Quantum Computers

David Zaret

Fall 2016

Contents

1 Quantum Bits

The quantum analogue of a classical bit is a **quantum bit**, or **qubit**. A qubit is a two-state quantum system, represented by a vector in a two-dimensional Hilbert space H^2.

Possible physical realizations of a qubit include a spin-1/2 particle; the polarization states of a photon; or a hydrogen atom in its two lowest energy levels.

For any physical realization of a qubit, we assume that the qubit's state space features a special basis $\{|0\rangle, |1\rangle\}$, termed the **computational basis**. The computational basis is distinguished by two fundamental properties. First, we assume that our quantum computer is equipped with instrumentation that enables us to set the state of a qubit to either of the basis states, $|0\rangle$ or $|1\rangle$, at our own choosing. And second, we assume that the computer is equipped with instrumentation that measures the state of the qubit accurately, when the qubit is in one of these distinguished states.

For example, suppose that we represent a qubit as a spin-1/2 particle. Then one possible computational basis consists of the eigenstates for the z-component of spin, $|-\rangle_z$ and $|+\rangle_z$. In this case the relevant "instrumentation" includes a Stern-Gerlach magnet oriented along the z-axis. We arbitrarily select one of the basis states as corresponding to the outcome 0, and the other as corresponding to the outcome 1. For example, we can identify

$$\begin{aligned} |0\rangle &= |+\rangle_z \\ |1\rangle &= |-\rangle_z. \end{aligned}$$

According to this convention, if we perform a measurement and the particle is deflected up, we return 0; if the particle is deflected down, we return 1.

Unless otherwise specified, we always assume that a measurement is performed in the coordinate basis. In other words, the default assumption is that we're measuring the observable with respect to which $\big|0\big\rangle$ and $\big|1\big\rangle$ are eigenvectors.

The general state of a qubit in the computational basis has the form,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{1}$$

where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. As we have discussed, (1) represents a physically possible state for any α and β that satisfy the normality condition. However, we do *not* assume that our quantum computer is capable of setting the state of a qubit to any arbitrary state in a single elementary step. To do so may in general take many steps, because α and β are in general infinite precision numbers.

When we measure a qubit whose state is described by $|\psi\rangle$ in (1), we obtain the result 0 with probability $|\alpha|^2$, and 1 with probability $|\beta|^2$. Hence a single measurement reveals only one bit of information about the prior state of the qubit. In particular, a single measurement tells us nothing at all about the values of α and β. We would need an infinite number of measurements on systems prepared in the same state in order to obtain these values.

This elementary observation points to a fundamental feature of quantum computation. The state of a qubit may contain a great deal of information (think of a 2^n-dimensional analogue of (1)). But our only way of getting at this information is by making a measurement; and the quantum mechanical projection postulate ensures that a measurement can provide us only with a single bit of information. The fundamental challenge of quantum computation is somehow to exploit in a useful way the information which is embedded in a quantum state, despite the limitations imposed by the projection postulate.

It will often be convenient to work with the following coordinate representation:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$\alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

Unless specified otherwise, we'll always assume that the coordinates refer to the computational basis.

2 Quantum Gates and Circuits

We will represent a quantum computer formally using **quantum gates**, which are the quantum mechanical analogues of classical logic gates.

Our ultimate goal is to implement quantum algorithms; and we will represent these algorithms using sequences of quantum gates, or *quantum circuits.* The quantum circuits we need in order to solve non-trivial problems can get pretty complex. But, just as in the classical case, we will construct our complex quantum circuits from a small number of elementary gates. In what follows, we will start building up a collection of the elementary quantum gates that we'll need for the implementation of quantum algorithms.

2.1 Single Qubit Gates

A **one-qubit gate** is an operation on a qubit that consists of a unitary transformation

$$U : H^2 \to H^2.$$

In other words, a one-qubit quantum gate specifies a linear operation,

$$\begin{aligned} |0\rangle &\mapsto a|0\rangle + c|1\rangle \\ |1\rangle &\mapsto b|0\rangle + d|1\rangle, \end{aligned}$$

such that the matrix

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

is unitary. Recall that to say this matrix is unitary is to say that it satisfies

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} a^* & c^* \\ b^* & d^* \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

So the basic operation in quantum computation looks like this. We take a unitary transformation U, apply it to a qubit

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix},$$

and wind up with something of the form,

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} a \cdot \alpha + b \cdot \beta \\ c \cdot \alpha + d \cdot \beta \end{pmatrix}$$

This is certainly simple enough; but what does it *mean*? What does the mathematical description tell us about what's going on in the physical world?

First, recall that the time evolution of a quantum system can be described by applying a unitary operator to the state vector of the system. So when we're talking about quantum gates, we're talking about the deterministic time evolution of the system comprised by the qubit and the gate, as specified by Postulate 5.

Of course, the fact that the time evolution of a quantum system can be described by a unitary transformation doesn't mean that just any old unitary transformation corresponds to a physically possible time evolution operator. Nature gives us the Hamiltonians for physical systems; we can't just make them up. Or can we? It turns out that any arbitrary unitary transformation can be expressed (not exactly, but with arbitrary accuracy) as the product of certain elementary unitary transformations. In particular, there is a set S of unitary transformations such that each element of S is a one or two qubit transformation that does correspond to a physically realizable system, and any n-dimensional unitary transformation can be decomposed into the product of a finite number of members of S. Such a set S (it's not unique) is called a *universal* set. In the real world, this means that any unitary transformation can be physically realized in principle by stringing together the physical devices that correspond to members of a universal set of transformations.

We'll return later to the idea of universal sets of transformations, and to the question of how one can physically realize the members of a universal set. For now, we'll just assume that any unitary transformation we define can be physically realized. We'll proceed, then, to look at some of the unitary transformations that are of interest from the standpoint of quantum computation.

2.1.1 Quantum NOT

The quantum-NOT or X gate inverts the computational basis states $|0\rangle$ and $|1\rangle$. In other words, if we apply quantum-NOT to a state

$$\alpha|0\rangle + \beta|1\rangle,$$

we wind up with the state

$$\beta|0\rangle + \alpha|1\rangle.$$

Formally, the quantum-NOT gate is a unitary transformation X defined by

$$\begin{aligned} X|0\rangle &= |1\rangle \\ X|1\rangle &= |0\rangle. \end{aligned}$$

The input to X need not be a basis state; it can be any state vector in the two-dimensional space we're working in. But, as we have seen, a linear operator is completely characterized by its action on the basis states. Thus, when the input to X is an arbitrary superposition, we calculate the output by applying the linearity of X:

$$X(\alpha|0\rangle + \beta|1\rangle) = \alpha X|0\rangle + \beta X|1\rangle = \alpha|1\rangle + \beta|0\rangle.$$

The matrix representation of X in the computational basis is

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

Note that this is precisely the Pauli X-matrix. We have, for example,

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle.$$

In the coordinate representation, the action of X on a general input qubit looks like this:

$$X \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}.$$

It's often helpful to draw "quantum circuit diagrams" when discussing quantum gates and circuits. In such a diagram, lines represent qubits, the input qubit is on the left, and the output qubit is on the right. Diagrams of this sort become especially useful when we start talking about multiple-qubit gates, or about circuits (sequences of gates).

The circuit diagram for the NOT-gate is displayed in Figure 1. In the figure, b refers to an arbitrary bit (0 or 1).

$$|b\rangle - \boxed{X} - |1-b\rangle$$

Figure 1: X-gate

2.1.2 Hadamard Gate

The Hadamard Gate H plays a fundamental role in quantum algorithms. Its action on the computational basis is specified by

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ H|1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle), \end{aligned}$$

from which we can read off the matrix representation

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

For example, applying H to the superposition

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

gives us

$$\begin{aligned} H(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) &= \frac{1}{\sqrt{2}}H|0\rangle + \frac{1}{\sqrt{2}}H|1\rangle \\ &= \frac{1}{2}(|0\rangle + |1\rangle) + \frac{1}{2}(|0\rangle - |1\rangle) \\ &= |0\rangle. \end{aligned}$$

Here is the same calculation, using the coordinate representation:

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

The H gate will appear like this in our quantum circuit diagrams:

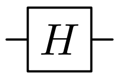

Figure 2: Hadamard gate

As mentioned, the H gate plays a fundamental role in quantum computation. While the basic definitions considered so far don't look very impressive, you'll be more impressed when you see some of the computational tricks that the H gate enables us to perform. We'll consider some of these tricks shortly.

2.2 Two Qubit Gates

So far we have been looking at gates that take a single qubit as input. Now we look at two qubit gates. In the single qubit case, a quantum gate is, when construed formally, a linear transformation on the two-dimensional state space over which the qubit is defined. Analogously, a two qubit gate is a linear transformation over a four-dimensional space: the space defined by taking the tensor product of the two dimensional spaces corresponding to the individual input qubits. More generally, an n-qubit gate is a linear transformation over a 2^n-dimensional tensor product space. But for now we focus on the two qubit case.

Let $|\psi_1\rangle$ and $|\psi_2\rangle$ be the qubits that comprise the input to our two-qubit gate, where

$$\begin{aligned} |\psi_1\rangle &= \alpha_0|0\rangle + \alpha_1|1\rangle \\ |\psi_2\rangle &= \beta_0|0\rangle + \beta_1|1\rangle \end{aligned}$$

We refer to $|\psi_1\rangle$ as the"input" qubit, and $|\psi_2\rangle$ as the"target" qubit. The import of this terminology is that the target qubit is the one that expresses the "output" of whatever procedure is being implemented.

When $|\psi_1\rangle$ and $|\psi_2\rangle$ are the inputs to a two-qubit gate, we characterize the total input by the tensor product,

$$|\psi_1\rangle \otimes |\psi_2\rangle = \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle.$$

The two-qubit gate itself is represented by a $4x4$ matrix that acts on the tensor-product input vector,

$$\begin{pmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{pmatrix}.$$

As our first example of a two-qubit gate, we look at the Controlled-NOT gate, CNOT:

$$CNOT(|x\rangle|y\rangle) = |x\rangle|y \oplus x\rangle.$$

In this equation, '$\oplus$' denotes the classical XOR operation (or, equivalently, addition modulo 2 without the carry). Thus CNOT flips the target bit if the input bit equals 1:

$$\begin{aligned} |0\rangle|0\rangle &\longmapsto |0\rangle|0\rangle \\ |0\rangle|1\rangle &\longmapsto |0\rangle|1\rangle \\ |1\rangle|0\rangle &\longmapsto |1\rangle|1\rangle \\ |1\rangle|1\rangle &\longmapsto |1\rangle|0\rangle \end{aligned}$$

It follows that the matrix representation for CNOT is given by

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

Note that the CNOT gate is a quantum mechanical analogue of the classical XOR gate. To see this, recall the truth table for classical XOR:

α	β	$\alpha \oplus \beta$
1	1	0
1	0	1
0	1	1
0	0	0

(2)

In other words, if $\alpha = 1$, then XOR flips the value of β; if $\alpha = 0$, then XOR leaves β unchanged. And this is exactly what CNOT does.

Here is the circuit diagram for CNOT:

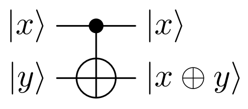

Figure 3: CNot-gate

Of course, what distinguishes CNOT from XOR is that, unlike its classical counterpart, CNOT is defined for nonclassical inputs. In particular, let's see what happens when one or both of the inputs to CNOT is a superposition. In order to handle this case, we just have to keep in mind the following basic features of a 2-qubit gate. First, the input to a 2-qubit gate consists of the tensor product of the two input qubits; and second, the 2-qubit gate represents a linear transformation on this tensor product. For example,

$$\begin{aligned} CNOT(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes |1\rangle) &= \frac{1}{\sqrt{2}} CNOT(|0\rangle|1\rangle - |1\rangle|1\rangle) \\ &= \frac{1}{\sqrt{2}}(CNOT|0\rangle|1\rangle - CNOT|1\rangle|1\rangle) \\ &= \frac{1}{\sqrt{2}}(|0\rangle|1\rangle - |1\rangle|0\rangle). \end{aligned}$$

Note that the output here is precisely the EPR state! Thus CNOT has, in this case, converted a tensor product input state into an entangled output state.

2.3 Quantum Circuits

A *quantum circuit* consists of a sequence of one or more quantum gates, connected by "wires". In general, the wire is not an actual physical connection; it indicates only that information is being transmitted.

For example, suppose we apply an X-gate and then an H-gate to a qubit $|x\rangle$. The circuit for this sequence is displayed in Figure 4. In this graphical representation, any sequence of quantum gates is read from the left (input) to the right (output). Algebraically, this diagram corresponds to

$$|b\rangle = HX|a\rangle.$$

So the "direction" of the algebraic equation is opposite to that of the diagram. This is something you have to get used to.

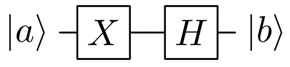

Figure 4: A Quantum Circuit

In terms of matrices, the circuit of Figure 4 represents the equation

$$\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}.$$

Here is a second, more interesting example: the circuit for the "quantum swap" operation, as displayed in Figure 5. This circuit takes two qubits as input; if the input is $|a\rangle \otimes |b\rangle$, then the output is $|b\rangle \otimes |a\rangle$.

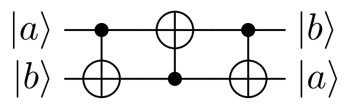

Figure 5: Quantum Swap

Let's think about what the quantum swap circuit is doing. First, note that the middle operation is an upside down $CNOT$. This means that if the *second bit* is 1, we flip the first bit; otherwise, the first bit is unchanged. Therefore, in terms of the coordinate basis, this operation is

$$\begin{array}{rcl}
|0\rangle|0\rangle & \longmapsto & |0\rangle|0\rangle \\
|0\rangle|1\rangle & \longmapsto & |1\rangle|1\rangle \\
|1\rangle|0\rangle & \longmapsto & |1\rangle|0\rangle \\
|1\rangle|1\rangle & \longmapsto & |0\rangle|1\rangle
\end{array}$$

Thus the matrix for the middle operation is

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

Therefore, the entire quantum swap circuit performs the following transformation on an input $|a\rangle \otimes |b\rangle$:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} a_0b_0 \\ a_0b_1 \\ a_1b_0 \\ a_1b_1 \end{pmatrix}.$$

You should verify that the output of this sequence of transformations is indeed

$$|b\rangle \otimes |a\rangle = \begin{pmatrix} b_0a_0 \\ b_0a_1 \\ b_1a_0 \\ b_1a_1 \end{pmatrix}.$$

We always have to perform measurements at the end of a quantum algorithm, in order to obtain any information at all. But sometimes we'll want

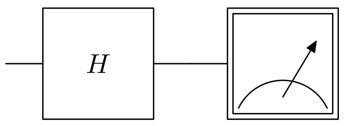

Figure 6: Representing a Measurement

to perform one or more measurements during the operation of an algorithm, and not just at the end. Figure 6 shows how we display a measurement operation; in this example, the measurement follows the application of a Hadamard gate.

2.4 Tensor Product Gates

As mentioned in the previous section, we represent the input to a two-qubit gate as a tensor product vector in a four-dimensional Hilbert space. Physically, however, the inputs are just a pair of qubits; and it is perfectly meaningful to apply single-qubit gates to the two input qubits individually. By considering the tensor product of these two single-qubit gates, we define a new two-qubit gate.

For example, consider the following circuit:

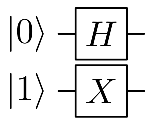

Figure 7: Simple two-qubit circuit

H is applied to the first qubit, in initial state $|0\rangle$, and X is applied to the second qubit, in initial state $|1\rangle$. So the output is

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

Pretty simple!

However, what happens if the input is a superposition? Then things get more complicated. They also get more interesting: this is where quantum computation starts to become *quantum*, non-classical computation. For example, consider the following situation: The input to the circuit is a two-

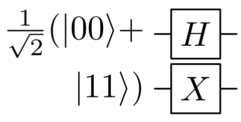

Figure 8: Simple circuit with entangled input

qubit system in the entangled state

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

In order to understand quantum computing, it's crucial to understand how to calculate the circuit output in this kind of situation. There are actually two methods for performing this calculation: I will work through both.

Method 1

1. Calculate the tensor product matrix $(H \otimes X)$

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

2. Multiply this tensor product matrix times the input vector; the result is the output of the circuit. In vector notation,

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

becomes

$$\frac{1}{\sqrt{2}}(1\ 0\ 0\ 1)^T$$

So the calculation is

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

Method 2: Use the linearity of the $(H \otimes X)$ operator. Recall that if A is any linear operator and

$$v = c \cdot (|v_1\rangle + |v_2\rangle + |v_3\rangle + |v_4\rangle)$$

a vector, then

$$\begin{aligned}
Av &= cA(|v_1\rangle + |v_2\rangle + |v_3\rangle + |v_4\rangle) \\
&= c(A|v_1\rangle + A|v_2\rangle + A|v_3\rangle + A|v_4\rangle).
\end{aligned}$$

The same linearity property holds for tensor product operators; the computation looks like this:

$$\begin{aligned}
(H \otimes X)\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) &= \frac{1}{\sqrt{2}}((H \otimes X)|00\rangle + (H \otimes X)|11\rangle) \\
&= \frac{1}{\sqrt{2}}((H|0\rangle \otimes X|0\rangle) + (H|1\rangle \otimes X|1\rangle)) \\
&= \frac{1}{2}(((|0\rangle + |1\rangle) \otimes |1\rangle) + ((|0\rangle - |1\rangle) \otimes |0\rangle)) \\
&= \frac{1}{2}((|0\rangle|1\rangle + |1\rangle|1\rangle) + (|0\rangle|0\rangle - |1\rangle|0\rangle)) \\
&= \frac{1}{2}(|00\rangle + |01\rangle - |10\rangle + |11\rangle),
\end{aligned}$$

which is the same as the method 1 result. We will use whichever method is more convenient for a given calculation–we'll wind up using both.

2.5 A Basic Trick

In this section, we look at the two-qubit gate defined by taking the tensor product of two H gates. In addition to further illustrating the idea of a tensor product gate, this example develops what turns out to be a fundamental technique in quantum computation: initializing the input state as an equally weighted superposition of all basis states.

To see how this works, let's start by looking again at a 2-qubit system. As we've already seen, such a system resides in a four-dimensional space, with computational basis

$$\begin{aligned}
|00\rangle &= |0\rangle \otimes |0\rangle \\
|01\rangle &= |0\rangle \otimes |1\rangle \\
|10\rangle &= |1\rangle \otimes |0\rangle \\
|11\rangle &= |1\rangle \otimes |1\rangle
\end{aligned}$$

From now on, when we encounter a vector of the form $|ij\rangle$, with $i, j \in \{0, 1\}$, we will often label this vector simply by the decimal integer which is represented in binary by ij. In other words, in the four-dimensional case, we'll write

$$\begin{aligned}
|0\rangle &= |00\rangle \\
|1\rangle &= |01\rangle \\
|2\rangle &= |10\rangle \\
|3\rangle &= |11\rangle \qquad (3)
\end{aligned}$$

We'll use a similar convention in higher dimensional spaces. Thus, for example, in H^8 we'll often write

$$|7\rangle$$

instead of

$$|111\rangle.$$

Recall now that the Hadamard transform H looks like this in the computational basis:

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

Now suppose that, in a 2-qubit circuit, an H-gate is applied to both of the input qubits. Formally, this means that we are applying the tensor product $(H \otimes H)$ to the tensor product of the input qubits, where

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

Something very interesting happens when we apply the tensor product operator $H \otimes H$ to input $|0\rangle \otimes |0\rangle$:

$$(H \otimes H)(|00\rangle) = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

To see why this is interesting, note that the output state vector is an equally weighted superposition of the vectors that comprise the computational basis. Using the notation (3), which associates an integer with each basis state, this output state can be written,

$$\frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle).$$

We can apply this idea to any number of input qubits. In the case of three qubits, we apply $(H \otimes H \otimes H)$ to $|000\rangle$ to obtain

$$\frac{1}{\sqrt{8}}(|0\rangle + |1\rangle + |2\rangle + |3\rangle + |4\rangle + |5\rangle + |6\rangle + |7\rangle).$$

And in general, the result of applying the N-fold tensor product operator $(H \otimes H \otimes \cdots \otimes H)$ to the N-fold tensor product $|00\cdots 0\rangle$ is the state

$$(H \otimes \cdots \otimes H)|00\cdots 0\rangle = \frac{1}{\sqrt{2^N}} \sum_{i=0}^{2^N-1} |i\rangle. \tag{4}$$

This is one of the fundamental tricks in quantum computing: create an equally weighted superposition of each of the integers 0 through $2^N - 1$. We will see our first application of this trick shortly, and will use it repeatedly in our discussion of quantum algorithms.

2.6 Reversible Functions and the U_f Operator

Consider a classical function of the form

$$f : \{0,1\}^n \to \{0,1\}^m. \tag{5}$$

Quite often, one has to build such a function into a quantum circuit. But there is a problem in doing so: quantum circuits have to be reversible, but classical functions typically are *not* reversible. For example, classical logical AND is not reversible:

α	β	$\alpha \wedge \beta$
1	1	1
1	0	0
0	1	0
0	0	0

If the output is 0, we cannot uniquely infer the input.

There is a standard technique for constructing a reversible version of a classical function. If the classical f takes an n-bit string as input and produces an m-bit string as output, then the reversible version of f will define the following mapping:

$$(x, y) \mapsto (x, y \oplus f(x)),$$

where x is an n-bit string and y an m-bit string. By calling the revised function with y the 0-string, we reproduce the original f. This is what the reversible-AND looks like:

x_1	x_2	y	x_1	x_2	$y \oplus (x_1 \wedge x_2)$
1	1	1	1	1	0
1	1	0	1	1	1
1	0	1	1	0	1
1	0	0	1	0	0
0	1	1	0	1	1
0	1	0	0	1	0
0	0	1	0	0	1
0	0	0	0	0	0

The output of this function is the triple $(x_1, x_2, y \oplus (x_1 \wedge x_2))$; you can verify that the function is indeed reversible. The rows with $y = 0$ reproduce the classical truth table.

Since we have constructed a reversible version of f, we can define a corresponding unitary operator U_f:

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

Moreover, it's a theorem that if the classical f can be computed in polynomial time, then there is a polynomial-sized circuit for U_f: the numbers of gates and qubits are polynomial functions of the input size (see section 6.2 of Rieffel and Polak's text for a detailed discussion).

Sometimes we'll want to keep track explicitly of the number of qubits we're working with. In order to do so, we use subscripts on brackets to denote dimensionality. Thus, for example, $|0\rangle_n$ denotes the n-fold tensor product of the basis state $|0\rangle$; in other words, $|0\rangle_n$ denotes a basis vector in a 2^n-dimensional complex vector space. In general, $|x\rangle_n$ denotes some vector x in such a space. Using this notation, we can characterize the $(n+m)$-qubit gate U_f that corresponds to a function of the form (5):

$$U_f|x\rangle_n|y\rangle_m = |x\rangle_n|y \oplus f(x)\rangle_m.$$

2.7 Quantum Parallelism

Let's look at what happens when we apply U_f to the input $|x\rangle|0\rangle$. Since $0 \oplus f(x) = f(x)$, for any value of $f(x)$, it follows that

$$U_f|x\rangle|0\rangle = |x\rangle|f(x)\rangle. \tag{6}$$

This may not look too impressive by itself, but when we combine (4) and (6), we get something really neat. For example, consider what happens when we apply U_f to

$$((H \otimes H) \otimes I)(|00\rangle \otimes |0\rangle). \tag{7}$$

Applying linearity together with (4) and (6), we obtain

$$\begin{aligned}
&U_f(((H \otimes H) \otimes I)(|00\rangle \otimes |0\rangle)) \\
&\qquad = U_f(\frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle) \otimes |0\rangle) \\
&\qquad = \frac{1}{2}(U_f|0\rangle|0\rangle + U_f|1\rangle|0\rangle + U_f|2\rangle|0\rangle + U_f|3\rangle|0\rangle) \\
&\qquad = \frac{1}{2}(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle + |2\rangle|f(2)\rangle + |3\rangle|f(3)\rangle).
\end{aligned}$$

Using a single quantum circuit with two H gates, we have simultaneously calculated $f(x)$ for all four values of x. In general,

$$\begin{aligned}
U_f(H^{\otimes n} \otimes I_m)(|0\rangle_n|0\rangle_m) &= \frac{1}{2^{n/2}} \sum_{0 \le x < 2^n} U_f|x\rangle_n|0\rangle_m \\
&= \frac{1}{2^{n/2}} \sum_{0 \le x < 2^n} |x\rangle_n|f(x)\rangle_m \qquad (8)
\end{aligned}$$

Using a single quantum circuit with n gates, we can simultaneously calculate $f(x)$ for all 2^n values of x. We can, in other words, *carry out an exponential number of operations in polynomial time, using a polynomial number of processors.* This is the essence of *quantum parallelism*, expressed in compact form by equation (8). Think of how different this is from the classical case where, if you have to carry out an exponential number of operations, you need either exponential time or an exponential number of processors.

However, quantum computation isn't magic. The peculiarities of the quantum world that allow us to carry out the operation expressed by (8), make it a very tricky business actually to *use* (8) in a meaningful way. The complication is the standard one that we have already discussed at length. We cannot know the state of a quantum system unless we perform a measurement on that system. But when we perform a measurement on a system in a state such as (8), the system is projected onto a normalized subset of the states that comprise the superposition. So what we "observe", ultimately, is just one of the values, $f(0), f(1), \ldots$. In other words, while all 2^n values of $f(x)$ are "encoded" in the quantum mechanical state (8), we can't get at this information without performing a measurement. And to perform a measurement is to lose most of the information.

The upshot of this discussion is not that it's impossible to apply equation (8) in a useful way, but just that it is far from straightforward to do so. David Deutsch was the first to figure out how to extract some useful information from a state such as (8) (he is also the person who dreamed up such states in the first place). Deutsch's insights allowed him to construct the first quantum algorithm, which we turn to next.

3 Quantum Algorithms: Deutsch's Problem

3.1 Deutsch's Problem

David Deutsch introduced the following problem to illustrate his conjecture that a quantum computer is inherently more powerful than a classical computer. When I say "inherently more powerful", I do *not* mean to imply that there are problems that a quantum computer can solve but a classical computer cannot. That there are no such problems follows from the fact that any quantum computation can be simulated by a classical computation; it's just that the classical computation takes longer. The reason that quantum computation is exciting is that it is believed that the class BQP of problems that can be solved in polynomial time by a quantum Turing machine is strictly larger that the class BPP of problems that can be solved in polynomial time by a classical probabilistic Turing machine. In other words, it is believed that quantum computers are more powerful than classical computers. However, this hasn't been proven.[1] Instead, what we have are polynomial time quantum algorithms for problems such as factoring, for which no polynomial time classical algorithms are known, and for which most researchers *believe* no polynomial time classical algorithm is possible.

Deutsch's problem itself is contrived. But it is conceptually interesting, because it is a problem that can be solved faster by a quantum computer than by any classical computer. In this case, however, the speedup is not exponential. Once Deutsch had presented his problem, the question arose of whether there are problems that

i. Can be solved in polynomial time by a quantum computer;

ii. Cannot be solved in polynomial time by any known classical algorithm;

iii. Are intrinsically interesting and important.

The first such "killer app" was Shor's quantum factoring algorithm, which we'll discuss in detail later. But for now we focus on Deutsch's problem, as a way of developing the idea of a quantum algorithm.

Consider a function f that takes 0 or 1 as input, and produces 0 or 1 as output. There are exactly four such functions:

	$x = 0$	$x = 1$
f_0	0	0
f_1	0	1
f_2	1	0
f_3	1	1

f_0 and f_3 are *constant* functions, while f_1 and f_2 are *balanced.* Suppose now that we're given a black box that executes one of these functions, but we don't know which. How much work (as measured by number of function calls) is required to determine whether the unknown f is constant or balanced?

[1]Note for complexity theorists: A proof that BPP is a proper subset of BQP would have major implications for classical complexity theory, because it would imply that BPP (and hence P) is a proper subset of $PSPACE$. This fact suggests that proving $BPP \subset BQP$ is likely to be very difficult.

Classically, we have to call the function twice: calculate $f(0)$ and $f(1)$, and then compare the results.

Remarkably, however, we can determine whether f is constant or balanced in a single run of a quantum computer. According to a quantum computational view of Deutsch's problem, we're given a black box that executes the unknown function f by applying the unitary transformation,

$$U_f(|x\rangle|y\rangle) = |x\rangle|y \oplus f(x)\rangle. \tag{9}$$

To compute $f(0)$ we set both input qubits to 0, while to compute $f(1)$ we set the first input qubit to 1 and the second to 0:

$$\begin{aligned} U_f(|0\rangle|0\rangle) &= |0\rangle|0 \oplus f(0)\rangle = |0\rangle|f(0)\rangle \\ U_f(|1\rangle|0\rangle) &= |1\rangle|0 \oplus f(1)\rangle = |1\rangle|f(1)\rangle. \end{aligned}$$

In both cases, we measure the second output qubit to obtain the value of f.

The interesting case occurs, as it usually does in the quantum domain, when one of the input qubits is a superposition of computational basis vectors. Specifically, suppose we apply U_f to the input

$$\frac{(|0\rangle + |1\rangle)}{\sqrt{2}} \otimes |0\rangle = \frac{(|00\rangle + |10\rangle)}{\sqrt{2}}.$$

By linearity,

$$\begin{aligned} U_f(\frac{(|00\rangle + |10\rangle)}{\sqrt{2}}) &= \frac{1}{\sqrt{2}}(U_f|00\rangle + U_f|10\rangle) \\ &= \frac{1}{\sqrt{2}}(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle). \end{aligned} \tag{10}$$

This is actually an instance of the trick (8), described above: we have calculated the values of $f(0)$ and $f(1)$ simultaneously. So what we're looking at here is our first example of quantum parallelism. But now we have to come to grips with the fundamental limitation of quantum parallelism, which was also discussed earlier. In order to learn anything about the state vector (10), we have to perform a measurement. And when we do perform a measurement, the state vector (10) collapses to either $|0\rangle|f(0)\rangle$ *or* $|1\rangle|f(1)\rangle$ (with equal probability). So we can only learn one of the values $f(0)$ or $f(1)$ directly from (10). We have to be more clever in order to learn anything useful.

To solve the function problem, Deutsch contrived the circuit displayed in Figure 7. We apply this circuit with input $|0\rangle \otimes |1\rangle$. Recall that

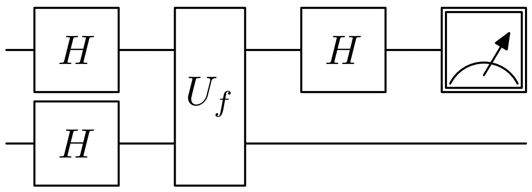

Figure 9: Deutsch's Circuit

$$|01\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

The circuit diagram tells us that $H \otimes H$ is applied directly to the input. As we have already seen,

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

Therefore, when the circuit is applied to the input $|01\rangle$, we obtain

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

Equation (11) is the state of the two-qubit system after it has been processed by the two H gates. Figure 7 tells us that the output of the two H gates is the input to U_f. So the next step is to calculate the effect of U_f on the vector (11).

In order to calculate the effect of U_f on (11), we apply linearity to obtain

$$\frac{1}{2}(U_f|0\rangle|0\rangle - U_f|0\rangle|1\rangle + U_f|1\rangle|0\rangle - U_f|1\rangle|1\rangle). \tag{12}$$

It's helpful at this point to introduce some additional notation. Let $b \in \{0,1\}$. Then $\overline{b} = (1-b)$ is the complement of b; i.e., $\overline{b} = 1$ if $b = 0$, and $\overline{b} = 0$ if $b = 1$. Using this notation, we can describe the behavior of U_f like this:

$$\begin{aligned} U_f|x\rangle|0\rangle &= |x\rangle|0 \oplus f(x)\rangle \\ &= |x\rangle|f(x)\rangle \\ \\ U_f|x\rangle|1\rangle &= |x\rangle|1 \oplus f(x)\rangle \\ &= |x\rangle|\overline{f(x)}\rangle. \end{aligned}$$

So we can now describe the output (12) of U_f, given (11) as input, in the following concise form:

$$\frac{1}{2}(|0\rangle|f(0)\rangle - |0\rangle|\overline{f(0)}\rangle + |1\rangle|f(1)\rangle - |1\rangle|\overline{f(1)}\rangle). \tag{13}$$

Now let's look at the two cases.

i. f **is constant** (i.e., $f(0) = f(1)$). Then (13) becomes

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

or equivalently

$$\frac{1}{2}(|0\rangle + |1\rangle)(|f(0)\rangle - |\overline{f(0)}\rangle). \tag{14}$$

ii. f **is balanced** (i.e., $f(0) \neq f(1)$). Then (13) becomes

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

or equivalently

$$\frac{1}{2}(|0\rangle - |1\rangle)(|f(0)\rangle - |\overline{f(0)}\rangle). \tag{15}$$

In both cases, the output (second) qubit is $(|f(0)\rangle - |\overline{f(0)}\rangle)$. So the two cases are distinguished by their different input qubits (qubits corresponding to the "top" wire in Figure 7). This is why the circuit has been designed to apply another Hadamard gate to the input qubit, after the application of U_f. In matrix notation, the result of applying the final Hadamard gate looks like this in the two cases:

Constant :

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

Balanced :

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

Putting the whole thing together, we see that Deutsch's circuit performs the following calculation on the input $|0\rangle|1\rangle$:

$$\begin{aligned}&(H \otimes 1)U_f(H \otimes H)(|0\rangle|1\rangle) \\ &\qquad = \begin{cases} \frac{1}{\sqrt{2}}|0\rangle(|f(0)\rangle - |\overline{f(0)}\rangle) & \text{if } f(0) = f(1) \\ \frac{1}{\sqrt{2}}|1\rangle(|f(0)\rangle - |\overline{f(0)}\rangle) & \text{if } f(0) \neq f(1) \end{cases}\end{aligned} \qquad (16)$$

So by measuring the input (first) qubit, we can answer the question of whether f is constant or balanced: if the measurement returns 0, then f is constant; if it returns 1, then f is balanced. Thus, with a single measurement, we answer a question that requires two measurements in the classical case.

It is interesting to observe that this measurement tells us nothing at all about the actual value of $f(0)$. The reason is that, in both cases, the two terms in the output superposition have the same magnitude. Thus if f is constant, the measurement projects the output superposition into one of the two states, $|0\rangle|f(0)\rangle$ or $|0\rangle|\overline{f(0)}\rangle$, with equal probability. The value of the output (second) qubit will be found to be 0 or 1; but how do we know whether this value represents $f(0)$ or $\overline{f(0)}$? How *could* we know this? Well, suppose our output state looked like this:

$$|0\rangle(\sqrt{2/3}|f(0)\rangle - \sqrt{1/3}|\overline{f(0)}\rangle).$$

Were we to repeat the experiment in this case, we would expect to obtain the output qubit value corresponding to $f(0)$ twice as often as the output register value corresponding to $\overline{f(0)}$. In the present case, however, the two values occur with equal probability.

Deutsch's algorithm nicely illustrates both the power and the subtlety of quantum computation.

3.2 Deutsch-Jozsa

The Deutsch-Jozsa algorithm solves a problem that is a straightforward generalization of the problem solved by Deutsch's algorithm. As before, we are given a reversible circuit that implements an unknown function f. But now f is a function from n-bit strings to a single bit:

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

We are also given the *promise* (this term has a specific meaning for complexity theorists) that f is either *constant* ($f(x)$ has the same value for all x) or *balanced* ($f(x) = 0$ for exactly half of the input strings x, and $f(x) = 1$ for the other half). The problem, once again, is to determine whether f is constant or balanced, by making queries to the oracle (black-box circuit).

Let's think about trying to solve this problem with a classical deterministic algorithm. Suppose we have used the oracle to determine $f(x)$ for exactly half of the the possible inputs x; that is, we have made 2^{n-1} queries to f. Suppose further that all queries have returned $f(x) = 0$. At this point we would strongly suspect that f is constant, but we couldn't be certain; it might happen that $f(x) = 1$ for each of the remaining 2^{n-1} inputs x. So in the worst case, classically, we need $2^{n-1}+1$ oracle calls to determine whether f is constant or balanced. (In the best case we'll only need two oracle calls; but it's the worst-case performance that determines the complexity of an algorithm.)

Once again, however, we can use a quantum algorithm to solve the problem by making just a single oracle query. The circuit is almost the same as it was for Deutsch's original version of the problem; the only difference is that the input register now consists of n qubits, each of which is passed through a Hadamard gate:

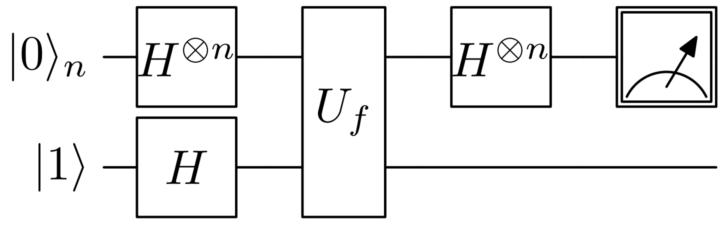

Figure 10: Deutsch-Jozsa Circuit

I'll leave the analysis of the Deutsch-Jozsa algorithm for you to think about; it's very similar to the analysis of Deutsch's algorithm.

So the Deutsch-Josza algorithm gives us an *exponential* speedup over the best possible classical deterministic algorithm! Does this prove that quantum computers are more powerful than classical computers? In complexity-theoretic terms, does it establish that $\mathbf{P} \subset \mathbf{BQP}$, where **BQP** is the set of problems that can be solved in polynomial time by a quantum computer? Actually, no. We're dealing here with what complexity theorists refer to as an "oracle separation" between **BQP** and **P**. In complexity theory, one often makes a fixed oracle available to a Turing machine with certain limited resources, and then studies how the oracle increases the computational power of the machine. In this context, a call to the oracle counts as a single step, even if the computation carried out by the oracle is very complex (or of unknown complexity). The Deutsch-Josza result shows that if we consider the call to the circuit for f as a single step, then there is something that can be done in polynomial time with a quantum computer, but cannot be done in polynomial time by a (deterministic) classical computer. This is extremely interesting, and can be viewed as evidence that quantum computers are more powerful than classical computers. But it doesn't prove that quantum computers are more powerful, because we're dealing with imaginary oracles and not real world computation. In general, an oracle separation result does not imply that the separation continues to hold when we remove the oracle. For example, there are oracles relative to which $\mathbf{P} \neq \mathbf{NP}$; but nobody views the **P** vs. **NP** problem as solved! (To make things interesting, there are also oracles relative to which $\mathbf{P} = \mathbf{NP}$.)

Continuing this line of thought, suppose that someone did prove that **P** is a proper subset of **BQP**. This would be a very big deal for complexity theory (for one thing, it would imply that **P** is a proper subset of **PSPACE**), but it wouldn't establish that quantum computers are more powerful than classical computers. The reason is that the largest class of problems that can be solved in polynomial time by a classical computer is **BPP**: the class of problems that can be solved in *randomized* polynomial time. Deutsch-Jozsa only gives an oracle separation between **BQP** and **P**, not between **BQP** and **BPP**. There is another result–Simon's algorithm–that does give an oracle separation between **BQP** and **BPP**. This result can be viewed as providing even stronger evidence that quantum computers are more powerful than classical computers. But again, it is only an oracle and not an absolute result.

4 Some Real World Complications

4.1 How additional qubits can mess things up

The real-life computational process will actually require the use of many qubits, in addition to the input and output registers that are usually displayed. In particular, the circuit used to compute f will use lots of extra qubits (beyond the input and output qubits). In general, the input and output qubits will become entangled with these additional qubits during the computational process, and as a result the input and output registers won't even have well-defined states. The result is that the given quantum algorithm won't actually work as advertised.

In the case of Deutsch's algorithm, we've been assuming a nice clean unitary transformation for f:

$$U_f : \big|x, y\big\rangle \to \big|x, y \oplus f(x)\big\rangle$$

To illustrate what can happen when input and/or output qubits become entangled with extra workspace qubits, let's replace U_f with a circuit V_f, which works like U_f but also entangles the input register with a temporary bit[2] (here the input and output registers are represented by x and y respectively, and w denotes the temporary bit):

$$V_f : \big|x, w, y\big\rangle \to \big|x, w \oplus x, y \oplus f(x)\big\rangle$$

So Deutsch's circuit now looks like the one in Figure 9.

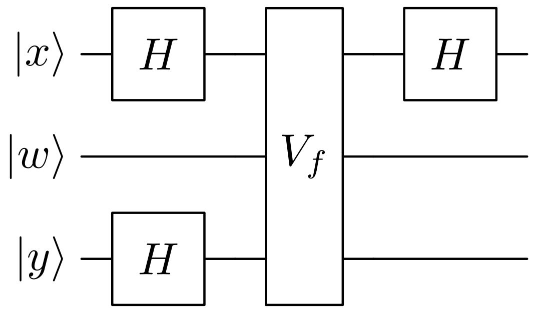

Figure 11: Deutsch's Circuit with an extra qubit

[2]See section 7.4.1 of Rieffel and Polak's text. In this section I'm working through the details of an example they present.

When this circuit is applied to the input $|xwy\rangle = |001\rangle$, we get

$$\frac{1}{2}(|000\rangle - |001\rangle + |100\rangle - |101\rangle),$$

after the application of the first two Hadamards, a state in which the three qubits are entangled.

Applying V_f to the previous state gives us

$$\begin{aligned} V_f(\frac{1}{2}(|000\rangle - |001\rangle + |100\rangle - |101\rangle)) \\ &= \frac{1}{2}(V_f|000\rangle - V_f|001\rangle \\ &\quad + V_f|100\rangle - V_f|101\rangle) \\ &= \frac{1}{2}(|0,0,f(0)\rangle - |0,0,\overline{f(0)}\rangle \\ &\quad + |1,1,f(1)\rangle - |1,1,\overline{f(1)}\rangle) \qquad (17) \end{aligned}$$

When f is constant, (17) becomes

$$\frac{1}{2}(|0,0,f(0)\rangle - |0,0,\overline{f(0)}\rangle + |1,1,f(0)\rangle - |1,1,\overline{f(0)}\rangle, \qquad (18)$$

and when f is balanced, (17) becomes

$$\frac{1}{2}(|0,0,f(0)\rangle - |0,0,\overline{f(0)}\rangle + |1,1,\overline{f(0)}\rangle - |1,1,f(0)\rangle. \qquad (19)$$

When we then try to follow Deutsch's logic by applying $H \otimes I \otimes I$ to (18) and (19) respectively, we get

$$\begin{aligned} &\frac{1}{\sqrt{8}}(|0,0,f(0)\rangle + |1,0,f(0)\rangle - |0,0,\overline{f(0)}\rangle - |1,0,\overline{f(0)}\rangle \\ &\quad + |0,1,f(0)\rangle - |1,1,f(0)\rangle - |0,1,\overline{f(0)}\rangle + |1,1,\overline{f(0)}\rangle) \qquad (20) \end{aligned}$$

and

$$\begin{aligned} &\frac{1}{\sqrt{8}}(|0,0,f(0)\rangle + |1,0,f(0)\rangle - |0,0,\overline{f(0)}\rangle - |1,0,\overline{f(0)}\rangle \\ &\quad + |0,1,\overline{f(0)}\rangle - |1,1,\overline{f(0)}\rangle - |0,1,f(0)\rangle + |1,1,f(0)\rangle). \qquad (21) \end{aligned}$$

We can no longer apply the technique of Section 1 (equations (14 - 16)), because that technique requires that the input and output registers be separable.

4.2 Solving the problem by uncomputing

We deal with this problem by using $CNOT$ gates to copy the input and output registers to some new registers, and then reversing the computational process that has been applied thus far:

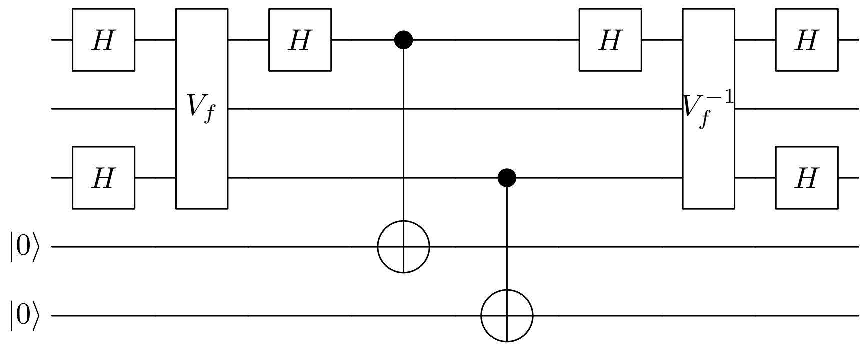

Figure 12: Unentangling the extra bits

For example, here is the state of the system for the constant function case, on input $|00100\rangle$, after the application of the two $CNOT$ gates (start with equation (20), tack on the final $|0\rangle$ bits, then apply the $CNOT$s):

$$\begin{aligned}
&\frac{1}{\sqrt{8}}\big(|0,0,f(0),0,f(0)\rangle + |1,0,f(0),1,f(0)\rangle - |0,0,\overline{f(0)},0,\overline{f(0)}\rangle \\
&\quad - |1,0,\overline{f(0)},1,\overline{f(0)}\rangle + |0,1,f(0),0,f(0)\rangle - |1,1,f(0),1,f(0)\rangle \\
&\quad - |0,1,\overline{f(0)},0,\overline{f(0)}\rangle + |1,1,\overline{f(0)},1,\overline{f(0)}\rangle\big) \qquad (22)
\end{aligned}$$

After we uncompute the first three qubits, as in the Figure 10 circuit, (22) becomes

$$\begin{aligned}
|0,0,1\rangle \otimes \frac{1}{\sqrt{8}}\big(&|0,f(0)\rangle + |1,f(0)\rangle - |0,\overline{f(0)}\rangle - |1,\overline{f(0)}\rangle \\
&+ |0,f(0)\rangle - |1,f(0)\rangle - |0,\overline{f(0)}\rangle + |1,\overline{f(0)}\rangle\big), \qquad (23)
\end{aligned}$$

which reduces to

$$\begin{aligned}
&|0,0,1\rangle \otimes \frac{1}{\sqrt{2}}\big(|0,f(0)\rangle - |0,\overline{f(0)}\rangle\big) \\
&\quad = |0,0,1\rangle \otimes |0\rangle \otimes \frac{1}{\sqrt{2}}\big(|f(0)\rangle - |\overline{f(0)}\rangle\big). \qquad (24)
\end{aligned}$$

Similarly, we get

$$\big|0,0,1\big\rangle \otimes \big|1\big\rangle \otimes \frac{1}{\sqrt{2}}\big(\big|f(0)\big\rangle - \big|\overline{f(0)}\big\rangle\big) \qquad (25)$$

for the case of a balanced function.

So we have now obtained a result analogous to the original equation (16): we can distinguish between the constant and balanced cases by performing a single measurement, on the copy of the original input qubit.

At this point the following argument may have occurred to you: "The circuit of Figure 10 makes two calls to the oracle for f. So while this circuit may deal with the problem of work qubits getting entangled with the input and output registers, it also negates the supposed advantage of quantum computing over classical computing for Deutsch's problem. Thus, the claim was that with a quantum computer we can solve this problem with a single call to the oracle; but now we're making two calls, the same number that are required in the classical case."

This is a good point. Since the uncomputation doubles the processing time, what have we gained? The reply is that what we are really interested in are quantum algorithms for which the speedup over the classical case is considerably larger than two. In particular, when we're dealing with an exponential speedup, as in the Deutsch-Jozsa algorithm (or more interesting algorithms that we will see later), doubling the processing time of the quantum algorithm in not significant. But yes, taking uncomputation into account does take some of the luster off of the original Deutsch algorithm.

An ongoing research program in quantum computing is to develop models that will enable us to provide an accurate estimate of the resources that would be needed to implement a given quantum algorithm on a real, physical quantum computer. Such estimates would explicitly take into account the number of "scratch" bits really needed to implement various circuits, the cost of uncomputation, the cost of quantum error correction, and other factors.