Quantum Computing Notes

Prof. Hamzeh Roumani
EECS, York University, Toronto, Ont., Canada

MULTI QUBIT SYSTEMS

1. Preamble

We learned that a single qubit is represented as a vector in a two-dimensional, complex Hilbert space, and that we can refer to it in the (0,1) standard basis using the ket:

$|s\rangle = \alpha|0\rangle + \beta|1\rangle$

or the column vector:

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

which, in the programming notation, is written as: [[α, β]]. How do we represent a composite system made up of more than one qubit? Is it also a vector, and if so, in what space—how many dimensions and what is its basis?

There are two common ways to combine vector spaces in linear algebra, one is the *direct sum* (denoted by ⊕) in which the dimensions add, and the other is the *tensor product* (denoted by ⊗) in which they multiply. For an n-qubit system, the former leads to a **2n-dimensional** space while the latter yields a **2^n dimensions**. Nature chose the latter: the vector space of the multi-qubit system is the **tensor product** of the spaces of the constituent qubits. This fact has a pivotal implication for quantum computing because it means the computing power and the information content *grows exponentially* rather than linearly with the number of qubits.

2. The Space

Given two complex vector spaces S1 and S2, with basis sets B1 and B2, their tensor product S is a complex vector space (with vector addition plus scaling by a complex number) of dimension equal to the product of the dimensions of S1 and S2. Its basis B is derived from B1 and B2 via a function *f* that maps each ordered pair in B1xB2 to a basis vector in B. The function is not linear, it does not map sums to sums or scaled arguments to scaled images in both arguments, but it is linear in each individual argument (aka *bilinear*): If a ∈B1 and b ∈B1 and n is a complex number, then:

f(a, b) = f(b,a)

f(a1 + a2, b) = f(a1, b) + f(a2, b)
f(a, b1 + b2) = f(a, b1) + f(a, b2)

f(na, b) = f(a, nb) = nf(a, b)

The above properties have the same algebraic structure as multiplication, viz. distributive over addition; scaling a product ais the same as scaling one of its factors; and commutative. This is why it is more convenient to switch from functional to operator notation:

$f(a, b) \rightarrow a \otimes b$

We use the same symbol to denote the tensor product for vectors, matrices, and spaces, e.g. we can write: S = S1 ⊗ S2. And B = B1 ⊗ B2.

Let us rewrite the properties of our bilinear map using the operator notation for the tensor product:

- Commutative:
  $|a\rangle \otimes |b\rangle = |b\rangle \otimes |a\rangle$
- Bilinear with respect to multiplication by a complex number n:
  $n \times (|a\rangle \otimes |b\rangle) = (n \times |a\rangle) \otimes |b\rangle) = |a\rangle \otimes (n \times |b\rangle)$
- Distributive over vector addition:
  $|a\rangle \otimes (|b_1\rangle + |b_2\rangle) = |a\rangle \otimes |b_1\rangle + |a\rangle \otimes |b_2\rangle$
  $(|a_1\rangle + |a_2\rangle) \otimes |b\rangle = |a_1\rangle \otimes |b\rangle + |a_2\rangle \otimes |b\rangle$

For 2 qubits, the four (standard) basis vectors of the tensor product are:

$|0\rangle_1 \otimes |0\rangle_2$, $|0\rangle_1 \otimes |1\rangle_2$, $|1\rangle_1 \otimes |0\rangle_2$, $|1\rangle_1 \otimes |1\rangle_2$

This (inconvenient) notation can be simplified through a number of successive steps:

- ◊ Drop the tensor operator symbol, e.g. $|0\rangle_1 |1\rangle_2$ for $|0\rangle_1 \otimes |1\rangle_2$.
- ◊ Drop the subscript, e.g. $|0\rangle |1\rangle$ for $|0\rangle_1 |1\rangle_2$.
- ◊ Put both vectors in one ket, e.g. $|01\rangle$ for $|0\rangle |1\rangle$.
- ◊ Put the decimal values of the bit pattern, e.g. $|1\rangle$ for $|01\rangle$.

We will adopt the very last simplification and revert to the explicit notation when needed to avoid ambiguity. As such, the four basis vectors for the tensor product of 2 qubits are:

$|0\rangle, |1\rangle, |2\rangle, |3\rangle$

And similarly, the eight basis vectors for the tensor product of 3 qubits are:

$|0\rangle, |1\rangle, |2\rangle, |3\rangle, |4\rangle, |5\rangle, |6\rangle, |7\rangle$

And for n qubits, the 2^n basis vectors are:

$|k\rangle, k = 1 \ldots 2^n$

Finally, we need to define an inner product on the composite space so we can turn it into a Hilbert space. For the basis vectors, we define it as the product of the two inner products:

$(\langle a| \otimes \langle b|) \cdot (\langle c| \otimes \langle d|) \equiv \langle a|c\rangle \langle b|d\rangle$

This implies that the dot product of two unit-vectors in S1 and S2 is a unit vector in S. It also implies that if B1 and B2 are orthonormal basis, then B = B1 ⊗ B2 is also orthonormal. And since the dot product is distributive over addition, the above allows us to compute the dot products of any two vectors in the space.

As indicated in the single qubit case, there is redundancy in the underlying vector space (vis-à-vis states) because vectors that differ by a scale represent the same state. For multiple qubits, the redundancy is even larger because the tensor product allows us to apply a phase to either factor and still end up with the same vector. This observation does not affect our calculations, because we always work in the full (redundant) vector space, but it should be kept in mind in order to correctly interpret vectors as states.

3. The State

Now that we understand the space in which a multi-qubit system resides, let us look at state vectors in such a system. For a two-qubit system, this is a 4d vector and we like to see how it relate to the two state vectors of the individual qubits. We'll start with an example: a vector in the 2-qubit space is expressed in the standard basis as something like this:

$|\psi\rangle = (0.6|00\rangle + 0.8|01\rangle + 0.6|10\rangle + 0.8|11\rangle) / \sqrt{2}$

Note that this state is normalized:

$(0.6^2 + 0.8^2 + 0.6^2 + 0.8^2)/2 = 1$

Note also that the state expression can be simplified using the properties of the tensor product:

$|\psi\rangle = [0.6 (|0\rangle + |1\rangle) \otimes |0\rangle + 0.8 (|0\rangle + |1\rangle) \otimes |1\rangle] / \sqrt{2}$

$|\psi\rangle = (0.6 |0\rangle + 0.8 |1\rangle) \otimes (|0\rangle + |1\rangle) / \sqrt{2}$

The composite state in this case is thus **separable** (aka a **product state**) and is easy to interpret: one qubit in state 0.6 |0> + 0.8 |1> and the other in |+>.

However, not all states in the composite space are separable. Take for example any of the following four states that are known as the **Bell states**:

$|\Phi+\rangle = (|00\rangle + |11\rangle) / \sqrt{2}$
$|\Phi-\rangle = (|00\rangle - |11\rangle) / \sqrt{2}$
$|\psi+\rangle = (|01\rangle + |10\rangle) / \sqrt{2}$
$|\psi-\rangle = (|01\rangle - |10\rangle) / \sqrt{2}$

Each is a normalized state, but no simplification can reduce it to a product state. Such states are known as **entangled**. It is difficult to interpret these states in terms of the states of their constituent qubits. For example, you may be tempted to interpret the first Bell state above as an equal superposition of |0> and |1> for each of the two constituent qubits, but this is wrong because such an interpretation describes the following different state:

$|\psi\rangle = [(|0\rangle + |1\rangle) / \sqrt{2}] \otimes [(|00\rangle + |11\rangle) / \sqrt{2}]$

We conclude from the above that an entangled state cannot be understood in terms of what its individual qubits are doing. In such a state, the system is somehow "bigger" than the sum of its parts and must therefore be viewed as one entity, regardless of the distances that separate its qubits. This is in direct contradiction with locality.

Most of the states in the tensor product space are in fact entangled and cannot be written as a product (although you can always write them as a linear combination of products), and few are separable. Here is why: for the most general composite state:

$|\varphi\rangle = a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$

to be separable, we should be able to write it as a product:

$|\varphi\rangle = (x|0\rangle + y|1\rangle) \otimes (z|0\rangle + t|1\rangle)$

Equating the above expressions and simplifying yields the following 4 equations:

$a = xz,\ b = xt,\ c = yz,\ d = yt$

The equations have no solution except when x=0, or y=0, or z=0, or t=0, or a/b = c/d. Only then would the state be separable. In all other cases, it would be entangled.

It is interesting to see how the components of a product state derives from the components of the states of its constituent qubits. Given the most general product state:

$|s\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes (\gamma|0\rangle + \boldsymbol{\delta}|1\rangle)$

use the properties of the tensor product (just think of it as multiplication):

$|s\rangle = \alpha\gamma|00\rangle + \alpha\boldsymbol{\delta}|01\rangle + \beta\gamma|10\rangle + \beta\boldsymbol{\delta}|11\rangle$

Or in column matrix form:

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

(Note that this is similar to an outer product but the 2d matrix is reshaped as a 4d vector.) This result allows us to easily switch from the bra-ket to the matrix representation. It works for any state, not just a separable one, because all states can be written as linear combinations of the basis states, which are separable. As an example, let us apply it to the first Bell state:

$|\psi\rangle = (\ |00\rangle + |11\rangle\ ) / \sqrt{2}$

$$|\psi\rangle = \begin{pmatrix}1\\ 0\end{pmatrix} \times \begin{pmatrix}1\\ 0\end{pmatrix} / \sqrt{2} + \begin{pmatrix}0\\ 1\end{pmatrix} \times \begin{pmatrix}0\\ 1\end{pmatrix} / \sqrt{2} = \begin{pmatrix}1\\ 0\\ 0\\ 0\end{pmatrix} / \sqrt{2} + \begin{pmatrix}0\\ 0\\ 0\\ 1\end{pmatrix} / \sqrt{2}$$

$$|\psi\rangle = \begin{pmatrix}1\\ 0\\ 0\\ 1\end{pmatrix} / \sqrt{2}$$

4. Transformations

As in the single-qubit case, a composite system can undergo a transformation that changes its state without leaking information, and these transformations are linear, unitary, and reversible. As such, these transformations are represented as operators or matrices. The matrix would be 4x4 in the 2-qubit case.

A key difference in the multi-qubit case is that the transforming device can operate on one of the qubits without affecting the other. We represent such a case mathematically by applying two operators (or two matrices) on the composite state: one performing the desired transformation on the affected qubit, and one doing nothing on the other (an identity operator). In the matrix representation, this amounts to computing the outer product of the transforming matrix with the unit matrix. Here is an example: a device that flips the state of the second qubit only. The corresponding operator would be:

Flip2 = $\mathbb{1}_1 X_2$

In matrix format, this would be:

$$Flip2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Note that the matrix is block diagonal when it affects only one qubit—no cross talk. Applying this to the first Bell state: $|\psi> = (|00> + |11>) / \sqrt{2}$ yields:

$$Flip2|\psi> = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} / \sqrt{2} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} / \sqrt{2} = (|01> + |10>)/\sqrt{2}$$

5. Measurement

Recall from the single-qubit case that each measuring device has a preferred basis, and it forces the state vector to become one of its basis vectors. The choice of which basis vector is random, with distribution governed by the amplitudes of the state vector in that basis. The same applies to a multi-qubit measurement except for the fact that not all qubits need to be measured, and the measurement need not be projective (are you 0 or 1) but can also be comparative (is qubit 1 and 2 in the same state). To represent such a measurement, we associate with each device a *decomposition* of the vector space. In other words, it views S as a *direct sum* of orthogonal subspaces and projects the state onto one of them. Which subspace is chosen is random with distribution governed by the amplitude of the state vector within that subspace. This approach, albeit more abstract, is not different from the one used in the single-qubit case, in which the device also partitioned the space into a direct sum of two subspaces each spanned by one of its two basis vectors.

The following examples illustrates the decomposition concept: consider the 2-qubit state:

$|\psi\rangle = (2|00\rangle + |01\rangle + 3|10\rangle + |11\rangle) / \sqrt{15}$

Measurement of both qubits in the standard basis
In this case decomposition is easy because the state vector is already written as a direct sum of four orthogonal subspaces each associated with an outcome. Hence, we can easily see that the outcome will be: 00 with probability 4/15, 01 with probability 1/15, 10 with probability 9/15, 11 with probability 1/15.

Measurement of the first qubit only in the standard basis
We rewrite the state as a linear combination of two normalized and orthogonal states, with the first qubit being 0 in one and 1 in the other:

$|\psi\rangle = 1/\sqrt{3} \times (2|00\rangle + |01\rangle) / \sqrt{5} + \sqrt{10}/15 \times (3|10\rangle + |11\rangle) / \sqrt{10}$

This looks like: $1/\sqrt{3} \times |\alpha\rangle + \sqrt{10}/15\ |\beta\rangle$, where $|\alpha\rangle$, $|\beta\rangle$, are orthonormal. Hence, this device will measure 0 for the first qubit with probability 1/3 and 1 with probability 10/15 or 2/3. And if 0 (or 1) was obtained, then the post-measurement state would be: $|\alpha\rangle$ (or $|\beta\rangle$).

Measurement of the equality of the two qubits
We rewrite the state as a linear combination of two normalized and orthogonal states, with the two qubits equal in one and unequal in the other:

$|\psi\rangle = 1/\sqrt{3} \times (2|00\rangle + |11\rangle) / \sqrt{5} + \sqrt{10}/15 \times (|01\rangle + 3|10\rangle) / \sqrt{10}$

Hence, this device will find the two qubits equal with probability 1/3 and unequal with probability 2/3. The post-measurement state would respectively be $|\alpha\rangle$ or $|\beta\rangle$.

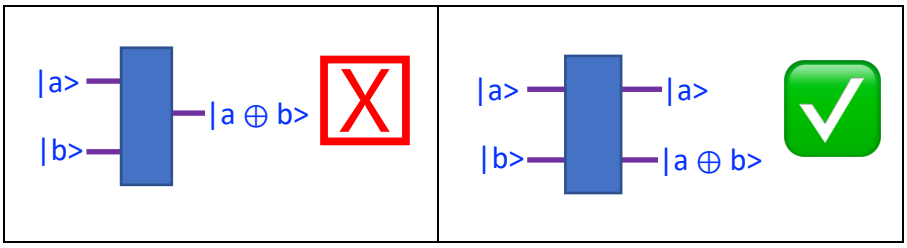
6. Gates

Recall that when transformations are performed as part of an algorithm), we call them gates. For single-qubit gates, we found that the rotation gate set is universal, and hence, we can build any gate out of its elements. For multi-qubit systems, it can be shown that any universal single-qubit set plus CNOT (discussed below) is approximately universal. Moreover, the Solovay-Kitaev theorem proves that we can approximate any n-qubit gate using only $2^n \lg^c(1/\varepsilon)$ gates from the universal set, where ε is the approximation error. Let us take a look at a few multi-qubit gates:

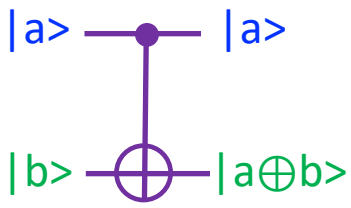
- **The CNOT (Controlled Not) Gate**
  As in the single-qubit case, we often define the gate's action on classical bits and then invoke linearity to infer its action on qubits. Given two bits a and b, this gate generates the NOT of b in a controlled manner: it generates it if a = 1 and does not generate it if a = 0. In a programming context, this gate gives us the power of selection—an if statement—and enables us to build elaborate logic in our algorithms.

  The function of this gate can be captured by the xor operation (addition mod 2) whose symbol is $\oplus$. We can therefore express the gate's output as $a \oplus b$.

In terms of design, our first attempt (shown below to the left) lets the gate take two bits as input and produce one output, being a ⊕ b. But this is not realizable because it is not reversible (given the output, we cannot reproduce the input. Hence, we must design it with two outputs, with one of them being a, as shown to the right. The input can be reproduced from this output, e.g. simply pass the output through the same gate again and you get the input back.



It is customary to refer to this gate as CX, since X is a NOT gate and C standard for "Controlled". It is also customary to draw this gate as shown in the figure to the right. The filled circle is on the **control** input (the one whose state controls whether we negate or not). The "xor" ⊕ symbol is placed on the so-called **target** input (the target of the possible negation).



It should be kept in mind, however, that the "control" / "target" terminology stems from treating all inputs as classical bits. Once we promote either input to a qubit in a superposition state, these terms lose their meanings, and can in fact become misleading, as we shall see.

Let us now look at the quantum case. If the input qubits are not in superposition, the output is similar to the classical case (we treat the top input as the leftmost qubit):

CX |0>|0> = |0>|0 ⊕ 0> = |00>

CX |0>|1> = |0>|0 ⊕ 1> = |01>

CX |1>|0> = |1>|1 ⊕ 0> = |11>

CX |1>|1> = |1>|1 ⊕ 1> = |10>

For inputs with superposition, we figure out the output through linearity, e.g.

CX 1/√2 (|0> + |1>) |0> = 1/√2 CX |00> + 1/√2 CX |10>) = 1/√2 (|00> + |11>)

Note that the 2-qubit input is a product (separable) state, yet the output is an entangled Bell state. Note also that the classical notions of an unchanging control in the upper output and a conditionally negated target in the lower have both lost their meanings.

To compute the CNOT matrix, we can compute its elements from the classical definition since its rows and columns are labeled by classical 0/1 labels (non-superposition). This leads to:

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

A second way to derive the matrix would be to start with an operator description. Considering the classical basis states, if the upper input is |0>, then the upper output is also |0>, and this leads us to |0><0|. In this case, the lower input is not negated and emerges unchanged, i.e. its operator is the identity. We therefore have |0><0| ⊗ 𝟙 for the 0 case. For the 1 case, the upper output is also 1 but the lower is the negation of the lower input. We thus have:

CX = |0><0| ⊗ 𝟙 + |1><1| ⊗ (|1><0| + |0><1|)

We can turn this into a matrix manually or via this short program:

```
import numpy as np
ket0 = np.matrix([[1],[0]]); bra0 = np.conjugate(ket0.T)
ket1 = np.matrix([[0],[1]]); bra1 = np.conjugate(ket1.T)
term1 = np.kron(np.kron(ket0, bra0), np.eye(2))
x = np.kron(ket1, bra0) + np.kron(ket0, bra1)
term2 = np.kron(np.kron(ket1, bra1), x)
cnot = term1 + term2
print("\nCNOT:\n", cnot)
```

The eye(2) function returns the 2x2 unit matrix.

The CNOT gate is an example of singly controlled gates, i.e. regular single-qubit gates that acts on the second qubit if the (classical) control is 1 and leaves it unchanged if the control is 0. For example, you can build a **Controlled-Phase gate** or a **Controlled Z gate**.

- **The Multi-Qubit Hadamard Gate**

We know that H transforms |0> to |+> and |1> to |−> and we have determined its matrix for a single qubit. To obtain the matrix for 2 qubits, we simply compute the tensor product:

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

We can also do this via operators. Here is an outline:

H |0>|0> = (|0> + |1>)/√2 ⊗ (|0> + |1>)/√2 = ½ (|00> + |01> + |10> + |11>)

which agrees with the top row of the matrix above. Similarly:

H |0>|1> = (|0> + |1>)/√2 ⊗ (|0> − |1>)/√2 = ½ (|00> − |01> + |10> − |11>)

which agrees with the second row, and so on.

Generalizing this to n bits leads to the so-called **Walsh-Hadamard Transform W**:

$$W|0>^{\otimes n} = H^{\otimes n}|0>^{\otimes n} = \frac{|0>+|1>}{\sqrt{2}} \otimes \frac{|0>+|1>}{\sqrt{2}} \ldots \frac{|0>+|1>}{\sqrt{2}}$$

Hence:

$$H^{\otimes n}|0>^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k>$$

Similarly:

$$W|11111\ldots1> = \frac{|0> - |1>}{\sqrt{2}} \otimes \frac{|0> - |1>}{\sqrt{2}} \ldots \frac{|0> - |1>}{\sqrt{2}}$$

Hence:

$$W|11111\ldots1> = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} (-1)^{f(k)}|k>$$

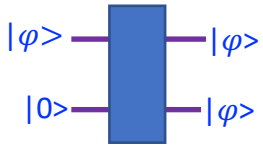
where *f(k)* is the number of 1-bits in the binary representation of k. This can also be captured by the dot notation: x.y is the number of common 1's in the binary representations of the integer x and the integer y. Hence, our *f(k)* can be expressed as $k.2^{n-1}$ because 2^{n-1} has all 1's in its binary representation.

The dot notation allows us to write an expression for the transformation of any state by W, as follows:

$$W|m> = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} (-1)^{k.m}|k>$$

- **The IMPOSSIBLE Gate (No Cloning Theorem)**

  Let us try to design a gate that clones an unknown state. We imagine feeding it some input |a> and getting two outputs both of them are equal to x. To ensure reversibility, we need a second input to the gate, say a qubit in state |0>.



We start with classical bits. If we denote the gate's operator by CL (for Clone) then:

CL |0>|0> = |00> and CL |1>|0> = |11>

For an arbitrary (unknown) superposition |φ> = a|0> + b|1>, we compute the output of the sought gate in two ways:

◊ From linearity:

CL |φ>|0> = (a|0> + b|1>) ⊗|0> = a|00> + b|10>

◊ From the gate's definition:

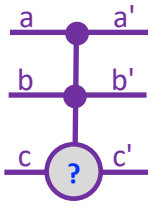
CL |φ>|0> = |φ> |φ> = (a|0> + b|1>) ⊗ (a|0> + b|1>) = a^2|00> + ab|01> + ba|10> + b^2|11>

For these two expressions to be equal, we must have:$a^2 = a$, $b^2 = b$, $ab = 0$, and $ba = 0$, which is clearly impossible unless ab=0, which means no superposition, just a classical state.

This is the proof of the **no-cloning theorem**.

- **Multiply Controlled Gates**

Building on CNOT, we can take any single-qubit gate and control it by multiple lines so that it acts only if all these controls are 1. For example, the **Toffoli gate T** is a doubly controlled NOT, and its (classical) action is:



a' = a;

b' = b;

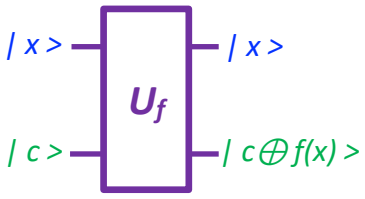
c' = NOT c if a ∧ b = 1 else c

This gate is very useful because you can adapt it to implement AND, XOR, NAND, etc.

- **The Function Gate U_f**

Classically, we know that any Boolean function of one or more Boolean variables can be implemented in hardware using building block gates from a universal set such as AND/OR/NOT. How can we do the same on a quantum computer? The classical gates can be replaced with quantum gates (reversible) using, for example, Toffoli gates. But how do we invoke the function, i.e. have the computer compute it for a given x? This is where the U_f gate is needed.

As in CNOT, we make this gate reversible by supplying x and some ancillary qubit as input. The gate contains within it the converted classical circuit that computes *f(x)*. On output, the gate generates x in the upper output and the xor of the ancillary with *f(x)*. Reversibility is manifest since passing this output through a second U_f would reproduce the input.



As in all other gates, the specified mechanism is applied per basis vector (a classical bit) and the results are then combined, with amplitudes, to handle the quantum case. For example, if x is a superposition, then we compute the output for |0> and |1> separately and then combine. The same applies if c is also a superposition. Here is a more specific example in which x is an equal superposition and c=0:

$U_f\,(|0\rangle + |1\rangle)/\sqrt{2} \otimes |0\rangle = 1/\sqrt{2}\,(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle)$

Notice that *f(0)* and *f(1)* were computed in just one pass. This is the power of superposition (we saw this in the Apple gedanken when the person traversed both corridors in one pass). Note as well that the two outputs are entangled.

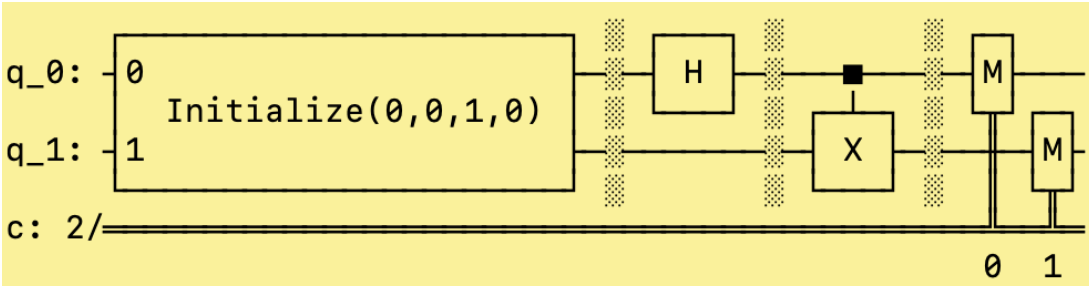
The gate can be generalized to handle the case of an integer function of an integer variable. If the integer variable can be represented in m bits and the function value in n bits, then the gate would have m+n inputs and outputs—the number of qubits in c and f must be the same.

7. Circuits and Qiskit

As we did in the single-qubit case, we can design, draw and visualize, simulate, and actually run multi-qubit circuits using Qiskit. We just need to be careful here about specifying which qubit is being targeted by a particular gate. As an example, here is a program that operates on 2 qubits initialized to |10>. The circuit applies H to the top qubit only and then apply CNOT on both.

```
import numpy as np
from qiskit import *
backend = Aer.get_backend('qasm_simulator')
q = QuantumRegister(2,'q')
c = ClassicalRegister(2,'c')
qc = QuantumCircuit(q,c)
qc.initialize([0,0,1,0], q)
qc.save_statevector(label='v0')
qc.h(q[0])
qc.save_statevector(label='v1')
qc.cx(q[0], q[1])
qc.save_statevector(label='v2')
qc.measure(q,c)
print(qc.draw(output='text'))
job = execute(qc, backend, shots=1000)
result = job.result()
counts = result.get_counts()
print(counts)
for i in range(0,3):
    print("At barrier "+str(i)+":")
    print(np.asarray(result.data(0)['v'+str(i)]))
```

Here is the output:




```
{'10': 507, '01': 493}
At barrier 0:
[0.+0.j 0.+0.j 1.+0.j 0.+0.j]
At barrier 1:
[0.        +0.j 0.        +0.j 0.70710678+0.j 0.70710678+0.j]
At barrier 2:
[0.        +0.j 0.70710678+0.j 0.70710678+0.j 0.        +0.j]
```

- Create a circuit that takes two qubits and applies 3 CNOT gates on them such that the top qubit is the control in the first and the last gate, while the bottom qubit is the control in the middle gate. Show that this circuit swaps the two input states.

  *Note: the two input states are unknown, and this circuit swaps them without leaking any information about them, i.e. it is a transformation, not a measurement. For this reason, it is referred to as a **SWAP Gate**.*

- Create a circuit for the **Fredkin Gate F**, which is a singly controlled Swap gate. Show that F can be adapted to implement NOT, OR, and AND.

Remarks

1. The state vector of an n-qubit system is a linear combination of 2^n terms, each of which has an amplitude—a complex number. Hence, there are in total $2x2^n$ real numbers.
2. Imposing normalization on the above, and factoring in the irrelevance of the global phase, still leaves $2^{n+1} - 2$ real numbers.
3. Suppose we have only 100 qubits (a ridiculously small number since there are more than 10^{23} atoms in a few grams of any material). To represent their state, we need to keep track of over 2^{100} real numbers. That is more than the number of atoms in the entire observable Universe!! How / where is that number stored in the quantum world?
4. The above sheds light on the immense richness of the information content, and the vast computing potential of the quantum world.
5. Note that the number of amplitudes in the most general entangled state drops from 2^n to merely 2xn if the state is separable (because it would be a product of two factors each of which has up to n terms). Hence, it is the entanglement of multiple qubits, not the super-position of individual qubits, that is responsible for the exponential growth.