# PHY256 Lecture notes: Introducing Quantum Computing

A. C. Quillen

April 25, 2021

## Contents

## 1 Universal gate sets on a classical computer

We start with a Boolean function

$$f(x) \rightarrow \;\; 1 \;\text{ or }\; 0 \tag{1}$$

where $x$ is a string of $n$ bits;

$$x = i_0, i_1, i_2, i_3....i_{n-1}$$

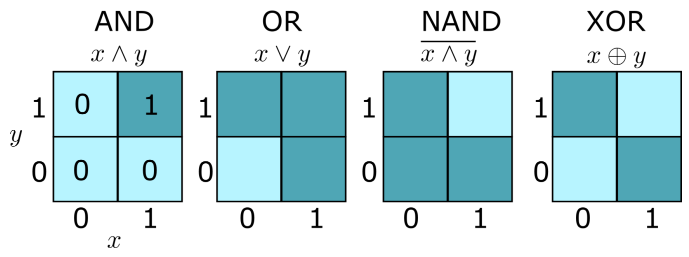

Figure 1: Various logical operations. Here $x, y \in \{0, 1\}$.

and the digits $i_b \in \{0, 1\}$.

Any Boolean function in the form of equation 1 can be written as a set of conditions involving the Boolean operations AND, OR and NOT. We make a list of all strings that satisfy the function $f(x) = 1$. Suppose that $x_a$ is a specific string that satisfies the function so $f(x_a) = 1$. Suppose our $x_a = 010....$ We can construct a Boolean function like this

$$\bar{i}_0 \wedge i_1 \wedge \bar{i}_i ...$$

that is only satisfied if the first bit is 0 and the second bit is 1 and the third bit is 0, etc. Here $\wedge$ is the AND function and $\bar{i}$ we use for NOT $i$ for digit $i$. I can adjust the position of the NOTs so that the function is only 1 for the specific string $x_a$. This gives me a function $f_a(x)$ that is only 1 if $x = x_a$.

A Boolean version of $f(x)$ can be written as

$$f_0(x) \vee f_1(x) \vee v_2(x) \vee f_3(x)...$$

where $\vee$ is the OR gate. where each of the functions $f_a()$ is for a specific string $x_a$ that satisfies the function $f(x_a) = 1$.

We have used three gates (AND, OR, NOT). With these we could construct any Boolean function in the form of equation 1 In this sense the set (AND, OR, NOT) is a **universal** gate set for classical computing.

AND and OR alone are not a universal gate set. The set (AND, OR) is not universal as it is linear or monotone or affine mod 2. One way to see this is to look at Figure 1. You can't combine them to make a XOR.

Any Boolean logical operation involving $n$ bits can be constructed of NAND operations involving 2 bits. Thus NAND is a **universal gate set** for Boolean logic circuits and so for classical computing. The NAND gate is **universal** as all other logic computations can be constructed from NANDs. For example, NOT is constructed by connecting both inputs together, the output is NAND(x,x) = NOT x. An AND is constructed by with NOT(NAND(x,y)). The other gates follow from combinations of NANDs, ANDs and NOTS. Some examples are shown in Figure 1.

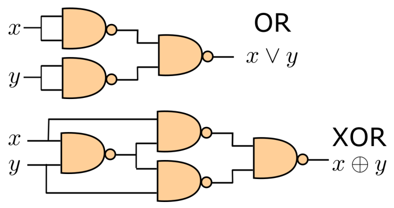

Figure 2: Constructing OR and XOR from NANDs. Here $x, y \in \{0, 1\}$.

## 2 Logical operations on a Quantum computer

Can a quantum computer do the same calculations as a classical computer? We start with Boolean operations and devise ways to carry them out on a quantum computer.

### 2.1 Classical logical operations on a two bit quantum computer

The NOT function can be done on a single qubit with the Pauli-X operator as $\sigma_x |0\rangle = |1\rangle$ and $\sigma_x |1\rangle = |0\rangle$.

The CNOT (controlled NOT) gate takes two bits $x, y$ and computes $x, x + y$ (with circuit shown in Figure 3. Here $x, y \in \{0, 1\}$.

$$\text{CNOT} : |xy\rangle \to |x, x + y\rangle$$

It flips the second bit only if the first one is 1. Here $x + y$ is computed mod 2 so it is 1 for bits $xy = 01$ or 10 and is 0 for bits $xy = 00$ or 11. The CNOT logic operation on the second bit output is equivalent to the XOR (exclusive OR) of $x, y$. The XOR logical operation is sometimes written as $x \oplus y$.

| $x$ | $y$ | $x$ XOR $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

The CNOT is invertible as $\text{CNOT}^2 = \mathbf{I}$ gives back the identity.

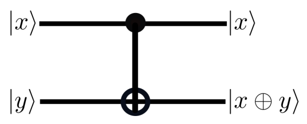

Figure 3: The CNOT (controlled NOT) performs a XOR operation and puts the result in the top qubit. Here $x, y \in \{0, 1\}$.

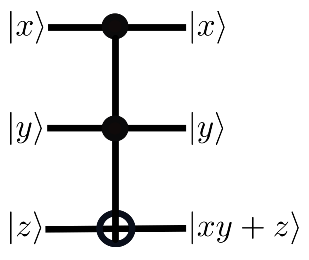

Figure 4: The circuit notation for the 3 bit Toffoli gate. The bottom bit is flipped if both the other two bits are $|1\rangle$. Here $x, y, z \in \{0, 1\}$.

Unitary transformations are reversible as each unitary transformation has an inverse. However, some logical functions on two bits are **not** reversible.

How many possible functions take two classical bits and give two bits? We take bits $x, y \to x', y'$. There are 4 possible inputs $00, 01, 1011$, and 4 possible outputs $00, 01, 10, 11$ for each input. The number of possible functions is $4^4 = 256$. So the number of possible functions is 256.

How many of these possible operations can be done to two qubits and with unitary transformations so that they are reversible? It turns out that there are only 24 possible reversible two bit gates that act like Boolean functions of two bits. The reversible ones are all affine, so AND, which is not affine, is missing. With only two qubits we cannot simulate all classical Boolean logic operations.

**This is not all that clear.**

## 2.2 3-bit gates: the Toffoli gate

The Toffoli gate is a controlled-controlled NOT or a ccNOT. It is a 3-qubit gate that flips the third bit if and only if the first two bits are both 1 (see Figure 4). Another way to write the Toffoli gate is

$$|x, y, z\rangle \to |x, y, xy + z\rangle$$

where the arithmetic is mod 2 and $x, y, z \in \{0, 1\}$. The Toffoli gate is special in that it has a product in the output and so it is non-linear! This differs from controlled two bit gates which are affine.

The 3 qubit Toffoli gate can be used to construct a NAND. The third bit is the output of a NAND given the inputs of the other two and setting the $z$ bit to 1. In other words

$$\text{To}\, |x, y, 1\rangle = |x, y, xy + 1\rangle = |x, y, \overline{xy}\rangle \,.$$

With three qubits on a quantum computer we can construct a NAND gate. Because of the classical universality of the NAND gate, our quantum computer has a universal gate set for classical logical operations. A quantum computer that can perform a NAND can execute all operations that are possible on a classical computer.

**Question:** Is the Toffoli gate reversible?

To answer this question we compute the following for $\text{To}(0,1,2)$ where control bits are 0,1 and the target bit is 2.

$$\begin{aligned}
\text{To}\,|000\rangle &= |000\rangle \\
\text{To}\,|001\rangle &= |001\rangle \\
\text{To}\,|010\rangle &= |010\rangle \\
\text{To}\,|011\rangle &= |011\rangle \\
\text{To}\,|100\rangle &= |100\rangle \\
\text{To}\,|101\rangle &= |100\rangle \\
\text{To}\,|110\rangle &= |111\rangle \\
\text{To}\,|111\rangle &= |110\rangle\,.
\end{aligned}$$

The Toffoli gate resembles the identity except $|110\rangle \leftrightarrow |111\rangle$. The matrix representation for this Toffoli gate using the order given above for the basis vectors is

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

As the Toffoli gate is unitary, the operation is reversible, so the answer to the above question of whether it is invertible is **Yes**.

The logical operation alone (the AND) need not be invertible. Three bits are used as the extra information is stored in the other two bits.

**Question:** Is it possible to construct a 3-bit ccNOT (or Toffoli gate) using 2-qubit gates? You can apply the 2-qubit gates to different pairs of bits.

Answer: Yes. But it requires at least 5 2-qubit gates. Apparently, the Toffoli gate can be constructed from single qubit gates and six CNOTs. All you need to achieve all logical operations on a quantum computer is a set of single qubit gates and a set of pairs of controlled gates.

## 2.3 Quantum versions of some classical bit operations

Using the Toffoli gate

$$\text{To}\,|x,y,z\rangle = |x,y,z+xy\rangle$$

we construct NOT, XOR, AND and NAND operations.

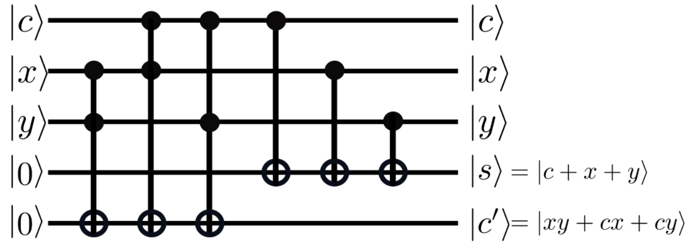

Figure 5: The quantum circuit notation for a 1-bit adder. Here $x, y$ are added. The output bit is $s$. Input is also a carry bit $c$ and a new carry bit is output as $c'$. The circuit is built from Toffoli gates and CNOTs. A series of these could be used to make a multi-bit adder. Here $x, y, z, c, c', s \in \{0, 1\}$.

| Logical operations with a 3-bit Toffoli gate | | | |
|---|---|---|---|
| NOT | $x$ | To $\|x,x,1\rangle = \|x,x,x+1\rangle$ | $\|x,x,\bar{x}\rangle$ |
| XOR | $x \oplus y$ | To $\|1,x,y\rangle = \|1,x,x+y\rangle$ | $\|1,x,x \oplus y\rangle$ |
| AND | $x \wedge y$ | To $\|x,y,0\rangle = \|x,y,xy\rangle$ | $\|x,y,x \wedge y\rangle$ |
| NAND | $\overline{x \wedge y}$ | To $\|x,y,1\rangle = \|x,y,xy+1\rangle$ | $\|x,y,\overline{x \wedge y}\rangle$ |

Here $x, y \in \{0, 1\}$, and we are using arithmetic mod 2, so $x + y$ is $x$ XOR $y$, and is not $x$ OR $y$.

Other logical operations can be constructed from these. For example

$$x \text{ OR } y = x \vee y = x + y + xy = (x \text{ XOR } y) \text{ XOR } (x \wedge y).$$

Another example

$$x \text{ OR } y = \text{NOT}(\text{NOT}x \wedge \text{NOT}y)$$

## 3 Realizing Unitary Transformations and Universality on a Quantum Computer

### 3.1 What is meant by universality on a quantum computer?

**Question:** Will we able to solve any problem on a quantum computer that can't be solved on a classical computer, regardless of resources?

Answer: No. However, the number of states in an $N$ qubit machine is $2^N$. This number gets large quickly as $N$ increases. So a few qubit system could be a powerful computer.

Consider all unitary transformations of an $N-$qubit system. These form a continuous manifold that has dimension $2^N \times 2^N$. We differentiate between unitary transformations of an $N$-qubit system and measurements which are not unitary. Computations are usually assumed to first put qubits in a particular set of states, then carry out a unitary transformation, and then measure all the qubits. Measurement in the middle of a computation is not necessary though some algorithms use this capability.

**Question:** It is possible to implement *any* unitary transformation, using many simple operators or gates that only involve 1, 2 or 3 qubits.

Answer: Yes. More on this below!

A minimal set of gates that achieves **complete control** of a quantum system is called a **universal gate set**.

**Question:** What is meant by **complete control** of a quantum system?

Answer: By complete control we mean that any unitary transformation can be approximated to a desired accuracy with a finite sequence of gates in a universal gate set. A notion of distance between unitary operations is required to know exactly how many gates are required to achieve this.

## 3.2 Single qubit unitary transformations

To build up a universal gate set we first must be able to perform any unitary possible transformation on a single qubit, or at least we should be able to perform a unitary transformation that is sufficient near a desired one.

Any unitary transformation on a single qubit can be decomposed as a sequence of three types of operations with each type depending on an angle. The three types of transformations are:

| Unitary Transformations on a single qubit | | | |
|---|---|---|---|
| $K(\delta)$ | $e^{i\delta}\mathbf{I}$ | A global phase shift by $\delta$ | |
| $R(\beta)$ | $\begin{pmatrix} \cos\beta & \sin\beta \\ -\sin\beta & \cos\beta \end{pmatrix}$ | A rotation by $\beta$ | Rotates about y-axis on Bloch sphere |
| $T(\alpha)$ | $\begin{pmatrix} e^{i\alpha} & 0 \\ 0 & e^{-i\alpha} \end{pmatrix}$ | A phase rotation by $\alpha$ | Rotates about z-axis on Bloch sphere |

Any unitary transformation $Q$ can be decomposed into a sequence involving 4 angles

$$Q = K(\delta)T(\alpha)R(\beta)T(\gamma). \tag{2}$$

Let's multiply this out

$$Q = e^{i\delta} \begin{pmatrix} \cos\beta\ e^{i(\alpha+\gamma)} & \sin\beta\ e^{i(\alpha-\gamma)} \\ -\sin\beta\ e^{-i(\alpha-\gamma)} & \cos\beta\ e^{-i(\alpha+\gamma)} \end{pmatrix}. \tag{3}$$

We have said that any unitary transformation can be written in this way. Consider

$$Q = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}.$$

For $Q$ to be unitary,

$$\begin{aligned} QQ^\dagger &= \mathbf{I} \\ &= \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix} \begin{pmatrix} u_{00}^* & u_{10}^* \\ u_{01}^* & u_{11}^* \end{pmatrix} \\ &= \begin{pmatrix} u_{00}u_{00}^* + u_{01}u_{01}^* & u_{00}u_{10}^* + u_{01}u_{11}^* \\ u_{00}^*u_{10} + u_{01}^*u_{11} & u_{10}u_{10}^* + u_{11}u_{11}^* \end{pmatrix} \end{aligned}$$

and

$$u_{00}u_{00}^* + u_{01}u_{01}^* = 1 \tag{4}$$
$$u_{00}u_{10}^* + u_{01}u_{11}^* = 0 \tag{5}$$
$$u_{10}u_{10}^* + u_{11}u_{11}^* = 1. \tag{6}$$

These imply that

$$u_{00}u_{00}^* = u_{11}u_{11}^* \tag{7}$$
$$u_{01}u_{01}^* = u_{10}u_{10}^*. \tag{8}$$

We now show that any unitary transformation of a qubit can be written in the form of equation 3. Let's write component $u_{00}$ in terms of an amplitude and phase and similarly for the other matrix components. Equation 4 implies that we can set the amplitudes of $u_{00}$ and $u_{01}$ to depend on an angle $\beta$ giving

$$\begin{aligned} u_{00} &= \cos\beta e^{i\theta_{00}} \\ u_{01} &= \sin\beta e^{i\theta_{01}} \end{aligned}$$

where $\theta_{00}, \theta_{01}$ are the phases of $u_{00}$ and $u_{01}$. Similarly equation 6 implies that we can chose an angle $\beta'$

$$\begin{aligned} u_{01} &= \sin\beta' e^{i\theta_{01}} \\ u_{11} &= \cos\beta' e^{i\theta_{11}} \end{aligned}$$

where $\theta_{01}, \theta_{11}$ are the phases of $u_{10}$ and $u_{11}$. Equation 7 implies that $\cos^2\beta = \cos^2\beta'$ and equation 8 implies that $\sin^2\beta = \sin^2\beta'$. We can set $\beta = -\beta'$ and let the phases absorb the sign ambiguity. We are left with 4 phases that can be chosen and we have a remaining constraint from equation 5. The constraint from equation 5 gives

$$\theta_{00} - \theta_{10} = \theta_{01} - \theta_{11}$$

and lets us restrict the choice of possible phases to three phases, $\alpha, \gamma, \delta$ by solving

$$\begin{aligned}\delta + \alpha + \gamma &= \theta_{00}\\ \delta + \alpha - \gamma &= \theta_{01}\\ \delta - \alpha + \gamma &= \theta_{10}\\ \delta - \alpha - \gamma &= \theta_{11}.\end{aligned}$$

The result can be written as in equation 3.

## 3.3 How to approximate any unitary transformation of a single qubit using a small set of gates

We show how the following set the Hadamard, $H$, $P_{\frac{\pi}{2}}$ and $P_{\frac{\pi}{4}}$ are enough to approximate any unitary transformation of a single qubit.

The three gates are

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \qquad P_{\frac{\pi}{2}} = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix} \qquad P_{\frac{\pi}{4}} = \begin{pmatrix}1 & 0\\ 0 & e^{\frac{i\pi}{4}}\end{pmatrix}.$$

The phase gate $P_{\frac{\pi}{2}}$ involves $\pi/2$ because $e^{\frac{i\pi}{2}} = i$. The two phase gates can be written in terms of the phase rotation gate $T()$

$$P_{\frac{\pi}{2}} = e^{\frac{i\pi}{4}} T\left(-\frac{\pi}{4}\right) \qquad P_{\frac{\pi}{4}} = e^{\frac{i\pi}{8}} T\left(-\frac{\pi}{8}\right).$$

The gate $P_{\frac{\pi}{2}}$ is also called the phase gate and sometimes it is given the symbol $S$.

Suppose we consider transformations of an angle $\theta$ on a circle. We translate the angle

$$f(\theta) = \left(\theta + \frac{c}{2\pi}\right)_{\text{mod } 2\pi}.$$

We start with $\theta_0$ and consider its orbit $f(\theta_0), f^2(\theta_0), f^3(\theta_0)$..... If $c$ is a rational number $c = p/q$ (with $p, q$ integers) then $f^q(\theta_0) = \theta_0$. The orbit contains at most $q$ points. However if $c$ is irrational then the orbit never repeats. Furthermore after many iterations of $f$ the distribution of points is uniform. Suppose we would like to get to $\theta_*$. For any initial $\theta_0$ and any desired $\theta_*$ if we do enough iterations we can get arbitrarily close. That means we can find an iteration number $i$ that gives $|f^i(\theta_0) - \theta_*| < \epsilon$ for any small number $\epsilon$. The smaller the value of $\epsilon$, the larger the number $i$ of iterations is required to reach any $\theta_*$.

Why is this relevant for our problem of being able to approximate any unitary transformation of a single qubit? If we can use our set of gates to construct a unitary transformation that is not periodic and is like an irrational rotation and our set of gates are not restricted to give a single great circle on the Bloch sphere, then we will be able to approximate any unitary transformation.

The transformation $H$ rotates by $\pi/2$ about the $y$ axis on the Bloch sphere. In the standard basis, the Hadamard is real so it cannot introduce any complex phases if the original wave vector has real coefficients.

The transformation $P_{\frac{\pi}{4}}$ rotates by $\pi/4$ about the $z$-axis on the Bloch sphere. This follows as $|0\rangle, |1\rangle$ are eigenvectors and $(P_{\frac{\pi}{4}})^4 = \mathbf{I}$.

The transformation

$$A = HP_{\frac{\pi}{2}}H = \frac{1}{2}\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix}$$

rotates by $\pi/2$ about the $x$-axis on the Bloch sphere. This follows as $|+\rangle, |-\rangle$ are eigenvectors and $A^4 = H(P_{\frac{\pi}{2}})^4H = \mathbf{I}$. (Recall that $H^2 = \mathbf{I}$).

The transformation

$$\begin{aligned} B &= HP_{\frac{\pi}{4}}H \\ &= \frac{1}{2}\begin{pmatrix} 1+e^{\frac{i\pi}{4}} & 1+e^{-\frac{i\pi}{4}} \\ 1-e^{\frac{i\pi}{4}} & 1-e^{-\frac{i\pi}{4}} \end{pmatrix} \end{aligned}$$

rotates by $\pi/4$ about the $x$-axis on the Bloch sphere. The transformations $H, P_{\frac{\pi}{2}}, P_{\frac{\pi}{4}}, A, B$ are rational as if we apply them 4 or 8 times we get the identity.

Consider a rotation by $\pi/4$ about the $x$ axis followed by a rotation of $\pi/4$ about the $z$-axis on the Bloch sphere. The transformation

$$V = P_{\frac{\pi}{4}}B$$

is irrational! The product of two rational rotations of a sphere can be irrational if the two transformations rotate the sphere about different axes.

Since $H, P_{\frac{\pi}{4}}, B$ and $V$ have different rotation axes and $V$ is irrational, we can reach all possible rotations of the Bloch sphere with a series of these gates. With application of enough gates, we can match any desired unitary transformation (of a single qubit) to any desired level of precision.

## 3.4 Universal gate sets

Now that we can reach any unitary transformation for a single qubit, we explore what is needed to reach any unitary transformation on an N-qubit system.

We chose a gate set that involves pairs or triplets of qubits and unitary transformations on single qubits.

**Why does each gate act on only a few qubits?**

Physical interactions which are often local. On classical computers, complex calculations are built from simple ones and we expect that we can do the same on a quantum computer. Realizations of quantum computers may have a limited set of possible operations. In other words, it may be easier to build a quantum computer that has a limited number of possible operations.

**What is the role of interference in quantum computing and what is the role of entanglement?** The CNOT entangles two qubits but the Hadamard gives a superposition of states in a single qubit. If we place the wave vector in a superposition and arrange the calculation so results constructively interfere and unwanted results cancel themselves out, we can simultaneously calculate more than one thing. This implies that we need superposition. If we don't use entanglement then our quantum computer is not really a quantum computer. We only reach the full capability of the quantum computer if we use the full space of possible wave vectors. This implies that we also need entanglement.

### 3.4.1 Non-universal gate sets

A bad (or non-universal) gate set has no superposition. If we only use CNOTs (between pairs of bits) we interchange states but we can't create a superposition. A Hadamard gate gives superposition so we need a gate like the Hadamard gate.

A bad gate set has no entanglement. The quantum computer is like a classical one. A CNOT gives entanglement so we need a two-bit controlled gate like the CNOT gate.

A bad gate set has only real gates. It would not even be able to reach all unitary transformations on a single qubit. A phase gate $P_{\frac{\pi}{2}}$ is complex so we need a gate like the phase gate.

A bad gate set gives a finite subgroup of $U(2^N)$ where $N$ is the number of qubits. An example is the gate set

$$\{\mathrm{CNOT}, H, P_{\frac{\pi}{2}}\} \tag{9}$$

where the phase gate $P_{\frac{\pi}{2}} = \mathrm{diag}(1, i)$. Here the two qubit CNOT gate can operate on any pair of gates and the single qubit Hadamard $H$ and phase $P_{\frac{\pi}{2}}$ gates can operate on any qubit. This gate set generates a finite group known as the Clifford group[1]. Relevant is the Gottesman-Knill theorem which states that a quantum circuit constructed using only these gates can be simulated efficiently (in polynomial time) on a classical computer. So despite entanglement (from the CNOT) and superposition (from the Hadamard), the gate set is not universal. While it would appear this gate set is too small to be useful, it turns out to be quite useful for error correction. In this setting the gate set in equation 9 is used to correct both bit flipping and phase flipping errors.

[1]The Clifford group contains normalizers of the Pauli spin matrices for each qubit. The group contains unitary operators $V$ that satisfy $VPV^\dagger = P$ for all $P \in \{\ I, \sigma_x, \sigma_y, \sigma_z\}$. The Hadamard and phase gate transform an axis direction on the Bloch sphere to another axis direction. The Clifford group on a single qubit is like the symmetry group of a cube.

### 3.4.2 Examples of universal gate sets

Apparently the set that generates the Clifford group, along with a single other transformation that is not in the Clifford group, gives a universal gate set. The $R_{\frac{\pi}{8}}$ gate

$$R_{\frac{\pi}{8}} = \begin{pmatrix} \cos\frac{\pi}{8} & -\sin\frac{\pi}{8} \\ \sin\frac{\pi}{8} & \cos\frac{\pi}{8} \end{pmatrix}$$

is not in the Clifford group.

An example of a universal gate set that uses the $R_{\frac{\pi}{8}}$ gate is

$$\{\text{CNOT}, H, P_{\frac{\pi}{2}}, R_{\frac{\pi}{8}}\}. \tag{10}$$

Another example of a universal gate set (by Yaoyun Shi) is

$$\{\text{Toffoli}, H, P_{\frac{\pi}{2}}\}. \tag{11}$$

## 3.5 Solovay-Kitaev Theorem

The **Solovay-Kitaev Theorem** states that we can approximate any unitary transformation on $N$ qubits to within precision $\epsilon$ using $O(4^N \text{polylog}\frac{1}{\epsilon})$ gates. In other words the complexity scales only like some power of $\log\frac{1}{\epsilon}$. The distance between two unitary transformations can be computed using a sum of the norm of the difference in each matrix entry.

(In future more details on the theorem!)

## 4 Circuits that reclaim unused qubits

In Figure 7 we show a circuit that computes using 3 Toffoli gates the AND of four input bits $|n_0\rangle, |n_1\rangle, |n_2\rangle$ and $|n_3\rangle$ The result is in the first or bottom bit. There are two additional

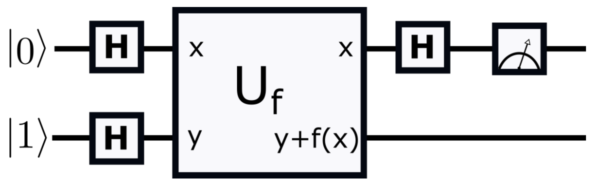

Figure 6: The Deutsch algorithm is the following quantum circuit applied to an initial state of $|01\rangle$. The $U_f$ transformation is $|xy\rangle \to |x, y + f(x)\rangle$ where $x, y \in \{0,1\}$ and the $+$ is mod 2. Here $f()$ is a Boolean function (returning 0 or 1). The goal is to determine if $f(x)$ is constant (with $f(0) = f(1)$) or balanced (with $f(0) = \text{NOT } f(1)$) with a single call of the operator $U_f$.

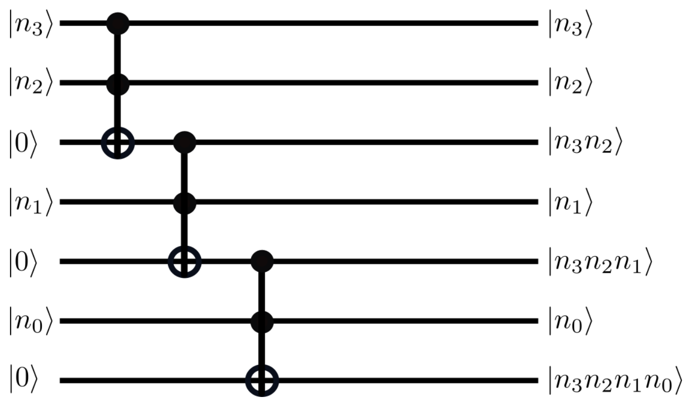

Figure 7: The quantum circuit for a 4 bit conjunction. The output bit is the lowest one and returns $n_0 \wedge n_1 \wedge n_2 \wedge n_3$ or the product base 2 of the four input bits $|n_0\rangle, |n_1\rangle, |n_2\rangle, |n_3\rangle$. Three Toffoli gates are used.

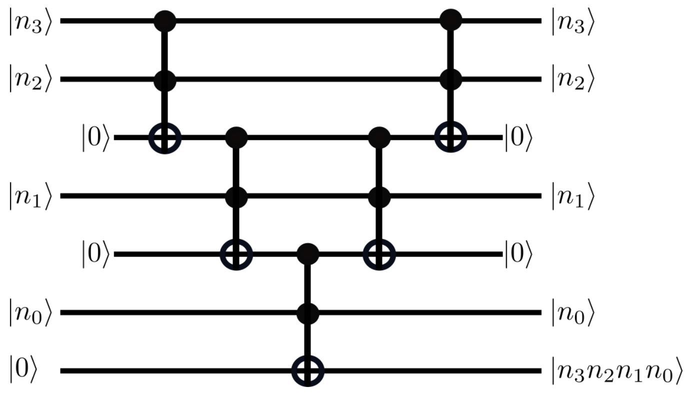

Figure 8: The quantum circuit for a 4 bit conjunction that reclaims temporary bits. Much of the computation is reversed.

bits that were initially set to zero but now contain information that we no longer need. We would rather be able to reuse these bits. Figure 8 shows how most of the circuit is reversed allowing the temporary bits to be reclaimed. The cost is approximately doubling the number of steps. Much of the computation is reversed.

**Why do we need to reclaim lost bits?** Consider the circuit diagram shown in Figure 6 for the Deutsch black box algorithm. A measurement is done on the upper bit but the unitary transformation inside the black box, $U_f$, which is a controlled gate, is only applied on the bottom bit. The circuit looks like the upper bit will not be affected by the unitary transformation $U_f$, however because of the combined actions of the Hadamard transformations, introducing superposition, and the controlled gate, the upper bit is affected by transformations that affect the bottom bit. To reuse bits we need to ensure that they are unentangled from all other bits. Reversing parts of the computation to reclaim bits is a way to ensure that bits can be reused without affecting other parts of the quantum calculation.

## 5 Quantum Random Walks

A popular way to do a random walk is to start with an initial state vector $|\psi\rangle$ in a product Hilbert space and then repetitively perform a series of unitary transformations. This is the type introduced by Aharonov, Davidovich, and Zagury 1993. Quantum random walks can also be called *quantum cellular automata*. Unlike a classical and stochastic random walk, all transformations are unitary and hence reversible. There is no actual stochasticity. The walk does not lose its recollection of the initial state and it cannot converge to a stationary distribution. For some problems, such as propagation along a random tree, quantum walks can give exponential speedups over classical walks. And some quantum algorithms look like quantum random walks.

We explore a discrete quantum random walk of a spin (a two state system) on a circle. The system can be at any location of a discrete set of points on the circle and it can have be in either spin up or spin down states.

Take a state space that is a tensor product of a space with two states (a qubit), $\mathcal{H}_2$, and a space with $N$ states, $\mathcal{H}_N$ (the discrete points on the circle). A basis for this space is

$$|jn\rangle$$

where $j \in [0,1]$ is spin up or down and $n \in [0, N-1]$. The $N$ dimensional Hilbert space can be described as $N$ possible particle positions.

The entire Hilbert space is a tensor product space $\mathcal{H} = \mathcal{H}_2 \otimes \mathcal{H}_N$.

We start with $|\psi\rangle = |00\rangle$ and alternate applying a spin mixing operator

$$\mathbf{H} \otimes \mathbf{I}$$

and a position change that depends on the spin

$$\mathbf{C} = \mathbf{P}_0 \otimes \mathbf{U}_+ + \mathbf{P}_1 \otimes \mathbf{U}_-$$

**H** is the Hadamard operator on the spin state (in $\mathcal{H}_2$) and it takes

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

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

The transformation $\mathbf{H} \otimes \mathbf{I}$ shifts the spin state but operates on a state in $\mathcal{H}$.

$\mathbf{P}_0$ projects the spin state onto $|0\rangle$ and $\mathbf{P}_1$ projects the spin state onto $|1\rangle$,

$$\mathbf{P}_0|0\rangle = |0\rangle$$
$$\mathbf{P}_0|1\rangle = 0$$
$$\mathbf{P}_1|0\rangle = 0$$
$$\mathbf{P}_1|1\rangle = |1\rangle$$

These two operate on spin states, or those in $\mathcal{H}_2$. We can also write

$$P_0 = |0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$
$$P_1 = |1\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

The operators $\mathbf{U}_+$ and $\mathbf{U}_-$ raise and lower $n$

$$\mathbf{U}_+|n\rangle = |n + 1 \bmod N\rangle$$
$$\mathbf{U}_-|n\rangle = |n - 1 \bmod N\rangle$$

These two operate on states in $\mathcal{H}_N$. We can also write

$$\mathbf{U}_+ = \sum_{j=0}^{N-2} |j+1\rangle\langle j| + |0\rangle\langle N-1| = \begin{pmatrix} 0 & 0 & 0 & .. & 0 & 1 \\ 1 & 0 & 0 & .. & 0 & 0 \\ 0 & 1 & 0 & .. & 0 & 0 \\ 0 & 0 & 1 & .. & 0 & 0 \\ : & : & : & : & : & : \\ 0 & 0 & 0 & .. & 1 & 0 \end{pmatrix}$$

$$\mathbf{U}_- = \sum_{j=1}^{N-1} |j-1\rangle\langle j| + |N-1\rangle\langle 0| = \begin{pmatrix} 0 & 1 & 0 & .. & 0 & 0 \\ 0 & 0 & 1 & .. & 0 & 0 \\ 0 & 0 & 0 & .. & 0 & 0 \\ : & : & : & : & : & : \\ 0 & 0 & 0 & .. & 0 & 1 \\ 1 & 0 & 0 & .. & 0 & 0 \end{pmatrix}$$

The operator $\mathbf{C}$ moves the particle to the right if the spin is up and moves the particle to left if the spin is down. It simulates a coin flip. This is why the procedure can be called a random walk. Because $\mathbf{U}_+|N-1\rangle = |0\rangle$ and $\mathbf{U}_-|0\rangle = |N-1\rangle$, the end of the $N$ state space is connected with the beginning of it, so our position space is equivalent to $N$ equidistant points on a circle.

We can think of operator $\mathbf{C}$ as defining a network of connections between states in the N-dimensional Hilbert space $\mathcal{H}_N$. The $\mathbf{C}$ operator entangles the spin with the particle position.

Combining the two operators

$$\mathbf{V} = \mathbf{C} * (\mathbf{H} \otimes \mathbf{I})$$

One can plot the probability of being in each state $|n\rangle$ at each iteration.

The resulting vector is has even/odd parity and spreads out ballistically (rather than diffusively) with more iterations. The shape of the probability distribution differs from that of a classical random walk.

Let's evaluate the first few states. Starting with $|\psi_0\rangle = |00\rangle$ we first apply the Hadamard to the spin state and we get

$$\mathbf{H} \otimes \mathbf{I}\,|\psi_0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$$

We now apply the controlled raising and lower operator, $\mathbf{C}$. $|00\rangle \to |01\rangle$ and $|10\rangle \to |1, N-1\rangle$.

$$\mathbf{V}\,|\psi_0\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |1, N-1\rangle)$$

Let's operate again with the Hadamard

$$(\mathbf{H} \otimes \mathbf{I})\mathbf{V}\,|\psi_0\rangle = \frac{1}{2}(|01\rangle + |11\rangle + |0, N-1\rangle - |1, N-1\rangle)$$

Now we operate again with $\mathbf{C}$.

$$\mathbf{V}^2 |\psi_0\rangle = \frac{1}{2}(|02\rangle + 2\,|00\rangle - |0, N-2\rangle)$$

Notice the even/odd parity developing! Apply the Hadamard again

$$(\mathbf{H} \otimes \mathbf{I})\mathbf{V}^2 |\psi_0\rangle = \frac{1}{\sqrt{8}}(|02\rangle + |12\rangle + 2\,|00\rangle + 2\,|10\rangle - |0, N-2\rangle - |1, N-2\rangle)$$

Apply $\mathbf{C}$ again,

$$\mathbf{V}^3 |\psi_0\rangle = \frac{1}{\sqrt{8}}(|03\rangle + |11\rangle + 2\,|01\rangle + 2\,|1, N-1\rangle - |0, N-1\rangle - |1, N-3\rangle)$$

And so on.

A quantum random walk might be used to find a short path on a complicated network. They also are interesting as a class of cellular automata. One can study the influence of imperfections and external perturbations on the behavior of a quantum random walk. While static spatial random changes of the coin may lead to Anderson localization temporal randomness in the coin operator can cause decoherence resulting in a transition to classical random walking behavior. Their transport and percolation behavior is dependent on the network or graph of the shift operator. Some times people approximate continuous (in time) Hamiltonian evolution with a series of discretely (in time) applied operators. Quantum random walks can be considered relevant for numerical techniques aiming to approximate continuous systems.