# Lecture notes on Quantum Computation[1]

*by Man-Hong Yung*

*February 19, 2020*

[1] Comments/suggestions/corrections can be sent to me directly.
Email: yung@sustech.edu.cn

"I think I can safely say that nobody understands quantum mechanics."

Richard Feynman, in The Character of Physical Law (1965)

## Contents

# Reference(s)


- Nielsen & Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition. (Cambridge University Press, 2011).
- Kaye, Laflamme, & Mosca, An introduction to quantum computing. (Oxford University Press, USA, 2007).
- Kitaev, Shen, and Vyalyi, Classical and quantum computation. Vol. 47. American Mathematical Society Providence, 2002.
- Lecture notes of Preskill (http://www.theory.caltech.edu/people/preskill/ph229/)
- Lecture notes of Watrous (https://cs.uwaterloo.ca/∼watrous/LectureNotes.html)
- Lecture notes of Aaronson ( http://arxiv.org/abs/1607.05256)

**Part I**

# Fundamental concepts

# 1 Overview

## 1.1 Quantum bits

- The **bit** is the fundamental concept of classical computation and classical information.
- Quantum computation and quantum information are built upon an analogous concept, the quantum bit, or **qubit** for short.

  - What then is a qubit? Just as a classical bit has a state, either 0 or 1, a qubit also has a state. Two possible states for a qubit are the states

    $$|0\rangle \quad \text{and} \quad |1\rangle\,, \tag{1}$$

    in the **Dirac notation**.

- Examples of qubits:
  - two polarizations (or paths) of a photon;
  - two nuclear-spin states;
  - two electronic orbitals of an atom.

## 1.2 Single qubit: superposition and measurement

- It is also possible to form linear combinations of states, often called **superpositions**:

  $$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\ , \tag{2}$$

  where the numbers $\alpha$ and $\beta$ are **complex** numbers.

- Put another way, the state of a qubit is a **vector** in a two-dimensional complex vector space.

- The special states $|0\rangle$ and $|1\rangle$ are known as **computational basis states**, and form an orthonormal basis for this vector space.
- Therefore, a qubit can exist in a **continuum** of states between $|0\rangle$ and $|1\rangle$, **until it is observed**.

- When we measure a qubit we get either the result 0, with **probability**,

$$\Pr(0) = |\alpha|^2 \,, \tag{3}$$

or the result 1, with probability,

$$\Pr(1) = |\beta|^2 \,. \tag{4}$$

- Naturally, we have

$$\Pr(0) + \Pr(1) = |\alpha|^2 + \beta|^2 = 1 \,, \tag{5}$$

since the probabilities must sum to one.

## 1.3 Measurement with different bases

- Note that the states $|0\rangle$ and $|1\rangle$ represent just one of many possible choices of basis states for a qubit.
- There are infinitely many choices. For example, a possible choice is the set,

$$|+\rangle \equiv \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right), \quad |-\rangle \equiv \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \,. \tag{6}$$

- An arbitrary state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ can be re-expressed in terms of the states $|+\rangle$ and $|-\rangle$:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \frac{\alpha+\beta}{\sqrt{2}}\,|+\rangle + \frac{\alpha-\beta}{\sqrt{2}}\,|-\rangle \,. \tag{7}$$

- Naturally, measuring with respect to the $|+\rangle$, $|-\rangle$ basis results in the result '$+$' with probability,

$$\Pr\left(+\right) = |\alpha+\beta|^2/2 \,, \tag{8}$$

and the result '$-$' with probability,

$$\Pr\left(-\right) = |\alpha-\beta|^2/2 \,, \tag{9}$$

with corresponding post-measurement states $|+\rangle$ and $|-\rangle$, respectively.

- The same result can be generalized to any orthnormal set of basis vectors.

## 1.4 Single qubit: Bloch sphere

- Because $|\alpha|^2 + \beta|^2 = 1$, we may rewrite Equation (2) as[2]

$$|\psi\rangle = \cos\tfrac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\tfrac{\theta}{2}\,|1\rangle \ , \tag{10}$$

where the numbers $\theta$ and $\phi$ define a point on the unit three-dimensional sphere.

- This sphere is often called the **Bloch sphere**.

[2] Note that there is no observable consequence if we add a **global phase** to $|\psi\rangle$, which means that the state $e^{i\varphi}|\psi\rangle$ is physically the same as $|\psi\rangle$.

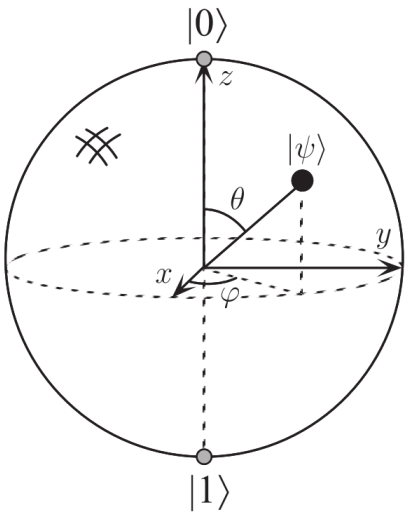



Figure 1: Bloch sphere representation of a qubit.

- Unfortunately, the bloch-sphere picture is only good for a single qubit; there is **no simple generalization** of the Bloch sphere known for multiple qubits.

- How much information is represented by a qubit? Paradoxically, there are an **infinite number** of points on the unit sphere, so that in principle one could store an entire text of Shakespeare in the infinite binary expansion of $\theta$.

- However, recall that (i) a measurement of a qubit will give only **1 bit of information**, i.e., either 0 or 1. (ii) measurement changes the state of a qubit, **collapsing** it from its

superposition of $|0\rangle$ and $|1\rangle$ to the specific state consistent with the measurement result.

- Therefore, a large number of **identically-prepared** qubits are needed to be measured, in order to determine $\alpha$ and $\beta$ for a qubit in the state given in Equation (2).

## 1.5 Multiple qubits

- Suppose we have two classical bits, then there would be **four possible states**, 00, 01, 10, and 11.
- However, a two-qubit system has **four computational basis** states:

$$|00\rangle, |01\rangle, |10\rangle, |11\rangle \ . \tag{11}$$

- A pair of qubits can also exist in superpositions of these four states; so the quantum state of two qubits involves a **complex coefficient** (sometimes called an amplitude) with each computational basis state, i.e.,

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle \ . \tag{12}$$

- Similar to the case for a single qubit, the measurement result, e.g. 00, occurs with probability $\Pr(00) = |\alpha|^2$, with the state of the qubits after the measurement being $|00\rangle$.
- The condition that probabilities sum to one is therefore expressed by the **normalization condition**:

$$|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1 \ . \tag{13}$$

- For a two qubit system, we could measure just a **subset** of the qubits, say the first qubit. For example, we get the result '0' with a probability

$$\Pr(0) = \Pr(00) + \Pr(01) = |\alpha|^2 + |\beta|^2 \ . \tag{14}$$

- In addition, the **post-measurement state** is given by[3]

$$|\psi_{\text{after}}\rangle = \frac{\alpha|00\rangle + \beta|01\rangle}{\sqrt{\Pr(0)}} = \frac{\alpha|00\rangle + \beta|01\rangle}{\sqrt{|\alpha|^2 + |\beta|^2}} \ . \tag{15}$$

[3] Note that we have **re-normalized** the quantum state by a factor $\sqrt{\Pr(0)} = \sqrt{|\alpha|^2 + |\beta|^2}$.

- An important two qubit state is the **Bell state** or **EPR pair**,
$$|\psi_{\text{Bell}}\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) , \tag{16}$$
which is the key ingredient in **quantum teleportation** and **super-dense** coding.

- For the Bell state, the measurement outcomes are **correlated**; a measurement of the second qubit always gives the same result as the measurement of the first qubit, i.e.,
$$\text{either} \quad |00\rangle , \quad \text{or} \quad |11\rangle \ . \tag{17}$$

- Bell proved an amazing result: the measurement correlations in the Bell state are **stronger than classical correlations**.

- More generally, we may consider a system of $n$ qubits. The computational basis states of this system are of the form $|x_1x_2...x_n\rangle$, i.e.,
$$|\psi_n\rangle = \sum_{\boldsymbol{x}\in\{0,1\}^n} c_{x_1x_2\cdots x_n}\,|x_1x_2\cdots x_n\rangle , \tag{18}$$
where $\boldsymbol{x} \equiv x_1x_2\cdots x_n$, and $\{0,1\}^n$ means the set of strings of length $n$ with each letter being either zero or one.

- So a quantum state of $n$ qubits is specified by $2^n$ amplitudes. For $n = 500$, this number is larger than the estimated number of atoms in the Universe! Trying to store all these complex numbers would not be possible on any conceivable classical computer.

# 2 Quantum Computation

- Analogous to the way a classical computer is built from an electrical circuit containing wires and logic gates, a quantum computer is built from a **quantum circuit** containing wires and elementary quantum gates to carry around and manipulate the quantum information.

## 2.1 Single-qubit gates

- Consider, for example, classical single bit logic gates. The only non-trivial member of this class is the NOT gate, which **exchanges 0 and 1**,

$$\text{NOT gate}: \quad 1 \to 0, \quad 0 \to 1 \,. \tag{19}$$

**LINEARITY OF QUANTUM GATES**

- In addition to flipping qubits, the quantum NOT gate, labelled as $X$, acts linearly, i.e.,

$$X(\alpha|0\rangle + \beta|1\rangle) = \alpha X|0\rangle + \beta X|1\rangle = \alpha|1\rangle + \beta|0\rangle \,. \tag{20}$$

- Linearity is an **intrinsic assumption** in quantum mechanics; nonlinear behavior can lead to apparent paradoxes such as time travel, faster-than-light communication, and violations of the second laws of thermodynamics.

- In fact, we can represent a **quantum state as a vector**, e.g.,

$$\alpha|0\rangle + \beta|1\rangle \quad \Leftrightarrow \quad \begin{bmatrix} \alpha \\ \beta \end{bmatrix} . \tag{21}$$

- In the matrix form, the NOT gate becomes,

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} . \tag{22}$$

- In other words, we can write

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} . \tag{23}$$

**UNITARITY OF QUANTUM GATES**

- In general, the appropriate condition on the matrix representing a quantum gate is that the matrix, $U$, describing the single qubit gate be **unitary**, i.e.,

$$U^\dagger U = I \,. \tag{24}$$

where $U^\dagger$ is the adjoint[4] of $U$, and $I$ is the two-by-two identity matrix.

[4] Obtained by transposing and then complex conjugating $U$.

- For example, for the NOT gate, it is easy to verify that $X^\dagger X = I$.
- Amazingly, this unitarity constraint is the **only constraint** on quantum gates; any unitary matrix specifies a valid quantum gate!
- Unlike the classical counterpart, there are many non-trivial single qubit gates. For example, the $Z$-gate,

$$Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} , \tag{25}$$

where

$$Z(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle - \beta|1\rangle \ . \tag{26}$$

**PAULI MATRICES**

- The operators $X$ and $Z$ are two members of the famous **Pauli matrices**. The remaining one is $Y$

$$Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} . \tag{27}$$

- For the Pauli matrices $\{X, Y, Z\}$, it is useful to remember the following identity:

$$X\,Y\,Z = i\,, \tag{28}$$

which implies that (using $X^2 = Y^2 = Z^2 = I$)

$$Y = iXZ\,. \tag{29}$$

- The Pauli matrices are often associated with the **rotation operators**, along respectively the $x$, $y$, and $z$ axis.

$$R_x(\theta) \equiv e^{-i\theta X/2},\ R_y(\theta) \equiv e^{-i\theta Y/2},\ R_z(\theta) \equiv e^{-i\theta Z/2}\,, \tag{30}$$

- By defining

$$c_\theta \equiv \cos\tfrac{\theta}{2} \quad \text{and} \quad s_\theta \equiv \sin\tfrac{\theta}{2}\,, \tag{31}$$

we have (double check the signs):

$$R_x(\theta) \equiv \begin{bmatrix} c_\theta & -is_\theta \\ -is_\theta & c_\theta \end{bmatrix}, \quad R_y(\theta) \equiv \begin{bmatrix} c_\theta & -s_\theta \\ s_\theta & c_\theta \end{bmatrix}\,, \tag{32}$$

and

$$R_z(\theta) \equiv \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix} . \tag{33}$$

- Note that **exercise**

$$XR_z(\theta)X = R_z(-\theta) = R_z(\theta)^{-1} = R_z(\theta)^\dagger , \tag{34}$$

and similarly for $R_y(\theta)$.

- More generally, If $\boldsymbol{n} = (n_x, n_y, n_z)$ is a real unit vector in three dimensions, then we generalize the previous definitions by defining a rotation by $\theta$ about the $\boldsymbol{n}$ axis by the equation **exercise**

$$R_{\boldsymbol{n}}(\theta) = e^{-i\theta \boldsymbol{n}\cdot\boldsymbol{\sigma}/2} = c_\theta I - is_\theta \left(n_x X + n_y Y + n_z Z\right) , \tag{35}$$

where $\boldsymbol{n}\cdot\boldsymbol{\sigma} \equiv n_x X + n_y Y + n_z Z$.

## HADAMARD, S, AND T GATES

- On the other hand, we also have the **Hadamard gate** $H$,

$$H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} , \tag{36}$$

where

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \& \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) . \tag{37}$$

- We note that, in addition to being unitary,

$$X^2 = Z^2 = Y^2 = H^2 = I , \tag{38}$$

- The Hadamard gate can be decomposed into a sum of the $X$ and $Z$ gates:

$$H = (X + Z)/\sqrt{2} . \tag{39}$$

- It would be useful to know the following identities **exercise**:

$$HXH = Z, \quad HZH = X, \quad HYH = -Y . \tag{40}$$

- The other two interesting single-qubit gates are the phase gate $S$, and the $T$ gate, defined by

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} , \tag{41}$$

- Of course, it is true that

$$S = T^2 \ . \tag{42}$$

**Z-Y DECOMPOSITION FOR A SINGLE QUBIT**

- Any arbitrary unitary matrix $U$ can be decomposed into the following form (with notation introduced in Equation (31)):

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta) \ , \tag{43}$$

or explicitly,

$$U = e^{i\alpha} \begin{bmatrix} e^{-i\beta/2} & 0 \\ 0 & e^{i\beta/2} \end{bmatrix} \begin{bmatrix} c_\gamma & -s_\gamma \\ s_\gamma & c_\gamma \end{bmatrix} \begin{bmatrix} e^{-i\delta/2} & 0 \\ 0 & e^{i\delta/2} \end{bmatrix} , \tag{44}$$

where $\alpha, \beta, \gamma$, and $\delta$ are real-valued.

*Proof.* Recall that a unitary transform preserves orthogonality, i.e., for any $U$, we define $|\psi\rangle \equiv U|0\rangle$ and $|\phi\rangle \equiv U|1\rangle$, such that $\langle\phi|\psi\rangle = \langle 1|U^\dagger U|0\rangle = \langle 1|0\rangle = 0$. In general, we can parametrize the states as follows: $|\psi\rangle = e^{i\phi} c_\gamma |0\rangle + e^{i\varphi} s_\gamma |1\rangle$, and $|\phi\rangle = e^{i\omega}\left(-e^{i\phi} s_\gamma |0\rangle + e^{i\varphi} c_\gamma |1\rangle\right)$. The explicit form of $U$ is given by the following:

$$U = \left(|\psi\rangle\langle 0| + |\phi\rangle\langle 1|\right) \ . \tag{45}$$

Now, by choosing an angle $\beta = (\varphi - \phi)$, we can see that $R_z(-\beta) U = (e^{-i\mu}|\psi_0\rangle\langle 0| + e^{i(\omega+\mu)}|\phi_0\rangle\langle 1|)$, where we defined $\mu \equiv (\phi + \varphi)/2$, and the real vectors, $|\psi_0\rangle = c_\gamma|0\rangle + s_\gamma|1\rangle$ and $|\phi_0\rangle = -s_\gamma|0\rangle + c_\gamma|1\rangle$. Next, we can construct a rotation $R_y(-\gamma)$ such that $R_y(-\gamma) R_z(-\beta) U = e^{-i\mu}|0\rangle\langle 0| + e^{i(\omega+\mu)}|1\rangle\langle 1|$. Finally, we can apply a $R_z(-\delta)$ to eliminate the extra phase, giving: $e^{-i\alpha} R_z(-\delta) R_y(-\gamma) R_z(-\beta) U = I$, which implies the result. □

- This decomposition can be used to give an **exact prescription** for performing an arbitrary single qubit quantum logic gate.

**COROLLARY OF Z-Y DECOMPOSITION**

- Another important result we need is as follows: given a unitary matrix $U$, there exist single-qubit rotational operators $A$, $B$, and $C$, of the form $R_y$ and $R_z$, such that

$$ABC = I, \quad \text{and} \quad U = e^{i\alpha} AXBXC \,. \tag{46}$$

*Proof.* From above, we have $B = A^{-1}C^{-1}$, and $AXBXC = R_z(\beta)\, R_y(\gamma)\, R_z(\delta)$ from Equation (43). Thus, we have,

$$AXBXC = AXA^{-1}C^{-1}XC = R_z(\beta)\, R_y(\gamma)\, R_z(\delta) \,. \tag{47}$$

Assume $C$ is of the form $R_z$, meaning that $XC^{-1}X = C$, and hence[5] $XA^{-1}C^{-1}X = (XA^{-1}X)(XC^{-1}X) = XA^{-1}XC$. Therefore, we have $AXA^{-1}XC^2 = R_z(\beta)\, R_y(\gamma)\, R_z(\delta)$. Now we set $C = R_z(\frac{\delta-\beta}{2})$, and get $AXA^{-1}X = R_z(\beta)\, R_y(\gamma)\, R_z(\beta)$. Now assume $A$ is of the form $R_zR_y$, meaning that $AXA^{-1}X = (R_zR_y)X(R_y^{-1}R_z^{-1})X = R_zR_y^2R_z$. Therefore, we have $A = R_z(\beta)\, R_y(\gamma/2)$. □

[5] If $A = UBU^\dagger$ is true, then $A^2 = UBU^\dagger UBU^\dagger = UB^2U^\dagger$.

## 2.2 CNOT gates

- In classical computing, any Boolean function can be computed from the composition of NAND gates alone, which is thus known as a **universal gate**.
- The quantum version is the controlled-NOT, or simply the CNOT gate, which has two input qubits, known as the **control qubit** and the **target qubit**, respectively.

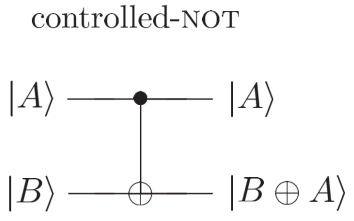



Figure 2: The CNOT gate.

- Alternatively, we can all represent the CNOT gate as shown in the following figure.

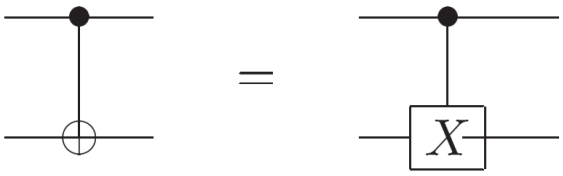



Figure 3: Alternative representatons of the CNOT gate.

- If the control qubit is set to 0, then the target qubit is left alone, i.e.,

$$|00\rangle \rightarrow |00\rangle \quad , \quad |01\rangle \rightarrow |01\rangle \ . \tag{48}$$

  But if the control qubit is set to 1, then the target qubit is flipped, i.e.,

$$|10\rangle \rightarrow |11\rangle \quad , \quad |11\rangle \rightarrow |10\rangle \ . \tag{49}$$

- Another way of describing the the action of the gate may be summarized as

$$|x, y\rangle \quad \rightarrow \quad |x, y \oplus x\rangle \ , \tag{50}$$

  where $\oplus$ is addition modulo two, which is exactly what the XOR gate does.

- The matrix representation of CNOT is,

$$U_{\text{CNOT}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \ . \tag{51}$$

- We note that in standard classical computation, which is irreversible, there is always a loss of information. For example, given the output $y \oplus x$ of the XOR gate, it is not possible to determine what the bit values of the input $x$ and $y$.

- However, unitary quantum gates are always invertible, since the inverse of a unitary matrix is also a unitary matrix, and thus a quantum gate can always be inverted by another quantum gate.

- Finally, we remark that **any multiple qubit logic gate may be composed from CNOT and single qubit gates**.

## 2.3 Quantum circuits

- A simple quantum circuit containing three quantum gates is shown in the figure below:

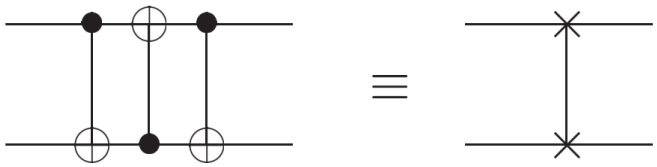


Figure 4: Circuit swapping two qubits, and an equivalent schematic symbol notation for this common and useful circuit.

- The circuit is to be read from left-to-right. It describes the equivalence of three CNOT gates and a SWAP gate exercise.
- The wires in the quantum circuit are not physical wires; they may correspond instead to the passage of time, or perhaps to a physical particle such as a photon moving from one location to another through space.
- There are a few features allowed in classical circuits that are not usually present in quantum circuits.
  1. In general, quantum circuits are **acyclic**; there is no loop.
  2. Classical circuits allow wires to be 'joined' together, **an operation known as FANIN**, with the resulting single wire containing the bitwise OR of the inputs.
  3. The inverse operation, FANOUT, whereby several copies of a bit are produced is also not allowed in quantum circuits; quantum mechanics forbids the copying of a qubit, known as **no-cloning theorem**.

**CONTROLLED-U OPERATION**

- Suppose $U$ is any unitary matrix acting on some number $n$ of qubits, so $U$ can be regarded as a quantum gate on those qubits. Then we can define a controlled-$U$ gate which is a natural extension of the CNOT gate.
- Such a gate has a single control qubit, indicated by the line with the black dot, and $n$ target qubits, indicated by the boxed $U$.

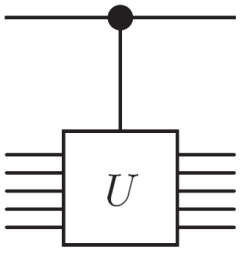



Figure 5: A controlled-$U$ gate.

- If the control qubit is set to 0 then nothing happens to the target qubits. If the control qubit is set to 1 then the gate $U$ is applied to the target qubits.
- As a special case, the matrix representation of the controlled-Z gate is of the following form

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} . \tag{52}$$

  and is symmetric in a quantum circuit:

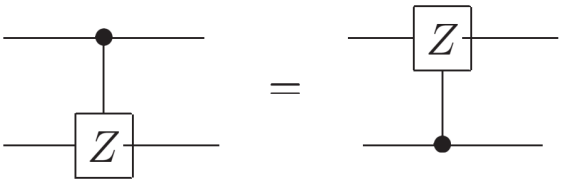



Figure 6: A controlled-$Z$ gate.

**IMPLEMENTING CONTROLLED-PHASE GATE**

- Before we consider how to implement the full controlled-$U$ gate, let us discuss how to take care of the phase factor $e^{i\alpha}$ in the decomposition of $U$ (see Equation (43)).
- In summary, we are to construct a gate to perform the following operation: (i) nothing happens if the first qubit is in $|0\rangle$, i.e.,

$$|00\rangle \to |00\rangle , \quad |01\rangle \to |01\rangle , \tag{53}$$

  but (ii) a phase factor is applied for $|1\rangle$, i.e.,

$$|10\rangle \to e^{i\alpha} |10\rangle , \quad |11\rangle \to e^{i\alpha} |11\rangle . \tag{54}$$

- Since the phase factor $e^{i\alpha}$ is not an operator, we can also interpret it as a local gate acting on the first qubit only.

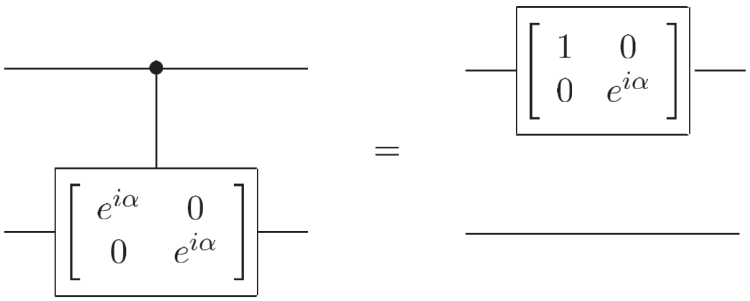



Figure 7: Controlled phase gate and an equivalent circuit.

## IMPLEMENTING CONTROLLED-U GATE

- Recall (see Equation (46)) that for any unitary gate $U$ acting on a single qubit, we can always decompose it into local gates $A$, $B$, and $C$, such that

$$ABC = I, \quad \text{and} \quad U = e^{i\alpha} AXBXC\,. \tag{55}$$

- If we replace the $X$ gates by controlled-NOT gates, then if the control qubit is in $|0\rangle$, then $ABC = I$ is applied (i.e. no change is made), but if the control qubit is in $|1\rangle$, then $AXBXC$ is applied. This is the basic idea of implementing the controlled-$U$ gate.

- Together with the controlled phase gate, the full implementation of the controlled-$U$ gate is shown in the following figure.

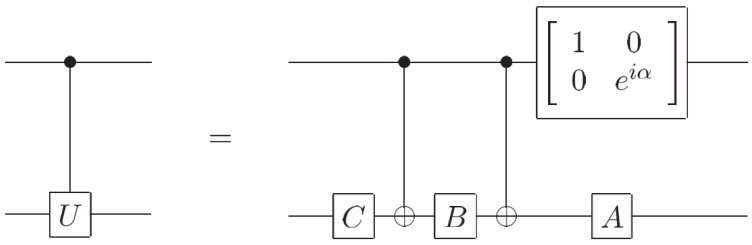



Figure 8: Circuit implementing the controlled-$U$ operation for a single qubit $U$. $\alpha$, $A$, $B$, and $C$ satisfy $ABC = I$ and $U = e^{i\alpha} AXBXC$.

## MULTIPLE CONTROLLED OPERATIONS

- The Toffoli gate flips the third qubit if and only if the first two qubits are both in the $|1\rangle$ state. This is an example of multiple controlled operation.

- More generally, we define the multiple controlled operation

as

$$C_n(U)\,|x_1x_2\cdots x_n\rangle\,|\psi\rangle = |x_1x_2\cdots x_n\rangle\,U^{x_1x_2\cdots x_n}\,|\psi\rangle\ , \quad (56)$$

which applies the operation $U$, if and only if the values of the first $n$ qubits are all 1, i.e., the product,

$$x_1x_2\cdots x_n = 1\ . \quad (57)$$

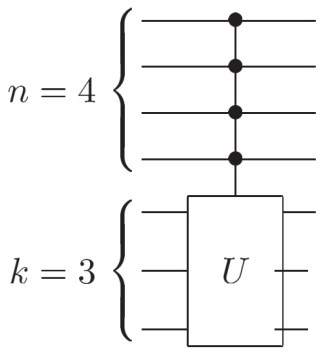



Figure 9: Sample circuit representation for the $C_n(U)$ operation, where $U$ is a unitary operator on $k$ qubits.

- As an other example, for a given $U$, suppose we can achieve the following decomposition:

$$U = V^2\ , \quad (58)$$

then the $C_2(U)$ can be implemented with a gate sequence described by the following quantum circuit.

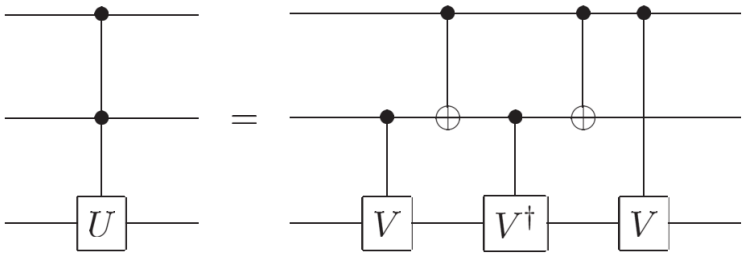



Figure 10: Circuit for the $C_2(U)$ gate. $V$ is any unitary operator satisfying $V^2 = U$. The special case $V \equiv (1-i)(I+iX)/2$ corresponds to the Toffoli gate.

- To understand this decomposition, let us eliminate the first and the last controlled-$V$ by applying a pair of controlled-$V^\dagger$ to both sides of the equation in the figure.
- The resulting gate performs the following operation:

$$|00\rangle \to I, \quad |11\rangle \to I\ , \quad (59)$$

and

$$|10\rangle \to V^\dagger, \quad |01\rangle \to V^\dagger\ , \quad (60)$$

which corresponds to exactly the three middle gates on the right hand side of the figure.

- Specifically, for the **Toffoli gate**, where $U = X$, we make

$$V \equiv (1-i)(I+iX)/2\,, \tag{61}$$

so that $V^2 = X$.

- Finally, we are ready to discuss how a general $C_n(U)$ gate can be implemented, which requires the help of $(n-1)$ ancilla qubits, as shown in the following figure.

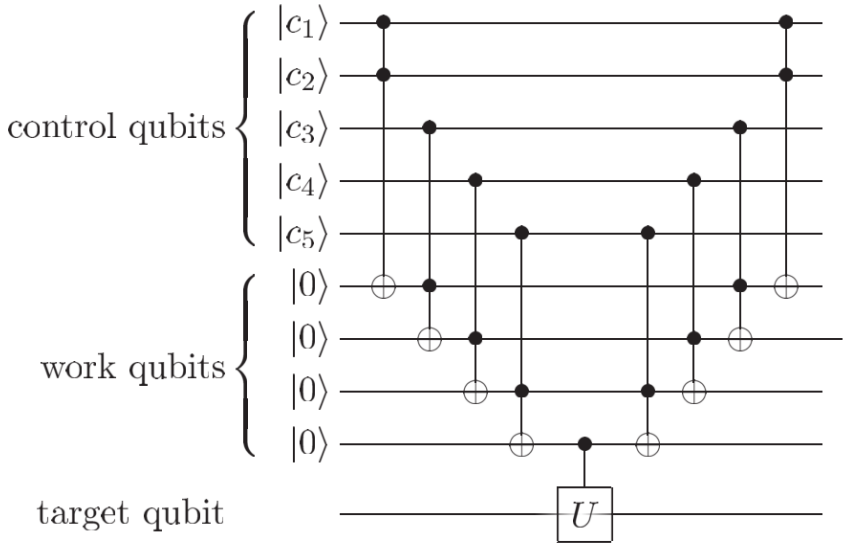



Figure 11: Network implementing the $C_n(U)$ operation, for the case $n = 5$.

- For a given configuration, $|c_1, c_2, ..., c_n\rangle$, the Toffoli gate in the circuits records the value of the product $c_1 \cdot c_2$ into the first ancilla qubit, i.e.,

$$|0\rangle \to |c_1 \cdot c_2\rangle\ . \tag{62}$$

- Similarly, the next Toffoli gate records the product $c_1 \cdot c_2 \cdot c_3$ into the second ancilla qubit, and so on.

- At the end, the value of the total product, $c_1 \cdot c_2 \cdots c_n$ is recorded at the last ancilla qubit, which is 1 iff all $c_k = 1$.

- In other words, the controlled-$U$ gate is applied if and only if all $c_k = 1$.

- The same set of gates is then applied in the reverse sequence to reset all the ancilla qubits to $|0\rangle$.

- Overall, the net effect is exactly the same as the action of the multi-controlled-$U$ gate.

**ADDITIONAL NOTATIONS IN QUANTUM CIRCUITS**

- Suppose we wish to implement a two qubit gate in which the second ('target') qubit is flipped, conditional on the first ('control') qubit being set to zero.
- Generically, we shall use the open circle notation to indicate conditioning on the qubit being set to zero, while a closed circle indicates conditioning on the qubit being set to one.
- A more elaborate example of this convention, involving three control qubits, is illustrated in the following figure:

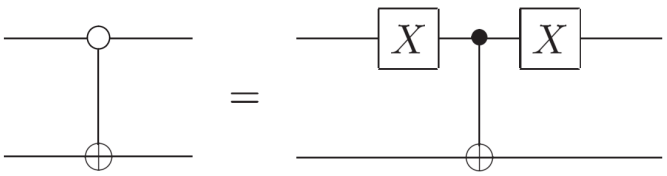



Figure 12: Controlled operation with a NOT gate being performed on the second qubit, conditional on the first qubit being set to zero.

- Another convention which is sometimes useful is to allow controlled-multiple targets, as shown in the following figure:

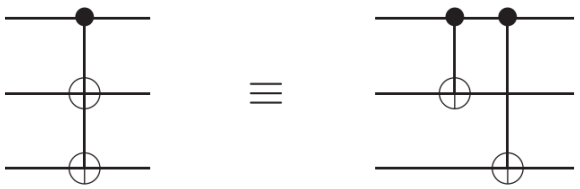


Figure 13: Controlled-NOT gate with multiple targets.

## 2.4 Creation of Bell states

- In fact, there are four Bell states (or known as EPR states or EPR pairs),

$$|B_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \quad |B_{01}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \tag{63}$$

and

$$|B_{10}\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}, \quad |B_{11}\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}} . \tag{64}$$

- For example, starting from $|00\rangle$, $|B_{00}\rangle$ can be generated by a Hadamard gate followed by a CNOT gate,

$$U_{\text{CNOT}} H |00\rangle = U_{\text{CNOT}}(|0\rangle + |1\rangle)|0\rangle / \sqrt{2} = |B_{00}\rangle . \tag{65}$$

- In a similar way, other Bell states are generated by different initial states with the same quantum circuit.

| In | Out |
|---|---|
| $\|00\rangle$ | $(\|00\rangle + \|11\rangle)/\sqrt{2} \equiv \|\beta_{00}\rangle$ |
| $\|01\rangle$ | $(\|01\rangle + \|10\rangle)/\sqrt{2} \equiv \|\beta_{01}\rangle$ |
| $\|10\rangle$ | $(\|00\rangle - \|11\rangle)/\sqrt{2} \equiv \|\beta_{10}\rangle$ |
| $\|11\rangle$ | $(\|01\rangle - \|10\rangle)/\sqrt{2} \equiv \|\beta_{11}\rangle$ |

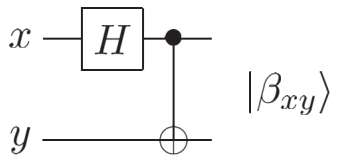



Figure 14: Quantum circuit to create Bell states, and its input-ouput quantum 'truth table'.

## 2.5 Quantum teleportation

- Quantum teleportation is a technique for moving quantum states around, even in the absence of a quantum communications channel linking the sender of the quantum state to the recipient.
- First, Alice has a qubit in a quantum state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ unknown to her. She wants to send the state to Bob who is located at a far distance from Alice.
- Alice is not allowed to send the qubit directly to Bob. Fortunately, Alice and Bob are shared with a EPR pair $|B_{00}\rangle$.

**Step 0:** The initial state of the total system is therefore,

$$|\psi_0\rangle = |\psi\rangle\,|B_{00}\rangle = \frac{1}{\sqrt{2}}\left(\alpha\,|0\rangle + \beta\,|1\rangle\right)\left(|00\rangle + |11\rangle\right)\ , \tag{66}$$

where the first two qubits belong to Alice and the last qubit belongs to Bob.

**Step 1:** First, Alice applies a CNOT gate to her qubits. The state becomes

$$|\psi_1\rangle = \frac{\alpha}{\sqrt{2}}\,|0\rangle\left(|00\rangle + |11\rangle\right) + \frac{\beta}{\sqrt{2}}\,|1\rangle\left(|10\rangle + |01\rangle\right)\ . \tag{67}$$

**Step 2:** Alice then applies a Hadamard gate to the first qubit, which yields the following state **exercise**:

$$|\psi_2\rangle \propto \left(|00\rangle\,|\psi\rangle + |01\rangle\,X\,|\psi\rangle + |10\rangle\,Z\,|\psi\rangle + |11\rangle\,XZ\,|\psi\rangle\right)\ . \tag{68}$$

**Step 3:** Alice perform a measurement on both of her qubits, and tell Bob the measurement result. For example, if $|01\rangle$ is obtained, then

$$|\psi_3\rangle = |01\rangle\,X\,|\psi\rangle\ . \tag{69}$$

**Step 4:** Since Bob now knows the measurement result, he can always apply a unitary correct the extra operation applied to the state. In the case above, $X(X\,|\psi\rangle) = X^2\,|\psi\rangle = |\psi\rangle$, which means that, finally, Bob can receive the unknown perfectly,

$$|\psi_4\rangle = |01\rangle\,|\psi\rangle\ . \tag{70}$$

- The whole process can be summarized by the following figure.
- Note that quantum teleportation **does not enable faster-than-light communication**, because to complete the teleportation Alice must transmit her measurement result to Bob over a classical communications channel.
- Furthermore, only one copy of the quantum state can exist

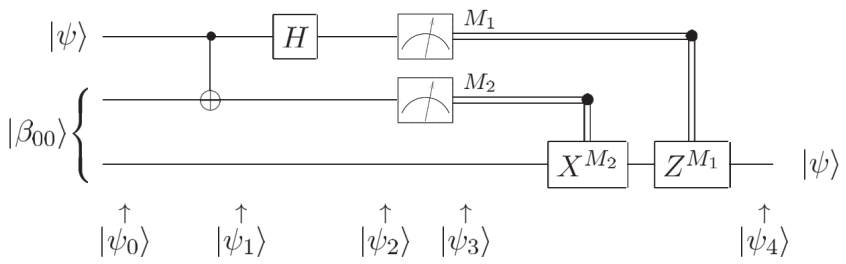



Figure 15: Quantum circuit for teleporting a qubit. The two top lines represent Alice's system, while the bottom line is Bob's system. The meters represent measurement, and the double lines coming out of them carry classical bits (recall that single lines denote qubits).

at any moment of time. Once the state $|\psi\rangle$ is transferred to Bob from Alice, Alice no longer has any information about $|\psi\rangle$.

## 2.6 Measurement

- In general, quantum measurement represents an interface between the quantum and classical worlds.
- A measurement operation may be represented by a 'meter' symbol, which converts a single-qubit state, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ into a probabilistic classical bit[6] $M$, which is 0 with probability $|\alpha|^2$, or 1 with probability $|\beta|^2$.

[6] It is distinguished from a qubit by drawing it as a double-line wire.

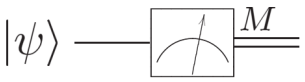



Figure 16: Quantum circuit symbol for measurement.

- Measurement is generally considered to be an irreversible operation, destroying quantum information and replacing it with classical information, except for carefully designed cases.
- In order for a measurement to be reversible, it must reveal no information about the quantum state being measured! For example, quantum teleportation and quantum error correction.

**PRINCIPLE OF DEFERRED MEASUREMENT**

- Often, quantum measurements are performed as an intermediate step in a quantum circuit, and the measurement results are used to conditionally control subsequent quantum gates.

- However, such measurements can always be moved to the end of the circuit. This is known as the **principle of deferred measurement**.
- For example, in the quantum circuit of quantum teleportation, all the classical conditional operations can be replaced by the corresponding quantum conditional operations.

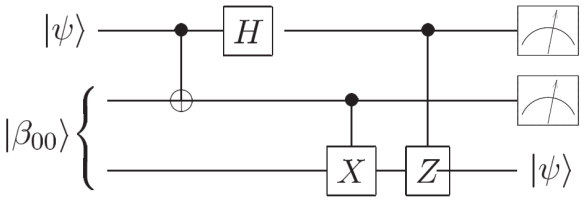



Figure 17: Quantum teleportation circuit in which measurements are done at the end, instead of in the middle of the circuit.

**PRINCIPLE OF IMPLICIT MEASUREMENT**

- Without loss of generality, any unterminated quantum wires (qubits which are not measured) at the end of a quantum circuit may be assumed to be measured. This is refereed to as **principle of implicit measurement**.
- For example, for a quantum circuit containing just two qubits, and only the first qubit is measured at the end of the circuit. The reduced density matrix of the first qubit is not affected by performing a measurement on the second.

## 2.7 No quantum-copying circuit

- Classical information can be copied using the CNOT gate, which takes in the bit to copy (in some unknown state $x \in \{0,1\}$) and a 'scratchpad' bit initialized to zero. The output is two bits, both of which are in the same state $x$, i.e.,

$$U_{\text{CNOT}} |x\rangle |0\rangle = |x\rangle |x\rangle \ . \tag{71}$$

- When applied to a qubit in a general quantum state $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$, we have

$$U_{\text{CNOT}} (\alpha |0\rangle + \beta |1\rangle) |0\rangle = \alpha |0\rangle |0\rangle + \beta |1\rangle |1\rangle \ , \tag{72}$$

which is not quite the same as what we want (two copies of $|\psi\rangle$), i.e.,

$$U_{\text{CNOT}} |\psi\rangle |0\rangle \neq |\psi\rangle |\psi\rangle \ , \tag{73}$$

unless either $\alpha = 0$ or $\beta = 0$.

- In general, it is impossible to make a copy of an unknown quantum state, known as the **no-cloning theorem**, which is one of the chief differences between quantum and classical information.

**Theorem 1** (No-cloning theorem)**.** *Suppose we are given an unknown n-qubit state $|\psi\rangle$. Let us try to duplicate it, through a unitary operator $U_{\text{clone}}$, into a register of qubit all initialized in the zero state $|0^n\rangle \equiv |000...0\rangle$, i.e.,*

$$U_{\text{clone}} |\psi\rangle |0^n\rangle = |\psi\rangle |\psi\rangle \ . \tag{74}$$

*The no-cloning theorem states that the operator $U_{\text{clone}}$ cannot exist.*

*Proof.* Assume such a operation $U_{\text{clone}}$ exists and that it works for all unknown states. Let us consider two general states, $|\psi\rangle$ and $|\phi\rangle$. We have,

$$U_{\text{clone}} |\phi\rangle |0^n\rangle = |\phi\rangle |\phi\rangle \ , \tag{75}$$

in addition to Equation (74). Now, by taking the inner product for the two resulting states, we have

$$\langle\phi| U_{\text{clone}}^{\dagger} U_{\text{clone}} |\psi\rangle = \langle\phi|\ \psi\rangle = \langle\phi|\ \psi\rangle^2 \ , \tag{76}$$

which cannot be true for all pairs of quantum states. For example, consider $|\psi\rangle = |0\rangle$ and $|\phi\rangle = |+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. □

## 2.8 Universal quantum gates (exact)

- In classical computing, we say that a set of gates is universal for computation, if the same set of gates can be used to compute any function. For example, the Toffoli gate is universal for classical computation.
- Consider a set of $m$ unitary gates,

$$G = \{U_1, U_2, ..., U_m\} \ , \tag{77}$$

where the gate $U_j$ acts on $k_j$ qubits, for $k_j \leq k$ (a constant) for each $j$.

**Definition:** The set of gates is said to be universal for quantum computation if any unitary operation may be **approximated to arbitrary accuracy** by a quantum circuit involving only those gates.

- More precisely, for any $n$-qubit gate $V \in U(2^n)$ and any given $\delta > 0$, there is a unitary $V_{\rm qc}$ achieved by a finite quantum circuit such that

$$\left\| V_{\rm qc} - e^{i\phi} V \right\| \leqslant \delta \ . \tag{78}$$

- We shall present the following facts:

  1. An arbitrary unitary operator may be expressed exactly as a product of unitary operators that each acts non-trivially only on a subspace spanned by two computational basis states.
  2. It then follows that an arbitrary unitary operator may be expressed exactly using single qubit gates and CNOT gates.
  3. Single qubit operation may be approximated to arbitrary accuracy using the Hadamard and $T$ gates.

- Therefore, the CNOT gate together with Hadamard and $T$ gates, are universal for quantum computation.

- In fact the CNOT gate is not special in this respect. Any "entangling" two-qubit gate, when combined with arbitrary single-qubit gates, is universal[7].

- In other words, every two-qubit unitary which is not local or locally equivalent to SWAP is entangling, and hence universal when combined with arbitrary single-qubit gates.

[7] We say a two-qubit gate is entangling if it maps some product state to a state which is not a product state.

## UNIVERSALITY OF $2 \times 2$ UNITARY GATES

**Definition of $2 \times 2$ gates:** For a fixed orthonormal basis, labelled by $\{|0\rangle, |1\rangle, |2\rangle, ..., |N-1\rangle\}$ in an $N$-dimensional space. We say a unitary transformation $U$ is a two-level unitary, or $2 \times 2$, if it acts nontrivially only in the two-dimensional subspace spanned by two basis elements $|i\rangle$ and $|j\rangle$.

- Let us see how to express any $U \in U(N)$ as a product of $2 \times 2$ unitaries.

**Step 1:** Consider the action of $U$ on the basis state $|0\rangle$. We can always write,

$$U|0\rangle = \sum_{i=0}^{N-1} a_i |i\rangle \ . \tag{79}$$

**Step 2:** The next step is easy; we can always construct another unitary matrix $W_0$ that can reproduce the same result as $U$, for the input $|0\rangle$,

$$W_0|0\rangle = \sum_{i=0}^{N-1} a_i |i\rangle \ , \tag{80}$$

where

$$W_0 = w_0^{(N-2)} \cdots w_0^{(1)} w_0^{(0)} \tag{81}$$

is a product of $N-1$ gates of $2 \times 2$ unitaries. These gates are defined such that $w_0^{(k)}$, acting non-trivially on the basis $\{|k\rangle, |k+1\rangle\}$, gives the correct coefficient $a_k$ associated with $|k\rangle$. More precisely,

$$w_0^{(0)}|0\rangle = a_0|0\rangle + b_0|1\rangle \ , \tag{82}$$

where $b_0$ can be determined by the normalization condition. Similarly,

$$w_0^{(1)}|1\rangle = \frac{a_1}{b_0}|1\rangle + \frac{b_1}{b_0}|2\rangle \ , \tag{83}$$

and

$$w_0^{(2)}|2\rangle = \frac{a_2}{b_1}|2\rangle + \frac{b_2}{b_1}|3\rangle \ , \tag{84}$$

and so on.

**Step 3:** Define another unitary $U_1 \equiv W_0^{-1} U$. Note that

$$U_1 |0\rangle = W_0^{-1} U |0\rangle = W_0^{-1} W_0 |0\rangle = |0\rangle \ , \qquad (85)$$

which means that $U_1$ acts nontrivially only in the $(N-1)$-dimensional span of $\{|1\rangle, |2\rangle, ..., |N-1\rangle\}$.

**Step 4:** Then, we repeat the same argument above by defining a product of $2 \times 2$ unitaries $W_1 = w_1^{(N-1)} \cdots w_1^{(1)} w_1^{(0)}$, such that $W_1 |0\rangle = |0\rangle$ and $W_1 |1\rangle = U_1 |1\rangle$.

**Step 5:** Now, we define another unitary $U_2 \equiv W_1^{-1} U_1 = W_1^{-1} W_0^{-1} U$, which is diagonal in the basis of $|0\rangle$ and $|1\rangle$.

**Step 6:** If we keep doing it this way, we will be able to construct the following identity: $W_{N-2}^{-1} W_{N-3}^{-1} ... W_1^{-1} W_0^{-1} U = I$, or equivalently,

$$U = W_0 W_1 ... W_{N-3} W_{N-2} \ , \qquad (86)$$

which consists of a total of $(N-1) + (N-2) + ... + 2 + 1 = N(N-1)/2$ of unitaries in the $2 \times 2$ form.

- The remaining task is to show that $2 \times 2$ unitaries can be constructed by two-qubit unitaries.
- We shall show that single-qubit and CNOT gates are sufficient to reproduce any $2 \times 2$ unitary gate and are therefore universal for quantum computation.

## GRAY CODES

- To do that, we need to first introduce *Gray codes*.

> **Definition:** A Gray code connecting, two binary numbers $s$ and $t$, is a sequence of binary numbers, starting with $s$ and concluding with $t$, such that adjacent members of the list differ in exactly one bit.

- For example, if $s = 101001$ and $t = 110011$, then the Gray code is as follows:

$$\begin{aligned} g_1 &\equiv 1\,0\,1\,0\,0\,1 \ &= s \\ g_2 &\equiv 1\,0\,1\,0\,1\,1 \\ g_3 &\equiv 1\,0\,0\,0\,1\,1 \\ g_4 &\equiv 1\,1\,0\,0\,1\,1 \ &= t \end{aligned} \qquad (87)$$

- Note that permutations between these Gray code can be achieved with higher-order Toffoli gates, denoted by $\Lambda^k(X)$, which flips the target bit iff all $k$ controlling bits are equal to 1.
- For example, to swap between $g_1$ and $g_2$, we can apply NOT gates[8] for the second and fourth qubit first, then apply $\Lambda^4(X)$, which applies to the fifth qubit iff all other qubits are in the $|1\rangle$ state.
- Recall (from page 16) that any higher-order Toffoli gate can be implemented by the standard 3-bit Toffoli gates and CNOT gate.
- Moreover, a 3-bit Toffoli gate can be decomposed by CNOT, Hadamard, and $T$ gates. See the following figure.

[8] Note that NOT gate can be readily realized by including an ancilla qubit initialized as $|1\rangle$.

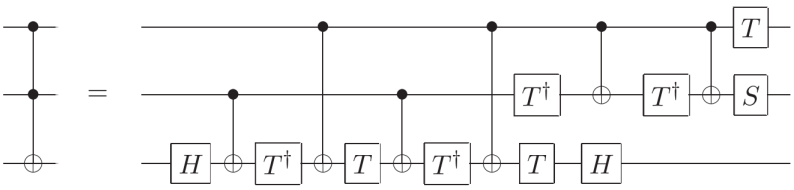



Figure 18: Decomposition of a Toffoli gate by CNOT, Hadamard, and $T$ gates

- Therefore, swapping any of the strings in the Gray code can be achieved by using only CNOT and single qubit gates.

## EXACT UNIVERSALITY OF SINGLE-QUBIT AND CNOT GATES

- Now, we are interested in a $2 \times 2$ unitary $V$ acting between $|s\rangle$ and $|t\rangle$, i.e.,

$$\begin{aligned} V|s\rangle &= \alpha|s\rangle + \beta|t\rangle \\ V|t\rangle &= \beta^*|s\rangle - \alpha^*|t\rangle \end{aligned} \tag{88}$$

- We can always first perform some permutation operations (labelled by $\Sigma$) following the gray code (with many control-control-control... NOT gates), such that[9]

$$\Sigma: \begin{array}{l} |s\rangle \to |111..10\rangle \\ |t\rangle \to |111..11\rangle \end{array} \tag{89}$$

- Consequently, the two-level unitary is now like a controlled-rotation $\Lambda^{n-1}(V)$ that applies to the last qubit, which can

[9] Recall that the SWAP gate can be realized by three CNOT gates (see the figure in page 12).

be achieved by CNOTs and single-qubit rotations through the relation (see equation (46) in page 10):

$$V = e^{i\alpha} AXBXC \,, \tag{90}$$

for a set of single-qubit gates $A$, $B$, and $C$, where $ABC = I$.

- As a result, we have proved that any $2 \times 2$ unitary can be implemented by CNOT and single qubit gates only, which implies that this set of gates are universal for quantum computation.

**Example 1**

- Suppose we wish to implement the following $2 \times 2$ unitary transformation[10]

$$U = \begin{bmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & c \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 & d \end{bmatrix} , \tag{91}$$

  which acts non-trivially only on the states $|000\rangle$ and $|111\rangle$.

- We write a Gray code connecting 000 and 111:

| $A$ | $B$ | $C$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |

(92)

- The following quantum circuit works:

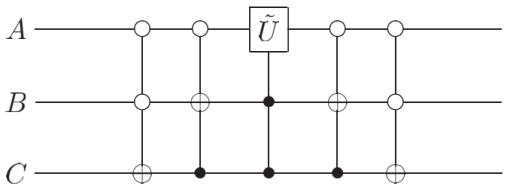



Figure 19: Quantum circuit implementing the two-level unitary operation.

- The first two gates shuffle the states so that $|000\rangle$ gets swapped with $|011\rangle$.
- Next, the operation

$$\tilde{U} \equiv \begin{bmatrix} a & c \\ b & d \end{bmatrix} \tag{93}$$

  is applied to the first qubit of the states $|011\rangle$ and $|111\rangle$, conditional on the second and third qubits being in the state $|11\rangle$.
- Finally, we unshuffle the states, ensuring that $|011\rangle$ gets swapped back with the state $|000\rangle$. (This step is optional.)

[10] This example is taken from Nielsen and Chuang.

- Obviously, the construction we have described does not provide terribly efficient quantum circuits! Furthermore, it is not fault-tolerant.

- However, the construction is close to optimal in the sense that there are unitary operations that require an exponential number of gates to implement.

## 2.9 Universal quantum gates (approximate)

- We now focus on a discrete set of gates which can be used to perform universal quantum computation, and importantly be implemented in an error-resistant fashion, using quantum error-correcting codes.
- Obviously, a discrete set of gates cannot be used to implement an arbitrary unitary operation exactly, since the set of unitary operations is continuous.

**ERRORS IN GATE APPROXIMATION**

**Definition of error:** Suppose $U$ and $V$ are two unitary operators on the same state space. $U$ is the target unitary operator that we wish to implement, and $V$ is the unitary operator that is actually implemented in practice. The error is defined as

$$\mathcal{E}(U,V) \equiv \max_{|\psi\rangle=1} \|(U-V)|\psi\rangle\| \ , \tag{94}$$

where the maximization is over all normalized quantum states $|\psi\rangle$.

  - The norm is called Euclidean norm, and is defined for any ket $|a\rangle$ by,

$$\| \, |a\rangle \, \| \equiv \sqrt{\langle a \, | a \rangle} \ . \tag{95}$$

- This measure of error has the interpretation that if $\mathcal{E}(U,V)$ is small, then any measurement performed on the state $V|\psi\rangle$ will give approximately the same measurement statistics as a measurement of $U|\psi\rangle$, for any initial state $|\psi\rangle$.

**Theorem 2.** *More precisely, if $M$ is a POVM element in an arbitrary POVM, $P_U = \langle\psi| U^\dagger M U |\psi\rangle$ (or $P_V = \langle\psi| V^\dagger M V |\psi\rangle$) is the probability of obtaining this outcome if $U$ (or $V$ ) were performed with a starting state $|\psi\rangle$, then*

$$|P_U - P_V| \leqslant 2\mathcal{E}(U, V) \ . \tag{96}$$

*Proof.* Define a unnormalized vector $|\Delta\rangle \equiv (U - V)|\psi\rangle$. Then, we have $|P_U - P_V| = \left|\langle\psi| U^\dagger M |\Delta\rangle + \langle\Delta| MV |\psi\rangle\right|$, which means that

$$|P_U - P_V| \leqslant |\langle\psi| U^\dagger M |\Delta\rangle| + |\langle\Delta| MV |\psi\rangle| \tag{97}$$

$$\tag{98}$$

□

- Suppose we perform a sequence of $m$ unitary quantum gates, $V_1, V_2, ..., V_m$, aiming to approximate some other sequence of gates, $U_1, U_2, ..., U_m$.

  - The error caused by the entire sequence of imperfect gates is at most the sum of the errors in the individual gates, i.e.,

  $$\mathcal{E}(U_m U_{m-1} ... U_1, V_m V_{m-1} ... V_1) \leqslant \sum_{j=1}^{m} \mathcal{E}(U_j, V_j) \ . \tag{99}$$

  *Proof.* Consider the case of two gates where $m = 2$, the total error is given by $\mathcal{E}(U_2 U_1, V_2 V_1) = \|(U_2 U_1 - V_2 V_1)|\psi\rangle\|$, which means that

  $$\tag{100}$$

  The more general cases then follow by induction. □

- In order that the probabilities of different measurement outcomes obtained from the approximate circuit be within a tolerance $\Delta > 0$ of the correct probabilities, it suffices that

$$\mathcal{E}(U_j, V_j) \leq \Delta/(2m) \ . \tag{101}$$

## UNIVERSALITY OF HADAMARD $H$ AND $T$ GATES

- In the following, we are going to prove an important result that Hadamard $H$ and $T$ (or $\pi/8$) gates[11],

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \text{ and } \quad T = \begin{bmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{bmatrix}, \quad (102)$$

can be used to approximate any single qubit unitary operation to arbitrary accuracy.

[11] The $T$-gate is historically often been referred to as the $\pi/8$ gate.

- First, we have to generalize the Z-Y decomposition in equation (43): any single-qubit unitary $U$ can be decomposed into the following (up to a global phase) **exercise**:

$$U = e^{i\alpha} R_{\hat{n}}(\beta_1) R_{\hat{m}}(\gamma_1) R_{\hat{n}}(\beta_2) R_{\hat{m}}(\gamma_2) \cdots, \quad (103)$$

where the rotation operators $R$'s are defined in equation (35) in page 8, $\hat{m}$ and $\hat{n}$ are non-parallel real unit vectors in three dimensions.

- Now, we set $\boldsymbol{m} \equiv (\cos(\frac{\pi}{8}), -\sin(\frac{\pi}{8}), \cos(\frac{\pi}{8}))$, and $\boldsymbol{n} = (\cos(\frac{\pi}{8}), \sin(\frac{\pi}{8}), \cos(\frac{\pi}{8}))$, specifically.

- In other words, our goal is to show that we can approximate any $R_{\hat{n}}(\delta)$ or $R_{\hat{m}}(\gamma)$ with $T$ and $H$ only.

### UNIT ROTATIONAL OPERATOR

- To do this, define an angle $\theta_{\text{unit}}$ such that $\cos(\theta_{\text{unit}}/2) \equiv \cos^2(\pi/8)$, and a 'unit' rotation operator

$$R_{\hat{n}}(\theta_{\text{unit}}) \equiv \cos\left(\frac{\theta_{\text{unit}}}{2}\right) I - i \sin\left(\frac{\theta_{\text{unit}}}{2}\right) (\boldsymbol{n} \cdot \boldsymbol{\sigma}). \quad (104)$$

- Note that the angle $\theta_{\text{unit}}$ is an **irrational** multiple of $2\pi$. This important fact will not be proved here, due to limited space (and energy).

- Apart from an overall phase factor, the above rotation operator can be constructed exactly by $H$ and $T$ as,

$$R_{\hat{n}}(\theta_{\text{unit}}) \propto THTH. \quad (105)$$

*Proof.* The $T$ gate is equivalent to the $\pi/4$ gate (up to a global phase), i.e., $T \propto e^{-i(\pi/8)Z}$. The composite gate[12], $HTH$, is a rotation of an angle of $\pi/4$ along the $x$-axis, i.e.,

[12] Note that we have the well-known identity, $HZH = X$.

$HTH \propto e^{-i(\pi/8)X}$. Thus, we have $T \cdot HTH \propto \cos^2\left(\frac{\pi}{8}\right) I - i \sin\left(\frac{\pi}{8}\right)\left[\cos\left(\frac{\pi}{8}\right)(X+Z) + \sin\left(\frac{\pi}{8}\right) Y\right] = R_{\hat{n}}(\theta_{\text{unit}})$. □

- Now, we show that one can apply the unit rotational operator $R_{\hat{n}}(\theta_{\text{unit}})$ repeatedly to approximate the rotation $R_{\hat{n}}(\theta)$ for any value of the angle $\theta$, along the same axis.

- More precisely, for any $\varepsilon > 0$, there exists an integer $k$, such that

$$E\left(R_{\hat{n}}(\theta), R_{\hat{n}}(\theta_{\text{unit}})^k\right) < \varepsilon\,. \tag{106}$$

- To understand this result, we need to discuss the concept of **denseness on the circle**.

**DENSENESS ON THE CIRCLE**

- We first present the following facts:

**Fact 1:** If $\alpha \in [0,1)$ is a irrational number, then the points, $n\alpha \ (mod\ 1)$, are **all distinct**.

*Proof.* For integers $k$ and $n \neq m$, the condition: $n\alpha - m\alpha = k$, implies that $\alpha$ is a rational number, i.e., $\alpha = k/(n-m)$. □

**Fact 2:** Consider open intervals of width $\varepsilon$, centered on each of the $N$ points $n\alpha \ (mod\ 1)$ for $n = 1, 2, 3, ..., N$ on a unit circle. Then, for $N\varepsilon > 1$, **at least two of these intervals must intersect.**

*Proof.* If these intervals do not intersect, then it is necessary that $N\varepsilon \leq 1$. Note that $N\varepsilon = 1$ is the condition where the intervals uniformly fill up the unit circuit. □

- Combining these two facts, there exist distinct positive integers $n$ and $m$ such that $|n-m|\ \alpha \ (\text{mod } 1) < \varepsilon$.

- In other words, the positive integer $r = |n-m|$ satisfies

$$r\ \alpha \ (\text{mod } 1) < \varepsilon\,. \tag{107}$$

- Note that the positive integer multiples of $r\ \alpha \ (\text{mod } 1)$ are **equally spaced points** on the unit interval separated by less than $\varepsilon$.

- In other words, for sufficiently large $M$, the intervals of width $\varepsilon$ centered on the points, $kr\alpha \pmod 1$ for $k = 1, 2, ..., M$, fill the unit interval, so that adjacent members of the sequence are no more than $\varepsilon$ apart.

**PROVING UNIVERSALITY OF HADAMARD $H$ AND $T$ GATES**

- We have now seen that for a sufficiently large $k$, the angles, $\theta$ and $\theta^k_{\text{unit}} (\text{mod} 2\pi)$ for the rotation operator $R_{\hat{n}}$, can be made arbitrarily close.
- Therefore, the error bound in equation (106) can always be achieved **exercise**. (hint: first prove the following inequalities, $E\left(R_{\hat{n}}(\theta_1), R_{\hat{n}}(\theta_2)\right) \leqslant \left|e^{i\theta_1} - e^{i\theta_2}\right| \leqslant |\theta_1 - \theta_2|$)
- How about the other operator $R_{\hat{m}}(\gamma)$ in equation (103)? Actually, the argument above can be applied readily as

$$H\, R_{\hat{n}}(\gamma)\, H = R_{\hat{m}}(\gamma)\ , \tag{108}$$

  which is true for any value of $\gamma$.
- Let us suppose the decomposition of $U$ in Eq. (103) has a total number of $c$ terms.
- Consequenlty, for suitable positive integers $n_1$, $n_2$, $n_3$,... we have (using equation (99)),

$$E\left(U, R_{\hat{n}}(\theta)^{n_1} \cdot H R_{\hat{n}}(\theta)^{n_2} H \cdot R_{\hat{n}}(\theta)^{n_3} \cdots\right) < c\varepsilon\ , \tag{109}$$

  which means that given any single qubit unitary operator $U$ and any $\varepsilon > 0$ it is possible to approximate $U$ to within $c\varepsilon$, using a circuit composed of Hadamard gates and $T$ gates alone.
- In summary, given a quantum circuit, we can first approximate it with CNOT gates and the set of single qubit gates exactly. The latter can then be approximated with Hadamard and $T$ gates to any accuracy.

## 2.10 Solovay-Kitaev algorithm

- The above results concern only the "reachability" of arbitrary $n$-qubit unitaries; they say nothing about the circuit size needed for a good approximation.

- Fortunately, there exists an algorithm due to Solovay and Kitaev, which suggests that an arbitrary single qubit gate may be approximated to an accuracy $\varepsilon$ efficiently.

- More precisely, for a given unitary gate $U$ acting on a single qubit, accuracy $\varepsilon$, and a set of non-commuting gates $\{g_k\}$, there exists a gate sequence of length $l$ producing a $\tilde{U}_l$, such that

$$\left\| U - \tilde{U}_l \right\| \leqslant \varepsilon \ , \tag{110}$$

  where $\left\| U - \tilde{U}_l \right\| = \mathrm{tr}\sqrt{(U - \tilde{U}_l)^\dagger (U - \tilde{U}_l)}$ is the trace distance.

- Furthermore, the length of the sequence $l$ varies as

$$l = O\left(\log_c (1/\varepsilon)\right) \ , \tag{111}$$

  for some constant $c$, which means that the sequence is logarithmic in length.

**CONCEPT OF $\varepsilon$-NET**

**Definition:** A set of unitary gates $\mathcal{V}_\varepsilon$ forms an "$\varepsilon$-net" in $U(N)$ (the set of all $N$-dimensional unitary transformation operators), if every elements in $U(N)$ is no more than a distance $\varepsilon$ away from an element in the set.

- More precisely, if we are given an $\varepsilon$-net, then for a given unitary operator, $U$, there exists a unitary gate $V \in \mathcal{V}_\varepsilon$ in the set such that

$$\|U - V\| \leqslant \varepsilon, \quad \text{or} \quad \|UV^{-1} - I\| \leqslant \varepsilon \ . \tag{112}$$

- Additionally, we require the set of gates to be **closed under inverse**, which means that for every element $V \in \mathcal{V}_\varepsilon$, there exists an inverse $V^{-1} \in \mathcal{V}_\varepsilon$ in the set.

- The key idea of the Solovay-Kitaev algorithm is as follows: one can construct a unitary operator $W$ from a sequence of gates in $\mathcal{V}_\varepsilon$, such that the error becomes significantly smaller.

- Explicitly, we are going to show that

$$\|UV^{-1} - W\| = O(\varepsilon^{3/2}) \,. \tag{113}$$

- Note that this result implies the following: if we choose from a set of gates with an $O(\varepsilon)$ (systematic) error, one can approximate a given unitary to a much smaller error $O(\varepsilon^{3/2})$.

**Proof of equation (113).**

– First, we define a Hermitian operator $A$ such that

$$UV^{-1} \equiv e^{iA} \,, \tag{114}$$

where $\|A\| = O(\varepsilon)$, or we may simply write $A = O(\varepsilon)$.

– In principle, one can always find[13] a pair of matrices, $B$ and $C$, such that exercise

$$[B, C] = -iA \,, \tag{115}$$

and $\|B\| = \|C\| = O\left(\varepsilon^{1/2}\right)$.

[13] The solution is not unqiue.

– Now, consider the unitaries $e^{iB}$ and $e^{iC}$. Given the $\varepsilon$-net and from equation (112), we can approximate them with $e^{i\tilde{B}}$ and $e^{i\tilde{C}}$ in $\mathcal{V}_\varepsilon$, such that

$$\|e^{i(B-\tilde{B})} - I\| = O(\varepsilon) \,, \;\&\; \|e^{i(C-\tilde{C})} - I\| = O(\varepsilon) \,, \tag{116}$$

which implies that

$$\left\|B - \tilde{B}\right\| = O(\varepsilon) \,, \;\&\; \left\|C - \tilde{C}\right\| = O(\varepsilon) \,. \tag{117}$$

– The $W$ matrix is constructed by these unitaries as exercise

$$W \equiv e^{i\tilde{B}} e^{i\tilde{C}} e^{-i\tilde{B}} e^{-i\tilde{C}} = I - \left[\tilde{B}, \tilde{C}\right] + O(\varepsilon^{3/2}) \,. \tag{118}$$

– Note that from equation (115) and (117), the commutator $\left[\tilde{B}, \tilde{C}\right]$ is in fact related to $A$, up to a small correction, i.e., exercise

$$\left[\tilde{B}, \tilde{C}\right] = -iA + O(\varepsilon^{3/2}) \,, \tag{119}$$

which implies that

$$W = I + iA + O(\varepsilon^{3/2}) = e^{iA} + O(\varepsilon^{3/2}) \,. \tag{120}$$

□

## ITERATIVE APPLICATION OF $\varepsilon$-NET

- Now, the result obtained above can be applied iteratively to arrive at the result of Solovay and Kitaev.
- First, suppose we have found an $\varepsilon_0$-net $V_{\varepsilon_0}$, closed under inverse.
- Second, we also assume that each element in $V_{\varepsilon_0}$ can be constructed by a set of universal gates with no more than $L_0$ gates.
- From the result in equation (113), we can construct a more dense $\varepsilon_1$-net, where for some constant $c$,

$$\varepsilon_1 = c\varepsilon_0^{3/2} , \tag{121}$$

using at most $L_1 = 5L_0$ elementary gates.

- The same argument can be applied to any pair of $\varepsilon$-nets, which is written as follows,

$$c^2\varepsilon_k = (c^2\varepsilon_{k-1})^{3/2} , \tag{122}$$

which gives

$$c^2\varepsilon_k = (c^2\varepsilon_0)^{(3/2)^k} . \tag{123}$$

- At the $k$-level approximation, one needs a number of

$$L_k = 5^k L_0 \tag{124}$$

elementary gates.

- Fortunately, we do not need $k$ to be very large, as we can write $5 = (3/2)^{\log 5/\log(3/2)}$, which gives

$$\frac{L_k}{L_0} = \left(\frac{\log\left(1/c^2\varepsilon_k\right)}{\log\left(1/c^2\varepsilon_0\right)}\right)^{\log 5/\log(3/2)} . \tag{125}$$

- In other words, to achieve a given accuracy $\varepsilon$, we can choose a value of $k$ such that $\varepsilon_k \leqslant \varepsilon$, which requires a number of gates scaling as

$$L_k = O\left(\left[\log\left(1/\varepsilon_k\right)\right]^{3.97}\right) , \tag{126}$$

which is the Solovay-Kitaev approximation.

# 3 Elementary Quantum Algorithms

## 3.1 Classical computations on a quantum computer

- Quantum circuits cannot directly simulate the results of a standard classical circuit that is irreversible.
- However, any classical circuit can be replaced by an equivalent circuit containing only reversible elements, by making use of a reversible gate known as the **Toffoli gate**.

### TOFFOLI GATE AS A UNIVERSAL REVERSIBLE GATE

- The Toffoli gate has three input bits and three output bits. Two of the bits are control bits that are unaffected by the action of the Toffoli gate. The third bit is a target bit that is flipped if both control bits are set to 1, and otherwise is left alone.

| Inputs | | | Outputs | | |
|---|---|---|---|---|---|
| $a$ | $b$ | $c$ | $a'$ | $b'$ | $c'$ |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

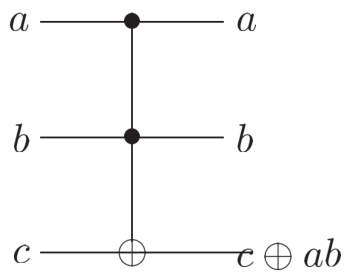



Figure 20: Truth table for the Toffoli gate, and its circuit representation.

- The Toffoli gate can be used to simulate the NAND gate, by initialize the target bit as 1.

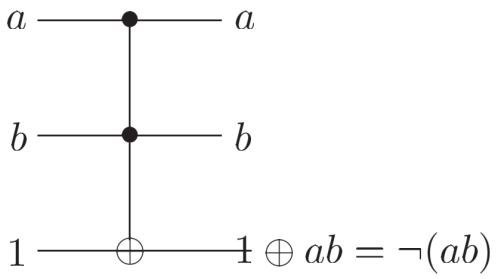



Figure 21: Classical circuit implementing a NAND gate using a Toffoli gate. The top two bits represent the input to the NAND gate, while the third bit is prepared in the standard state 1, sometimes known as an ancilla state. The is on the third bit output from the NAND gate is on the third bit.

- The Toffoli gate can also be implemented to simulate the FANOUT gate, by initializing the first and the third bits as 1 and 0, respectively. The value of the second bit will then be **duplicated to the third bit**.
- With these two operations it becomes possible to simulate all other elements in a classical circuit, and thus an arbitrary classical circuit can be simulated by an equivalent reversible circuit.
- Therefore, the quantum Toffoli gate can be used to simulate irreversible classical logic gates, which ensures that quantum computers are capable of performing any computation which a classical (deterministic) computer may do.

**SIMULATING NON-DETERMINISTIC CLASSICAL COMPUTATION**

- In principle, classical computer can be equipped with the ability to generate random bits to be used in the computation, e.g., in Monte Carlo methods.
- Quantum computer can generate random bits easily: simply prepare the state $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$, then perform measurement, which gives 50% chance of getting 0 or 1.
- This provides a quantum computer with the ability to efficiently simulate a non-deterministic classical computation.

## 3.2 Quantum parallelism

- Suppose $f(x) : \{0,1\} \to \{0,1\}$ is a function with a one-bit domain and range.

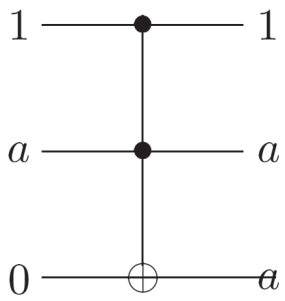



Figure 22: FANOUT with the Toffoli gate, with the second bit being the input to the FANOUT (and the other two bits standard ancilla states), and the output from FNAOUT appearing on the second and third bits.

- With an appropriate sequence of logic gates, labelled as $U_f$, it is possible to transform state $|x,y\rangle$, where $x,y \in \{0,1\}$, into

$$U_f |x,y\rangle = |x, y \oplus f(x)\rangle \ , \tag{127}$$

where $\oplus$ indicates addition modulo 2.

- If $y = 0$, then the final state of the second qubit is just the value $f(x)$.
- Now, if the initial state is prepared in a state with a superposition, e.g., $(|0\rangle + |1\rangle)/\sqrt{2}$, then the resulting state becomes

$$U_f \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) |0\rangle = \frac{1}{\sqrt{2}} (|0, f(0)\rangle + |1, f(1)\rangle) \ . \tag{128}$$

- It is almost as if we have evaluated $f(x)$ for two values of $x$ simultaneously[14], a feature known as **quantum parallelism**.

[14] In classical parallelism, multiple circuits are built to compute $f(x)$ simultaneously.

### HADAMARD TRANSFORM

- In order to extend the application of quantum parallelism for $n$ qubits, we need to prepare an equal superposition of all input states.
- For example, for two qubits where $n = 2$, we can apply two Hadamard gates, labelled as $H \otimes H = H^{\otimes 2}$, to each qubit initialized in the $|0\rangle$ state, i.e.,

$$H \otimes H |00\rangle = \frac{1}{2} (|00\rangle + |01\rangle + |10\rangle + |11\rangle) \ , \tag{129}$$

which is known as the **Hadamard transform**.

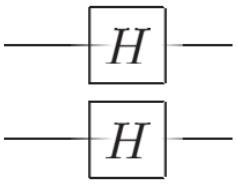



Figure 23: The Hadamard transform $H^{\otimes 2}$ on two qubits.

- More generally, the result of performing the Hadamard transform on $n$ qubits initialized in the $|0^{\otimes n}\rangle$ state is,

$$H^{\otimes n}\left|0^{\otimes n}\right\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle\ , \tag{130}$$

where $N = 2^n$ and $x$ labels the binary values of the bit string.

- Consequently, for $n$ qubits, all $2^n$ possible values can be computed with a single run of the circuit,

$$\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}U_f\,|x\rangle\,|0\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle\,|f\,(x)\rangle\ . \tag{131}$$

- However, this parallelism is not immediately useful; for a single qubit, measurement of the state gives only either $|0\rangle\,|f\,(0)\rangle$ or $|1\rangle\,|f\,(1)\rangle$. Similarly, in the general case, measurement of the state $\sum_x |x\rangle\,|f\,(x)\rangle$ would give only $f(x)$ for a single value of $x$.

## 3.3 Deutsch's algorithm

- Deutsch's algorithm combines quantum parallelism with a property of quantum mechanics known as *interference*.

**Step 1:** Let us first prepare the initial state $|\psi_0\rangle = |01\rangle$. Then, we apply Hadamard gates $H \otimes H$ for both qubits,

$$|\psi_1\rangle = \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) \equiv |+\rangle|-\rangle \ . \quad (132)$$

- Now, the non-trivial part is that, for $f(x) \in \{0,1\}$ and with $U_f : |x,y\rangle \to |x, y \oplus f(x)\rangle$, we have

$$U_f |x\rangle (|0\rangle - |1\rangle) = (-1)^{f(x)} |x\rangle (|0\rangle - |1\rangle) \ , \quad (133)$$

which comes from the fact that

$$|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle = \begin{cases} |0\rangle - |1\rangle & f(x) = 0 \\ -(|0\rangle - |1\rangle) & f(x) = 1 \end{cases} . \quad (134)$$

**Step 2:** Therefore, when applied to the full quantum state $|\psi_1\rangle$, we have

$$|\psi_2\rangle = U_f |\psi_1\rangle = \frac{1}{\sqrt{2}} \left( (-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle \right) |-\rangle \ . \quad (135)$$

**Step 3:** Finally, a Hadamard gate is applied to the first qubit, we have $|\psi_3\rangle = H \otimes I |\psi_2\rangle$ with

$$|\psi_3\rangle = s_+ |0\rangle |-\rangle + s_- |1\rangle |-\rangle \ , \quad (136)$$

where $s_\pm \equiv [(-1)^{f(0)} \pm (-1)^{f(1)}]/2$.

- Note that either (i) $f(0) = f(1)$, or (ii) $f(0) \neq f(1)$ is true. A measurement on the first qubit can unambiguously reveal the two cases; $|0\rangle$ implies case (i) and $|1\rangle$ implies the case (ii).

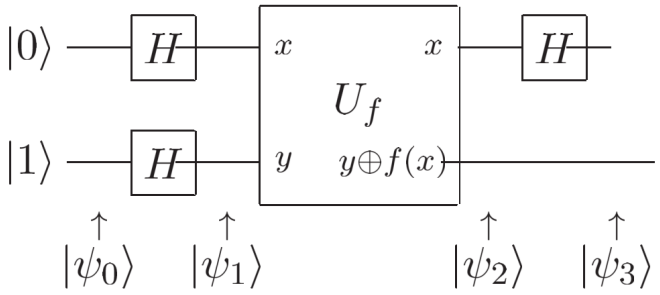



Figure 24: Quantum circuit implementing Deutsch's algorithm.

- This is very interesting indeed: the quantum circuit has given us the ability to determine a global property of $f(x)$, namely $f(0) \oplus f(1)$, using only one evaluation of $f(x)$!
- This is faster than is possible with a classical apparatus, which would require at least two evalutions.
- The essence of the design of many quantum algorithms is that a clever choice of function and final transformation allows efficient determination of useful global information about the function, information which cannot be attained quickly on a classical computer.

## 3.4 Deutsch-Jozsa algorithm

- Let us consider the following **Deutsch's problem:**

  **DEUTSCH'S PROBLEM**

  - Alice, in Amsterdam, selects a number $x$ from 0 to $2^n - 1$, and mails it in a letter to Bob, in Boston.
  - Bob calculates some function $f(x)$ and replies with the result, which is either 0 or 1.
  - Now, Bob has promised to use a function $f$ which is of one of two kinds; either (i) $f(x)$ is constant[15] for all values of $x$, or else (ii) $f(x)$ is balanced[16], that is, equal to 1 for exactly half of all the possible $x$, and 0 for the other half.
  - Alice's goal is to determine with certainty whether Bob has chosen a constant or a balanced function.

[15] For example, $f(00) = f(11) = f(01) = f(10) = 0$ or $f(00) = f(11) = f(01) = f(10) = 1$ for two bits.

[16] For example, $f(00) = f(11) = 0$ and $f(01) = f(10) = 1$.

- In the classical case, Alice may only send Bob one value of $x$ in each letter.
- At worst, Alice will need to query Bob at least $2^n/2 + 1$ times, since she may receive $2^n/2$ zeros (0s) before finally getting a one (1), telling her that Bob's function is balanced.
- The best deterministic classical algorithm she can use therefore requires $2^n/2 + 1$ queries.
- If Bob and Alice were able to exchange qubits, instead of just classical bits, and if Bob agreed to calculate $f(x)$ using

a unitary transform $U_f$ , then Alice could achieve her goal in just one correspondence with Bob, using the Deutsch-Jozsa algorithm.

**DEUTSCH-JOZSA ALGORITHM**

**Step 0:** Let us consider the following initial state,

$$|\psi_0\rangle = \left|0^{\otimes n}\right\rangle |1\rangle \ . \tag{137}$$

**Step 1:** Applying $n+1$ Hadamard gates, we have

$$|\psi_1\rangle = H^{\otimes n+1} |\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{x\in\{0,1\}^n} |x\rangle |-\rangle \ . \tag{138}$$

**Step 2:** Next, the function $f$ is evaluated (by Bob) using $U_f : |x,y\rangle \to |x, y\oplus f(x)\rangle$,

$$|\psi_2\rangle = U_f |\psi_1\rangle = \frac{1}{\sqrt{N}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |x\rangle |-\rangle \ . \tag{139}$$

- Before we move on, let us look at the effect of applying $H^{\otimes n}$ to a computational basis $|x\rangle$[17],

$$H^{\otimes n} |x\rangle = \frac{1}{\sqrt{N}} \sum_{z_1,z_2...,z_n} (-1)^{x_1z_1+x_2z_2+...+x_nz_n} |z_1z_2\cdots z_n\rangle \ . \tag{140}$$

[17] For example, we can write $H^{\otimes n}|110...\rangle = H^{\otimes n} X_1 \otimes X_2 |0^{\otimes n}\rangle$. Together with, $HX = ZH$, we have $Z_1Z_2H^{\otimes n}|0^{\otimes n}\rangle = \sum_z Z_1Z_2|z\rangle$, where we can write $Z_1Z_2|z\rangle = (-1)^{x_1+x_2}|z\rangle$.

**Step 3:** After applying the $H^{\otimes n}$ gate to the first $n$ qubits (and ignoring the last qubit in the $|-\rangle$ state),

$$|\psi_3\rangle = H^{\otimes n} |\psi_2\rangle = \sum_{z_1,z_2...,z_n} s(z_1z_2\cdots z_n) |z_1z_2\cdots z_n\rangle \ , \tag{141}$$

where by defining the bitwise product, $x\cdot z \equiv x_1z_1 + x_2z_2 + ... + x_nz_n$, we have

$$s(z_1z_2\cdots z_n) \equiv \frac{1}{N}\sum_x (-1)^{x\cdot z+f(x)} \ . \tag{142}$$

- Let us focus on the value of $s(00\cdots0) = (1/N)\sum_x (-1)^{f(x)}$. (i) If $f(x)$ is constant, then $s(00\cdots0) = \pm1$, which implies that the resulting state is in $|00...0\rangle$ for sure. However, (ii) if

$f(x)$ is balanced, then $s\,(00\cdots0) = 0$ for sure, which means that when we perform a measurement, we can find at least one '1', instead of all zeros.

- Therefore, by checking if the final state is in $|00...0\rangle$, we can unambiguous check if the function is constant or balanced, with a single run of the quantum algorithm.
- There Deutsch-Jozsa algorithm demonstrates how quantum computer may be useful. However, there is no known applications.
- Furthermore, if Alice is allowed to use a probabilistic classical computer, then by asking Bob to evaluate $f(x)$ for a few randomly chosen $x$ she can very quickly determine with high probability whether $f$ is constant or balanced. This probabilistic scenario is perhaps more realistic than the deterministic scenario we have been considering

# 4 Quantum Simulation

- Simulation of quantum systems by classical computers is possible, but generally only very inefficiently.
- The dynamical behavior of many simple quantum systems is governed by Schrödinger's equation,

$$i\hbar\frac{d}{dt}\,|\psi(t)\rangle = H\,|\psi(t)\rangle\ , \tag{143}$$

  where we shall set $\hbar = 1$ for simplicity.
- The key challenge in simulating quantum systems is the exponential number of differential equations which must be solved.
- The quantum simulation problem is concerned with the solution of equation (143). For a time-independent $H$, it is just

$$|\psi\,(t)\rangle = e^{-iHt}\,|\psi\,(t=0)\rangle\ . \tag{144}$$

## 4.1 Error in short-time evolution

- In many scenarios, we would want to study the error of the time-evolutionary operator, resulting from a short-time evolution.

**Definition** Given any operator $X$, and its deviation $X + \Delta X$, let us adopt the following notation,

$$X + O(\varepsilon) \quad \Leftrightarrow \quad \|\Delta X\| = O(\varepsilon) \ , \tag{145}$$

to mean that the operator norm of the extra part is $O(\varepsilon)$.

- Now, we can write,

$$e^{iXt} = I + iXt + \sum_{m=2}^{\infty} \frac{(iXt)^m}{m!} \ . \tag{146}$$

- Note that there exist a triangle inequality for the operator norm **exercise**:

$$\|X + Y\| \leqslant \|X\| + \|Y\| \ , \tag{147}$$

which implies that the last term can be bounded by

$$\left\| \sum_{m=2}^{\infty} \frac{(iXt)^m}{m!} \right\| \leqslant (\|X\| t)^2 \sum_{m=0}^{\infty} \frac{(\|X\| t)^m}{(m+2)!} \ . \tag{148}$$

- Now, suppose the evolution time $t$ is short, in the sense that

$$\|X\| t \leqslant 1 \ . \tag{149}$$

- In this case, we can bound the last term by[18]

$$\sum_{m=0}^{\infty} \frac{(\|X\| t)^m}{(m+2)!} \leqslant \sum_{m=0}^{\infty} \frac{1}{(m+2)!} = e - 1 - 1 < 1 \ , \tag{150}$$

where we have set $x = 1$ in the expansion, $e^x = 1 + x + \sum_{m=1}^{\infty} 1/(m+2)!$.

- As a result, we can write

$$e^{iXt} = I + iXt + O(\|X\|^2 t^2) \ . \tag{151}$$

[18] This elegant observation is found by my undergraduate student Cheng bin.

## 4.2 k-Local Hamiltonians

- In many physical systems with $n$ spins/particles, the Hamiltonian can be written as a sum over many local interactions, i.e.,
$$H = \sum_{j=1}^{L} H_j \ , \tag{152}$$
where each local term $H_j$ acts on at most a constant $k$ number of systems (not required to be physically local).

- For Hamiltonian with terms containing exactly $k$ bodies each[19] **exercise**,
$$L \leqslant \left( \begin{array}{c} n \\ k \end{array} \right) = O(n^k) = \text{poly}\,(n) \ , \tag{153}$$
scales as a polynomial number of the input.

[19] You may want to show that $\left( \begin{array}{c} n \\ k \end{array} \right) = n^k \left(1 + O\left(1/n\right)\right)/k!$

- The important point is that although $e^{-iHt}$ is difficult to compute. However, $e^{-iH_j t}$ acts on a much smaller subsystem, and is straightforward to approximate using quantum circuits.

- More specifically, from the Solovay-Kitaev theorem (see equation (111)), each local evolution $e^{-iH_j t}$ can be approximated by a small number $O\left(\log^c\left(1/\varepsilon\right)\right)$ of gate from a set of universal gates within an $\varepsilon$ error.

**COMMUTING CASE**

- If all the terms in the Hamiltonian commute with one another, i.e., $\left[H_j, H_m\right] = 0 \ \forall \ j, m$ , then the total evolution is simply given by
$$e^{-iHt} = e^{-iH_1 t} e^{-iH_2 t} \cdots e^{-iH_L t} \ . \tag{154}$$

- To ensure the error in the total evolution is bounded by $\varepsilon$, each gate is required to have an error $\varepsilon/L$ (see equation (99)).

- From the Solovay-Kiteav theorem, each gate then needs a total of $O\left(\log^c\left(L/\varepsilon\right)\right)$ elementary gates, and the total number of gates required to simulate the full Hamiltonian dynamics is $O\left(L \log^c\left(L/\varepsilon\right)\right)$.

**NON-COMMUTING CASE: LIE-TROTTER PRODUCT FORMULA**

- In general, the local terms may not commute, i..e, $\left[H_j, H_m\right] \neq 0$, which means that

$$e^{-iHt} \neq e^{-iH_1 t} e^{-iH_2 t} \cdots e^{-iH_L t} \,. \tag{155}$$

- It is still possible to simulate the time-evolution operator if we perform a pieces-wise simulation, i.e.,

$$e^{-iHt} = \lim_{n\to\infty} \left( e^{-iH_1 t/n} e^{-iH_2 t/n} \cdots e^{-iH_L t/n} \right)^n \,. \tag{156}$$

- This is possible because of the *Lie-Trotter product formula*:

**Theorem 3 (Lie-Trotter product formula).** *For a pair of Hermitian operators $A$ and $B$, where $\|A\| \leqslant s$ and $\|B\| \leqslant s$ for some $s < 1$, we have*

$$e^{-iA} e^{-iB} = e^{-i(A+B)} + O(s^2) \,. \tag{157}$$

*Proof.* From equation (151), we can expand the factors $e^{-iA} e^{-iB}$ separately, i.e., $\left(I - iA + O(s^2)\right)\left(I - iB + O(s^2)\right)$, which gives

$$e^{-iA} e^{-iB} = I - i\left(A + B\right) + O(s^2) = e^{-i(A+B)} + O(s^2) \,. \tag{158}$$

□

- Now, for simplicity, let us set $t = 1$. If each local term $\|H_j\| < s$, then

$$\|H_1 + H_2 + ... + H_L\| < Ls \,. \tag{159}$$

- In order to apply the Lit-Trotter product formula, we require that, $s < 1/L$.

## 4.3 Applying the Lie-Trotter formula

- Consequently, the product,

$$\Pi_H \equiv e^{-iH_1} e^{-iH_2} \cdots e^{-iH_L} \,, \tag{160}$$

can be written as

$$\Pi_H = \left(e^{-i(H_1+H_2)} + O(s^2)\right) e^{-iH_3} \cdots e^{-iH_L} \,. \tag{161}$$

- Note that for any unitary transformation $U$, $\|AU\| = \|A\|$, which means that we can simplify

$$O(s^2)\ e^{-iH_3} \cdots e^{-iH_L} = O(s^2)\ . \tag{162}$$

- Consequently, we can group the first two terms as

$$\Pi_H = e^{-i(H_1+H_2)} e^{-iH_3} \cdots e^{-iH_L} + O(s^2)\ . \tag{163}$$

**GROUPING MULTIPLE TERMS**

- Note that from Eq. (151), we have

$$e^{-i(H_1+H_2)} = I - i(H_1 + H_2) + O(2^2 s^2)\ . \tag{164}$$

- Therefore, we can group the first three terms as

$$\Pi_H = e^{-i(H_1+H_2+H_3)} e^{-iH_4} \cdots e^{-iH_L} + O(2^2 s^2) + O(s^2)\ . \tag{165}$$

- If we keep doing this, we will have

$$\Pi_H = e^{-i(H_1+H_2+\ldots+H_L)} + O(L^3 s^2)\ , \tag{166}$$

where the correction term comes from the fact that

$$O(L^3 s^2) = O(s^2) + O(2^2 s^2) + O(3^2 s^2) + \ldots + O((L-1)^2 s^2)\ . \tag{167}$$

**ERROR IN LONG-TIME EVOLUTION**

- To generalize above discussion for arbitrarily-long time, we now write, $U(t) \equiv e^{i(H_1+H_2+\ldots+H_L)t}$, where

$$e^{i(H_1+H_2+\ldots+H_L)t} = [e^{i(H_1 t/N + H_2 t/N + \ldots + H_L t/N)}]^N\ , \tag{168}$$

which means that we divide the time interval into $n$ divisions.

- From equation (149), we need to choose the value of $N$ such that,

$$\frac{s\ t}{N} < 1\ , \tag{169}$$

where $\|H_j\| < s$ for all $j$'s.

- Suppose now we want to simulate the evolution $U$ subject to an error $O(\varepsilon)$, it means that we need to be able to simulate each each term, $e^{i(H_1 t/N + H_2 t/N + ... + H_L t/N)}$, up to an error $O(\varepsilon/N)$, (see equation (99)).
- From equation (166), we replace $s \to st/N$, which implies that

$$O\left(L^3\, s^2\, t^2/N^2\right) = O\left(\varepsilon/N\right)\ , \tag{170}$$

- Therefore, in order to obtain $U(t) + O(\varepsilon)$, the number of division $N$ is required to be

$$N = O\left(\frac{L^3\, s^2\, t^2}{\varepsilon}\right)\ . \tag{171}$$

- Since there are $L$ terms for each small-time interval $t/N$, the **circuit size** (i.e., $N_{\text{gate}}$ number of elementary gates) is given by $N_{\text{gate}} = NL$, which scales as $O\left(L^4\, s^2\, t^2/\varepsilon\right)$.
- Recall from equation (153) that for $k$-local Hamiltonians, $L = O(n^k)$; this means that the circuit size can be written as

$$N_{\text{gate}} = O\left(\frac{n^{4k}\, s^2\, t^2}{\varepsilon}\right)\ . \tag{172}$$

**SIMULATION WITH UNIVERSAL GATE SET**

- Suppose we still want to simulate the overall time evolution operator with an overall error $\varepsilon$, i.e., $U(t) + O(\varepsilon)$, from the Solovay-Kitaev theorem, each of the short-time evolution, $e^{iH_j t/N}$, should be simulated by $O\left(\log^c\left(N_{\text{gate}}/\varepsilon\right)\right)$ elementary gates.
- Therefore, the total number of elementary gates scales as $N_{\text{gate}} \log^c\left(N_{\text{gate}}/\varepsilon\right)$, with $N_{\text{gate}}$ given by equation (172).

## 4.4 Simulating multi-body interactions

- Natural physical interactions usually involve two bodies.
- In the context of quantum simulation, it is not difficult to simulate Hamiltonian with an interaction term involving multiple systems.

- Let us consider the following "Ising" Hamiltonian,

$$\mathcal{H} = Z_1 \otimes Z_2 \otimes ... \otimes Z_n \ , \tag{173}$$

  actin on a system of $n$ qubits.

- As an example, for $n = 3$, the time evolution $e^{-i\mathcal{H}\Delta t}$ can be simulated with the following circuit.

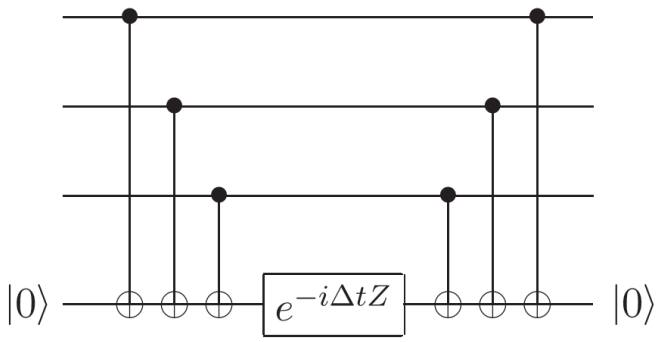



Figure 25: Quantum circuit for simulating the Hamiltonian $H = Z_1 \otimes Z_2 \otimes Z_3$ for $\Delta t$.

- If we look at the Hamiltonian carefully, we can see that the phase factor $e^{-i\Delta t}$ (or $e^{i\Delta t}$) is applied if the parity of the qubit is even (or odd).

- The parity information can be stored in the bit value of the ancilla qubit (at the bottom).

- Following the series of CNOT gates, a local phase factor $e^{-iZ\Delta t}$ gives exactly the same action as $e^{-i\mathcal{H}\Delta t}$.

- The last three CNOT is required to ensure the ancilla qubit returns to $|0\rangle$.

- Using the "Ising" Hamiltonian in equation (173), we can simulate a more general class of Hamiltonians, involving an arbitrary product of the Pauli matrices, $\{I, X, Y, Z\}$.

- For example, since $X = HZH$, we simulate the Hamiltonian $\mathcal{H} = X_1 \otimes Z_2 \otimes X_3$ by

$$e^{-i\mathcal{H}\Delta t} = (H \otimes I \otimes H)\, e^{-iZ_1 \otimes Z_2 \otimes Z_3 \Delta t}\, (H \otimes I \otimes H) \ . \tag{174}$$

## 4.5 Schrödinger's wave equation

- Simulating Schrödinger's wave equation can be achieved with a somewhat different method.

- As an example, consider the problem of a particle with a mass $m$, living in a 1D potential $V(x)$, where the Hamiltonian $H$ is given by

$$\mathcal{H} = \frac{P^2}{2m} + V(X) \ . \tag{175}$$

- Here $P$ is the momentum operator.

- To simulate this system, we consider a total length $2L$ and discretize the continuous space into discrete sites with a unit length $a$, i.e.,

$$|\psi\rangle = \sum_{k=-L/a}^{L/a} \phi_k |ka\rangle \ . \tag{176}$$

- There are totally $2L/a + 1$ number of sites. If we use $n$ qubits, we can represent $2^n$ points. Therefore, we require only $n \leqslant \log_2(2L/a + 1)$ number of qubits.

- Let us now write $\mathcal{H} = \mathcal{H}_0 + \mathcal{H}_1$, where

$$\mathcal{H}_0 = \frac{P^2}{2m}, \quad \& \quad \mathcal{H}_1 = V(X) \ . \tag{177}$$

- From the previous section, since $[P, V(x)] \neq 0$ in general, we need to simulate $e^{-i\mathcal{H}_0\Delta t}$ and $e^{-i\mathcal{H}_1\Delta t}$ individually.

**SIMULATION OF THE POTENTIAL TERM**

- Let us focus on the simulation of the evolution of the potential term $e^{-iV(X)\Delta t}$, which is diagonal in the position basis, i.e.,

$$e^{-iV(x)\Delta t} |ka\rangle = e^{-iV_k\Delta t} |ka\rangle \ , \tag{178}$$

  where $V_k \equiv V(ka)$.

- Since the potential term, $V(x)$, is supposed to be an analytic function, the required operation can be achieved in the following way:

- Suppose we are given a function $f$, which takes a $m$-bit string to an $n$-bit string, i.e.,

$$f : \{0, ..., 2^m - 1\} \to \{0, ..., 2^n - 1\} \ . \tag{179}$$

- Suppose the function $f$ can be computed reversibly i.e.,

$$(x, y) \to (x, y \oplus f(x)) \ , \tag{180}$$

using $T$ Toffoli gates (see section 3.1).

- From equation (127), one can also implement such an operation on a quantum computer, i.e.,

$$U_f |x\rangle |0\rangle = |x\rangle |f(x)\rangle \equiv |x\rangle |b_0\rangle |b_1\rangle |b_2\rangle \cdots \tag{181}$$

where we labeled

$$f(x) = b_0 b_1 b_2 \cdots \tag{182}$$

- For each state $|b_k\rangle$, we apply the following gate to each qubit, namely $e^{-i\theta_k \sigma_0}$, where $\sigma_0 |0\rangle = 0$ and $\sigma_0 |1\rangle = |1\rangle$, and $\theta_k = 2^{-k}$.

- Consequently, we have

$$\prod_k e^{-i\theta_k \sigma_0} |x\rangle |f(x)\rangle = e^{-if(x)/2^n} |x\rangle |f(x)\rangle \ . \tag{183}$$

- Finally, an inverse operation $U_f^{-1}$ is required to clean up the memory, in order to achieve the overall action,

$$|x\rangle \quad \to \quad e^{-if(x)/2^n} |x\rangle \ , \tag{184}$$

which is all we need for simulating the potential term in equation (178).

**SIMULATION OF THE KINETIC TERM**

- The other term, $e^{-i(P^2/2m)\Delta t}$, involves the momentum operator $P$.

- As a matter of fact, there is a relationship between the position $X$ and momentum operator $P$,

$$P = U_{FT} \ X \ U_{FT}^\dagger \ , \tag{185}$$

which means that we can implement the evolution in the following way:

$$e^{-i(P^2/2m)\Delta t} = U_{FT} \ e^{-i(X^2/2m)\Delta t} \ U_{FT}^\dagger \ . \tag{186}$$

- The quantum Fourier transform operator is discussed in another section.

# 5 Classical Computational Complexity

- A **complexity class** can be thought of as **a collection of computational problems**, all of which share some common feature with respect to the computational resources needed to solve those problems.

## 5.1 Elementary theory of classical computation

- A deterministic classical computer works as follows: it takes $n$-bits of input and produces $m$-bits of output, i.e.,

$$f : \{0,1\}^n \to \{0,1\}^m \ . \tag{187}$$

**BOOLEAN FUNCTIONS**

- A function with an $m$-bit output is equivalent to $m$ functions, each with a one-bit out, which means that we can also consider the basic task of a classical computer is to do the following:

$$f : \{0,1\}^n \to \{0,1\} \ . \tag{188}$$

- A function that takes an $n$-bit input to a one-bit output is called a **Boolean function**.

- For a string of $n$-bits, there are totally $2^{2^n}$ possible functions **exercise** [20].

- A more precise definition: a Boolean function contains a subset $\Sigma$,

$$\sum = \left\{ x^{(1)}, x^{(2)}, x^{(3)}, ... \right\} \ , \tag{189}$$

of the $n$-bit strings $x^{(i)}$, where all $x^{(i)}$ satisfy the condition of $f(x^{(i)}) = 1$; we say these strings are **accepted** by $f$.

- The complementary set $\sum_{\text{com}}$ contains values of $x$ where $f(x) = 0$, which is said to be "**rejected**" by $f$.

**BOOLEAN FUNCTIONS BY LOGICAL OPERATIONS**

- A Boolean function $f$ can be reduced to a sequence of simple logical operations.

[20] There are totally $k = 2^n$ possible inputs. For each input, we can assign either '0' or '1', which gives a total of $2^k$ possible assignments.

- For each $x^{(a)}$, we define a function $f^{(a)}(x) : \{0,1\}^n \to \{0,1\}$ such that

$$f^{(a)}(x) = \begin{cases} 1 & x = x^{(a)} \\ 0 & \text{otherwise} \end{cases} \tag{190}$$

- Then $f$ can be expressed as a logical OR "$\vee$" of the functions $f^{(a)}(x)$:

$$f(x) = f^{(1)}(x) \vee f^{(2)}(x) \vee f^{(3)}(x) \vee ... \tag{191}$$

- We now label each string $x$ as follows:

$$x = x_{n-1}x_{n-2}...x_2x_1x_0 \ . \tag{192}$$

- For each $f^{(a)}(x)$, if the solution is a simple form like this, $x^{(a)} = 11...111$, then we can simply write

$$f^{(a)}(x) = x_{n-1} \wedge x_{n-2} \wedge ... \wedge x_2 \wedge x_1 \wedge x_0 \tag{193}$$

  as a result of a series of the AND "$\wedge$" operations.

- For general $x^{(a)}$, we can have a similar result by applying the NOT "$\neg$" operation for the bits that are not 1; for example, we write

$$f^{(a)}(x) = ... (\neg x_3) \wedge x_2 \wedge x_1 \wedge (\neg x_0) \tag{194}$$

  if $x^{(a)} = ...0110$.

- Consequently, any function $f(x)$ can be constructed by a using logical operations OR, AND, and NOT. Th expression obtained from them is called the "**disjunctive normal form**" (DNF) of $f(x)$.

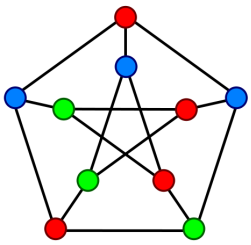
- A computation is a finite sequence of such operations, a **circuit**, applied to a specified string of input bits. Each operation is called a **gate**.

## USING "LANGUAGES" TO SPECIFY DECISION PROBLEMS

- There are four major types of computational problems, namely
  - decision problems
  - search problems

  - optimization problems
  - counting problems

- Here we are mostly interested in decision problems, which have only two possible outputs, yes or no (or alternately 1 or 0) on any input.
- A Boolean function $f$ may be said to *encode* a solution to a "**decision problem**" - the function examines the input and issues a **YES** or **NO** answer.
- In other words, a decision problem can be viewed as a Boolean function on binary strings.
- An example of decision problem is the 3-coloring problem: given an undirected graph, determine whether there is a way to assign a "color" chosen from {**R**ed, **G**reen, **B**lue} to each vertex in such a way that no two adjacent vertices have the same color.



- What might not be stated colloquially as a question with a YES/NO answer can be "repackaged" as a decision problem. For example, we can always turn an optimization problem into a decision problem.
- Now we invoke some useful definitions from the formal language machinery:
- An **alphabet** $\Sigma$ is *a finite set of symbols*.
- A **language** $L$ over $\Sigma$ is *any set of strings made up of symbols from* $\Sigma$.
  - For example, if $\Sigma = \{0, 1\}$, the set $L = \{10, 100, 110, 1000, 1010, ...\}$ is the language of the binary representation of even positive integers.

- A decision problem, where the input $x$ is accepted by a function family $f(x)$, can now be described by a language, , i.e.,

$$L = \left\{ x \in \{0,1\}^* : f(x) = 1 \right\} , \tag{195}$$

  where $\{0,1\}^*$ denotes strings with variable length.

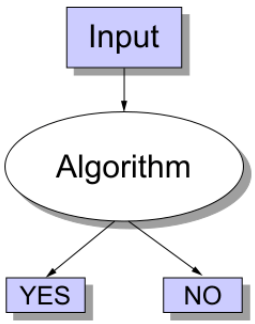
- For example, if we call 3COL the subset of $\{0,1\}$ containing (descriptions of) 3-colorable graphs, then 3COL is the language that specifies the 3-coloring problem.

- Very often, people would use the term decision problem and language interchangeably.

## 5.2 Algorithms of decision problems

- A method for solving a decision problem, given in the form of an **algorithm**, is called a decision procedure for that problem.
  - An algorithm $A$ is said to accept a string $x \in \{0,1\}^*$ if, given input $x$, the algorithm's output $A(x)$ is 1, i.e., $A(x) = 1$.
  - An algorithm $A$ rejects a string $x$, if $A(x) = 0$.

- In other words, decision problems consist of three parts: input, algorithm, and (YES/NO) output.




- Even if language $L$ is accepted by an algorithm $A$, the algorithm will not necessarily reject a string $x \notin L$ provided as input to it. For example, the algorithm may loop forever.

- A language $L$ is **decided** by an algorithm $A$ if every binary string in $L$ is accepted by $A$ and every binary string not in $L$ is rejected by $A$.

**Example 2**

- Let us consider the problem of deciding if a number is a prime number, which is called **primality test** and can be solved efficiently.
- Denote a language,

$$\mathsf{PRIME} = \{10, 11, 101, 111, 1011, \ldots\} , \tag{196}$$

  which contains all prime numbers in the binary form.
- If a binary string $x$ is a prime number, i.e., $x \in \mathsf{PRIME}$, then the corresponding algorithm $A$ should accept $x$, i.e., $A(x) = 1$ (YES!).
- On the other hand, if $x$ is not a prime number, i.e., $x \notin \mathsf{PRIME}$, then the algorithm should reject it, i.e., $A(x) = 0$ (NO!).

## 5.3 Classical Complexity Classes

- Associated with a family of functions $\{f_n\}$ (where $f_n$ has $n$-bit input) are circuits $\{C_n\}$ that compute the functions.
- We say that a circuit family $\{C_n\}$ is "polynomial size" if the size $|C_n|$ of $C_n$ grows with $n$ no faster than a power of $n$.

**COMPLEXITY CLASS P**

- The class of decision problems that can have a polynomial size circuit is defined to be the class P, which means "polynomial time". More precisely,

- A language $L$ (or a set of decision problems) is in the complexity class P if and only if there exists a (worst-case) polynomial-time algorithm $A$ such that
  - for all input $x$ in $L$, $A(x) = 1$ (accept), and
  - for all input $x$ not in $L$, $A(x) = 0$ (reject).

- In other words, P represents the sets of languages that can be decided by an algorithm $A$ efficiently, or more precisely,

$$\mathsf{P} = \left\{ L \subseteq \{0,1\}^* : \exists \text{ an algorithm } A \text{ that decides } L \text{ in polynomial time} \right\} \tag{197}$$

- Decision problems in P are considered to be "easy" to solve. Problems require an exponentially growing circuit to solve are "hard".

- Of course, logically, a polynomial can be made larger than an exponential. Practically, the classification of the hardness of computational problems based on the polynomial scaling vs exponential scaling works very well so far.
- Any problem in P does not depend which universal gate set to use. Any universal set of gate can simulate another set efficiently.

## COMPLEXITY CLASS NP

- An other important class of decision problems is called "NP", which stands for "nondeterministic polynomial time", which is defined as

---

- A language L is in NP iff there is a polynomial-size verifier $V(x, y)$ such that
  - for all $x \in L$, then there exists $y$ such that $V(x, y) = 1$ (completeness),
  - for all $x \notin L$, then, for all $y$, $V(x, y) = 0$ (soundness).

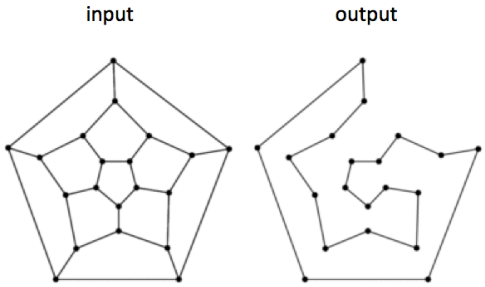
---

- Completeness means that for each input in the language (for which the answer is YES), there is an appropriate "witness" such that the verifier accepts the input if that witness is provided.
- Soundness means that for each input not in the language (for which the answer is NO) the verifier rejects the input no matter what witness is provided.
- It is implicit that the witness is of polynomial length, $|y| = poly(|x|)$; since the verifier has a polynomial number of gates, including input gates, it cannot make use of more than a polynomial number of bits of the witness.

**Example 3**

---

- A hamiltonian cycle of an undirected graph $G(V, E)$ is a simple cycle that contains each vertex in $V$ .
- A graph that contains a hamiltonian cycle is said to be **hamiltonian**; otherwise, it is nonhamiltonian.
- Problem: Find an ordering of the vertices such that each vertex is visited exactly once (i.e., the path passing each corner exactly once and returning to the starting corner).
- The following is an example of a Hamilton cycle




## 5.4 P versus NP problems

- In short, **P** is the class of computational problems that can be **solved quickly on a classical computer**; **NP** is the class of problems which have solutions which can be quickly checked on a classical computer.
- Consider the problem of finding the prime factors of an integer, $n$. So far, there is no efficient way of solving this problem on a classical computer, which suggests that the problem is not in **P**.
- On the other hand, if somebody tells you that some number $p$ is a factor of $n$, then we can quickly check that this is correct by dividing $p$ into $n$, so factoring is a problem in **NP**.

**PROBLEMS IN 'P' IS ALSO IN 'NP'**

- It is clear that **P** is a subset of **NP**, i.e., $\mathsf{P} \subseteq \mathsf{NP}$, since the ability to solve a problem implies the ability to check potential solutions.
- What is not so clear is whether or not there are problems in **NP** that are not in **P**.
- Perhaps the most important unsolved problem in theoretical computer science is to determine whether these two classes are different:

$$\mathbf{P} \neq \mathbf{NP} \quad (\text{true?}) \,. \tag{198}$$

- Most researchers believe that **NP** contains problems that are not in **P**.

## NP-COMPLETE PROBLEMS

- There is an important subclass of the **NP** problems, the **NP**-complete problems.
- In fact, there are thousands of problems known to be **NP**-complete.
- Any given **NP**-complete problem is in some sense 'at least as hard' as all other problems in **NP**.
- More precisely, an algorithm to solve a specific **NP**-complete problem can be adapted to solve any other problem in **NP**, with a small (polynomail) overhead.
- In particular, if $\mathbf{P} \neq \mathbf{NP}$, then it will follow that no **NP**-complete problem can be efficiently solved on a classical computer.

## NP PROBLEMS FOR QUANTUM COMPUTERS

- It is not known whether quantum computers can be used to quickly solve all the problems in **NP**, despite the fact that they can be used to solve some problems, e.g. factoring.
- Many people believe that factoring is in **NP** but not in **P**.
- Note that factoring is not known to be **NP**-complete, otherwise we would already know how to efficiently solve all problems in **NP** using quantum computers.
- It would certainly be very exciting if it were possible to solve all the problems in **NP** efficiently on a quantum computer.

## A NEGATIVE RESULT

- There is a very interesting negative result known in this direction which rules out using a simple variant of quantum parallelism to solve all the problems in **NP**.
- Specifically, one approach to the problem of solving problems in **NP** on a quantum computer is to try to use some form of quantum parallelism to search in parallel through all the possible solutions to the problem.

- It has been proved that no approach based upon such a search-based methodology can yield an efficient solution to all the problems in **NP**.

## 5.5 Randomized computation

- A gate in a probabilistic circuit might act in either one of two ways, and flip a fair coin to decide which action to execute. For example, Markov-chain Monte Carlo, or Metropolis sampling.
- In general, a probabilistic algorithm may not directly yield a high probability.
- However, for a decision problem, one can always amplify it as long as,
  - for $x \in L$, the probability of accepting $x$ is a bit greater than $1/2$, i.e., $\Pr[\text{accept}] = 1/2 + \delta$, for any real number $\delta > 0$, and
  - for $x \notin L$, the probability of accepting $x$ is smaller than $\Pr[\text{accept}] = 1/2 - \delta$.
- More precisely, we would simply run the same computation many times and take the **majority vote**, then we have the following results from the **Chernoff bound**:
  - For $x \in L$, if we run the computation $N$ times, the probability of rejecting in more than half the runs is no more than $e^{-2N\delta^2}$.
  - For $x \notin L$, the probability of accepting in the majority of $N$ runs is no more than $e^{-2N\delta^2}$

---

*Proof.* For $x \in L$:

- There are all together $2^N$ possible sequences of outcomes in the $N$ trials.
- The probability of any particular sequence with $N_w$ wrong answers is
$$\left(\frac{1}{2} - \delta\right)^{N_w} \left(\frac{1}{2} + \delta\right)^{N - N_w}, \tag{199}$$
where the majority is wrong only if $N_w \geq N/2$.

- The probability of any sequence with an incorrect majority is no larger than

$$\left(\frac{1}{2}-\delta\right)^{N/2}\left(\frac{1}{2}+\delta\right)^{N/2}=\frac{1}{2^N}\left(1-4\delta^2\right)^{N/2}. \qquad (200)$$

- Multiplying by the total number of sequences $2^N$ , we obtain the Chernoff bound:

$$\Pr\left(\text{wrong majority}\right)\leqslant\left(1-4\delta^2\right)^{N/2}\leqslant e^{-2N\delta^2}, \qquad (201)$$

where we used the inequality: $1-x\leqslant e^{-x}$.

□

---

- In other words, for $x\in L$, if we want an overall probability of error bounded by a small $\epsilon>0$, i.e., $\epsilon\geqslant\Pr\left(\text{wrong majority}\right)$, then we need to have

$$N\geqslant\frac{1}{2\delta^2}\ln\left(\frac{1}{\epsilon}\right). \qquad (202)$$

- Therefore, problems with different values of $\delta$ does not differ much in terms of the computational complexity.
- The standard convention is to specify $\delta=1/6$, so that $x\in L$ is accepted with probability at least $2/3$ and $x\notin L$ is accepted with probability no more than $1/3$.
- More precisely, the complexity class of BPP (bounded-error probabilistic polynomial time) is defined as follows:
- A language $L$ (or a set of decision problems) is in the complexity class BPP if and only if there exists a (worst-case) polynomial-time algorithm $A$ such that
  - for all input $x$ in $L$, $\Pr[A(x)=1]\geq 2/3$, and
  - for all input $x$ not in $L$, $\Pr[A(x)=1]\leq 1/3$.
- Note that P is contained in BPP, i.e., $\mathsf{P}\subseteq\mathsf{BPP}$, which is true because P is a special case of BPP, where the additional resource of randomness is not applied.
- A randomized class analogous to NP, called MA ("Merlin-Arthur"), containing languages that can be checked when a randomized verifier is provided with a suitable witness.

# 6 Quantum Computational Complexity

- We are ready to formulate a mathematical model of a quantum computer.
- It will help us formulate a framework to answer the question "how powerful are quantum computers?" No one can answer it unambiguously yet.
- It is still possible that quantum computers are no more powerful than classical computers, in the sense that any problem which can be efficiently solved on a quantum computer can also be efficiently solved on a classical computer. It will very nice if we can eliminate such a possibility.

## 6.1 Complexity class BQP

- Furthermore, we can define a new complexity class BQP ("bounded-error quantum polynomial time") — the class of languages that can be decided with high probability by polynomial-size uniform quantum circuit families.

---

- A language $L$ is in BQP if and only if there exists a polynomial-time uniform family of quantum circuits $Q_n(x)$, which takes $n$ qubits and output 1 qubit, such that
  - For all $x \in L$, $\Pr\left(Q_n\left(x\right)=1\right) \geqslant 2/3$
  - For all $x \notin L$, $\Pr\left(Q_n\left(x\right)=1\right) \leqslant 1/3$

---

- Roughly speaking, BQP represents the class of problems that can be solved efficiently with a quantum computer.
- Notice that a quantum computer can easily simulate a probabilistic classical computer: it can prepare $\left(|0\rangle+|1\rangle\right)/\sqrt{2}$ and then project to $\{|0\rangle, |1\rangle\}$.
- Therefore, BQP contains the class of BPP.

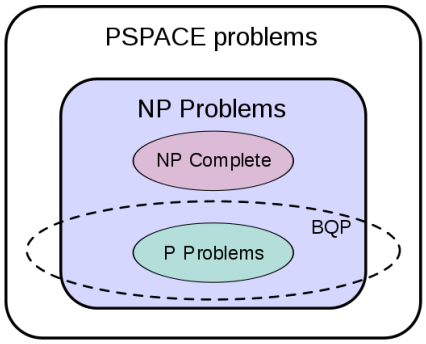
## 6.2 Complexity class PSPACE

- A common misconception is to believe that quantum computers are more powerful because of the vast Hilbert space

it can provide through quantum superposition.

- Despite the vastness of Hilbert space, a classical computer can simulate an $n$-qubit quantum computer even if limited to an amount of memory space that is polynomial in $n$.
- A Language $L$ is in PSPACE if there exists a poly-space Turing machine (or an algorithm) $M$ such that for all $x$, $x \in L$ if and only if $M(x)$ accepts.
- This means the BQP is contained in the complexity class PSPACE, the decision problems that can be solved with polynomial space, which may or may not require an exponential time.
- Note that NP is also contained in PSPACE, because we can determine whether a verifier $V(x, y)$ accepts the input $x$ for any witness $y$ by running the verifier for all possible witnesses. Though there are an exponential number of candidate witnesses to interrogate, each one can be checked in polynomial time and space.
- The following diagram demonstrates the relationship that is consistent with our current understanding of the complexity classes[21]:

[21] No rigorous result has been discovered to justify the diagram.




- Quantum computers pose a serious challenge to the **strong (or extended) Church-Turing thesis**[22], which contends that any physically reasonable model of computation can be simulated by probabilistic classical circuits with at worst a polynomial slowdown.

[22] A thesis here means an assumption.

## 6.3 Proof that BQP is in PSPACE

- We will now show that the classical simulation of a quantum computer can be done to acceptable accuracy (albeit very slowly!) in polynomial space.
- This means that the quantum complexity class BQP is contained in the class PSPACE of problems that can be solved with polynomial space.
- We can always assume that the initial state is fixed to be in the $|0\rangle \equiv |000..0\rangle$ state.
- The goal of the classical simulation is to compute the probability for each possible outcome a of the final measurement:
$$p(a) = |\langle a| U |0\rangle|^2 , \tag{203}$$
where
$$U = U_T U_{T-1} ... U_2 U_1 \tag{204}$$
is a product of $T$ unitary gates.
- Note that $T = T(n)$ is a polynomial of $n$.
- Each $U_t$ acts on $n$ qubits and can be represented by a $2^n \times 2^n$ unitary matrix, characterized by the complex matrix elements
$$\langle y| U_t |x\rangle , \tag{205}$$
where $x, y \in \{0, 1, ..., 2^n - 1\}$.
- Explicitly, we have a path-integral like expression:
$$\langle a| U |0\rangle = \sum_{\{x_t\}} \langle a| U_T |x_{T-1}\rangle ... \langle x_2| U_2 |x_1\rangle \langle x_1| U_1 |0\rangle . \tag{206}$$
- Here each $U_t$ is chosen to be a member in some set of universal quantum gates, which acts on at most two qubits.
- The total number of paths is equal to
$$\underbrace{2^n \times 2^n \times ... \times 2^n}_{T-1 \text{ terms}} = 2^{n(T-1)} . \tag{207}$$
- The classical computer has to calculate exponential number of paths with a bounded precision, as the matrix elements $\langle y| U_t |x\rangle$ may not be rational numbers.

- If for each term we keep $m$ digits, which means that the error for each term is less than $1/2^m$. There are $T$ terms for each path, and totally $2^{n(T-1)}$ paths. So, we have to deal with $T2^{n(T-1)}$. The total errors is less than

$$\epsilon \leqslant \frac{2^{n(T-1)}}{2^m} T. \tag{208}$$

- To make it small, we need to choose a large enough $m$, such that $m > \log_2 T + n\,(T-1)$, which is polynomial in $n$, as required by PSPACE. □
- Although it may sound trivial now, establishing BQP $\subseteq$ PSPACE is one of the most significant results in quantum computational complexity.
- Now, together with what we discussed previously that BPP $\subseteq$ BQP, we have the chain of relations:

$$\mathsf{BPP} \subseteq \mathsf{BQP} \subseteq \mathsf{PSPACE}\ . \tag{209}$$

- If we can prove that BPP $\neq$ BQP, then it implies that BPP $\neq$ PSPACE, which is no known to be true or false in the field of traditional computer science.
- Therefore proving that quantum computers are more powerful than classical computers would have some very interesting implications for classical computational complexity! Unfortunately, it also means that providing such a proof may be quite difficult.

## 6.4 PSPACE and BPP problems

**PSPACE PROBLEMS**

- Another important complexity class is **PSPACE**. Roughly speaking, **PSPACE** consists of those problems which can be solved using resources which are few in spatial size (that is, the computer is 'small'), but not necessarily in time ('long' computations are fine).
- More precisely, **PSPACE** is defined as the class of decision problems which can be solved on a Turing machine using **space polynomial in the problem size** and an arbitrary amount of time.

- **PSPACE** is believed to be strictly larger than both **P** and **NP** although, again, this has never been proved.

**BPP PROBLEMS**

- In computational physics, it is quite common to rely on random resource for solving physical problems, e.g. Monte Carlo methods.
- The complexity class **BPP** is the class of problems that can be solved using randomized algorithms in polynomial time, if a bounded probability of error (say $1/4$) is allowed in the solution to the problem.
- **BPP** is widely regarded as being, the class of problems which should be considered efficiently soluble on a classical computer.

## 6.5 BQP problems

**BQP PROBLEMS**

- We can define **BQP** to be the class of all computational problems which can be **solved efficiently on a quantum computer**, where a bounded probability of error is allowed.
- More formally, a language $L$ is in **BQP** if there is a family of polynomial size quantum circuits which decides the language such that

  (i) it accepts strings in the language with probability at least $3/4$, and

  (ii) it rejects strings which are not in the language with probability at least $3/4$.
- Additionally, the size of the circuits should only grow polynomially with the size of the input string $x$ for which we are trying to determine whether $x \in L$.
- What this means is that the quantum circuit takes as input binary strings, and tries to determine whether they are elements of the language or not.

- At the conclusion of the circuit one qubit is measured, with 0 indicating that the string has been accepted, and 1 indicating rejection.
- By testing the string a few times to determine whether it is in $L$, we can determine with very high probability whether a given string is in $L$.
- Strictly speaking this makes **BQP** more analogous to the classical complexity class **BPP** than to **P**.
- What is known is that quantum computers can solve all the problems in **P** efficiently, but that there are no problems outside of **PSPACE** which they can solve efficiently.

### COMPLEXITY OF LOCAL HAMILTONIAN TIME EVOLUTION

- Since we can employ a set of local Hamiltonians to perform quantum computation, the problem of simulating Hamiltonian time evolution is **BQP-hard**.
- On the other hand, we have discussed that each time-evolution operator can be simulated with high accuracy using elementary quantum gates, form the Solovay-Kitaev theorem. Therefore, a quantum computer is capable of simulating the time evolution of local Hamiltonians, which means that the latter is a problem in **BQP**.
- Combing these two arguments, the problem of simulating local-Hamiltonian time evolution is **BQP-complete**.

### BQP vs PSPACE

- In fact, one of the most significant results in quantum computational complexity is that **BQP** $\subseteq$ **PSPACE**.

*Proof (informal) for* **BQP** $\subseteq$ **PSPACE**. Suppose we have an $n$-qubit quantum computer initialized in the state $|0^n\rangle$, and do a computation involving a sequence of $p_n = \text{poly}(n)$ polynomial number of gates, i.e., $U_1, U_2, ..., U_{p_n}$. The transition amplitude for obtaining $|y\rangle$ at the end is given by

$$A_y = \langle y| U_{p_n} \cdots U_2 U_1 |0^n\rangle \ . \tag{210}$$

Inserting the completeness relation, $\sum_x |x\rangle \langle x| = I$, many times, we obtain a path integral:

$$A_y = \sum_{\{x\}} \langle y| U_{p_n} |x_{p_n-1}\rangle \cdots \langle x_2| U_2 |x_1\rangle \langle x_1| U_1 |0^n\rangle \ . \quad (211)$$

Since the gates are local, it is clear that each term in the sum can be calculated to high accuracy using only polynomial space on a classical computer. Thus the sum as a whole can be calculated using polynomial space, since individual terms in the sum can be erased after being added to the running total. □

- Therefore, **BQP** lies somewhere between **P** and **PSPACE**.

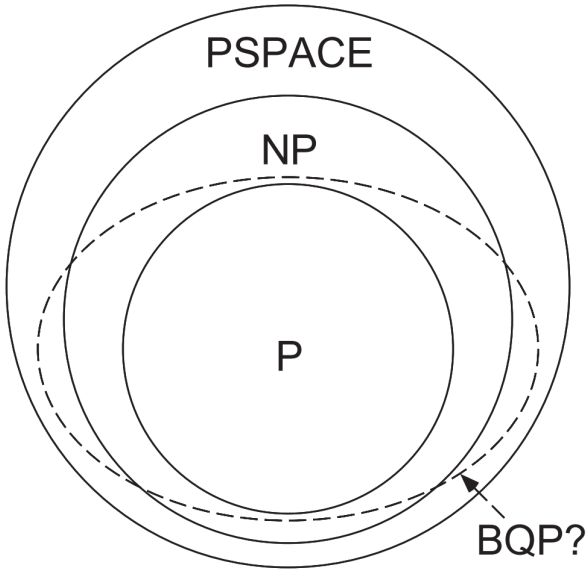



Figure 26: The relationship between classical and quantum complexity classes. Quantum computers can quickly solve any problem in **P**, and it is known that they can't solve problems outside of **PSPACE** quickly. Where quantum computers fit between **P** and **PSPACE** is not known, in part because we don't even know whether PSPACE is bigger than **P**!

- Quantum computers do not violate the Church-Turing thesis that any algorithmic process can be simulated using a Turing machine.
- Quantum computers may be much more efficient than their classical counterparts, thereby challenging the strong Church-Turing thesis that any algorithmic process can be simulated efficiently using a probabilistic Turing machine.
- If it is proved that quantum computers are strictly more powerful than classical computers, then it will follow that **P** is not equal to **PSPACE**. Proving this latter result has been attempted without success by many computer scientists.

### BQP vs BPP

- It is also true that **BPP** $\subseteq$ **BQP**, where **BPP** is the classical complexity class of decision problems which can be solved with bounded probability of error using polynomial time on a classical Turing machine.
- Thus we have the chain of inclusions

$$\mathbf{BPP} \subseteq \mathbf{BQP} \subseteq \mathbf{PSPACE} \,. \tag{212}$$

- If we believe that quantum computer is more powerful than classical computers, then we should expect that **BQP** $\neq$ **BPP**.
- Unfortunately, no such a proof exists so far. If it is true that **BQP** $\neq$ **BPP**, then one can prove that **BPP** $\neq$ **PSPACE**, which is still an outstanding major open problem in computer science.
- Therefore, proving that quantum computers are more powerful than classical computers would have some very interesting implications for classical computational complexity!
- Unfortunately, it also means that providing such a proof may be quite difficult.