangle \left\langle\psi_i\right|,$$

where $n = |\Sigma|$, $(p_1, \ldots, p_n)$ is a probability vector, and $\{|\psi_1\rangle, \ldots, |\psi_n\rangle\}$ is an orthonormal basis of $\mathcal{X}$. Every density matrix has such a decomposition. For $\mathcal{Y} = \mathcal{X}$, define a unit vector $|\phi\rangle \in \mathcal{X} \otimes \mathcal{Y}$ as follows:

$$|\phi\rangle = \sum_{i=1}^{n} \sqrt{p_i} \left|\psi_i\right\rangle \left|\psi_i\right\rangle.$$

Then $\mathrm{Tr}_{\mathcal{Y}} |\phi\rangle \langle\phi| = \rho$ (which you should check for yourself). The vector $|\phi\rangle$ (or the state $|\phi\rangle \langle\phi|$) is called a *purification* of ρ. Another purification is

$$|\phi\rangle = \sum_{i=1}^{n} \sqrt{p_i} \left|\psi_i\right\rangle |i\rangle$$

for $\mathcal{Y} = \mathbb{C}^n$, which has the same dimension as $\mathcal{X}$.

The Schmidt decomposition

By considering reduced states of bipartite systems, we can learn something interesting about pure states of those systems.

Suppose that $|\phi\rangle \in \mathcal{X} \otimes \mathcal{Y}$ is a pure quantum state of registers (X, Y), and let $\rho \in \mathrm{D}(\mathcal{X})$ denote the reduced state of X:

$$\rho = \mathrm{Tr}_{\mathcal{Y}} |\psi\rangle \langle\psi| \in \mathrm{D}(\mathcal{X}).$$

Like all density matrices, ρ is positive semidefinite and has trace equal to 1. It is therefore possible to write ρ as

$$\rho = \sum_{j=1}^{n} p_j \left|\mu_j\right\rangle \left\langle\mu_j\right|$$

for some orthonormal basis $\{|\mu_1\rangle, \ldots, |\mu_n\rangle\}$ of $\mathcal{X}$ and some probability vector $(p_1, \ldots, p_n)$. The vectors $|\mu_1\rangle, \ldots, |\mu_n\rangle$ are eigenvectors of ρ, and $p_1, \ldots, p_n$ are the associated eigenvalues. (We know the eigenvalues are all nonnegative because ρ is positive semidefinite, and they sum to 1 and therefore form a probability distribution because ρ has trace equal to 1.)

Suppose now that we write

$$|\psi\rangle = \sum_{j=1}^{n} \left|\mu_j\right\rangle \left|\nu_j\right\rangle$$

for some choice of vectors $|\nu_1\rangle, \ldots, |\nu_n\rangle \in \mathcal{Y}$. This is always possible, following from the fact that $\{|\mu_1\rangle, \ldots, |\mu_n\rangle\}$ is a basis of $\mathcal{X}$. The vectors $|\nu_1\rangle, \ldots, |\nu_n\rangle$ are uniquely determined given that the basis $\{|\mu_1\rangle, \ldots, |\mu_n\rangle\}$ has been fixed: it must be that $|\nu_j\rangle = (\langle\mu_j| \otimes I)|\psi\rangle$. A quick calculation now shows that because

$$\mathrm{Tr}_{\mathcal{Y}} |\psi\rangle\langle\psi| = \rho = \sum_{j=1}^{n} p_j |\mu_j\rangle\langle\mu_j|,$$

it must be the case that

$$\langle\nu_i|\nu_j\rangle = \begin{cases} p_i & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Let us throw away all of the vectors for which $p_j = 0$ and re-number the remaining ones so that

$$\rho = \sum_{j=1}^{k} p_j |\mu_j\rangle\langle\mu_j|$$

for $p_1, \ldots, p_k \neq 0$. We now see that

$$|\psi\rangle = \sum_{j=1}^{k} |\mu_j\rangle|\nu_j\rangle$$

for $\{|\nu_1\rangle, \ldots, |\nu_k\rangle\}$ an orthogonal set of nonzero vectors. By normalizing these vectors, i.e., letting $|\gamma_j\rangle = |\nu_j\rangle / \sqrt{p_j}$ for each $j = 1, \ldots, k$, we have that

$$|\psi\rangle = \sum_{j=1}^{k} \sqrt{p_j} |\mu_j\rangle|\gamma_j\rangle$$

for both $\{|\mu_1\rangle, \ldots, |\mu_k\rangle\}$ and $\{|\gamma_1\rangle, \ldots, |\gamma_k\rangle\}$ *orthonormal* sets.

An expression of $|\psi\rangle$ in this form is known as a *Schmidt decomposition*. (It is really only a cosmetic variant of the *singular value decomposition* of matrices, which was not discovered by Schmidt. Nevertheless, people use the term Schmidt decomposition, so that is what we will use.)

You may wonder why this is interesting. It turns out to be an incredibly useful fact, and we will see later some uses for it. Here is one consequence of the above proof.

Fact. Suppose $|\phi\rangle, |\psi\rangle \in \mathcal{X} \otimes \mathcal{Y}$ satisfy $\mathrm{Tr}_{\mathcal{Y}} |\phi\rangle\langle\phi| = \mathrm{Tr}_{\mathcal{Y}} |\psi\rangle\langle\psi|$. Then there exists a unitary operator $U \in \mathrm{L}(\mathcal{Y})$ such that $(I_{\mathcal{X}} \otimes U)|\phi\rangle = |\psi\rangle$.

To see that this fact holds, consider performing the previous construction of the Schmidt decomposition for two pure states $|\phi\rangle$ and $|\psi\rangle$ satisfying $\mathrm{Tr}_{\mathcal{Y}} |\phi\rangle\langle\phi| = \mathrm{Tr}_{\mathcal{Y}} |\psi\rangle\langle\psi|$. Because the reduced state of register X is the same in both cases, you could take $\{|\mu_1\rangle, \ldots, |\mu_n\rangle\}$ and $(p_1, \ldots, p_n)$ to be identical for the two cases. Following through with the construction, you would then have

$$|\phi\rangle = \sum_{j=1}^{k} \sqrt{p_j} |\mu_j\rangle|\gamma_j\rangle,$$

$$|\psi\rangle = \sum_{j=1}^{k} \sqrt{p_j} |\mu_j\rangle|\gamma'_j\rangle,$$

for two (possibly different) orthonormal sets $\{|\gamma_1\rangle, \ldots, |\gamma_k\rangle\}$ and $\{|\gamma'_1\rangle, \ldots, |\gamma'_k\rangle\}$. There is always at least one unitary transformation U mapping the first set to the second, given that both sets are orthonormal.

It happens to be the case that there are also approximate versions of the above fact, but because we have not discussed meaningful distance measures for quantum states it will not be possible to go into greater detail about this. Expressed informally, the approximate versions are of this form: if

$$\mathrm{Tr}_{\mathcal{Y}} |\phi\rangle \langle\phi| \approx \mathrm{Tr}_{\mathcal{Y}} |\psi\rangle \langle\psi|$$

then there exists a unitary operator $U \in \mathrm{L}(\mathcal{Y})$ such that $(I \otimes U) |\phi\rangle \approx |\psi\rangle$.

Example 15. It was claimed above that all four Bell states result in the totally mixed state when one of the two qubits is traced out:

$$\mathrm{Tr}_{\mathcal{A}} |\phi^+\rangle \langle\phi^+| = \mathrm{Tr}_{\mathcal{A}} |\phi^-\rangle \langle\phi^-| = \mathrm{Tr}_{\mathcal{A}} |\psi^+\rangle \langle\psi^+| = \mathrm{Tr}_{\mathcal{A}} |\psi^-\rangle \langle\psi^-| = \frac{1}{2} I$$

if the spaces are $\mathcal{A}$ and $\mathcal{B}$. This means that they are all purifications of the totally mixed state. It follows that for any two of them, there is a unitary operator acting just on Alice's qubit that maps the first to the second. This is essentially what is happening in super-dense coding—Alice decides to transform $|\phi^+\rangle$ to one of the four Bell states depending on her two classical input bits, sends her qubit to Bob, and he performs a measurement to determine which of the four states it is to determine Alice's classical bits.

Measurements

Finally, the last remaining piece of the general formalism of quantum information is measurements. There are more general types of measurements than the simple type we have discussed previously. Mathematically speaking, they are described in a manner similar to admissible operations.

Suppose that X is a quantum register with classical state set Σ and corresponding vector space $\mathcal{X} = \mathbb{C}(\Sigma)$. A *measurement* on X can have any finite, nonempty set Γ of possible outcomes. Formally, a measurement is described by a collection

$$\{M_a : a \in \Gamma\} \subset \mathrm{L}(\mathcal{X})$$

of matrices that satisfies

$$\sum_{a \in \Gamma} M_a^\dagger M_a = I.$$

If the register X is measured with respect to this measurement while it is in the state $\rho \in \mathrm{D}(\mathcal{X})$, the outcome must be some element of Γ. As before, the outcome may be random: for each $a \in \Gamma$, the outcome will be a with probability

$$\mathrm{Tr}\left(M_a \rho M_a^\dagger\right).$$

Conditioned on any given outcome a, the density matrix describing the state of X then becomes

$$\frac{M_a \rho M_a^\dagger}{\mathrm{Tr}\left(M_a \rho M_a^\dagger\right)}.$$

Example 16. Suppose X is an n qubit register, so that $\mathcal{X} = \mathbb{C}(\{0,1\}^n)$. Measuring the first qubit with respect to the "old" notion of measurement is now described by the set $\{M_0, M_1\}$ where

$$M_0 = |0\rangle\langle 0| \otimes I \quad \text{and} \quad M_1 = |1\rangle\langle 1| \otimes I.$$

In the special case where each of the matrices M_a of a measurement $\{M_a : a \in \Gamma\}$ is a *projection*, the measurement is said to be a *projective measurement* (or *von Neumann measurement*). (A projection is a matrix M that satisfies $M = M^\dagger$ and $M^2 = M$.) In this case, the number of possible outcomes $|\Gamma|$ can be at most $|\Sigma|$ (the dimension of $\mathcal{X}$). If we have that $M_a = |\psi_a\rangle\langle\psi_a|$ for $\{|\psi_a\rangle : a \in \Gamma\}$ an orthonormal basis of $\mathcal{X}$ (which requires $|\Gamma| = |\Sigma|$), then the measurement is a *complete projective measurement*. In this case we sometimes say that the measurement is *with respect to the basis* $\{|\psi_a\rangle : a \in \Gamma\}$.

When one doesn't care about the state of the register X after the measurement, it is sufficient to know just the operators $\{M_a^\dagger M_a : a \in \Gamma\}$ to determine the probabilities of the various outcomes:

$$\Pr[\text{outcome is } a] = \mathrm{Tr}\left(\left(M_a^\dagger M_a\right)\rho\right).$$

This follows from the fact that $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$ for any choice of matrices A and B. Sometimes these operators are called *POVM elements*, and the measurement is called a POVM (short for positive operator valued measure).

Lecture 16: Quantum error correction

March 21, 2006

Any physical realization of a quantum computer is likely to be susceptible to errors (such as noise and inaccuracies)—we cannot build perfect physical systems and isolate them from their environments while still maintaining control over them. In this lecture and the next, we will discuss quantum error correction, which aims to protect quantum information against such errors.

Classical repetition codes

We will start with a very simple classical error correcting code, the *3 bit repetition code*. Initially, just for the sake of having a concrete error model to think about, we will consider errors described by the *binary symmetric channel*. Here, we have a channel from Alice to Bob that carries 1 bit at a time. There is noise on the channel, so the bit sent by Alice is not always received correctly by Bob. Specifically, we suppose that the noise is parameterized by some real number $p \in [0,1]$ that represents the probability of an error: if Alice sends the bit b, then Bob receives b with probability $1-p$ and $\neg b$ with probability p:

$$b \longmapsto \begin{cases} b & \text{with probability } 1-p \\ \neg b & \text{with probability } p. \end{cases}$$

Suppose that Alice needs to send a very important bit to Bob, and the two of them do not wish to accept that Bob will receive the wrong bit with probability p. What can they do? The answer is that they can use an error correcting code to decrease the probability of error.

Possibly the simplest example of an error correcting code is the 3 bit repetition code. The encoding is simply to repeat the bit to be transmitted three times, and decoding is just taking the most frequently appearing bit (which is the same as computing the majority function).

encoding	transmission (3 bits)	decoding
$0 \to 000$	$\longrightarrow$	$abc \to \text{majority}(a,b,c)$
$1 \to 111$		

If 0 or 1 of the bits are flipped during transmission, then Bob will recover the correct bit sent by Alice. The probability of error shrinks from p to $3\,p^2 - 2\,p^3$:

$$\Pr[\text{correct transmission}] = (1-p)^3 + 3\,(1-p)^2 p = 1 - (3\,p^2 - 2\,p^3).$$

(Really this is only a reduction in error when $p < 1/2$. If $p = 1/2$, then the channel is useless for transmitting information, and if $p > 1/2$ then Bob can simply flip every bit which effectively replaces p with $1-p$.)

Another way of explaining what this code does (without making any reference to the binary symmetric channel) is to say that it encodes 1 bit into 3 in a way that protects against one bit-flip. Of course much more efficient (and more complicated) error correcting codes exist—this is indeed a very simple code.

Protecting quantum information against bit-flips

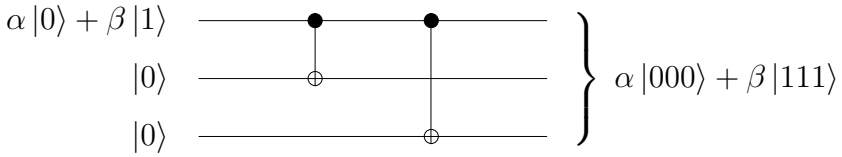
Can we encode quantum information in a similar way to the previous example? For instance, suppose Alice has a qubit in the pure state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ that she wishes to send to Bob through a noisy quantum channel. We will have to discuss exactly what a noisy quantum channel is, but let us just think about this notion informally for now. Can something analogous to the repetition code be used?

First let us note that it is not acceptable to encode $|\psi\rangle$ as $|\psi\rangle|\psi\rangle|\psi\rangle$. In case Alice does not know α and β, this would violate the no-cloning theorem. It is also not clear how Bob would decode.

A more sensible analogue of the repetition code is to perform this encoding:

$$\alpha|0\rangle + \beta|1\rangle \mapsto \alpha|000\rangle + \beta|111\rangle\,.$$

This encoding could be performed by the following simple circuit:



Now, this encoding protects against some errors, but actually makes things worse for others as we will see. Let us start off with a type of error that it does correct against: bit-flip errors. Specifically, consider a *quantum* channel that is similar to the classical binary symmetric channel, again parameterized by a number $p \in [0, 1]$:

> With probability $1 - p$, the channel acts as the identity. With the remaining probability p, the channel applies a bit-flip error, meaning that the unitary operation
>
> $$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$
>
> is performed.

This channel can therefore be described by the admissible operation

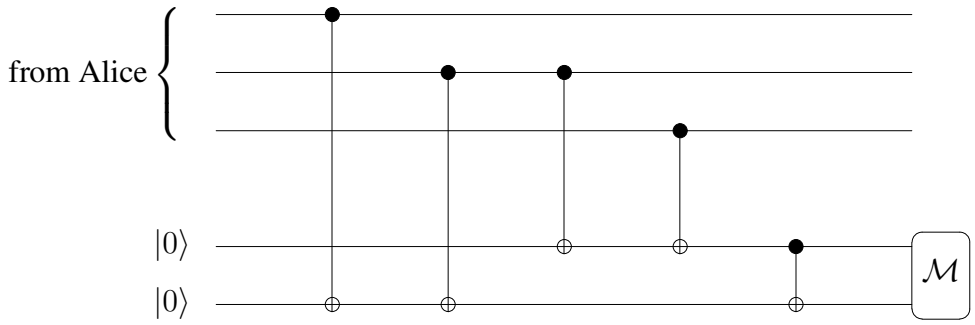
$$\Phi(\rho) = (1-p)\,\rho + p\,\sigma_x\rho\sigma_x^\dagger = (1-p)\,\rho + p\,\sigma_x\rho\sigma_x.$$

Suppose that Alice encodes $\alpha|0\rangle + \beta|1\rangle$ as $\alpha|000\rangle + \beta|111\rangle$, and sends the three qubits through the channel (one at a time). Imagine that it happens that the second qubit experiences a bit-flip error, but the first and third are unaffected. The pure state of the three qubits therefore becomes

$$\alpha|010\rangle + \beta|101\rangle\,.$$

How would Bob decode? Of course he does not know that the bit-flip occurred, and he does not want to measure the three qubits and compute the majority because that would ruin the qubit sent by Alice. Bob's goal is to recover the qubit $\alpha|0\rangle + \beta|1\rangle$.

Consider the following circuit:



Consider what this circuit does in the present case (where a bit-flip error occurred on the second qubit):

$$\begin{aligned}(\alpha|010\rangle + \beta|101\rangle)|00\rangle &= \alpha|010\rangle|00\rangle + \beta|101\rangle|00\rangle \\ &\mapsto \alpha|010\rangle|10\rangle + \beta|101\rangle|10\rangle \\ &= (\alpha|010\rangle + \beta|101\rangle)|10\rangle\,.\end{aligned}$$

The measurement would give outcome 10 (with certainty). This output string is called the *syndrome*; in this case it tells us that a bit-flip error occurred on qubit number 2 (or 10 in binary). So, Bob corrects the error by applying σ_z to qubit number 2:

$$\alpha|010\rangle + \beta|101\rangle \mapsto \alpha|000\rangle + \beta|111\rangle\,.$$

Now, by applying the inverse of the encoding procedure he recovers $\alpha|0\rangle + \beta|1\rangle$:

$$\alpha|000\rangle + \beta|111\rangle \mapsto (\alpha|0\rangle + \beta|1\rangle)|00\rangle\,.$$

The same procedure works in the case that the first or third qubit experiences a bit-flip as well. To see this, consider the syndrome that would result from any classical state:

Classical state	Syndrome
$\|000\rangle$	00
$\|001\rangle$	11
$\|010\rangle$	10
$\|011\rangle$	01
$\|100\rangle$	01
$\|101\rangle$	10
$\|110\rangle$	11
$\|111\rangle$	00

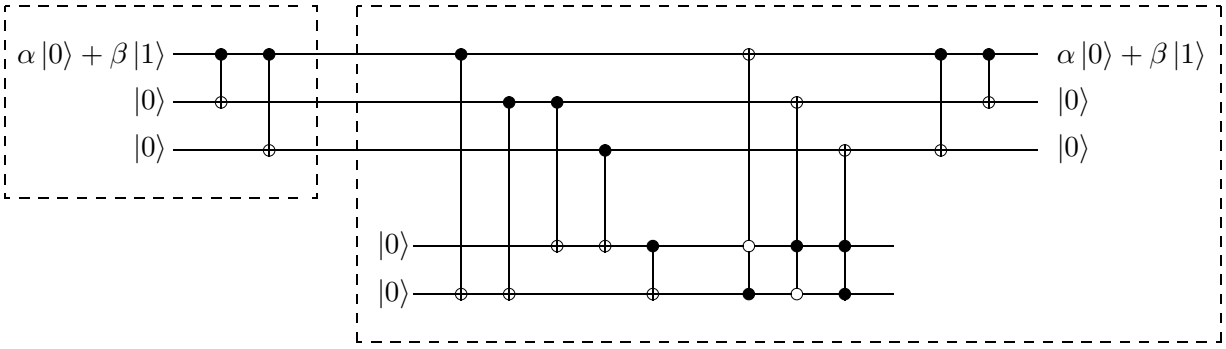
If no errors occur, or a single bit-flip occurs, the syndrome will correctly diagnose the errors (or lack of errors):

- No errors $\Rightarrow$ Syndrome = 00.
- σ_x applied to qubit number $j \in \{1, 2, 3\} \Rightarrow$ Syndrome = j (in binary).

In general, the decoding procedure is as follows.

1. If the syndrome is 00, do nothing. Otherwise, if the syndrome is $j \in \{1, 2, 3\}$ (in binary), apply a σ_x operation to qubit number j.
2. Apply the reverse of the encoding procedure.

The entire code can be represented by a diagram as follows:



The code corrects up to one bit-flip error. If two or more bit-flip errors occurred, there are no guarantees...

You could analyze the precise behavior for the channel described above, but we will not do this. You would find (assuming $p < 1/2$) that the probability of a correct transmission would increase. (There are more precise technical ways of characterizing how close the output density matrix would be to $|\psi\rangle\langle\psi|$, but this would be too much of a digression at this point in the course.)

Other quantum errors

Although the above code protects against bit-flips, this is not good enough in general. The problem is that there are different types of quantum errors. For example, a *phase-flip* error could occur:

$$|a\rangle \mapsto (-1)^a |a\rangle$$

for $a \in \{0, 1\}$. This error is therefore represented by the σ_z matrix:

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

The previous code does nothing to protect against phase-flips. For instance, we have

$$\alpha|000\rangle + \beta|111\rangle \mapsto \alpha|000\rangle - \beta|111\rangle$$

if any odd number of phase-flips occur. If, say, each phase-flip occurs with probability $p \in (0, 1/2)$, then the probability that an odd number occur is

$$3p(1-p)^2 + p^3 > p.$$

In other words, the code is making things *worse* rather than better.

Suppose that we are interested in protecting against phase errors. We could change the encoding slightly in order to do this:

$$\alpha |0\rangle + \beta |1\rangle \mapsto \alpha |+\rangle |+\rangle |+\rangle + \beta |-\rangle |-\rangle |-\rangle .$$

This is easily done by encoding $\alpha |0\rangle + \beta |1\rangle$ as $\alpha |000\rangle + \beta |111\rangle$ just like before, followed by a Hadamard transform on each qubit. The effect of a phase-flip on the basis $\{|+\rangle , |-\rangle\}$ is similar to the effect of a bit-flip on the standard basis:

$$\sigma_z |+\rangle = |-\rangle , \quad \sigma_z |-\rangle = |+\rangle .$$

Therefore, Bob can easily correct against a phase-flip on a single qubit by first applying Hadamard transforms to all three qubits, and then correcting as before. For instance, if a phase-flip happens on qubit number 1, then the encoding

$$\alpha |+\rangle |+\rangle |+\rangle + \beta |-\rangle |-\rangle |-\rangle$$

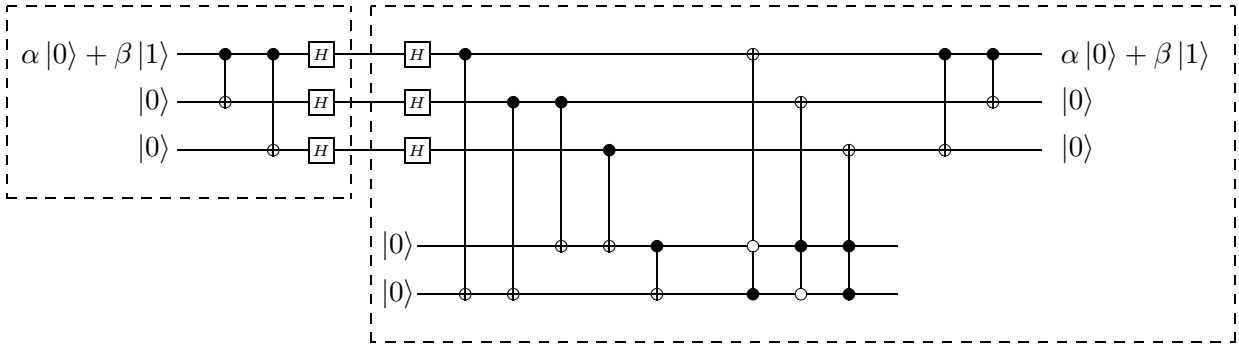
changes to

$$\alpha |-\rangle |+\rangle |+\rangle + \beta |+\rangle |-\rangle |-\rangle .$$

Bob applies Hadamard transforms to all three qubits to obtain

$$\alpha |100\rangle + \beta |011\rangle .$$

He then corrects just as before to obtain $\alpha |0\rangle + \beta |1\rangle$. This process can be described by the following circuit:

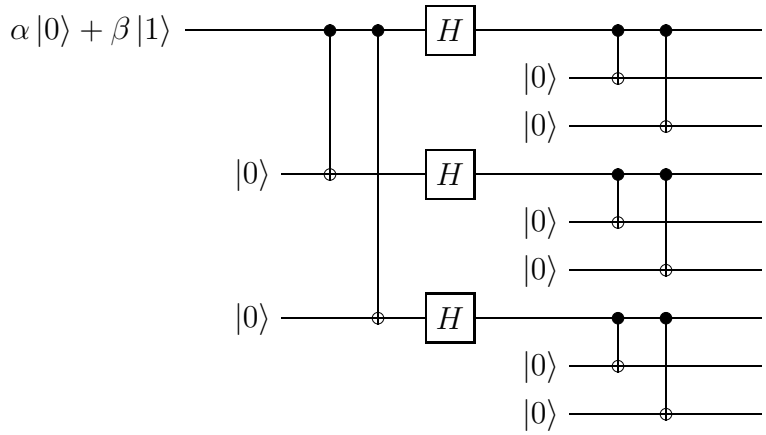


As you would expect, although the new code protects against phase-flips, it fails to protect against bit-flips (and in fact makes matters worse for bit-flips just as happened previously for phase-flips).

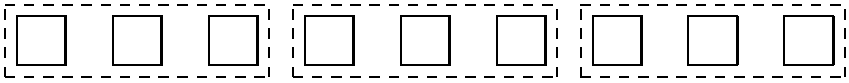
Is there any way to protect against both bit flips and phase flips simultaneously? The answer is yes, but we need to *concatenate* the codes. In other words, we first apply the code that protects against phase-flips, and then encode *each* of the three resulting qubits using the code that protects against bit-flips. In other words, we will encode $\alpha|0\rangle + \beta|1\rangle$ as

$$\alpha \frac{(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}}$$
$$+\beta \frac{(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)}{2\sqrt{2}}.$$

This is known as Shor's 9 qubit code, because it was first discovered by Shor. A circuit for the encoding procedure looks like this:



Now, how does Bob decode? Intuitively it is easy—he first imagines that the 9 qubits form 3 blocks of 3 qubits:

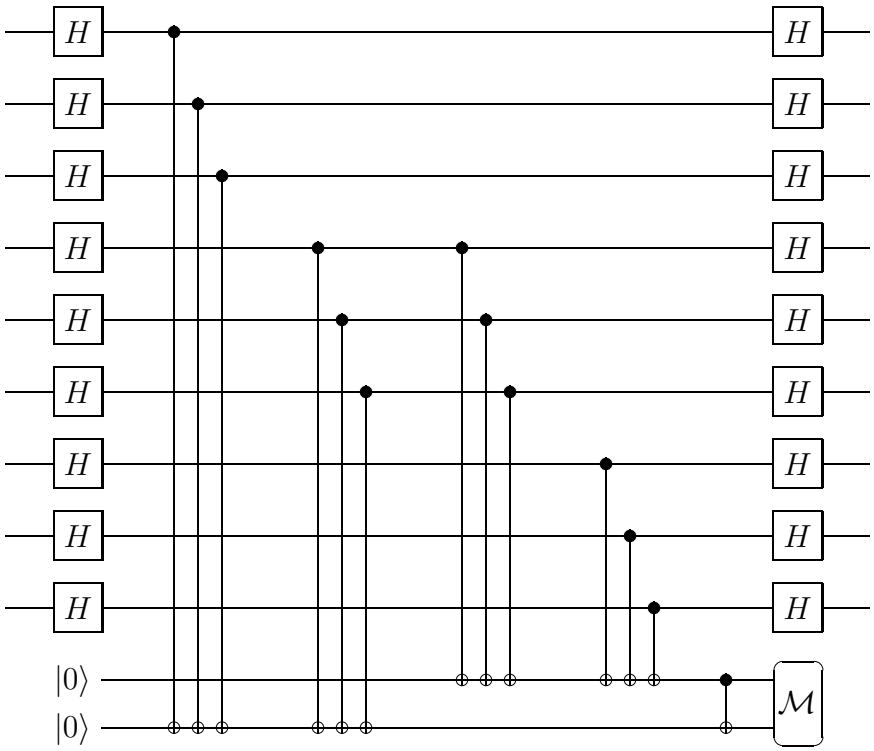


Each block represents an encoding of one of the three qubits of the state

$$\alpha|+\rangle|+\rangle|+\rangle + \beta|-\rangle|-\rangle|-\rangle,$$

where the encoding is with respect to the bit-flip code. He corrects any bit flips as before, and decodes each block to give a total of three qubits (the first in each block). Assuming at most one bit-flip occurred in each block, the remaining 6 qubits will have all returned to the state $|0\rangle$, and can be discarded or re-used. When it is time to correct phase flips, the process is again as before.

Sometimes it is convenient to separate the decoding phase into two parts—the first being to correct errors so that the original *encoding* is recovered, and the second being to transform the encoding back to the original qubit (which is obtained by running the encoding procedure in reverse). In the present case this is done by correcting bit-flip errors in each block as above and then running a slightly modified procedure to correct phase errors:



The measurement gives a syndrome that specifies, in binary, which of the three blocks has received a phase-flip error. Any phase-flip error can be corrected by applying a σ_z operation to **any one** of the three qubits in that block (or all three if you prefer).

Fact. The Shor 9 qubit code can simultaneously protect against up to 1 bit-flip error in each block and any number of phase-flip errors contained in any one block. In particular, it can protect against any one of the four "errors"

$$I,\ \sigma_x,\ \sigma_z,\ \sigma_x\sigma_z$$

occurring on a single qubit.

Example 17. Suppose the error $\sigma_x\sigma_z$ occurs on qubit number four. The encoded state becomes

$$\alpha\,\frac{(|000\rangle+|111\rangle)(|100\rangle-|011\rangle)(|000\rangle+|111\rangle)}{2\sqrt{2}}$$
$$+\beta\,\frac{(|000\rangle-|111\rangle)(|100\rangle+|011\rangle)(|000\rangle-|111\rangle)}{2\sqrt{2}}.$$

The bit-flip code correcting procedure is applied to each block. This results in three 2-bit syndromes, which in this case are 00, 01, and 00, respectively—the first block has no bit-flip errors, the second has a bit-flip error in position 1, and the third block has no bit-flip errors. The appropriate correction for each block results in the state

$$\alpha\,\frac{(|000\rangle+|111\rangle)(|000\rangle-|111\rangle)(|000\rangle+|111\rangle)}{2\sqrt{2}}$$
$$+\beta\,\frac{(|000\rangle-|111\rangle)(|000\rangle+|111\rangle)(|000\rangle-|111\rangle)}{2\sqrt{2}}.$$

Now the phase-flip code correcting procedure is applied, and this gives a single 2-bit syndrome; you can check that the procedure above gives syndrome 10 in this case. This implies that a phase-flip happened somewhere in block 2. Applying a σ_z gate to any qubit in this block results in

$$\alpha \frac{(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}}$$
$$+\beta \frac{(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)}{2\sqrt{2}},$$

which is the original encoding.

Next time we will see how arbitrary errors on a single qubit can be corrected by this code (and more generally by any code that can correct the four errors represented by the Pauli matrices: $I,\ \sigma_x,\ \sigma_z,\ \sigma_y = -i\sigma_x\sigma_z$).

Lecture 17: General quantum errors; CSS codes

March 23, 2006

General quantum errors

In the previous lecture we discussed Shor's 9 qubit code, which can be viewed as an encoding of one qubit into three blocks of three qubits. It protects against up to 1 bit-flip error in each block and any number of phase-flip errors contained in any one block. In particular, it can protect against any one of the three errors

$$\sigma_x,\ \sigma_z,\ \sigma_x\sigma_z$$

occurring on a single qubit. The code of course does not require that an error takes place to work correctly, so we can add the identity matrix to the above list. Finally, because $\sigma_y = -i\sigma_x\sigma_z$ and global phase factors are irrelevant, we may say that Shor's 9 qubit code protects against any of the four possible errors

$$I,\ \sigma_x,\ \sigma_y,\ \sigma_z$$

on any one qubit.

It turns out that this is enough to conclude that the code protects against an *arbitrary* one-qubit error, represented by any single-qubit admissible operation, and the same may be said of any quantum error correcting code. The first part of this lecture will be devoted to demonstrating this fact. The situation generalizes to multiple-qubit errors and codes that can protect against multiple qubit errors, but for simplicity we will focus on the single-qubit case.

Suppose A is any 2×2 complex matrix. Then

$$A = aI + b\sigma_x + c\sigma_y + d\sigma_z$$

for some choice of $a, b, c, d \in \mathbb{C}$. Forget about the fact that A may not be unitary for a moment, and imagine that A is applied to the j-th qubit in some n qubit state $|\psi\rangle$, which we will view as encoding one or more qubits in some quantum error correcting code. We obtain

$$A^{(j)}|\psi\rangle = a\ |\psi\rangle + b\,\sigma_x^{(j)}|\psi\rangle + c\,\sigma_y^{(j)}|\psi\rangle\, d\,\sigma_z^{(j)}|\psi\rangle\,,$$

where the superscript (j) indicates which qubit each mapping acts on.

Assuming that the code protects against any one of the four errors given by the Pauli matrices, the first step of error correction would result in the state

$$\begin{aligned} & a\ |\psi\rangle\,|I \text{ syndrome}\rangle \\ +& b\,\sigma_x^{(j)}|\psi\rangle\,|\sigma_x^{(j)} \text{ syndrome}\rangle \\ +& c\,\sigma_y^{(j)}|\psi\rangle\,|\sigma_y^{(j)} \text{ syndrome}\rangle \\ +& d\,\sigma_z^{(j)}|\psi\rangle\,|\sigma_z^{(j)} \text{ syndrome}\rangle\,. \end{aligned}$$

Correcting according to the syndrome then gives

$$|\psi\rangle \left(a \ |I \text{ syndrome}\rangle + b \ |\sigma_x^{(j)} \text{ syndrome}\rangle + c \ |\sigma_y^{(j)} \text{ syndrome}\rangle + d \ |\sigma_z^{(j)} \text{ syndrome}\rangle\right)$$

Alternately, if the syndrome is measured and the error is corrected appropriately, the state $|\psi\rangle$ is recovered regardless of the measurement outcome.

Now, if an arbitrary one-qubit *admissible* operation occurs on qubit j, we obtain

$$\Phi^{(j)}(|\psi\rangle\langle\psi|) = \sum_{k=1}^{N} A_k^{(j)} |\psi\rangle\langle\psi| \left(A_k^{(j)}\right)^\dagger$$

for some collection of matrices $A_1^{(j)}, \ldots, A_N^{(j)}$, which can each be written

$$A_k^{(j)} = a_k I + b_k \sigma_x^{(j)} + c_k \sigma_y^{(j)} + d_k \sigma_z^{(j)}.$$

A slightly messy (but conceptually simple) calculation shows that if the syndrome is computed and measured, we obtain the mixed state

$$\begin{aligned}\sum_{k=1}^{N} \Big(& |a_k|^2 |\psi\rangle\langle\psi| \otimes |I \text{ syndrome}\rangle\langle I \text{ syndrome}| \\ & |b_k|^2 \sigma_x^{(j)} |\psi\rangle\langle\psi| \sigma_x^{(j)} \otimes |\sigma_x^{(j)} \text{ syndrome}\rangle\langle\sigma_x^{(j)} \text{ syndrome}| \\ & |c_k|^2 \sigma_y^{(j)} |\psi\rangle\langle\psi| \sigma_y^{(j)} \otimes |\sigma_y^{(j)} \text{ syndrome}\rangle\langle\sigma_y^{(j)} \text{ syndrome}| \\ & |d_k|^2 \sigma_z^{(j)} |\psi\rangle\langle\psi| \sigma_z^{(j)} \otimes |\sigma_z^{(j)} \text{ syndrome}\rangle\langle\sigma_z^{(j)} \text{ syndrome}| \Big).\end{aligned}$$

The fact that there are no cross-terms in this expression is a consequence of measuring the syndrome (with respect to the standard basis).

Although the above expression might look unappealing to you, it is expressing something quite remarkable. The act of computing the syndrome and measuring has effectively projected the arbitrary error represented by Φ into one of the four discrete errors I, σ_x, σ_y, or σ_z. Correcting according to the syndrome then gives

$$\begin{aligned}|\psi\rangle\langle\psi| \otimes \sum_{k=1}^{N} \Big(& |a_k|^2 |I \text{ syndrome}\rangle\langle I \text{ syndrome}| \\ & + |b_k|^2 |\sigma_x^{(j)} \text{ syndrome}\rangle\langle\sigma_x^{(j)} \text{ syndrome}| \\ & + |c_k|^2 |\sigma_y^{(j)} \text{ syndrome}\rangle\langle\sigma_y^{(j)} \text{ syndrome}| \\ & + |d_k|^2 |\sigma_z^{(j)} \text{ syndrome}\rangle\langle\sigma_z^{(j)} \text{ syndrome}| \Big).\end{aligned}$$

Here we are of course relying on the fact that the code corrects the four Pauli operators correctly. Throwing the syndrome in the trash gives $|\psi\rangle\langle\psi|$ as desired.

CSS codes

The last thing we will discuss on the topic of quantum error correction is CSS codes, named after their co-discoverers Robert Calderbank, Peter Shor, and Andrew Steane. These codes are interesting because they illustrate how classes of classical error correcting codes can sometimes be turned into classes of quantum codes. They are also useful for one of the simpler proofs of security of the BB84 quantum key-distribution protocol, which we will discuss next week.

Classical linear codes

In order to discuss CSS codes, we will need to first learn a little bit about classical linear codes. Some terminology will be helpful when doing this.

First, when we refer to an $[n, k]$ code, we mean an error correcting code that encodes k bits into n (so we will always have $n > k$). Formally, we define an $[n, k]$ code simply to be a subset of $\mathbb{Z}_2^n$ of size 2^k. Here, and throughout this discussion, we are identifying strings of length n with (column) vectors in $\mathbb{Z}_2^n$ in the most straightforward way. The elements of such a code $C \subseteq \mathbb{Z}_2^n$ are called *codewords*.

An $[n, k]$ code C is said to have distance d if the *Hamming distance* between any two distinct code words $y, z \in C$ is at least d, and d is the largest positive integer that satisfies this property. The Hamming distance between strings y and z is the number of positions in which they disagree. An $[n, k]$ code with minimum distance d is sometimes referred to as an $[n, k, d]$ code. Such a code can correct bit-flips on any number of bits that is strictly less than half of the code's minimum distance. An $[n, k]$ *linear code* is a code that happens to form a subspace of $\mathbb{Z}_2^n$ of dimension k, where all arithmetic takes place in $\mathbb{Z}_2$. In the case of a linear code, the minimum distance is the same as the minimal *Hamming weight* (or number of 1s) of any nonzero codeword.

There are two matrices that can be associated with a linear code that make encoding and error detection/correction easier: a *generator matrix* and a *parity check matrix*.

Generator matrix. A generator matrix G for an $[n, k]$ linear code C is an $n \times k$ matrix over $\mathbb{Z}_2^n$ such that

$$C = \mathrm{range}(G) \stackrel{\text{def}}{=} \{Gx \,:\, x \in \mathbb{Z}_2^k\}.$$

To encode a string $x \in \mathbb{Z}_2^k$, we simply multiply by G, i.e., $x \in \mathbb{Z}_2^k$ is encoded as $Gx \in \mathbb{Z}_2^n$.

Parity check matrix. A parity check matrix K for an $[n, k]$ linear code C is an $(n-k) \times n$ matrix over $\mathbb{Z}_2$ such that

$$C = \ker(K) \stackrel{\text{def}}{=} \{y \in \mathbb{Z}_2^n \,:\, Ky = 0\}.$$

The syndrome for a given string $y' \in \mathbb{Z}_2^n$ is $Ky' \in \mathbb{Z}_2^{n-k}$.

Once either of these two matrices is chosen, the code is determined. It is helpful, however, to have both.

Let us briefly discuss why the syndrome given by the parity check matrix is helpful. Suppose that we have a codeword $y \in C$ and errors occur on one or more of the bits. We can write the resulting string y' as $y + e$ where $e \in \mathbb{Z}_2^n$ indicates where the errors took place (a 1 in each position where an error takes place, and 0 everywhere else). Then we see that $Ky' = K(y + e) = Ke$ is

a function only of the errors and not the original codeword. If the Hamming weight of e is small enough, the syndrome Ke will uniquely determine the error vector e. More specifically, there cannot be two distinct strings e and e' both of Hamming weight less than $d/2$ such that $Ke' = Ke$.

Example 18. Let us define a $[7, 4]$ linear code C as follows. The generator matrix is

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

and the parity check matrix is

$$K = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

This particular code is called a $[7, 4]$ Hamming code. (More generally, Hamming codes are $[2^m - 1, 2^m - m - 1]$ codes obtained by taking the parity check matrix to have as columns all of the nonzero strings of length $n - k$.) All Hamming codes including the above one have distance 3. Because the distance is 3, the code can tolerate up to one bit-flip.

For example, the encoding of the string 1001 is 100100 and the encoding of 1111 is 1111111, because

$$G \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad \text{and} \qquad G \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Dual codes

Suppose that we have an $[n, k]$ linear code C. The *dual code* to C is denoted $C^\perp$, and contains all strings that have modulo 2 inner product 0 with all codewords of C. In symbols:

$$C^\perp = \{y \in \mathbb{Z}_2^n \,:\, y \cdot x = 0 \text{ for all } x \in C\}.$$

Therefore, $C^\perp$ is an $[n, n - k]$ linear code. If we have a generator matrix G and a parity check matrix K for C, then the generator matrix for $C^\perp$ is K^T and the parity check matrix is G^T. A code C is *weakly self dual* if $C \subseteq C^\perp$ and is *strictly self dual* (or just *self dual*) if $C = C^\perp$.

Example 19. The code $C = \{00, 11\}$ is a very simple self dual $[2, 1]$ linear code. The $[7, 4]$ Hamming code above is not weakly self dual. In order for C to be weakly self dual, it is necessary and sufficient that the generator matrix G for C satisfies $G^{\mathsf{T}}G = 0$. This is not the case for the $[7, 4]$ Hamming code. However, the dual code $C^{\perp}$ of that code is weakly self dual. This follows from the fact that the generator matrix of $C^{\perp}$ is K^{T}, which satisfies $(K^{\mathsf{T}})^{\mathsf{T}}K^{\mathsf{T}} = KK^{\mathsf{T}} = 0$.

CSS codes

Now we are ready to define CSS codes. In order to define a CSS code, we must have two classical linear codes C_1 and C_2 that satisfy certain properties:

1. C_1 must be an $[n, k_1]$ linear code and C_2 must be an $[n, k_2]$ linear code for $k_2 < k_1$.
2. It must be that $C_2 \subseteq C_1$.
3. If both C_1 and $C_2^{\perp}$ can correct up to t errors, then the resulting CSS code will be an $[n, k_1 - k_2]$ code that corrects up to t quantum errors (meaning an arbitrary error confined to t qubits).

Example 20. Let C_1 be the $[7, 4]$ Hamming code discussed above and let $C_2 = C_1^{\perp}$. Then C_2 is a $[7, 3]$ (weakly self dual) linear code. We have $C_2 \subsetneq C_1$ by virtue of the fact that C_2 is weakly self dual and is clearly not equal to C_1 (because it has fewer vectors). Because C_1 corrects up to one error and $C_2^{\perp} = C_1$, the CSS code constructed will be a $[7, 1]$ quantum code that can correct 1 quantum error.

The encoding for the CSS code constructed from codes C_1 and C_2 as above is as follows.

1. Let $N = 2^{k_1 - k_2}$. The space spanned by all possible encodings will have dimension N. Choose codewords $x_0, \ldots, x_{2^N - 1} \in C_1$ satisfying the condition

$$x_i + x_j \notin C_2$$

   for $i \neq j$. (This will always be possible because C_1/C_2 is a space of dimension $k_1 - k_2$ over $\mathbb{Z}_2$, and each element of this space gives rise to at least one good choice of x_j.)

2. Identify the classical states of the $k_1 - k_2$ qubits to be encoded with numbers $0, \ldots, N - 1$ in binary. Then the encoding corresponds to the linear mapping defined by

$$|j\rangle \mapsto |x_j + C_2\rangle = \frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} |x_j + y\rangle .$$

   The fact that $\langle x_i + C_2 | x_j + C_2 \rangle = 0$ for $i \neq j$ follows from the fact that $x_i + x_j \notin C_2$ for $i \neq j$.

Example 21. Let C_1 and C_2 be as in the previous example. Then $N = 2$, so we need just two codewords x_0 and x_1 in C_1 so that $x_0 + x_1 \notin C_2$. We may take $x_0 = 0000000$ and $x_1 = 1111111$. Enumerating all elements of $C_2 = C_1^{\perp}$ is not so hard in this case—we have

$$C_2 = \{0000000, 0001111, 0110011, 1010101, 0111100, 1011010, 1100110, 1101001\}.$$

Thus, the encoding for the corresponding CSS code will be:

$$\begin{aligned}|0\rangle \mapsto \frac{1}{\sqrt{8}}(&|0000000\rangle + |0001111\rangle + |0110011\rangle + |1010101\rangle \\ &+ |0111100\rangle + |1011010\rangle + |1100110\rangle + |1101001\rangle) \\ |1\rangle \mapsto \frac{1}{\sqrt{8}}(&|1111111\rangle + |1110000\rangle + |1001100\rangle + |0101010\rangle \\ &+ |1000011\rangle + |0100101\rangle + |0011001\rangle + |0010110\rangle).\end{aligned}$$

This code is known as the Steane 7 qubit code.

It remains to discuss error correction for CSS codes. Suppose we have an n qubit state contained in a register R that we want to correct. The procedure for doing this is very simple:

1. First, reversibly compute the syndrome of R for code C_1, which corresponds to the reversible transformation
$$|y\rangle |00\cdots 0\rangle \mapsto |y\rangle |s_1(y)\rangle ,$$
where $s_1(y)$ denotes the syndrome of y with respect to code C_1. Measure this syndrome, and correct bit-flips by applying NOT gates to the appropriate qubits of R.
2. Apply a Hadamard transform to every qubit of R.
3. Repeat the same procedure as in step 1, except using the code $C_2^{\perp}$.
4. Again apply Hadamard transforms to all qubits of R.

Under the assumption that at most t bit-flips and t phase-flips occurred on X starting from some valid encoding $|\psi\rangle$, the result will be that the encoding $|\psi\rangle$ is recovered. Let us see why this is the case.

First, some notation is needed. Suppose that bit-flips are represented by a vector $e \in \mathbb{Z}_2^n$ and phase-flips are represented by $f \in \mathbb{Z}_2^n$, with both vectors containing at most t ones. If, for a given vector $v \in \mathbb{Z}_2^n$ we denote $X^v = X^{v[1]} \otimes \cdots \otimes X^{v[n]}$ and similarly $Z^v = Z^{v[1]} \otimes \cdots \otimes Z^{v[n]}$, this means that the overall error is given by $X^e Z^f$. It will be helpful to note that for any choice of $u, v \in \mathbb{Z}_2^n$ the following simple formulas hold:

(a) $X^u Z^v = (-1)^{u \cdot v} Z^v X^u$,

(b) $H^{\otimes n} X^u = Z^u H^{\otimes n}$, and

(c) $H^{\otimes n} Z^u = X^u H^{\otimes n}$.

Now, consider an arbitrary valid encoded state with respect to the CSS code:

$$\sum_{j=0}^{N-1} \alpha_j |x_j + C_2\rangle .$$

If the errors represented by e and f occur, the state becomes

$$\sum_{j=0}^{N-1} \alpha_j X^e Z^f \left| x_j + C_2 \right\rangle = \sum_{j=0}^{N-1} \alpha_j (-1)^{e \cdot f} Z^f \left| x_j + e + C_2 \right\rangle .$$

Step (1) of the error correction procedure computes and measures the syndrome with respect to C_1. Error correction is performed by applying NOT gates to the qubits indicated by the syndrome. Because e has at most t ones, the syndrome reveals the vector e, and after correction the state becomes

$$\sum_{j=0}^{N-1} \alpha_j (-1)^{e \cdot f} X^e Z^f \left| x_j + C_2 + e \right\rangle = \sum_{j=0}^{N-1} \alpha_j Z^f \left| x_j + C_2 \right\rangle .$$

Notice that here is where we require that $C_2 \subseteq C_1$—it must be that every string in $x_j + C_2$ is a codeword with respect to C_1. It will be convenient for the next step to write the above state as

$$\sum_{j=0}^{N-1} \alpha_j Z^f X^{x_j} \left| C_2 \right\rangle .$$

Next, the Hadamard transforms are performed. It is a straightforward calculation to show that $H^{\otimes n} \left| C_2 \right\rangle = \left| C_2^{\perp} \right\rangle$, and so the state becomes

$$\sum_{j=0}^{N-1} \alpha_j H^{\otimes n} Z^f X^{x_j} \left| C_2 \right\rangle = \sum_{j=0}^{N-1} \alpha_j X^f Z^{x_j} \left| C_2^{\perp} \right\rangle = \sum_{j=0}^{N-1} \alpha_j (-1)^{f \cdot x_j} Z^{x_j} \left| f + C_2^{\perp} \right\rangle .$$

Step (3) of the error correction procedure detects the vector f in a similar way to e being detected in step (1). The correction works by applying NOT gates in the positions indicated by f, which results in the state

$$\sum_{j=0}^{N-1} \alpha_j (-1)^{f \cdot x_j} X^f Z^{x_j} \left| f + C_2^{\perp} \right\rangle = \sum_{j=0}^{N-1} \alpha_j Z^{x_j} \left| C_2^{\perp} \right\rangle .$$

Finally, the second collection of Hadamard transforms map this state to

$$\sum_{j=0}^{N-1} \alpha_j H^{\otimes n} Z^{x_j} \left| C_2^{\perp} \right\rangle = \sum_{j=0}^{N-1} \alpha_j X^{x_j} \left| C_2 \right\rangle = \sum_{j=0}^{N-1} \alpha_j \left| x_j + C_2 \right\rangle .$$

This is the original encoding as required.

Lecture 18: Quantum Key Distribution

March 28, 2006

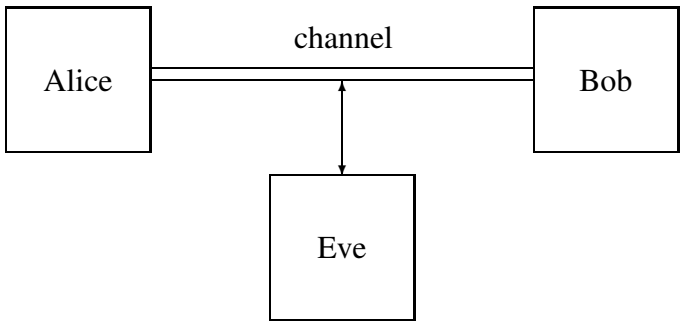
In this lecture we will begin discussing quantum cryptography. There are indeed many aspects to cryptography, and many different cryptographic tasks that one may consider. Likely the first cryptographic task that comes to mind is *private communication*: Alice and Bob wish to communicate, and they do not want an eavesdropper (Eve) to learn any information about their communication. There are many other cryptographic tasks or protocols one may consider, such as *message authentication*, *digital signatures*, and *voting schemes*. Often these tasks are broken down into more primitive operations, such as *bit-commitment* and *oblivious transfer*.

In this lecture we will just consider private communication, and will discuss bit commitment in the next lecture.

A perfectly secure cryptosystem

Let us first consider a purely classical situation. Suppose that Alice and Bob can communicate through a classical channel, and a third party Eve is able to monitor the communications on that channel. Alice wishes to transmit a particular message to Bob, and wants Eve to learn as little as possible about it. Eve wants to learn as much as possible about the message.

In cryptography we always need to be precise about our assumptions and model. In this example, we will assume that the channel Alice and Bob are communicate through is an *authenticated* channel: when Bob receives a message, he can verify that it came from Alice (and vice versa). We will also assume that Eve cannot jam the channel or modify messages sent over it, meaning that Alice and Bob always receive one another's messages without error.



There exists a provably secure classical scheme, known as the *one-time pad*, for accomplishing the task at hand. It requires that Alice and Bob share a sequence of random bits, called a *private key*, that is secret from Eve. This private key could have been created a long time ago, and is completely independent of the message Alice wishes to send. The key must be of the same length as the message. The scheme works as follows.

One-time Pad

Alice and Bob share a private key $K \in \{0,1\}^n$.

Alice wishes to send Bob a message $M \in \{0,1\}^n$. She computes $E = K \oplus M$ (bitwise exclusive OR) and sends E (the encrypted message) to Bob.

Bob receives E, and computes $E \oplus K$. The result is $E \oplus K = M \oplus K \oplus K = M$.

Eve can learn nothing (aside from the length of the message) by looking at E. If K is uniformly distributed, then so too is E.

One very important note is that the key K can only be used once (hence the name "one-time pad"). If they were to use it twice with two different messages, the encrypted texts would be

$$E_1 = M_1 \oplus K$$
$$E_2 = M_2 \oplus K.$$

Eve could then compute

$$E_1 \oplus E_2 = M_1 \oplus M_2,$$

which could contain a lot of information about M_1 and M_2. This could also allow Eve to learn something about K that could be exploited if Alice and Bob were to use the key a third time.

The one-time pad is unbreakable (if used properly), but it is very inefficient or impossible in many situations. For instance, a single entity such as a bank or online business may wish to communicate securely with many different individuals, and may not have met with them previously to exchange a key. It is impossible for two parties to classically generate a secure private key over a public channel.

Quantum key distribution

The aim of quantum key distribution is to allow Alice and Bob to generate a secure private key that can be used for the one-time pad without having to meet privately. They will be able to accomplish this task by using quantum information. There are a few different schemes for doing this, such as:

1. BB84. Easy to implement, relatively hard to prove security.
2. B92. Similar to BB84.
3. Lo-Chau. Hard to implement, easy to prove security.

We will only look at BB84, so named because its inventors Charles Bennett and Gilles Brassard proposed it in 1984. Proofs of security actually took a long time; Mayers gave a (complicated) proof in 1998, and Shor and Preskill gave a fairly simple proof based on the theory of quantum error correcting codes in 2000.

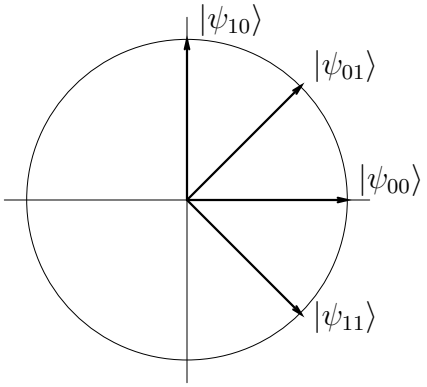
Once again, we need to be precise about our assumptions. We will assume that as before, Alice and Bob can communicate over a classical channel that is authenticated and cannot be tampered

with, but that is readable by Eve.[4] We will call this the *public* channel. They can also send qubits over a quantum channel. Eve has full access to this channel, meaning that she can intercept messages sent by Alice or Bob, perform measurements or quantum computations on these messages, and transmit the modified qubits to whichever party was to receive them in the first place.

The BB84 Key Exchange Protocol (first stage)

Alice randomly generates two strings of bits $x, y \in \{0,1\}^m$.

Define $|\psi_{00}\rangle = |0\rangle$, $|\psi_{10}\rangle = |1\rangle$, $|\psi_{01}\rangle = |+\rangle$, and $|\psi_{11}\rangle = |-\rangle$. These four states may be pictured as follows:



Alice prepares m qubits in the state

$$|\psi_{x,y}\rangle = |\psi_{x_1 y_1}\rangle \, |\psi_{x_2 y_2}\rangle \cdots |\psi_{x_m y_m}\rangle$$

and sends these m qubits over a quantum channel to Bob.

Bob receives m qubits, although they may not longer be in the state $|\psi_{x,y}\rangle$ because Eve may have tampered with them, or possibly the channel is noisy. (To understand how the protocol works, it is helpful to imagine first that Eve is not present and the channel is not noisy, so Bob receives precisely the state $|\psi_{x,y}\rangle$.)

Bob randomly chooses $y' \in \{0,1\}^m$, and measures each qubit received from Alice as follows:

- If $y'_i = 0$, Bob measures qubit i (with respect to the standard basis).
- If $y'_i = 1$, Bob performs a Hadamard transform on qubit i and then measures it with respect to the standard basis. (Equivalently, Bob measures qubit i with respect to the *diagonal basis* $\{|+\rangle, |-\rangle\}$, interpreting the result as 0 or 1, respectively.)

Let $x' \in \{0,1\}^m$ be the string corresponding to the results of Bob's measurements. The important thing to note at this point is that if $y_i = y'_i$ for some i and there was no noise or eavesdropping, then it is certain that $x_i = x'_i$.

[4] It is not a trivial assumption that the classical channel is authenticated, and this assumption represents an inevitable weakness of quantum key distribution. In the lecture I have briefly discussed how authenticated channels can be implemented using very short private keys (that only need to remain private for the duration of the protocol).

Finally, Alice and Bob publicly compare y and y'. They discard all bits x_i and x'_i for which $y_i \neq y'_i$. The remaining bits of x and x' represent a "semi-private" (and possibly noisy) key that will go into the next stage of the protocol.

End of protocol (stage 1).

Example 22. Suppose $m = 8$. Alice randomly chooses

$$x = 01110100$$
$$y = 11010001.$$

She sends $|\psi_{x,y}\rangle$ to Bob.

Suppose there is no tampering of the message, so Bob receives precisely $|\psi_{x,y}\rangle$. He randomly chooses

$$y' = 01110110$$

and measures. The first qubit is in state $|\psi_{01}\rangle = |+\rangle$, and because $y'_1 = 0$ Bob measures in the standard basis. He will get a uniform random bit; say $x'_1 = 1$. The second qubit is in state $|\psi_{11}\rangle = |-\rangle$, and because $y'_2 = 1$ Bob measures in the diagonal basis. He gets $x'_2 = 1$ with certainty this time. Continuing in this way, we might obtain the following table:

x	y	x'	y'
0	1	1	0
1	1	1	1
1	0	0	1
1	1	1	1
0	0	0	0
1	0	1	1
0	0	1	1
0	1	1	0

Alice and Bob publicly compare y and y'. They agree at positions 2, 4, and 5, so they keep the bits of x and x' at these positions and discard the rest:

x	y	x'	y'
~~0~~	~~1~~	~~1~~	~~0~~
1	1	1	1
~~1~~	~~0~~	~~0~~	~~1~~
1	1	1	1
0	0	0	0
~~1~~	~~0~~	~~1~~	~~1~~
~~0~~	~~0~~	~~1~~	~~1~~
~~0~~	~~1~~	~~1~~	~~0~~

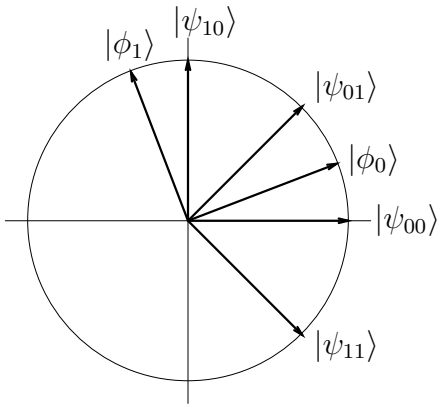
The three remaining bits of x and x' agree.

Of course the difficulty arises when Eve tries to extract information about x. The general principle working for Alice and Bob is the **uncertainty principle**—Eve cannot learn something about $|\psi_{x,y}\rangle$ without disturbing it. In particular, Eve does not know y, so she cannot hope to learn the bits of x without making some mistakes and causing some disturbance to $|\psi\rangle$.

For example, suppose that $y_i = y'_i$ for some index i, but Eve decided to measure qubit number i. Perhaps she measures the qubit in the *Breidbart basis*

$$|\phi_0\rangle = \cos(\pi/8)\,|0\rangle + \sin(\pi/8)\,|1\rangle\,, \quad |\phi_1\rangle = -\sin(\pi/8)\,|0\rangle + \cos(\pi/8)\,|1\rangle\,,$$

which is pictured as follows:



This is a good basis for Eve to choose—she will learn x_i with probability $\cos^2(\pi/8) \approx 0.85$. However, this measurement will cause the i-th qubit to be in state $|\phi_0\rangle$ or $|\phi_1\rangle$ after Eve's measurement. When Bob then measures, he will then get $x'_i \neq x_i$ with probability $\sin^2(\pi/8) \approx .15$, even though he should get $x_i = x'_i$ with certainty.

Informal and unsubstantiated claim: the more Alice learns about x, the more positions in which x and x' will disagree.

The difficulty in proving security in general is that Eve may use any strategy allowed by quantum mechanics—it is not sufficient to consider just a single attack such as measuring each qubit in the Breidbart basis.

Second stage of protocol.

Alice and Bob now need to estimate how much Eve might know about x and x'. They do this by sacrificing some of the remaining bits of x and x', say one-half of them (selected randomly). By comparing these bits publicly, they can estimate the error rate with high accuracy, and if it is too large they abort (Eve potentially has too much information for them to succeed). The maximum error rate that can be tolerated is about 11%.

If they have an acceptable error rate, Alice and Bob will have two strings x and x' (which now include only the bits that they did not sacrifice to estimate the error) that agree in a high percentage of positions with high probability. They have some bound on the amount of information Eve possesses about these strings.

Classical cryptography can now take over. They perform:

1. Information reconciliation.
2. Privacy amplification.

Information reconciliation is essentially distributed error correction that corrects x and x' so that they agree in all positions with very high probability. This may leak some information about these strings, but not too much. Privacy amplification transforms a shared string about which Eve has some bounded amount of information and compresses it to a shared string about which Eve has almost no information. Basically, Alice and Bob apply some suitable hash function to their shared strings.

End of protocol (stage 2).

There has been a lot of work done on information reconciliation and privacy amplification, but we will not have time to discuss these things in detail. In the lecture I have discussed some simplified versions of these problems to give you the feel for how they might work, but I will not include these examples in the notes.

Lecture 19: Impossibility of Quantum Bit Commitment

March 30, 2006

In the previous lecture we discussed the BB84 quantum key-distribution protocol, which allows two physically separated parties to construct a secure private key, and therefore communicate privately by means of the one-time pad. Although the problem of communicating privately is of paramount importance in cryptography, there are other cryptographic tasks that one may consider that key distribution protocols do not address. In this lecture we will discuss one of them: *bit commitment*. This is a particularly interesting problem from the point of view of quantum information because it turns out to be impossible, and this impossibility is due to a fundamental fact about bipartite quantum states (which we have already discussed).

Bit commitment

First let us define what bit commitment is. Alice has a bit b that she wishes to *commit* to Bob, but she doesn't want Bob to know what it is until she chooses to *reveal* it. Although Bob should not be able to determine b before Alice reveals, he should be sure that Alice cannot change the bit after it is committed. The two key properties of any bit commitment protocol are therefore that it must be:

- **Binding**. Alice should not be able to change the bit she committed.
- **Concealing**. Bob should not be able to identify the bit that Alice committed until she reveals it.

One can imagine a "mechanical" implementation of bit commitment as follows. Alice writes b on a piece of paper, locks it in a safe, and sends the safe to Bob. Bob receives the safe, but he doesn't have the key. He cannot open the safe without the key, so he cannot determine b, and therefore the concealing property holds. Because the safe is in Bob's possession, Alice cannot open it and change the bit, so the binding property also holds. When Alice wishes to reveal the bit, she sends the key to Bob.

Of course this mechanical interpretation is not satisfactory with respect to information processing purposes—we would like an implementation based on information. (It could also be argued that the binding and concealing properties, and in particular the concealing property, are based on strong physical assumptions. It is probably impossible to build a safe that can only be opened with a unique key. This is beside the point, however, because our real interest is with an information-based implementation.)

Similar to key distribution, it is impossible to implement bit commitment using classical information without using assumptions about computational intractability. It is a natural question to ask whether quantum information allows one to implement bit commitment.

Before addressing this question, it may be helpful to briefly motivate bit commitment. Why would we want to implement bit commitment? The answer is that it is a very interesting cryptographic primitive from which several interesting protocols can be built. For example, bit commitment allows for secure multi-party computations (such as voting), zero-knowledge proofs for NP-complete problems, and coin-flipping (as well as more complicated variants, such as playing poker).

Sketch of impossibility proof

The impossibility of quantum bit commitment relies on the following fact, which we proved in Lecture 15.

Fact. Suppose $|\phi\rangle, |\psi\rangle \in \mathcal{A} \otimes \mathcal{B}$ satisfy $\mathrm{Tr}_{\mathcal{A}} |\phi\rangle\langle\phi| = \mathrm{Tr}_{\mathcal{A}} |\psi\rangle\langle\psi|$. Then there exists a unitary operator $U \in \mathrm{L}(\mathcal{A})$ such that $(U \otimes I)|\phi\rangle = |\psi\rangle$.

It happens to be the case that there are approximate versions of the above fact, but because we have not discussed meaningful distance measures for quantum states it will not be possible to go into greater detail about this. Expressed informally, the approximate versions are of this form: if

$$\mathrm{Tr}_{\mathcal{A}} |\phi\rangle\langle\phi| \approx \mathrm{Tr}_{\mathcal{A}} |\psi\rangle\langle\psi|$$

then there exists a unitary operator $U \in \mathrm{L}(\mathcal{A})$ such that $(U \otimes I)|\phi\rangle \approx |\psi\rangle$.

Now let us apply the above fact to the problem of bit commitment. To begin, suppose that Alice and Bob use a "purely quantum" protocol to supposedly implement bit commitment. What we mean by this is that Alice and Bob apply only unitary operations and send quantum information back and forth—so assuming Alice and Bob are both being honest there are no measurements made until the end of the protocol, and there is no noise or other non-unitary transformations during the protocol.

There are necessarily two phases of any bit commitment protocol: the commit phase and the reveal phase. During the commit phase, Alice and Bob may perform some sequence of unitary operations and send qubits back and forth to one another any number of times. Assume that Alice and Bob's quantum systems at the end of the commit phase have corresponding vector spaces $\mathcal{A}$ and $\mathcal{B}$. There are two possible pure states of the entire system at this point: $|\psi_0\rangle$ or $|\psi_1\rangle$, depending on whether Alice intended to commit 0 or 1, respectively.

Under the assumption that the protocol is perfectly concealing, it is the case that

$$\mathrm{Tr}_{\mathcal{A}} |\psi_0\rangle\langle\psi_0| = \mathrm{Tr}_{\mathcal{A}} |\psi_1\rangle\langle\psi_1|;$$

if this were not so, Bob would be able to perform some measurement of his portion of $|\psi_0\rangle$ or $|\psi_1\rangle$ and gain at least partial information about which bit Alice committed. This implies the existence of a unitary operation $U \in \mathrm{L}(\mathcal{A})$ that Alice can perform on her qubits alone that would transform $|\psi_0\rangle$ to $|\psi_1\rangle$:

$$(U \otimes I)|\psi_0\rangle = |\psi_1\rangle.$$

Therefore, the binding property must completely fail to hold—Alice can switch back and forth between $|\psi_0\rangle$ and $|\psi_1\rangle$ without Bob's help or knowledge. For example, Alice may simply run the

original protocol and "commit" to 0, and later right before the reveal phase, she may either apply U (to switch her commitment to 1) or do nothing (leaving the commitment as 0).

Now, you might ask what happens when a protocol specifies that Alice and Bob must perform certain measurements or exchange classical rather than quantum information. It turns out that the situation is essentially the same. This is because the cheating party can decide to simulate all measurements and other non-unitary operations by using only unitary operations, which is always possible using auxiliary qubits as we discussed a few lectures ago. The fact that the non-cheating party may perform measurements or other non-unitary operations will not affect the cheater's ability to cheat—we may view that the non-cheater uses only unitary operations and simply chooses not to interact with the auxiliary qubits that would be used to do this.

Finally, you might ask what happens if perfect security is not required, but in instead only approximate security is permitted. This is indeed the more interesting situation, and it is handled by the approximate versions of the fact that was used above. Specifically, if it is the case that after the commit phase that Bob can learn a little bit, but not too much, about Alice's commitment, then we must have

$$\mathrm{Tr}_{\mathcal{A}} |\psi_0\rangle \langle\psi_0| \approx \mathrm{Tr}_{\mathcal{A}} |\psi_1\rangle \langle\psi_1| ,$$

where "$\approx$" has some technical meaning that we have not discussed. (Usually one uses either the notion of *fidelity* or *trace distance* to quantify the notion of approximate equality in such a situation.) This will not necessarily allow Alice complete freedom to change her commitment, but she will have almost complete freedom. There will exist U such that $(U \otimes I) |\psi_0\rangle \approx |\psi_1\rangle$, meaning that Alice will be able to change her commitment in a way that Bob will probably not be able to notice.

Example of an incorrect protocol

In order to illustrate the above discussion, let us consider a protocol that may initially appear to implement bit commitment (in a way that is perfectly concealing and approximately binding). This example, including a cheating strategy, was given in the same paper that proposed the BB84 key exchange protocol (so it was never really believed to be a correct protocol).

Example 23. Consider the following protocol, where we assume Alice wishes to commit to the bit $b \in \{0, 1\}$.

Commit phase. Let $S_0 = \{|0\rangle , |1\rangle\}$ and $S_1 = \{|+\rangle , |-\rangle\}$. Alice prepares a qubit X in a uniformly chosen state $|\phi\rangle \in S_b$, and sends X to Bob. (Bob does not need to do anything in the commit phase other than store the qubit sent by Alice.)

Reveal phase. When Alice wishes to reveal her commitment to Bob, she reveals a classical specification of $|\phi\rangle$, for instance:

$$\begin{aligned} 00 &\leftrightarrow |\phi\rangle = |0\rangle \\ 01 &\leftrightarrow |\phi\rangle = |+\rangle \\ 10 &\leftrightarrow |\phi\rangle = |1\rangle \\ 11 &\leftrightarrow |\phi\rangle = |-\rangle. \end{aligned}$$

In other words, the second bit is b, while the first bit specifies which element of S_b was selected. To check that Alice was being truthful, Bob measures the qubit X sent by Alice in the basis S_b. If the measurement result does not match with the state $|\phi\rangle$ sent by Alice, then Bob has caught Alice cheating. (Otherwise he has not.)

Let us examine the protocol to see what is wrong. (There has to be something wrong if we believe the previous argument that bit commitment is impossible.) First we check to see if it is concealing. If Alice wishes to commit 0, she sends Bob the qubit X in state $|0\rangle$ or state $|1\rangle$, each with equal probability. As Bob does not know which one Alice chooses, his description of the state of X is given by the density matrix

$$\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| = \frac{1}{2}I,$$

which is the totally mixed state. On the other hand, if Alice wishes to commit a 1, she sends either the state $|+\rangle$ or the state $|-\rangle$, again each with probability 1/2. In this case, the density matrix describing Bob's knowledge of X is

$$\frac{1}{2}|+\rangle\langle +| + \frac{1}{2}|-\rangle\langle -| = \frac{1}{2}I,$$

which is identical to the first case. So, the protocol is indeed perfectly concealing—Bob cannot determine any information about b at any time before the reveal phase.

This means the binding property must not hold. It is not difficult to come up with a faulty argument for why the binding property should hold (where Bob catches Alice cheating with some nonzero probability when she does so), under the assumption that Alice prepares X in some state $|\phi\rangle$. This is a bad assumption, though.

One way that Alice can cheat is as follows. She starts by preparing two qubits W and X in the state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, and sends X to Bob. The reduced state of Bob's qubit X at this point happens to be the totally mixed state, but this doesn't really matter—Bob is being honest, so he would not measure his qubit until the reveal phase. Alice has not really committed to anything. Time passes and the reveal phase comes. At this point, Alice may effectively decide to reveal that she "committed" $b = 0$ or $b = 1$, whichever she wants.

If she wishes to reveal $b = 0$, she measures W in the standard basis. If she gets the result 0, she sends 00 to Bob, and if she gets 1, she sends 10 to Bob. Bob measures and gets precisely the same outcome as Alice, leading him to believe that she was being honest and had committed $b = 0$ all along.

If Alice instead wishes to reveal $b = 1$, she measures W in the $\{|+\rangle, |-\rangle\}$ basis, interpreting the result as 0 or 1 respectively. (Alternately, she performs a Hadamard transform on W and measures.) She then sends the measurement outcome followed by $b = 1$ to Bob. Suppose Alice's measurement outcome was 0. Then the state of X becomes $|+\rangle$. Similarly, if Alice measures 1, the state of X becomes $|-\rangle$. This is because

$$\begin{aligned}(H \otimes I)\left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle\right) &= \frac{1}{2}|00\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|01\rangle - \frac{1}{2}|11\rangle \\ &= \frac{1}{\sqrt{2}}|0\rangle|+\rangle + \frac{1}{\sqrt{2}}|1\rangle|-\rangle.\end{aligned}$$

When Bob measures to "check" Alice's honesty, he gets always gets the outcome that leads him to believe Alice was honest.

The binding property has failed completely as we suspected.

There have been many attempts to bypass the proof that bit commitment is impossible using quantum information, but they never succeed. Indeed, there are some people who continue to write papers arguing that the proof of the impossibility of quantum bit commitment is possible to circumvent. It isn't—their arguments are always based on a misunderstanding of quantum information, cryptography, or mathematics more generally.

Lecture 20: Bell inequalities and nonlocality

April 4, 2006

So far in the course we have considered uses for quantum information in the settings of computation and cryptography. In this lecture and the next we will consider uses of quantum information in distributive settings. This lecture will focus on using entanglement to achieve stronger correlations than are possible classically between two or more parties that cannot communicate. In the next lecture we will discuss communication complexity, which considers the communication costs for performing various tasks, and we will see that quantum information can sometimes give very large reductions in these costs.

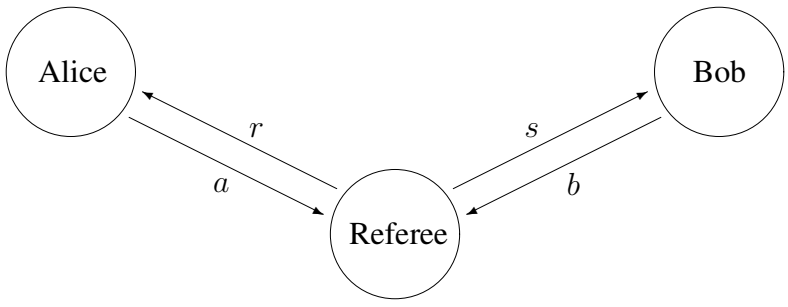
Nonlocal games

We saw very early in the course that entanglement is a useful resource when combined with other resources such as in teleportation and super-dense coding. However, we know that entanglement alone cannot be used by directly for communication. For instance, if Alice and Bob share some entangled state $|\psi\rangle \in \mathcal{A} \otimes \mathcal{B}$, and Alice performs any measurement on her part of the state, the results are completely determined by her reduced state $\mathrm{Tr}_{\mathcal{B}} |\psi\rangle \langle\psi|$ and are independent of any operation that Bob could perform on his half of the state. The situation is similar for Bob.

However, if both Alice and Bob perform measurements, then there can be interesting *correlations* between their measurement outcomes that would not be possible using classical information. These correlations can be studied by means of *nonlocal games*.

In a nonlocal game there are two or more *players* (Alice, Bob, Charlie, Diane, ...) who are viewed as cooperating with one another. A *referee* runs the game, and all communication in the game is between the players and the referee—no communication directly between any of the players is permitted. The referee randomly selects a *question* for each player and sends each question to the appropriate player. Each player sends an *answer* back to the referee, and based on all of the questions and answers the referee determines whether the players win or lose.

For example, if there are two players, a nonlocal game has the following structure:



Here, the referee chooses a pair of questions (r, s) (according to some prespecified distribution), sends r to Alice and s to Bob, and Alice and Bob answer with a and b, respectively. The referee evaluates some predicate on (r, s, a, b) to determine if they win or lose.

We can speak of *classical strategies* where the players must behave classically or *quantum strategies* which allow the players to share an entangled state and to perform quantum measurements to determine their answers to the referees questions. What is interesting is that sometimes quantum strategies can allow the players to win with a much higher probability than they could with any classical strategy.

Example 24 (The GHZ game). Consider the following game, where there are three players: Alice, Bob, and Charlie. Each question and each answer will be a single bit.

The referee chooses a three bit string $r\,s\,t$ uniformly from the set $\{000, 011, 101, 110\}$ and sends r to Alice, s to Bob, and t to Charlie. Their answers must be bits: a from Alice, b from Bob, and c from Charlie. They win if $a \oplus b \oplus c = r \vee s \vee t$ and lose otherwise. The following table lists the winning condition for each possible set of questions:

$r\,s\,t$	$a \oplus b \oplus c$
000	0
011	1
101	1
110	1

What is the maximum probability of winning if Alice, Bob, and Charlie use a classical strategy? First consider a *deterministic strategy*, where each answer is a function of the question received and no randomness is used by the players. Let us write a_r, b_s and c_t to denote the answers that would be given for each choice of r, s, and t. For example, if $a_0 = 1$ and $a_1 = 0$, then Alice always answers the question 0 with 1 and the question 1 with 0. The winning conditions can be expressed by the four equations

$$\begin{aligned} a_0 \oplus b_0 \oplus c_0 &= 0 \\ a_0 \oplus b_1 \oplus c_1 &= 1 \\ a_1 \oplus b_0 \oplus c_1 &= 1 \\ a_1 \oplus b_1 \oplus c_0 &= 1 \end{aligned}$$

Adding the four equations modulo 2 gives $0 = 1$, a contradiction. This means it is not possible for a deterministic strategy to win every time, so the probability of winning can be at most $3/4$ (because at least one of the four question sets will be answered incorrectly). It is easy to devise a strategy that wins $3/4$ of the time ($a_0 = a_1 = b_0 = b_1 = c_0 = c_1 = 1$, for instance), and so the maximum probability of winning is $3/4$.

What about probabilistic strategies? It turns out that for nonlocal games, probabilistic classical strategies can be no better than deterministic strategies. This is because the probability that a probabilistic strategy wins is just an average of the probabilities that some collection of deterministic strategies win. An average of a collection of numbers cannot be larger than the largest of the numbers.

Now let us turn to quantum strategies. Suppose that the three players share the entangled state

$$|\psi\rangle = \frac{1}{2}|000\rangle - \frac{1}{2}|011\rangle - \frac{1}{2}|101\rangle - \frac{1}{2}|110\rangle .$$

This is sometimes called a GHZ state—but also the term GHZ state sometimes refers to the state

$$\frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|111\rangle$$

which is equivalent to $|\psi\rangle$ up to local unitary operations. Each player will use the same strategy:

1. If the question is $q = 1$, then the player performs a Hadamard transform on their qubit of the above state. (If $q = 0$, the player does not perform a Hadamard transform.)
2. The player measures their qubit in the standard basis and returns the answer to the referee.

Alternately, the players measure with respect to the basis $\{|0\rangle, |1\rangle\}$ or $\{|+\rangle, |-\rangle\}$ depending on the question they received.

Let us see how well this strategy works. There are two cases:

Case 1: $r\,s\,t = 000$. In this case the players all just measure their qubit, and it is obvious that the results satisfy $a \oplus b \oplus c = 0$ as required.

Case 2: $r\,s\,t \in \{011, 101, 110\}$. All three possibilities will work the same way by symmetry, so let us assume $r\,s\,t = 011$. Notice that

$$\begin{aligned}|\psi\rangle &= \frac{1}{\sqrt{2}}|0\rangle\left(\frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle\right) - \frac{1}{\sqrt{2}}|1\rangle\left(\frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle\right) \\ &= \frac{1}{\sqrt{2}}|0\rangle|\phi^-\rangle - \frac{1}{\sqrt{2}}|1\rangle|\psi^+\rangle .\end{aligned}$$

Bob and Charlie both receive the question 1, so they both perform a Hadamard transform. It is easy to check that

$$(H \otimes H)|\phi^-\rangle = |\psi^+\rangle \qquad \text{and} \qquad (H \otimes H)|\psi^+\rangle = |\phi^-\rangle$$

so the state becomes

$$(I \otimes H \otimes H)|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle|\psi^+\rangle - \frac{1}{\sqrt{2}}|1\rangle|\phi^-\rangle = \frac{1}{2}\left(|001\rangle + |010\rangle - |100\rangle + |111\rangle\right).$$

When they measure, the results satisfy $a \oplus b \oplus c = 1$ as required. We have therefore shown that there is a quantum strategy that wins every time.

Remark. This example, as well as the others we will discuss, appear in the physics literature not as games but as hypothetical experiments. The game formulation is natural and appealing for theoretical computer scientists, as we are used to thinking about different parties optimizing their performance in various abstract settings. In physics, the notion of a classical strategy is typically

replaced by the notion of a *hidden variable theory*, and good quantum strategies simply coincide with the setting up of the experiment. Theoretically, one finds that the statistics of the experiment could not be predicted by a hidden variable theory.

Although the interpretation is critical in physics, the mathematics is the same in the two settings. Because our focus is on the mathematics, and this is a computer science course, I will stick mainly with the game formulation of these examples.

Example 25 (The CHSH game). The set-up for this game is similar to the GHZ game, except that there are only two players: Alice and Bob. The referee chooses questions $rs \in \{00, 01, 10, 11\}$ uniformly, and Alice and Bob must each answer a single bit: a for Alice, b for Bob. They win if $a \oplus b = r \wedge s$, i.e., the winning conditions are:

rs	$a \oplus b$
00	0
01	0
10	0
11	1

By similar reasoning to the GHZ game, the maximum probability with which a classical strategy can win is $3/4$.

There are two ways to describe the quantum strategy that we will consider. The first way will be more familiar, but the second way is useful and this is a good opportunity to learn something new.

Here is the first way. The entangled state shared by Alice and Bob will be $|\psi\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$. For a given angle $\theta \in [0, 2\pi)$, define

$$|\phi_0(\theta)\rangle = \cos(\theta)\,|0\rangle + \sin(\theta)\,|1\rangle\,,$$
$$|\phi_1(\theta)\rangle = -\sin(\theta)\,|0\rangle + \cos(\theta)\,|1\rangle\,.$$

If Alice receives the question 0, she will measure her qubit with respect to the basis

$$\{|\phi_0(0)\rangle\,, |\phi_1(0)\rangle\},$$

and if she receives the question 1, she will measure her qubit with respect to the basis

$$\{|\phi_0(\pi/4)\rangle\,, |\phi_1(\pi/4)\rangle\}.$$

Bob uses a similar strategy, except that he measures with respect to the basis

$$\{|\phi_0(\pi/8)\rangle\,, |\phi_1(\pi/8)\rangle\} \quad \text{or} \quad \{|\phi_0(-\pi/8)\rangle\,, |\phi_1(-\pi/8)\rangle\},$$

depending on whether his question was 0 or 1.

The second way uses the notion of an *observable*. This term has different meanings to different people, but here is what I mean. Suppose we have a projective measurement $\{\Pi_a : a \in \Gamma\}$ where

the set Γ of measurement outcomes is finite (as usual) and is a subset of the real numbers: $\Gamma \subset \mathbb{R}$. Then the *observable* that corresponds to this measurement is a single Hermitian matrix:

$$A = \sum_{a \in \Gamma} a \Pi_a.$$

In other words, the eigenspaces correspond to the projection operators, and the eigenvalues correspond to the measurement outcomes. If a pure state $|\psi\rangle$ is measured with respect to this observable, then the *expected value* of the outcome is simply $\langle \psi | A | \psi \rangle$.

For instance, if we measure with respect to the standard basis and associate the outcome 1 with $|0\rangle$ and -1 with $|1\rangle$, then the corresponding observable is $|0\rangle\langle 0| - |1\rangle\langle 1| = \sigma_z$. A similar example using the diagonal basis give observable $|+\rangle\langle +| - |-\rangle\langle -| = \sigma_x$.

Now, back to the CHSH game, let us imagine that Alice and Bob both answer ± 1, where we are making the identification $a \leftrightarrow (-1)^a$. Then Alice's observables are as follows:

$$A_0 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \sigma_z \qquad \text{and} \qquad A_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \sigma_x$$

while Bob's observables are

$$B_0 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = H \qquad \text{and} \qquad B_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ -1 & -1 \end{pmatrix} = ??.$$

(I don't know of any special name for B_1.) These observables simply come from translating the first description of the strategy to observables, using some very well known trigonometric inequalities along the way.

Okay, so how well does the strategy do? A boring case analysis will reveal that for all 4 possible question pairs, Alice and Bob will answer correctly with probability $\cos^2(\pi/8) \approx 0.85$, which is better than an optimal classical strategy that wins with probability $3/4$.

We can also use the observables to prove this. Consider the expression

$$\frac{1}{4} \langle \psi | A_0 \otimes B_0 + A_0 \otimes B_1 + A_1 \otimes B_0 - A_1 \otimes B_1 | \psi \rangle .$$

I claim that this is the probability that Alice and Bob win minus the probability they lose. This comes from the fact that $\langle \psi | A_r \otimes B_s | \psi \rangle$ is the expected value of the product of Alice and Bob's ± 1 measurement outcomes. When $rs \in \{00, 01, 10\}$ this is the probability of winning minus the probability of losing on questions (r, s) because they win when their measurement outcomes are the same (have product 1) and lose when they are different (have product -1). In the last case $rs = 11$, they win when their measurement outcomes disagree (have product -1) and lose if they are the same (have product 1), so we put a minus sign in front of the term $A_1 \otimes B_1$ to reflect this. A quick check shows that

$$\langle \psi | A_0 \otimes B_0 | \psi \rangle = \langle \psi | A_0 \otimes B_1 | \psi \rangle = \langle \psi | A_1 \otimes B_0 | \psi \rangle = - \langle \psi | A_1 \otimes B_1 | \psi \rangle = \frac{1}{\sqrt{2}}$$

and so the probability of winning minus the probability of losing is $\frac{1}{\sqrt{2}}$. This means the probability of winning is

$$\frac{1}{2} + \frac{1}{2\sqrt{2}} = \cos^2(\pi/8).$$

Bell Inequalities

You may be wondering why the title of the lecture is "Bell inequalities and nonlocality". In the physical interpretation of nonlocal games, the bounds on classical strategies (or on local hidden variable theories) are typically known as *Bell Inequalities*, and quantum strategies (or experiments) that beat these bounds are said to *violate* a Bell inequality.

Typically you would see the Bell Inequality (known as the CHSH inequality) associated with the second example stated as:

$$A_0B_0 + A_0B_1 + A_1B_0 - A_1B_1 \leq 2.$$

You can probably convince yourself easily that if A_0, A_1, B_0 and B_1 are treated as ordinary variables taking values in $[-1, 1]$, then the inequality must hold. This is equivalent to saying that there is no classical strategy that wins the CHSH game with probability exceeding 3/4. Because there exist observables A_0, A_1, B_0 and B_1 with eigenvalues in $[-1, 1]$ and a state $|\psi\rangle$ such that

$$\langle\psi|A_0B_0 + A_0B_1 + A_1B_0 - A_1B_1|\psi\rangle = 2\sqrt{2}$$

means that this Bell inequality is violated by the corresponding experiment. In both cases, the tensor product can be viewed as being implicit, or simply removed but the assumption that A_a and B_b commute for each choice of a and b.

Tsirelson's bound

You might wonder if it is possible to do better than $\cos^2(\pi/8)$ with a quantum strategy for the CHSH game. The answer is "no" due to Tsirelson's bound:

$$\langle\psi|A_0 \otimes B_0 + A_0 \otimes B_1 + A_1 \otimes B_0 - A_1 \otimes B_1|\psi\rangle \leq 2\sqrt{2}$$

for any choice of observables A_0, A_1, B_0 and B_1 with eigenvalues in $[-1, 1]$ and any state $|\psi\rangle$. This includes the possibility that $|\psi\rangle$ is a state on many qubits, not just two. This can be shown as follows, using the fact that $\|A_0\|, \|A_1\|, \|B_0\|, \|B_1\| \leq 1$ along with some other simple properties of the norm of an operator:

$$\begin{aligned}
&\langle\psi|A_0 \otimes B_0 + A_0 \otimes B_1 + A_1 \otimes B_0 - A_1 \otimes B_1|\psi\rangle \\
&\qquad \leq \|(A_0 \otimes B_0 + A_0 \otimes B_1 + A_1 \otimes B_0 - A_1 \otimes B_1)|\psi\rangle\| \\
&\qquad \leq \|(A_0 \otimes (B_0 + B_1))|\psi\rangle\| + \|(A_1 \otimes (B_0 - B_1))|\psi\rangle\| \\
&\qquad \leq \|(I \otimes B_0)|\psi\rangle + (I \otimes B_1)|\psi\rangle\| + \|(I \otimes B_0)|\psi\rangle - (I \otimes B_1)|\psi\rangle\| \\
&\qquad = \||\phi_0\rangle + |\phi_1\rangle\| + \||\phi_0\rangle - |\phi_1\rangle\|
\end{aligned}$$

for $|\phi_b\rangle = (I \otimes B_b)|\psi\rangle$. Because $\||\phi_b\rangle\| \leq 1$ we have

$$\||\phi_0\rangle + |\phi_1\rangle\| + \||\phi_0\rangle - |\phi_1\rangle\| \leq \sqrt{2 + 2\Re\langle\phi_0|\phi_1\rangle} + \sqrt{2 - 2\Re\langle\phi_0|\phi_1\rangle} = \sqrt{2 + 2x} + \sqrt{2 - 2x}$$

for $x \in [-1, 1]$. It is an easy calculus problem to find the maximum of this expression. It occurs at $x = 0$, giving $2\sqrt{2}$ as required.

Lecture 21: Quantum communication complexity

April 6, 2006

In this lecture we will discuss how quantum information can allow for a significant reduction in the communication costs associated with various distributed tasks. Before we discuss how this is possible, however, it is important to mention that quantum information cannot reduce the cost of direct communication. This is due to Holevo's Theorem.

Holevo's Theorem

Let us begin with a simplified version of the theorem, which is sufficient to give the spirit of Holevo's Theorem, but not too much more.

Holevo's Theorem (informal, simplified version). *If Alice and Bob share no prior entanglement, then it is necessary for them to exchange at least n qubits (or bits) in order for Alice to communicate n classical bits of information to Bob. If they do share prior entanglement, then they must exchange at least $n/2$ qubits.*

I would like to mention the formal version of Holevo's Theorem, even though we have not covered the necessary concepts to understand it properly. The reason is that anyone who plans to continue studying quantum information after this course should not be under the impression that Holevo's Theorem is as simple as the statement above: it is both qualitatively and quantitatively deeper than the simplified version suggests.

Holevo's Theorem (formal version). *Let Σ and Γ be finite, nonempty sets and let $\mathcal{X}$ be the space corresponding to some quantum system. Let $\{\rho_a : a \in \Sigma\} \subset \mathrm{D}(\mathcal{X})$ be a collection of density operators, let $\{M_b : b \in \Gamma\}$ be a POVM on $\mathcal{X}$, and let $p \in \mathbb{R}(\Sigma)$ be a probability vector. Let A and B be random variables taking values in Σ and Γ, respectively, such that $\Pr[A = a] = p[a]$ and $\Pr[B = b \,|\, A = a] = \mathrm{Tr}(M_b \rho_a)$. Then*

$$I(A : B) \leq S\left(\sum_{a \in \Sigma} p[a]\rho_a\right) - \sum_{a \in \Sigma} p[a] S(\rho_a).$$

In this theorem, $I(A : B)$ is the *mutual information* between the random variables A and B, and $S(\cdot)$ is the *von Neumann entropy* function. We may interpret the statement of the theorem as follows. The random variable A, which has associated probability vector p, determines the "symbol" a (or string a if you prefer) that Alice wishes to send to Bob. Alice samples A to find some a, prepares a quantum state ρ_a, and sends it to Bob. Bob then measures this state with respect to some POVM $\{M_b : b \in \Gamma\}$. The random variable B describes the outcome of this measurement. You could imagine that $\Gamma = \Sigma$, and Bob is hoping that $\Pr[A = B]$ is large in order

to learn a, but the theorem is more general. The theorem implies that the amount of information that B contains about A (which is $I(A:B)$) is at most the quantity on the right-hand-side of the inequality. It is a simple property of the von Neumann entropy that this quantity, sometimes called the *Holevo quantity* or *Holevo χ quantity*, cannot be larger than the logarithm of the dimension of $\mathcal{X}$. This puts a lower bound on the size of $\mathcal{X}$ required for any given amount of information to have been transmitted from Alice to Bob.

Holevo's Theorem is easily proved once a fundamental property of the von Neumann entropy, known as *strong subadditivity*, is proved. Proving strong subadditivity, on the other hand, is highly nontrivial but not beyond your reach—for instance it takes about two lectures to prove in my advanced quantum information theory graduate course.

Communication complexity

Although quantum information cannot reduce the direct cost of communication as Holevo's Theorem implies, there are many distributed computational tasks that require communication but not direct communication in an information theoretic sense. Communication complexity studies such problems.

The general type of problem considered by communication complexity is as follows. Alice receives an input string $x \in \{0,1\}^n$ and Bob receives an input string $y \in \{0,1\}^n$. Their goal is to compute $f(x,y)$ for some function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. We can also consider the more general case where Alice and Bob receive strings of different lengths and the output of the function f is longer than just one bit, but the simpler model will be sufficient for this lecture. They wish to accomplish this task using as little communication as possible. We can measure the amount of communication in the number of bits or qubits sent, depending on whether we are considering classical or quantum communication complexity. For simplicity let us be more precise and say that Bob should learn the value $f(x,y)$—if instead we required that both Alice and Bob learn $f(x,y)$, the amount of communication required would differ by at most one bit. We are typically interested in the cost for the *worst case* choice of x and y, and will consider the minimum such cost over all possible protocols.

There are several variants of the communication complexity model that one may consider. We may consider quantum or classical communication complexity; we may require the answer to be correct with probability 1 or allow a small error probability ε that the wrong answer is found; and we may allow or disallow shared randomness or entanglement. Other variations, such as requiring the communication to go only one way or only allowing communication from Alice and Bob to a third party referee, are also frequently studied.

It is worth mentioning one formality of the communication complexity model, to make sure the model is properly understood. A protocol for a given function should completely specify the structure of the communication: Alice sends k_1 bits (or qubits) to Bob, Bob sends k_2 bits (or qubits) to Alice, Alice sends k_3 bits (or qubits) to Bob, and so on for some fixed number of rounds. The inputs x and y only influence the contents of these messages, not the sizes or timing of the messages or the number of rounds. This disallows complications such as "communicating by not communicating".

We can gain some insight about communication complexity by considering some example functions. The following three functions are important examples.

1. **Equality:**
$$\mathrm{EQ}(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y. \end{cases}$$

2. **Modulo 2 inner product:**
$$\mathrm{IP}(x,y) = x \cdot y.$$

3. **Disjointness:**
$$\mathrm{DISJ}(x,y) = \begin{cases} 1 & \text{if } x_i \wedge y_i = 1 \text{ for some } i \in \{1,\ldots,n\} \\ 0 & \text{otherwise.} \end{cases}$$

Let us first consider the classical communication complexity of the equality function. If we restrict our attention to deterministic protocols (that are correct with certainty), then nothing non-trivial is possible for equality: n bits of communication are both necessary and sufficient. We express this as
$$D(\mathrm{EQ}) = n.$$
In general, $D(f)$ denotes the deterministic communication complexity. Note that all functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ have $D(f) \leq n$, because Alice can always send her entire input x to Bob, who then can just evaluate $f(x,y)$. It is usually not too hard to put lower bounds on $D(f)$ for various functions f—I won't show you how to prove $D(\mathrm{EQ}) \geq n$, but you can probably convince yourself that it is true.

In the probabilistic case, a significant improvement is possible. A probabilistic protocol that has error bounded by $\varepsilon > 0$ exists that requires just $O(\log(n/\varepsilon))$ bits of communication. We will express this as
$$R(\mathrm{EQ}) = O(\log n)$$
(with $\varepsilon = 1/3$ being implicit if we do not specify the error). In fact, if we assume that Alice and Bob share $O(\log n)$ random bits, then only a constant number of bits of communication are required for constant error.

Let us briefly discuss how this is possible. The idea is to use an error correcting code with the right properties. Specifically, we want an error correcting code
$$E : \{0,1\}^n \to \{0,1\}^m$$
where $m = cn$ for some constant c, and
$$\mathrm{dist}(E(x), E(y)) \geq \delta m \quad \text{(for all } x \neq y)$$
for some constant $\delta > 0$. Such codes are known to exist.[5] Now, Alice and Bob both encode their strings using this code, Alice randomly chooses $i \in \{1,\ldots,m\}$, and sends $(E(x)_i, i)$ to Bob.

[5] I believe Justesen codes were the first examples of such codes. For these codes we can, for instance, take $c = 4$ and $\delta = 1/12$. I presume there are better constructions known.

This requires $O(\log m) = O(\log n)$ bits of communication. Bob compares $E(x)_i$ with $E(y)_i$. If $E(x)_i = E(y)_i$, Bob concludes $x = y$, and otherwise he concludes $x \neq y$. If $x = y$, Bob will be right, while if $x \neq y$ he will conclude with constant probability that $x \neq y$. The process can be repeated a constant number of times to decrease the probability of error. In the setting where Alice and Bob have shared randomness, the reduction in the cost to a constant number of bits comes from the fact that Alice does not need to send the index i to Bob.

For the other two example functions, it turns out that even using randomness and allowing a small probability of error, no asymptotic reduction in communication complexity is possible over the trivial protocol: we have $R(\mathrm{IP}) = \Theta(n)$ and $R(\mathrm{DISJ}) = \Theta(n)$. Unlike the typical situation for deterministic communication complexity, proving lower bounds for randomized protocols can be very difficult—so I will not attempt to prove to you that any protocol for disjointness has communication cost $\Omega(n)$ (but it is true).

Quantum communication complexity

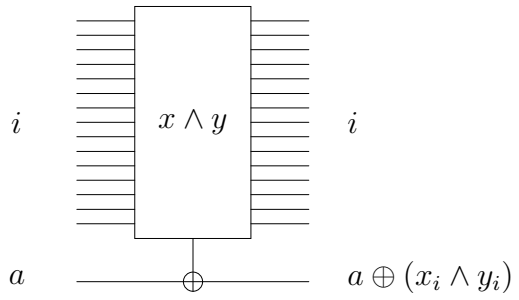
It is a natural question to ask whether quantum information can reduce communication complexity. In light of Holevo's Theorem, it may be tempting to think not, but in fact significant reductions are possible.

For example, let us show that the quantum communication complexity of disjointness, denoted $Q(\mathrm{DISJ})$, satisfies $Q(\mathrm{DISJ}) = O(\sqrt{n}\log n)$. More specifically, $Q(f)$ refers to quantum communication complexity allowing a small probability of error and disallowing shared entanglement. It turns out that $Q(\mathrm{DISJ}) = \Theta(\sqrt{n})$; the elimination of the logarithmic factor is clever optimization of the method we will see, while the lower bound requires a rather difficult proof.

Before describing the protocol, let us consider the following problem. Suppose that the strings x and y are fixed, but unknown to us, and imagine that we have access to a black box B_g for a function $g : \{0,1\}^n \to \{0,1\}$ satisfying

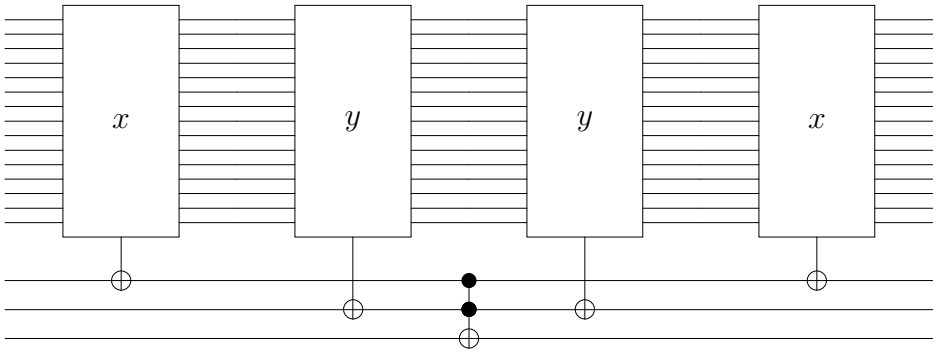
$$g(i) = x_i \wedge y_i.$$

As you would assume, the black box is defined on strings of length $k = \lceil \log_2 n \rceil$, each index i is encoded as a string of length k, and the black box implements the following reversible operation:



Using $O(\sqrt{n})$ queries to this black box, Grover's Algorithm could determine with high probability whether there exists an index i for which $g(i) = 1$ (which is equivalent to $x_i \wedge y_i = 1$).

Now, the way that Alice and Bob will solve the disjointness problem is essentially to run Grover's algorithm. Of course, neither of them has a black box for $x \wedge y$ as above, but they can implement this black box in a distributed fashion. Specifically, Alice can implement such a black box for x in place of $x \wedge y$ (because she knows x), and Bob can do likewise for y. Then they can combine their black boxes to simulate the one for $x \wedge y$ as follows:



Of course this requires communication: they must exchange $O(\log n)$ qubits in order to implement this circuit. However, that is good enough for what we need: Alice (say) can effectively run Grover's Algorithm to determine whether $x_i \wedge y_i = 1$ for some i, exchanging $O(\log n)$ qubits with Bob for each query. Because the total number of queries needed is at most $O(\sqrt{n})$, the total amount of communication is $O(\sqrt{n} \log n)$.

For other functions quantum information does not help. For instance, $Q(\mathrm{IP}) = \Theta(n)$. The idea behind the proof of this bound is that anything asymptotically less than $\Theta(n)$ would allow for a violation of Holevo's Theorem.

Exponential separation

It turns out that an exponential separation between quantum and classical communication complexity is possible, provided we allow for promises on the inputs. Ran Raz proved that the following problem gives such a separation.

Alice receives classical descriptions of (i) a unit vector $|\psi\rangle \in \mathbb{C}^n$ and (ii) a projective measurement $\{\Pi_0, \Pi_1\}$ on $\mathbb{C}^n$, and Bob receives a classical description of a unitary operator $U \in \mathrm{L}(\mathbb{C}^n)$. It is promised that one of $\|\Pi_0 U |\psi\rangle\|^2$ or $\|\Pi_1 U |\psi\rangle\|^2$ is close to 1 (and so the other is close to 0), and the goal is to output 0 or 1 accordingly. It is easy for Alice and Bob to solve this problem using $O(\log n)$ qubits of communication: Alice sends $|\psi\rangle$ to Bob, Bob applies U and sends it back to Alice, and Alice measures $U |\psi\rangle$ with respect to $\{\Pi_0, \Pi_1\}$. The challenging part is to prove that no classical probabilistic protocol can achieve this—it is possible to prove a lower bound of $\Omega(n^{1/4}/\log n)$ for the classical communication complexity. The proof is very difficult.

It is still an open question whether an exponential improvement of classical over quantum communication complexity is possible for a total function f (meaning that no promises are allowed in the corresponding problem).

Quantum fingerprinting

Finally, I would like to mention a variant of the communication complexity model where it is possible to prove an exponential separation between quantum and classical complexities for a total function (i.e., not a promise problem). It is a very weak model called the *simultaneous message passing model*. As before, Alice receives an input string x and Bob receives y, but now Alice and Bob cannot communicate with one another at all. Instead, they must both send a single message to a third party, called the referee. The referee does not receive any input, but is required to give the output of the function f based on the messages received from Alice and Bob.

One can consider several variants of this model—we will compare the classical probabilistic model with the quantum model, where the referee is allowed a small probability of error in both cases. No shared randomness or entanglement will be permitted between Alice and Bob. (If we do allow shared randomness or entanglement, the example to be considered no longer gives an interesting separation between quantum and classical.)

The separation will be for the equality function. If we write $R^{\|}(f)$ to denote the randomized communication complexity of f in the simultaneous message passing model, then we have

$$R^{\|}(\mathrm{EQ}) = \Theta(\sqrt{n}).$$

In contrast, using a similar notation for the quantum case, we have

$$Q^{\|}(\mathrm{EQ}) = O(\log n).$$

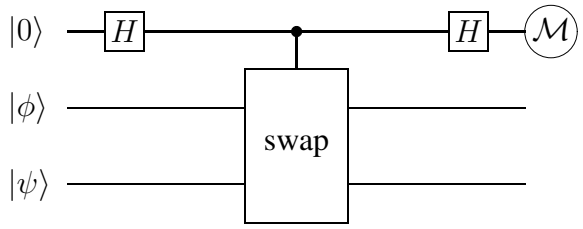
The technique that establishes that $Q^{\|}(\mathrm{EQ}) = O(\log n)$ is called *quantum fingerprinting*. The idea is that Alice will prepare a small quantum state $|\psi_x\rangle$ given x, and Bob will prepare $|\psi_y\rangle$ given y. We want that each $|\psi_x\rangle$ is a quantum state on a number of qubits that is logarithmic in n, but so that $|\psi_x\rangle$ and $|\psi_y\rangle$ are very different when $x \neq y$. The state $|\psi_x\rangle$ is a "fingerprint" of x—it does not contain enough information for someone to recover x from it, but it can be used to distinguish x from another string y with a different fingerprint. The referee will then attempt to check whether the fingerprints $|\psi_x\rangle$ and $|\psi_y\rangle$ are the same or different.

Of course it is not possible to come up with states $\{|\psi_x\rangle \,:\, x \in \{0,1\}^n\}$ that form an orthonormal set, because this would require that the fingerprints are too large: at least n qubits. So instead what we will do is to make them pairwise "almost orthogonal": $|\langle\psi_x|\psi_y\rangle|$ should be small whenever $x \neq y$. We can do this using exactly the same kind of error correcting codes we discussed in the first part of the lecture. Specifically, we let

$$|\psi_x\rangle = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} |i\rangle \, |E(x)_i\rangle \,.$$

Then, assuming $\mathrm{dist}(E(x), E(y)) \geq \delta m$ for $x \neq y$, we have $|\langle\psi_x|\psi_y\rangle| \leq 1 - \delta$. Replacing $|\psi_x\rangle$ with some constant number of copies of $|\psi_x\rangle$ allows this bound on the inner product to be decreased to an arbitrarily small constant. The number of qubits required for these fingerprints is small: $O(\log n)$ qubits.

Now, how can the referee determine whether $|\psi_x\rangle = |\psi_y\rangle$ or $|\langle\psi_x|\psi_y\rangle|$ is small? You've already answered this question (in a slightly simplified form) in question 3 of homework assignment 1. In that question, you were asked to consider this circuit:



There were many ways to formulate you answer, but one way was to say

$$\Pr[\text{outcome is } 0] = \frac{1}{2} + \frac{|\langle\phi|\psi\rangle|^2}{2}$$

$$\Pr[\text{outcome is } 1] = \frac{1}{2} - \frac{|\langle\phi|\psi\rangle|^2}{2}.$$

This generalizes to states $|\phi\rangle$ and $|\psi\rangle$ on any number of qubits. Thus, the referee can simply perform this procedure (sometimes called the *swap test*) on $|\psi_x\rangle$ and $|\psi_y\rangle$. If he measures 0, he determines that $x = y$, and if he measures 1 he determines that $x \neq y$.

If it were the case that $x = y$, then the referee always answers correctly because he will always measure 0. If $x \neq y$, the referee might make a mistake and conclude incorrectly that $x = y$, but if Alice and Bob send some constant number of copies of their fingerprints, the probability of error can be reduced to any desired positive constant in the usual way.

Lecture 22: Quantum computational complexity

April 11, 2006

This will be the last lecture of the course—I hope you have enjoyed the lectures. There is still much that we have not had time to discuss, and of course there is a great deal that remains to be discovered, but we have to stop somewhere...

In this lecture, I'm going to give a very brief introduction to quantum computational complexity theory. Because many of you have no prior background in classical computational complexity, we will be severely limited in terms of what we can cover, but I hope to at least give you the feel for some of the topics considered in quantum complexity.

Promise Problems and Complexity Classes

Let us start with the notion of a *promise problem*. We have talked about promise problems in the black box setting, but for this lecture we will assume the input is given explicitly and there are no black boxes. A promise problem is a computational problem where two disjoint sets of inputs (*yes* inputs and *no* inputs) must be distinguished. For example, here is the Graph Isomorphism problem:

Graph Isomorphism (GI)

Input:	Two simple, undirected graphs G_0 and G_1.
Yes:	G_0 and G_1 are isomorphic ($G_0 \cong G_1$).
No:	G_0 and G_1 are not isomorphic ($G_0 \ncong G_1$).

Formally, a promise problem is a pair $A = (A_{\text{yes}}, A_{\text{no}})$ with $A_{\text{yes}}, A_{\text{no}} \subseteq \{0,1\}^*$ (i.e., A_{yes} and A_{no} are sets of binary strings) that satisfies $A_{\text{yes}} \cap A_{\text{no}} = \varnothing$. Strings in $A_{\text{yes}} \cup A_{\text{no}}$ are said to *satisfy the promise*, and may be thought of as "valid" inputs. It is not required that $A_{\text{yes}} \cup A_{\text{no}} = \{0,1\}^*$. In the Graph Isomorphism problem, for example, a string $x \in \{0,1\}^*$ that encodes a pair of graphs (G_0, G_1) is an element of GI_{yes} if $G_0 \cong G_1$, and is an element of GI_{no} if $G_0 \ncong G_1$. A string that does not encode a pair of graphs at all is in neither set, and is said to *violate the promise*. You may think of such strings as "don't care" inputs.

Traditionally, complexity theory has dealt mostly with non-promise decision problems, where you would essentially require $A_{\text{yes}} \cup A_{\text{no}} = \{0,1\}^*$. In this case, the decision problem is identified with the *language* A_{yes}, and it is implicit that $A_{\text{no}} = \{0,1\}^* \backslash A_{\text{yes}}$. You don't generally bother writing a subscript "yes" in this case—you just talk about a language A, which contains all of the strings for which the correct answer is yes. There are some aspects of the complexity theory of languages that are lost when treating things in more generality by allowing promises, but for this lecture it won't make any difference. I have chosen to discuss promise problems rather than

non-promise problems mostly because the promise problem formulation turns out to be useful and this is a good opportunity to tell you about it.

A *complexity class* is just a set of promise problems, usually coinciding with some particular resource constraint and/or computational model. For example, the following complexity classes are defined by placing time or space constraints on the deterministic and probabilistic Turing machine models:

P	The class of promise problems solvable in polynomial time on a deterministic Turing machine.
BPP	The class of promise problems solvable in polynomial time on a bounded error probabilistic Turing machine (correct on every input with probability at least 2/3).
PP	The class of promise problems solvable in polynomial time on an unbounded error probabilistic Turing machine (correct on every input with probability strictly greater than 1/2).
PSPACE	The class of promise problems solvable in polynomial space on a deterministic Turing machine.
EXP	The class of promise problems solvable in exponential time on a deterministic Turing machine.

Other complexity classes can be defined based on the notion of efficient verification. One of the most important ones is NP; a promise problem $A = (A_{\text{yes}}, A_{\text{no}})$ is in the class NP if and only if there exists (i) a polynomial p, and (ii) a polynomial-time computable predicate V, such that these two properties are satisfied:

Completeness: If $x \in A_{\text{yes}}$ then there exists a string y of length $p(|x|)$ such that $V(x, y) = 1$. The string y is a *proof* (or *certificate* or *witness*) that $x \in A_{\text{yes}}$.

Soundness: If $x \in A_{\text{no}}$ then $V(x, y) = 0$ for every string y of length $p(|x|)$.

You can also define NP in terms of *nondeterministic* Turing machines, but the definition above is more intuitive.

One can define a similar class where the verification is probabilistic: a promise problem $A = (A_{\text{yes}}, A_{\text{no}})$ is in MA if and only if there exists: (i) a polynomial p, and (ii) a polynomial-time probabilistic algorithm V, such that these two properties are satisfied:

Completeness: If $x \in A_{\text{yes}}$ then there exists a string y of length $p(|x|)$ such that V accepts (x, y) with probability at least $99/100$.

Soundness: If $x \in A_{\text{no}}$ then V rejects (x, y) for every string y of length $p(|x|)$ with probability at least $99/100$.

Several other classes that abstract the notion of efficient verification can be defined based on the notion of an *interactive proof system*. These and other related classes have played an important role in complexity theory over the last 20 years.

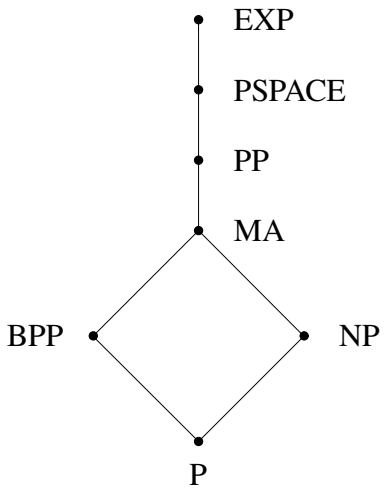


Figure 4: Relations among some classical complexity classes

Figure 4 describes known inclusions among the classes defined above. This is a typical type of figure for showing inclusions—one class is contained in another if you can get from the smaller class to the larger one by following lines upward. For example, BPP and NP are both contained in MA, and both contain P. The diagram gives no information about the relationship between NP and BPP, reflecting the fact that it is not known whether BPP $\subseteq$ NP, NP $\subseteq$ BPP, or if they are incomparable. The diagram also does not indicate proper inclusions, which are often hard to come by in complexity theory. For instance, there is only one known proper inclusion among the classes in the diagram: P $\subsetneq$ EXP. It is not known if P $=$ PSPACE or not, or if NP $=$ EXP or not, for instance. If you can prove P $\neq$ NP, which most people conjecture is the case, then you will have solved a major open problem and will receive a \$1 million prise from the Clay Mathematics Institute. Of course, the \$ 1 million prise would be nothing in comparison to having proved P $\neq$ NP—you can't buy mathematical immortality for a mere \$ 1 million.

Quantum complexity classes

Uniform quantum circuits

We can define quantum complexity classes in a similar way to the classes defined above, except based on quantum computations instead of classical (deterministic or probabilistic) computations. Although it is possible to define quantum Turing machines for the purpose of doing this, it turns out to be easier and equivalent to use the quantum circuit model. Let us briefly discuss some assumptions that we need to make concerning quantum circuits in order to have a good model for complexity theory.

One technical concern that must be addressed when we discuss quantum circuits for complexity classes is the notion of *uniformity*. Suppose that we have a collection of quantum circuits

$$\{Q_x : x \in \{0,1\}^*\},$$

one circuit Q_x for each possible string x. Then we say that this collection is *polynomial-time uniformly generated* (or sometimes people say *polynomial-time uniform* or *polynomial-time generated*) if there exists a deterministic polynomial-time algorithm that, on input x, outputs a classical description of Q_x.

Secondly, we need to fix a set of gates that are permitted. Although you could make different (but equivalent) choices, the following collection of gates works out nicely:

1. Toffoli gates,
2. Hadamard gates,
3. i-phase shift gates (which induce the transformations $|0\rangle \mapsto |0\rangle$ and $|1\rangle \mapsto i\,|1\rangle$ on a single qubit),
4. initialization gates (which input nothing and output a qubit in state $|0\rangle$), and
5. trace-out gates (which take one qubit as input and produce no output, effectively tracing out that qubit).

This collection is *universal* in the sense that any admissible operation from n qubits to m qubits can be approximated with arbitrarily good accuracy with a circuit composed of these gates. We will assume all circuits are composed of these gates for the rest of the lecture. A circuit will be said to be of type (n, m) if it has n input qubits and m output qubits.

BQP

Now we are ready to define our first quantum complexity class: BQP. It is intended that this class abstracts the notion of a promise problem that could be solved efficiently with a quantum computer. Formally, a promise problem $A = (A_{\text{yes}}, A_{\text{no}})$ is in BQP if there exists a polynomial-time generated family $\{Q_x : x \in \{0,1\}^*\}$ of type $(0, 1)$ quantum circuits such that:

1. If $x \in A_{\text{yes}}$, then $\langle 1|Q_x|1\rangle \geq 99/100$, and
2. if $x \in A_{\text{no}}$, then $\langle 1|Q_x|1\rangle \leq 1/100$.

The idea here is that the input string x is "hard coded" into the circuit Q_x. When you run Q_x, you start with no qubits, and end up with a 1 qubit mixed state (which we are simply denoting Q_x, because it is produced by the circuit Q_x). If you measure this qubit with respect to the standard basis, you get the outcome 1, which we interpret as "yes", with probability $\langle 1|Q_x|1\rangle$.

A decision problem version of factoring is is BQP because of Shor's algorithm:

Integer Factoring

Input:	Positive integers (N, k) with $2 \leq k \leq N$.
Yes:	There exists a proper factor of N in the range $\{2, \ldots, k\}$.
No:	There is no proper factor of N in the range $\{2, \ldots, k\}$.

It also happens to be the case that both Integer Factoring and its complement (where we switch the yes and no conditions) are in NP. We would typically state this as *Integer Factoring* $\in$ NP$\cap$co-NP.

A natural question to ask is: how does BQP compare with the classical complexity classes defined above? It is fairly easy to prove that BPP $\subseteq$ BQP, based on the observation that quantum circuits can simulate coin-flips and perform deterministic computations efficiently. It is also easy to prove BQP $\subseteq$ EXP: if all of the gates of Q_x are replaced by matrices and all of the matrix arithmetic is done explicitly, then at most exponential time (and space) is required to calculate $\langle 1|Q_x|1\rangle$ given a description of the circuit Q_x. This method can be improved to show BQP $\subseteq$ PSPACE by performing the same computations in a very space-efficient manner. It may not be obvious how this can be done, but there are well-known methods that can be used to accomplish it. In essence, if you are willing to repeatedly calculate certain numbers over and over, once each time you need them, you don't need to actually store much information to perform matrix arithmetic.

We can do somewhat better than PSPACE as an upper bound on BQP, though; we can prove BQP $\subseteq$ PP. To establish this, we will consider a classical probabilistic algorithm for simulating a quantum circuit. There is a catch, of course, which is that the probabilistic algorithm will have *unbounded error*: although it will work correctly a majority of the time, its probability of outputting yes and no will be so close to 1/2 that it will be useless from a practical point of view. This, however, is the way that the class PP is defined, and it should not be viewed as representing a practical notion of probabilistic computation. Instead, the class PP represents a different type of problem that can in some sense be solved if one has the ability to count an exponential number of things quickly.

Before describing this algorithm, it will be helpful to get rid of and change some gates in a given circuit Q_x. The first thing to notice is that the initialization and trace-out gates are useless for computation—they are only there so that we can say certain things more efficiently when we state theorems or discuss admissible operations. Therefore, you might as well imagine that Q_x has type (n, n) for some integer n, and only contains unitary (Toffoli, Hadamard, and i-phase shift) gates. You start the circuit on the state $|0^n\rangle$ and ignore all of the qubits except one of them (say the first one) after all of the gates are applied. Lastly, each of the i-phase shift gates can be replaced with a two qubit gate that induces the transformation

$$\begin{aligned}
|00\rangle &\mapsto |00\rangle \\
|01\rangle &\mapsto |01\rangle \\
|10\rangle &\mapsto |11\rangle \\
|11\rangle &\mapsto -|10\rangle .
\end{aligned}$$

Here, the first qubit is the qubit on which the phase shift originally acted and the second is a single new qubit that is added to the circuit. This new qubit can be thought of as the "real/imaginary qubit", and starts in the state $|0\rangle$ like all of the other qubits. You simply ignore it when the circuit is done. Effectively, this qubit is doubling the dimension of the space we are working with so that we can identify $\mathbb{C}$ with $\mathbb{R}^2$. (You don't actually even need the above two qubit gate if you are willing to do some fancy footwork with Toffoli and Hadamard gates alone, but it will not make the proof that BQP $\subseteq$ PP any harder to include it.)

Now we are ready to state the algorithm that probabilistically simulates a given quantum circuit Q_x. This algorithm will output a single bit, and we just need that the probability to output a 1 is greater than the probability to output 0 if and only if the same is true of Q_x.

1. Let $B \leftarrow 0$ be a single bit register that we will think of as a sign bit. Let $Z \leftarrow 0^n$ represent the initial state of the n qubits of Q_x. Simulate the effect of each gate of Q_x on Z as follows:
   - If the gate is a Toffoli gate, simply modify Z accordingly.
   - If the gate is a Hadamard gate, flip a coin to determine the new state of the corresponding bit of Z. If the induced transformation was $1 \mapsto 1$, toggle the sign bit.
   - If the gate is the two-qubit gate we used to replace the i-phase shift gate, modify Z appropriately and toggle the sign bit if the transformation induced was $11 \mapsto 10$.
2. Repeat step 1 a second time, using variables B' and Z' in place of B and Z.
3. If $Z = Z'$ and the first bit of Z and Z' is a 0, then output $B \oplus B'$.

   If $Z = Z'$ and the first bit of Z and Z' is a 1, then output $\neg B \oplus B'$.

   If $Z \neq Z'$, then "give up". This means: flip a fair coin and output 0 or 1 accordingly.

You might take a look at this algorithm and wonder how it could possibly work. The idea is that the probability that the algorithm outputs 1 minus the probability that it outputs 0 is proportional to $\langle 1|Q_x|1\rangle - \langle 0|Q_x|0\rangle$. The constant of proportionality happens to be tiny—exponentially small in the number of Hadamard gates—but that is okay. All that we care about is whether the algorithm outputs 1 with probability greater than or less than $1/2$.

Quantum proofs: QMA

Finally, I would like to just mention an interesting quantum complexity class that gives a quantum analogue of NP (or really MA because the verification is probabilistic). The class is called QMA. Formally, a promise problem $A = (A_{\text{yes}}, A_{\text{no}})$ is in QMA if there exists (i) a polynomial p, and (ii) a polynomial-time generated family $\{Q_x : x \in \{0,1\}^*\}$ of circuits, where each Q_x is of type $(p(|x|), 1)$, such that these two properties hold:

Completeness: If $x \in A_{\text{yes}}$, then there exists a state ρ on $p(|x|)$ qubits such that

$$\langle 1|Q_x(\rho)|1\rangle \geq \frac{2}{3}.$$

The state ρ is essentially a "quantum proof" or "quantum certificate" that establishes that $x \in A_{\text{yes}}$.

Soundness: If $x \in A_{\text{no}}$, then for every state ρ on $p(|x|)$ qubits it holds that

$$\langle 1|Q_x(\rho)|1\rangle \leq \frac{1}{3}.$$

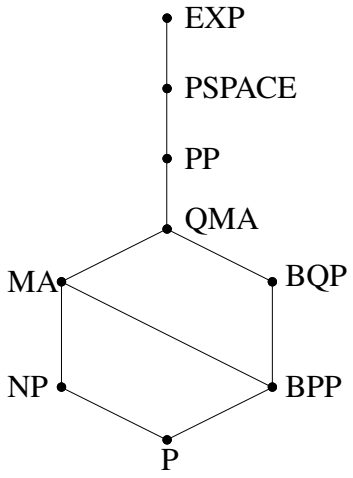


Figure 5: Containments among classes discussed in this lecture.

It can be shown that even this class is contained in PP. The proof is more difficult than the proof that BQP $\subseteq$ PP, so I won't have time to discuss it.

An interesting question is whether quantum certificates can be more powerful than classical ones. An example of a problem that is known to be in QMA but not known to be in MA (or even QMA but restricting the proof to be classical) is *Group Non-membership*: given a group and an element of some common super-group, answer "yes" if the element is not contained in the group, and "no" otherwise. This is another example of how the structure of groups can be exploited by quantum computational models.

Figure 5 contains a diagram of the classes we have discussed in this lecture.