Quantum Computing
Lecture 1

Bits and Qubits

Anuj Dawar

What is Quantum Computing?

Aim to use quantum mechanical phenomena that have no classical counterpart for computational purposes.

Central research tasks include:

- *Building devices* — with a specified behaviour.
- *Designing algorithms* — to use the behaviour.

Mediating these two are models of computation.

Bird's eye view

A computer scientist looks at Quantum Computing:

Algorithmic Languages

Theory/complexity

System Architecture

Specified Behaviour

Physics

𝔇𝔯𝔞𝔤𝔬𝔫𝔰

Why look at Quantum Computing?

- *The world is quantum*
  - classical models of computation provide a level of abstraction
  - discrete state systems
- *Devices are getting smaller*
  - Moore's law
  - the only descriptions that work on the very small scale are quantum
- *Exploit quantum phenomena*
  - using quantum phenomena may allow us to perform computational tasks that are not otherwise possible/efficient
  - understand capabilities/resources

Course Outline

A total of eight lecturers.

1. *Bits and Qubits* (this lecture).
2. *Linear Algebra*
3. *Quantum Mechanics*
4. *Models of Computation*
5. *Some Applications*
6. *Search Algorithms*
7. *Factorisation*
8. *Complexity*

Useful Information

Some useful books:

- Nielsen, M.A. and Chuang, I.L. (2010). *Quantum Computation and Quantum Information*. 2nd ed. Cambridge University Press.
- Mermin, N.D. (2007). *Quantum Computer Science*. CUP.
- Kitaev, A.Y., Shen, A.H. and Vyalyi, M.N. (2002). *Classical and Quantum Computation*. AMS.

Course website:

`http://www.cl.cam.ac.uk/teaching/1718/QuantComp/`

Bits

A building block of classical computational devices is a two-state system.

$$0 \quad \longleftrightarrow \quad 1$$

Indeed, any system with a finite set of *discrete, stable* states, with controlled transitions between them will do.

Qubits

Quantum mechanics tells us that any such system can exist in a *superposition* of states.

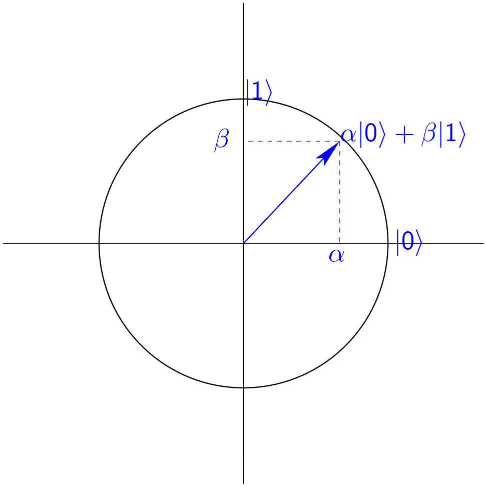
In general, the state of a *quantum bit* (or *qubit* for short) is described by:

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

where, α and β are complex numbers, satisfying

$$|\alpha|^2 + |\beta|^2 = 1$$

Qubits



A qubit may be visualised as a unit vector on the plane.

In general, however, α and β are *complex* numbers.

Measurement

Any attempt to measure the state

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

results in $|0\rangle$ with probability $|\alpha|^2$, and $|1\rangle$ with probability $|\beta|^2$.

After the measurement, the system is in the measured state!

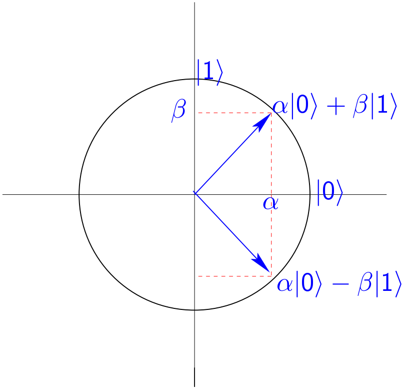
That is, further measurements will always yield the same value.

We can only extract one bit of information from the state of a qubit.

Measurement

$\alpha|0\rangle + \beta|1\rangle$ and $\alpha|0\rangle - \beta|1\rangle$ have the same probabilities for their measurement

However, they are *distinct* states which behave differently in terms of how they evolve.



$|1\rangle$
β
$\alpha|0\rangle + \beta|1\rangle$
$|0\rangle$
α
$\alpha|0\rangle - \beta|1\rangle$

Vectors

Formally, the state of a qubit is a unit vector in $\mathbb{C}^2$—the 2-dimensional complex *vector space*.

The vector $\left[\begin{array}{c}\alpha\\\beta\end{array}\right]$ can be written as

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

where, $|0\rangle = \left[\begin{array}{c}1\\0\end{array}\right]$ and $|1\rangle = \left[\begin{array}{c}0\\1\end{array}\right]$.

$|\phi\rangle$— a *ket*, Dirac notation for vectors.

Basis

Any pair of vectors $|\phi\rangle, |\psi\rangle \in \mathbb{C}^2$ that are linearly independent could serve as a basis.

$$\alpha|0\rangle + \beta|1\rangle = \alpha'|\phi\rangle + \beta'|\psi\rangle$$

The basis is determined by the measurement process or device.

Most of the time, we assume a standard (orthonormal) basis $|0\rangle$ and $|1\rangle$ is given.

This will be called the *computational basis*

Example

The vector $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ measured in the computational basis gives either outcome with probability $1/2$.

Measured in the basis

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix}$$

it gives the first outcome with probability 1.

Entanglement

An n-qubit system can exist in any superposition of the 2^n *basis* states.

$$\alpha_0|000000\rangle + \alpha_1|000001\rangle + \cdots + \alpha_{2^n-1}|111111\rangle$$

with $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$

Sometimes such a state can be decomposed into the states of individual bits

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

Entanglement

Compare the two (2-qubit) states:

$$\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle) \quad \text{and} \quad \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

If we measure the first qubit in the first case, we see $|0\rangle$ with probability 1 and the state remains unchanged.

In the second case (*an EPR pair*), measuring the first bit gives $|0\rangle$ or $|1\rangle$ with equal probability. After this, the second qubit is also determined.

Quantum Computing
Lecture 2

Review of Linear Algebra

Anuj Dawar

Linear Algebra

The state space of a quantum system is described in terms of a *vector space*.

Vector spaces are the object of study in *Linear Algebra*.

In this lecture we review definitions from linear algebra that we need in the rest of the course.

We are mainly interested in vector spaces over the *complex number field* – $\mathbb{C}$.

We use the *Dirac notation*—$|v\rangle, |\phi\rangle$ (read as *ket*) for vectors.

Vector Spaces

A vector space over $\mathbb{C}$ is a set $\mathbf{V}$ with

- a commutative, associative addition operation $+$ that has
    - an identity $\mathbf{0}$: $|v\rangle + \mathbf{0} = |v\rangle$
    - inverses: $|v\rangle + (-|v\rangle) = \mathbf{0}$
- an operation of multiplication by a scalar $\alpha \in \mathbb{C}$ such that:
    - $\alpha(\beta|v\rangle) = (\alpha\beta)|v\rangle$
    - $(\alpha + \beta)|v\rangle = \alpha|v\rangle + \beta|v\rangle$ and $\alpha(|u\rangle + |v\rangle) = \alpha|u\rangle + \alpha|v\rangle$
    - $1|v\rangle = |v\rangle$.

$\mathbb{C}^n$

$\mathbb{C}^n$ is the vector space of n-tuples of complex numbers: $\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}$.

with addition $\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} + \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} = \begin{bmatrix} \alpha_1 + \beta_1 \\ \vdots \\ \alpha_n + \beta_n \end{bmatrix}$

and scalar multiplication $z \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} z\alpha_1 \\ \vdots \\ z\alpha_n \end{bmatrix}$

Basis

A *basis* of a vector space $\mathbf{V}$ is a *minimal* collection of vectors $|v_1\rangle, \ldots, |v_n\rangle$ such that every vector $|v\rangle \in \mathbf{V}$ can be expressed as a linear combination of these:

$$|v\rangle = \alpha_1|v_1\rangle + \cdots + \alpha_n|v_n\rangle.$$

n—the size of the basis—is uniquely determined by $\mathbf{V}$ and is called the *dimension* of $\mathbf{V}$.

Given a basis, every vector $|v\rangle$ can be represented as an n-tuple of scalars.

Bases for $\mathbb{C}^n$

The standard basis for $\mathbb{C}^n$ is $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$
(written $|0\rangle, \ldots, |n-1\rangle$).
But other bases are possible: $\begin{bmatrix} 3 \\ 2 \end{bmatrix}, \begin{bmatrix} 4 \\ -i \end{bmatrix}$ is a basis for $\mathbb{C}^2$.

We'll be interested in *orthonormal* bases. That is bases of vectors of unit length that are mutually orthogonal. Examples are $|0\rangle, |1\rangle$ and $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

Linear Operators

A linear operator A from one vector space $\mathbf{V}$ to another $\mathbf{W}$ is a function such that:

$$A(\alpha|u\rangle + \beta|v\rangle) = \alpha(A|u\rangle) + \beta(A|v\rangle)$$

If $\mathbf{V}$ is of dimension n and $\mathbf{W}$ is of dimension m, then the operator A can be represented as an $m \times n$-matrix.

The matrix representation depends on the choice of bases for $\mathbf{V}$ and $\mathbf{W}$.

Matrices

Given a choice of bases $|v_1\rangle, \dots, |v_n\rangle$ and $|w_1\rangle, \dots, |w_m\rangle$, let

$$A|v_j\rangle = \sum_{i=1}^{m} \alpha_{ij}|w_i\rangle$$

Then, the matrix representation of A is given by the entries α_{ij}.

Multiplying this matrix by the representation of a vector $|v\rangle$ in the basis $|v_1\rangle, \dots, |v_n\rangle$ gives the representation of $A|v\rangle$ in the basis $|w_1\rangle, \dots, |w_m\rangle$.

Examples

A 45° rotation of the real plane that takes $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ to $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ to $\begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ is represented, in the standard basis by the matrix

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

The operator $\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$ does not correspond to a transformation of the real plane.

Inner Products

An inner product on $\mathbf{V}$ is an operation that associates to each pair $|u\rangle, |v\rangle$ of vectors a *complex number*

$$\langle u|v\rangle.$$

The operation satisfies

- $\langle u|\alpha v + \beta w\rangle = \alpha\langle u|v\rangle + \beta\langle u|w\rangle$
- $\langle u|v\rangle = \langle v|u\rangle^*$ where the * denotes the complex conjugate.
- $\langle v|v\rangle \geq 0$ (note: $\langle v|v\rangle$ is a real number) and $\langle v|v\rangle = 0$ iff $|v\rangle = \mathbf{0}$.

Inner Product on $\mathbb{C}^n$

The standard inner product on $\mathbb{C}^n$ is obtained by taking, for

$$|u\rangle = \sum_i u_i |i\rangle \quad \text{and} \quad |v\rangle = \sum_i v_i |i\rangle$$

$$\langle u|v\rangle = \sum_i u_i^* v_i$$

Note: $\langle u|$ is a *bra*, which together with $|v\rangle$ forms the *bra-ket* $\langle u|v\rangle$.

Norms

The *norm* of a vector $|v\rangle$ (written $\| \, |v\rangle \|$) is the *non-negative, real number*:

$$\| \, |v\rangle \| = \sqrt{\langle v|v\rangle}.$$

A *unit vector* is a vector with norm 1.

Two vectors $|u\rangle$ and $|v\rangle$ are *orthogonal* if $\langle u|v\rangle = 0$.

An *orthonormal* basis for an inner product space $\mathbf{V}$ is a basis made up of *pairwise orthogonal, unit vectors*.

the term *Hilbert space* is also used for an inner product space

Outer Product

With a pair of vectors $|u\rangle \in \mathbf{U}$, $|v\rangle \in \mathbf{V}$ we associate a linear operator $|u\rangle\langle v| : \mathbf{V} \to \mathbf{U}$, known as the *outer product* of $|u\rangle$ and $|v\rangle$.

$$(|u\rangle\langle v|)|v'\rangle = \langle v|v'\rangle|u\rangle$$

$|v\rangle\langle v|$ is the *projection* on the one-dimensional space generated by $|v\rangle$.

Any linear operator can be expressed as a linear combination of outer products:

$$A = \sum_{ij} A_{ij}|i\rangle\langle j|.$$

Eigenvalues

An *eigenvector* of a linear operator $A : \mathbf{V} \to \mathbf{V}$ is a non-zero vector $|v\rangle$ such that

$$A|v\rangle = \lambda|v\rangle$$

for some complex number λ
λ is the *eigenvalue* corresponding to the eigenvector v.

The eigenvalues of A are obtained as solutions of the characteristic equation:

$$\det(A - \lambda I) = 0$$

Each operator has at least one eigenvalue.

Diagonal Representation

A linear operator (over an inner product space) A is said to be *diagonalisable* if

$$A = \sum_i \lambda_i |v_i\rangle\langle v_i|$$

where the $|v_i\rangle$ are an orthonormal set of eigenvectors of A with corresponding eigenvalues λ_i.

Equivalently, A can be written as a matrix

$$\begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}$$

in the basis $|v_1\rangle, \ldots, |v_n\rangle$ of its eigenvectors.

Adjoints

Associated with any linear operator A is its *adjoint* $A^\dagger$ which satisfies

$$\langle v|Aw\rangle = \langle A^\dagger v|w\rangle$$

In terms of matrices, $A^\dagger = (A^*)^T$
where $*$ denotes complex conjugation and T denotes transposition.

$$\begin{bmatrix} 1+i & 1-i \\ -1 & 1 \end{bmatrix}^\dagger = \begin{bmatrix} 1-i & -1 \\ 1+i & 1 \end{bmatrix}$$

Normal and Hermitian Operators

An operator A is said to be *normal* if

$$AA^\dagger = A^\dagger A$$

Fact: An operator is diagonalisable if, and only if, it is normal.

A is said to be *Hermitian* if $A = A^\dagger$

A normal operator is Hermitian if, and only if, it has real eigenvalues.

Unitary Operators

A linear operator A is *unitary* if

$$AA^{\dagger} = A^{\dagger}A = I$$

Unitary operators are normal and therefore diagonalisable.

Unitary operators are norm-preserving and invertible.

$$\langle Au | Av \rangle = \langle u | v \rangle$$

All eigenvalues of a unitary operator have modulus 1.

Tensor Products

If $\mathbf{U}$ is a vector space of dimension m and $\mathbf{V}$ one of dimension n then $\mathbf{U} \otimes \mathbf{V}$ is a space of dimension mn.
Writing $|uv\rangle$ for the vectors in $\mathbf{U} \otimes \mathbf{V}$:

- $|(u + u')v\rangle = |uv\rangle + |u'v\rangle$
- $|u(v + v')\rangle = |uv\rangle + |uv'\rangle$
- $z|uv\rangle = |(zu)v\rangle = |u(zv)\rangle$

Given linear operators $A : \mathbf{U} \to \mathbf{U}$ and $B : \mathbf{V} \to \mathbf{V}$, we can define an operator $A \otimes B$ on $\mathbf{U} \otimes \mathbf{V}$ by

$$(A \otimes B)|uv\rangle = |(Au), (Bv)\rangle$$

Tensor Products

In matrix terms,

$$A \otimes B = \begin{bmatrix} A_{11}B & A_{12}B & \cdots & A_{1m}B \\ A_{21}B & A_{22}B & \cdots & A_{2m}B \\ \vdots & \vdots & \vdots & \\ A_{m1}B & A_{m2}B & \cdots & A_{mm}B \end{bmatrix}$$

Quantum Computing
Lecture 3

Principles of Quantum Mechanics

Anuj Dawar

What is Quantum Mechanics?

Quantum Mechanics is a framework for the development of physical theories.

It is not itself a physical theory.

It states *four mathematical postulates* that a physical theory must satisfy.

Actual physical theories, such as *Quantum Electrodynamics* are built upon a foundation of quantum mechanics.

What are the Postulates About

The four postulates specify a general framework for describing the behaviour of a physical system.

1. How to describe the state of a closed system.—*Statics* or *state space*
2. How to describe the evolution of a closed system.—*Dynamics*
3. How to describe the interactions of a system with external systems.—*Measurement*
4. How to describe the state of a composite system in terms of its component parts.

First Postulate

Associated to any physical system is a *complex inner product space* (or *Hilbert space*) known as the *state space* of the system.
The system is completely described at any given point in time by its *state vector*, which is a *unit vector* in its state space.

Note: Quantum Mechanics does not prescribe what the state space is for any given physical system. That is specified by individual physical theories.

Example: A Qubit

Any system whose state space can be described by $\mathbb{C}^2$—the two-dimensional complex vector space—can serve as an implementation of a qubit.

Example: An electron spin.

Some systems may require an infinite-dimensional state space.
We always assume, for the purposes of this course, that our systems have a *finite dimensional* state space.

Second Postulate

The time evolution of *closed* quantum system is described by the Schrödinger equation:

$$i\hbar \frac{d|\psi\rangle}{dt} = H|\psi\rangle$$

where

- $\hbar$ is Planck's constant; and
- H is a fixed Hermitian operator known as the *Hamiltonian* of the system.

Second Postulate—Simpler Form

The state $|\psi\rangle$ of a closed quantum system at time t_1 is related to the state $|\psi'\rangle$ at time t_2 by a unitary operator U that depends only on t_1 and t_2.

$$|\psi'\rangle = U|\psi\rangle$$

U is obtained from the Hamiltonian H by the equation:

$$U(t_1, t_2) = \exp[\frac{-iH(t_2 - t_1)}{\hbar}]$$

This allows us to consider time as discrete and speak of *computational steps*

Exercise: Check that if H is Hermitian, U is unitary.

Why Unitary?

Unitary operations are the only linear maps that preserve norm.

$$|\psi'\rangle = U|\psi\rangle$$

implies

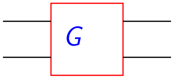
$$||\,|\psi'\rangle|| = ||\,U|\psi\rangle|| = ||\,|\psi\rangle|| = 1$$

Exercise: Verify that unitary operations are norm-preserving.

Gates, Operators, Matrices

In this course, most linear operators we will be interested in are unitary. They can be represented as matrices where each column is a *unit vector* and columns are pairwise orthogonal.

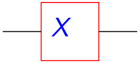
Another useful representation of unitary operators we will use is as gates:



A 2-qubit gate is a unitary operator on $\mathbb{C}^4$.

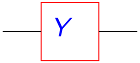
Pauli Gates

A particularly useful set of 1-*qubit* gates are the *Pauli Gates*.
The X gate



$X|0\rangle = |1\rangle \quad X|1\rangle = |0\rangle$ $\qquad$ $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

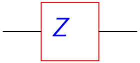
The Y gate



$Y|0\rangle = i|1\rangle \quad Y|1\rangle = -i|0\rangle$ $\qquad$ $Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$

Pauli Gates–*contd.*

The Z gate



$Z|0\rangle = |0\rangle \quad Z|1\rangle = -|1\rangle$ $\qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Sometimes we include the identity $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ as a fourth Pauli gate.

Third Postulate

A measurement on a quantum system has some set M of outcomes. Quantum measurements are described by a collection $\{P_m : m \in M\}$ of *measurement operators*. These are linear (not unitary) operators acting on the state space of the system.
If the state of the system is $|\psi\rangle$ before the measurement, then the probability of outcome m is:

$$p(m) = \langle\psi|P_m^\dagger P_m|\psi\rangle$$

The state of the system after measurement is

$$\frac{P_m|\psi\rangle}{\sqrt{\langle\psi|P_m^\dagger P_m|\psi\rangle}}$$

Third Postulate—*contd.*

The measurement operators satisfy the *completeness equation*.

$$\sum_{m \in M} P_m^\dagger P_m = I$$

This guarantees that the sum of the probabilities of all outcomes adds up to 1.

$$\sum_m p(m) = \sum_m \langle \psi | P_m^\dagger P_m | \psi \rangle = \langle \psi | I | \psi \rangle = 1$$

Measurement in the Computational Basis

We are generally interested in the special case where the measurement operators are projections onto a particular orthonormal basis of the state space (which we call the *computational basis*).

So, for a single qubit, we take measurement operators $P_0 = |0\rangle\langle 0|$ and $P_1 = |1\rangle\langle 1|$

This gives, for a qubit in state $\alpha|0\rangle + \beta|1\rangle$:

$$p(0) = |\alpha|^2 \quad p(1) = |\beta|^2$$

Exercise: Verify!

Global Phase

For any state $|\psi\rangle$, and any θ, we can form the vector $e^{i\theta}|\psi\rangle$.

Then, for any unitary operator U,

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

Moreover, for any measurement operator P_m

$$\langle\psi|e^{-i\theta}P_m^\dagger P_m e^{i\theta}|\psi\rangle = \langle\psi|P_m^\dagger P_m|\psi\rangle$$

Thus, such a global phase is unobservable and the states are physically indistinguishable.

Relative Phase

In contrast, consider the two states $|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.
Measured in the computational basis, they yield the same outcome probabilities.

However, measured in a different orthonormal basis (say $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$), the results are different.

Also, if $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, then

$$H|\psi_1\rangle = |0\rangle \quad H|\psi_2\rangle = |1\rangle$$

Fourth Postulate

The state space of a composite physical system is the tensor product of the state spaces of the individual component physical systems.

If one component is in state $|\psi_1\rangle$ and a second component is in state $|\psi_2\rangle$, the state of the combined system is

$$|\psi_1\rangle \otimes |\psi_2\rangle$$

Not all states of a combined system can be separated into the tensor product of states of the individual components.

Separable States

A state of a combined system is *separable* if it can be expressed as the tensor product of states of the components.
E.g.

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

If Alice has a system in state $|\psi_1\rangle$ and Bob has a system in state $|\psi_2\rangle$, the state of their combined system is $|\psi_1\rangle \otimes |\psi_2\rangle$.

If Alice applies U to her state, this is equivalent to applying the operator $U \otimes I$ to the combined state.

Entangled States

The following states of a 2-qubit system cannot be separated into component parts.

$$\frac{1}{\sqrt{2}}(|10\rangle + |01\rangle) \quad \text{and} \quad \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Note: Physical separation does not imply separability. Two particles that are physically separated could still be entangled.

Summary

Postulate 1: A closed system is described by a unit vector in a complex inner product space.
Postulate 2: The evolution of a closed system in a fixed time interval is described by a unitary transform.
Postulate 3: If we measure the state $|\psi\rangle$ of a system in an orthonormal basis $|0\rangle \cdots |n-1\rangle$, we get the result $|j\rangle$ with probability $|\langle j|\psi\rangle|^2$. After the measurement, the state of the system is the result of the measurement.
Postulate 4: The state space of a composite system is the tensor product of the state spaces of the components.

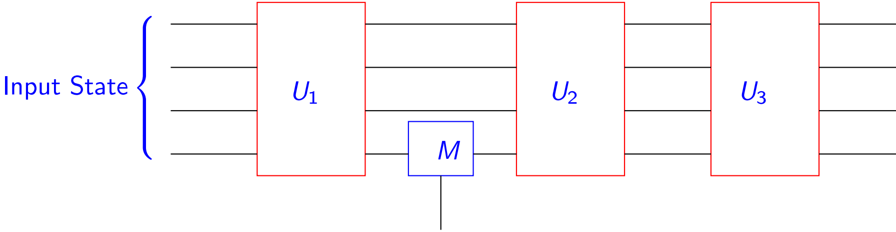
Quantum Computing
Lecture 4

Models of Quantum Computation

Anuj Dawar

Quantum Circuits

A *quantum circuit* is a sequence of unitary operations and measurements on an n-qubit state.



Note: each U_i is described by a $2^n \times 2^n$ matrix.

Algorithms

A *quantum algorithm* specifies, for each n, a sequence

$$\mathcal{O}_n = O_1 \dots O_k$$

of n-qubit operations.

The map $n \to \mathcal{O}_n$ must be computable.

i.e. the individual circuits must be generated from a common pattern.

All measurements can be deferred to the end (possibly, at the expense of increasing the number of qubits).

Model of Computation

As a model of computation, this is parasitic on classical models.

what is computable is not independently determined

Purely quantum models can be defined. We will see more on this in Lecture 8.

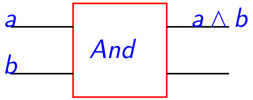
What computations can be performed in the model as defined?

What functions can be computed?
What decision problems are decidable?

Can all such computations be performed with some fixed set of unitary operations?

Simulating Boolean Gates

Could we find a quantum circuit to simulate a classical *And* gate?



This would require $\mathit{And}: \quad |00\rangle \mapsto |0x\rangle, \quad |01\rangle \mapsto |0y\rangle$
$\qquad |10\rangle \mapsto |0z\rangle, \quad |11\rangle \mapsto |1w\rangle$

There is no *unitary* operation of this form.
Unitary operations are reversible. No information can be lost in the process.

Computing a Function

If $f : \{0,1\}^n \to \{0,1\}^m$ is a *Boolean function*, the map

$$|x\rangle \mapsto |f(x)\rangle$$

may not be unitary.

We will, instead seek to implement

$$|x\rangle \otimes |0\rangle \mapsto |x\rangle \otimes |f(x)\rangle$$

Exercise: Describe a unitary operation that implements the Boolean *And* in this sense.

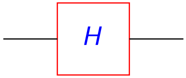
One-Qubit Gates

We have already seen the *Pauli Gates*:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

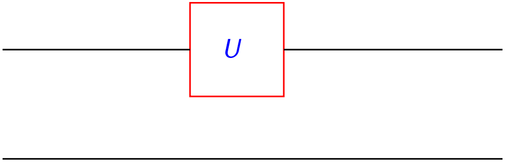
Another useful *one-qubit* gate is the *Hadamard gate*:

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



Gates on a Multi-Qubit State

When we draw a circuit with a one-qubit gate, this must be read as a unitary operation on the *entire state*.

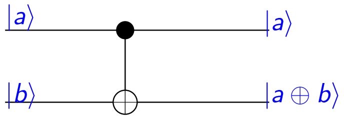


$$U \otimes I$$

This does not change measurement outcomes on the second qubit.

Controlled Not

The *Controlled Not* is a 2-qubit gate:

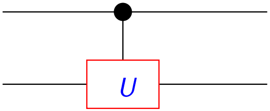


The controlled not flips the second qubit if the first qubit is $|1\rangle$ and leaves it unchanged if it's $|0\rangle$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

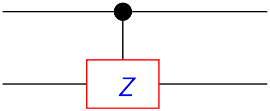
Controlled U

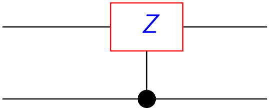
More generally, we can define, for any single qubit operation U, the *Controlled U* gate:



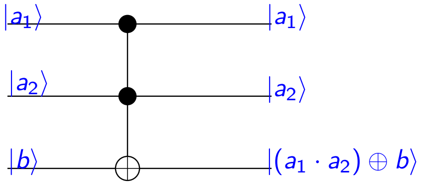
$$|0x\rangle \mapsto |0x\rangle$$
$$|1x\rangle \mapsto |1, Ux\rangle$$

Particularly useful is the controlled-Z gate:





Toffoli Gate



The *Toffoli Gate* is a 3-qubit gate.
It has a classical counterpart which can be used to simulate standard Boolean operations

A *permutation matrix* is a unitary matrix where all entries are 0 or 1.

Any $2^n \times 2^n$ permutation matrix can be implemented using only Toffoli gates.

Classical Reversible Computation

A Boolean function $f : \{0,1\}^n \to \{0,1\}^n$ is *reversible* if it's described by a $2^n \times 2^n$ permutation matrix.

For any function $g : \{0,1\}^n \to \{0,1\}^m$, there is a reversible function $g' : \{0,1\}^{m+n} \to \{0,1\}^{m+n}$ with

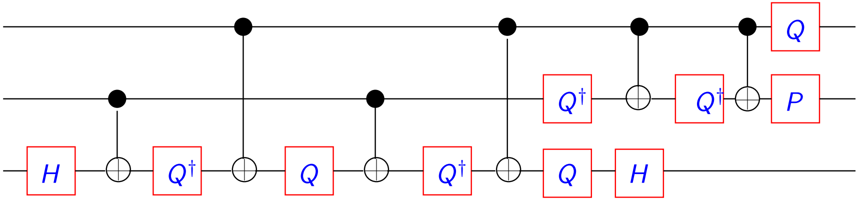
$$g'(x,0) = (x, g(x)).$$

Toffoli gates are *universal* for reversible computation.

The Toffoli gate cannot be implemented using 2-bit reversible classical gates.

Quantum Toffoli Gate

The Toffoli gate can be implemented using 2-qubit quantum gates.



where, $P = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$, $Q = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$.

Universal Set of Gates

Fact: Any unitary operation on n qubits can be implemented by a sequence of 2-qubit operations.

Fact: Any unitary operation can be implemented by a combination of C-NOTs and single qubit operations.

Fact: Any unitary operation can be *approximated* to any required degree of accuracy using only C-NOTs, H, P and Q.

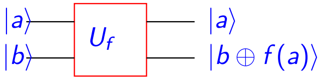
These can serve as our finite set of gates for quantum computation.

Deutsch-Jozsa Problem

Given a function $f : \{0,1\} \to \{0,1\}$, determine whether f is constant or balanced.

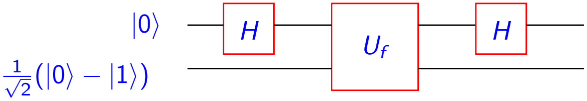
Classically, this requires *two* calls to the function f.

But, if we are given the *quantum black box*:



One use of the box suffices

Deutsch-Jozsa Algorithm



U_f with input $|x\rangle$ and $|0\rangle - |1\rangle$ is just a phase shift.
It changes phase by $(-1)^{f(x)}$.
When $|x\rangle = H|0\rangle$, this gives $(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle$.

Final result is $[(-1)^{f(0)} + (-1)^{f(1)}]|0\rangle + [(-1)^{f(0)} - (-1)^{f(1)}]|1\rangle$
which is $|0\rangle$ if f is constant and $|1\rangle$ if f is balanced.

Quantum Computing
Lecture 5

Applications of Quantum Information

Anuj Dawar

Some Applications

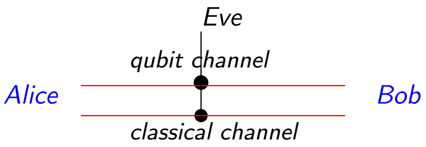
We look at some applications of the encoding of information in quantum states.

- *Quantum Cryptography*, or more accurately *Quantum Key Distribution*.
- *Superdense Coding*.
- *Quantum Teleportation*

These do not rely on *quantum computation* as such, but the properties of information encoded in quantum states: *superposition* and *entanglement*.

Quantum Key Distribution

A protocol for *quantum key distribution* was described by Bennett and Brassard in 1984 (and is known as BB84).



The protocol does not provide the means of transmitting an arbitrary message.

At the end of the protocol, there is a *random* sequence of bits that is shared between *Alice* and *Bob* but unknown to any third party.

Assumptions

The BB84 protocol relies on the following assumptions:

- *Alice* has a source of random (classical) bits.
- *Alice* can produce *qubits* in states $|0\rangle$ and $|1\rangle$.
- *Alice* can apply a *Hadamard operator H* to the qubits.
- *Bob* can measure incoming qubits
  - either in the basis $|0\rangle$, $|1\rangle$;
  - or in the basis $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

These conditions are satisfied, for instance, by a system based on polarised photons.

The Protocol

Alice sends *Bob* a stream of qubits.

For each qubit, before sending it, she

- *randomly* chooses a bit $|0\rangle$ or $|1\rangle$;
- *randomly* either applies H to the qubit or not; and
- sends it to *Bob*.

So, *Bob* receives a random sequence of qubits, each of which is in one of the four states:

$$|0\rangle, |1\rangle, H|0\rangle, H|1\rangle$$

The Protocol–*contd.*

- For each qubit, *Bob* *randomly* chooses either the basis $|0\rangle$, $|1\rangle$ or the basis $H|0\rangle$, $H|1\rangle$ and measures the qubit in the chosen basis.
- *Bob* announces (over the classical channel) which basis he used for each measurement.
- *Alice* tells *Bob* which measurements were made in the correct basis.
- The qubits which were measured in the wrong basis are discarded, while the rest form a shared key.

Attacks

Why not announce the bases for all qubits before transmission, thus avoiding the loss of half the bits?

> *This allows Eve to intercept, measure and re-transmit the bits.*

Why not wait until *Bob* has received all the qubits, then have *Alice* announce the basis for each one before *Bob* measures them?

- Requires *Bob* to store the qubits—currently technically difficult.
- If *Bob* can store the qubits, then *Eve* can too and then she can retransmit after measurement.

If we could fix the basis before hand, this could be used to transmit a *fixed* (rather than random) message.

Attack 2

What happens if *Eve* intercepts the qubits, measures each one randomly in either the basis $|0\rangle$, $|1\rangle$ or the basis $H|0\rangle, H|1\rangle$ and then retransmits it?

For half of the bits *that are shared between Alice and Bob*, *Eve* will have measured them in the wrong basis.
Moreover, these bits will have changed state, and so for approx. $\frac{1}{4}$ of the *shared* bits, the value measured by *Bob* will be different to the one encoded by *Alice*.
Alice and *Bob* can choose a random sample of their shared bits and publically check their values against each other and detect the presence of an eavesdropper.

Attack 3

Could *Eve* intercept the qubits, make a copy *without measuring them* and re-transmit to *Bob* and then wait for the basis to be announced?

No Cloning Theorem:
There is no unitary operation U which for an arbitrary state ψ gives

$$U|\psi 0\rangle = |\psi\psi\rangle.$$

Exercise: Prove the no-cloning theorem.

Key Distribution

Quantum key distribution relies on nothing more than

- linear superposition of states; and
- change of basis.

In particular, it does not rely on *entanglement*.
We next look at some applications of entanglement.

Bell States

Entanglement based protocols generally rely on using the following four states of a two-qubit system, known as the *Bell states*.

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \quad \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$$

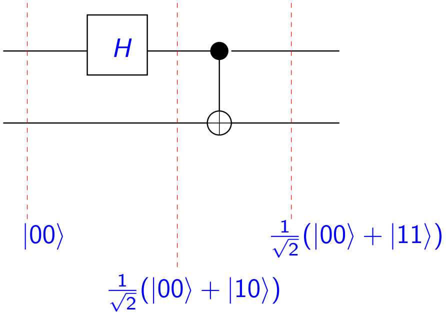
$$\frac{1}{\sqrt{2}}(|00\rangle - |11\rangle), \quad \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

These form an orthonormal basis for $\mathbb{C}^4$, known as the *Bell basis*.

Note that, in each of the states, measuring either qubit in the computational basis yields $|0\rangle$ or $|1\rangle$ with equal probability, but after the measurement, the other bit is determined.

Generating Bell States

We can generate the Bell states from the computational basis $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ using the following circuit:


H
$|00\rangle$
$\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$
$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$

Superdense Coding

In general, it is impossible to extract more than one classical bit of information from a single qubit.

However, if *Alice* and *Bob* is each in possession of one qubit of a pair in a known Bell state

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

Then *Alice* can perform an operation *solely* on her own qubit, and then send it to *Bob* to convey two bits of information.

Superdense Coding 2

Generating *Bell states* from $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ with only operations on the first qubit.

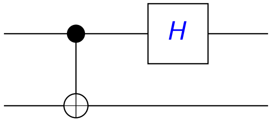
$$(X \otimes I)\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$$

$$(Z \otimes I)\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$

$$((ZX) \otimes I)\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

Superdense Coding 3

Once he has both qubits, *Bob* can convert back to the computational basis using the circuit.



After this, a measurement in the computational basis yields the two bits that *Alice* intended to convey.

Quantum Teleportation

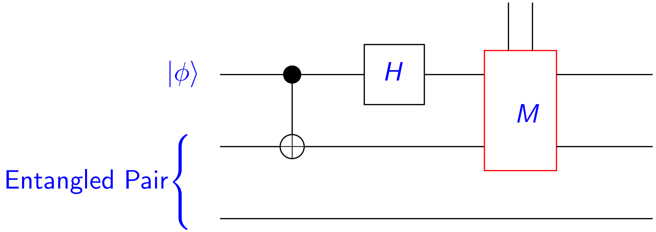
The *superdense coding* protocol allows *Alice* to send *Bob* two classical bits by transmitting a single qubit, *provided they already share an entangled pair*.

Conversely, the *quantum teleportation* protocol allows *Alice* to send *Bob* a qubit, by sending just *two classical bits* along a classical channel, *provided they already share an entangled pair*.

Contrast this with the *no-cloning theorem*, which tells us that we cannot make a copy of a qubit.

Quantum Teleportation 2

Alice has a state $|\phi\rangle$ that she wishes to transmit to *Bob*. The two already share a pair of qubits in state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.



Quantum Teleportation 3

Alice conveys to *Bob* the result of her measurement. Say the qubit in Bob's possession is in state $|\theta\rangle$, then:

- If *Alice* measures $|00\rangle$, then $|\phi\rangle = |\theta\rangle$.
- If *Alice* measures $|01\rangle$, then $|\phi\rangle = X|\theta\rangle$.
- If *Alice* measures $|10\rangle$, then $|\phi\rangle = Z|\theta\rangle$.
- If *Alice* measures $|11\rangle$, then $|\phi\rangle = XZ|\theta\rangle$.

Thus, *Bob* performs the appropriate operation and now has a qubit whose state is exactly $|\phi\rangle$.

Quantum Computing
Lecture 6

Quantum Searching

Anuj Dawar

Search Problems

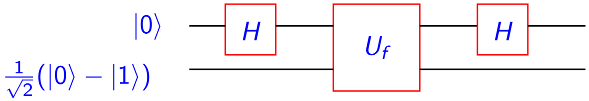
One of the two most important algorithms in quantum computing is *Grover's search algorithm*—first presented by Lov Grover in 1996.

This provides a means of searching for a particular value in an *unstructured search space*.
Compare

- searching for a name in a telephone directory
- searching for a phone number in a telephone directory

Given a black box which can take any of N inputs, and for each of them gives a yes/no answer, Grover's algorithm allows us to find the unique value for which the answer is yes in $O(\sqrt{N})$ steps (with high probability).

Deutsch-Jozsa Algorithm revisited

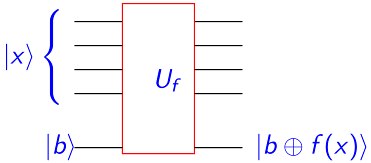


When the lower input to U_f is $|0\rangle - |1\rangle$, we can regard this as unchanged, and instead see U_f as shifting the phase of the upper qubit by $(-1)^{f(x)}$.

Oracle

Suppose we have $f : N \to \{0, 1\}$, and that $N = 2^n$, so we can think of f as operating on n bits.

We assume that we are provided a *black box* or *oracle* U_f for computing f, in the following sense:



Grover's Algorithm

Suppose further that there is exactly one n-bit value a such that

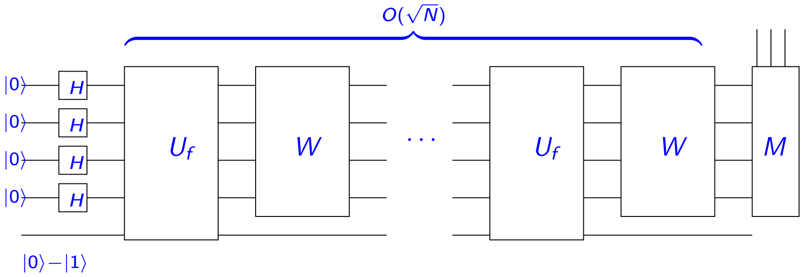
$$f(a) = 1$$

and for all other values x,

$$f(x) = 0.$$

Grover's algorithm gives us a way of using the black box U_f to determine the value a with $O(\sqrt{N}) = O(2^{n/2})$ calls to U_f.

Grover's Algorithm Schematic



The operator $G = (W \otimes I)U_f$ is known as the *Grover Iterate* (we will see soon what W is).

The input to the last bit is $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

The Action of U_f

As the "*output qubit*" is $|0\rangle - |1\rangle$, it remains unaffected by the action of U_f, which we can think of instead as a conditional phase change on the n input qubits.

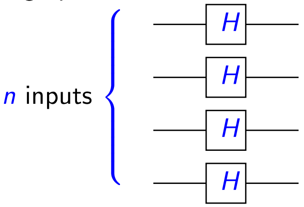
$$\begin{aligned} |a\rangle &\mapsto -|a\rangle \\ |x\rangle &\mapsto |x\rangle \qquad \text{for any } x \neq a \end{aligned}$$

We will ignore the output bit completely and instead talk of the n-bit operator V above.
Note: $V = I - 2|a\rangle\langle a|$.

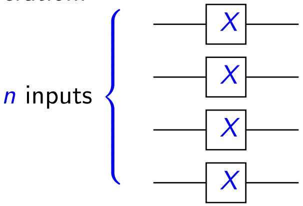
We now analyse the Grover iterate WV.

Components of W

We write $H^{\otimes n}$ for the following operation:



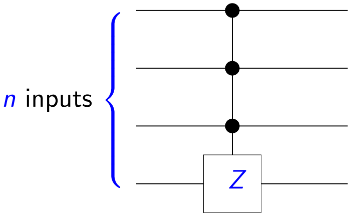
And $X^{\otimes n}$ for the following operation:



Each of these can, of course, be implemented by a series of n 1-qubit operations.

More Components of W

We write $cZ^{\otimes n}$ for the n-bit controlled-Z gate:



$$cZ^{\otimes n} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \end{bmatrix}$$

$cZ^{\otimes n}$ can be implemented using $O(n)$ cZ and *Toffoli* gates, using some workspace qubits (*Exercise*).

Defining W

Now, we can define W by:

$$\begin{aligned} W &= (-1)H^{\otimes n}(X^{\otimes n} cZ^{\otimes n} X^{\otimes n})H^{\otimes n}. \\ &= (-1)H^{\otimes n}(I - 2|0^n\rangle\langle 0^n|)H^{\otimes n} \end{aligned}$$

Write $|\Psi\rangle$ for the state

$$H^{\otimes n}|0^n\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle.$$

So, $W = (-1)(I - 2|\Psi\rangle\langle\Psi|)$, i.e.

$$W = 2|\Psi\rangle\langle\Psi| - I.$$

The Grover Iterate

Since $G = WV$, we have

$$G = (2|\Psi\rangle\langle\Psi| - I)(I - 2|a\rangle\langle a|).$$

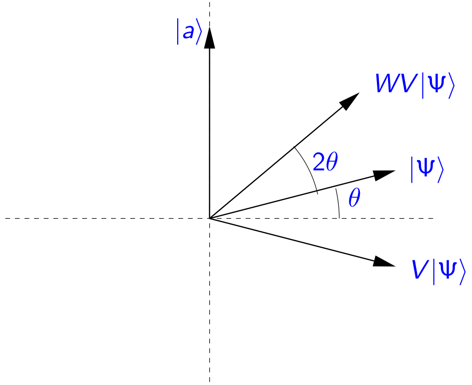
Consider the actions of W and V on the two states $|\Psi\rangle$ and $|a\rangle$.

$$\begin{aligned} W|\Psi\rangle &= |\Psi\rangle & \qquad W|a\rangle &= \tfrac{2}{\sqrt{N}}|\Psi\rangle - |a\rangle. \\ V|\Psi\rangle &= |\Psi\rangle - \tfrac{2}{\sqrt{N}}|a\rangle & \qquad V|a\rangle &= -|a\rangle \end{aligned}$$

Thus, as we start the algorithm in state $|\Psi\rangle$, the result of repeated applications of V and W will always give a *real* linear combination of $|a\rangle$ and $|\Psi\rangle$.

Geometric View

We can picture the action of W and V in the two-dimensional real plane spanned by the vectors $|a\rangle$ and $|\Psi\rangle$.



V is a *reflection* about the line perpendicular to $|a\rangle$.
W is a *reflection* about $|\Psi\rangle$.
The composition of two reflections of the plane is always a *rotation*.

The Rotation

It is clear from the picture that WV (the Grover iterate) is a rotation through an angle 2θ in the direction from $|\Psi\rangle$ to $|a\rangle$, where the angle between $|\Psi\rangle$ and $|a\rangle$ is $\frac{\pi}{2} - \theta$.

$|\Psi\rangle$ and $|a\rangle$ are *nearly orthogonal*, so θ is small (if N is large).

$$\sin\theta = \cos(\frac{\pi}{2} - \theta) = \langle a|\Psi\rangle = \frac{1}{\sqrt{N}} = \frac{1}{2^{n/2}}.$$

So,

$$\theta \sim \frac{1}{\sqrt{N}} = \frac{1}{2^{n/2}}$$

for large enough values of N.

Number of Iterations

After $t \sim \frac{\pi/2}{2\theta} \sim \frac{\pi}{4}\sqrt{N}$ iterations of the Grover iterate $G = WV$, the state of the system

$$G^t|\Psi\rangle$$

is within an angle θ of $|a\rangle$.

A measurement at this stage yields the state $|a\rangle$ with probability

$$|\langle G^t\Psi|a\rangle|^2 \geq (\cos\theta)^2 = 1 - (\sin\theta)^2 = \frac{N-1}{N}.$$

Note: Further iterations beyond t will *reduce* the probability of finding $|a\rangle$.

Multiple Solutions

Grover's algorithm works even if the solution $|a\rangle$ is not unique.

Suppose there is a set of solutions $S \subseteq \{0, \ldots, N-1\}$ and let $M = |S|$ be the number of solutions.
The Grover iterate is then a rotation in the space spanned by the two vectors

$$|\Psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \qquad |S\rangle = \frac{1}{\sqrt{M}} \sum_{j \in S} |j\rangle$$

As the angle between these is smaller, the number of iterations drops, but so does the probability of success.

Lower Bound

For classical algorithms, searching an unstructured space of solutions (such as given by a black box for f), it is easy to show a $\Omega(N)$ lower bound on the number of calls to the black box required to identify the unique solution.

Grover's algorithm demonstrates that a quantum algorithm can beat *any* classical algorithm for the problem.

It is possible to show a $\Omega(\sqrt{N})$ lower bound for the number of calls to U_f by *any* quantum algorithm that identifies a unique solution.

Grover's algorithm does not allow quantum computers to solve NP-complete problems in polynomial time.

Quantum Computing
Lecture 7

Quantum Factoring

Anuj Dawar

Quantum Factoring

A *polynomial time* quantum algorithm for factoring numbers was published by *Peter Shor* in 1994.

> *polynomial time here means that the number of gates is bounded by a polynomial in the number of bits n of the number being factored.*

The best known classical algorithms are exponential (in $n^{1/3}$).

Fast factoring would undermine public-key cryptographic systems such as RSA.

Period Finding

Suppose we are given a function $f : \mathbb{N} \to \{0, \ldots, N-1\}$ which we know is periodic, i.e.

$$f(x + r) = f(x) \quad \text{for some fixed } r \text{ and all } x.$$

Can we find the least value of r?

If we can find the period of a function efficiently, we can factor integers quickly.

Order Finding

Suppose we are given an integer N and an a with $a < N$ and

$$\gcd(a, N) = 1.$$

Consider the function $f_a : \mathbb{N} \to \{0, \ldots, N-1\}$ given by

$$f_a(x) \equiv a^x \pmod{N}$$

Then, f_a is periodic, and if we can find the period r, we can factor N.

Factoring

Suppose (for simplicity) $N = pq$, where p and q are prime. And, for some $a < N$, we know the period r of the function f_a.
Then, $a^{r+1} \equiv a \pmod{N}$, so $a^r \equiv 1 \pmod{N}$.

If r is even and $a^{r/2} + 1 \not\equiv 0 \pmod{N}$, then take $x^2 = a^r$.

$$\begin{array}{rcll} x^2 - 1 & \equiv & 0 & \pmod{N} \\ (x-1)(x+1) & \equiv & 0 & \pmod{N} \end{array}$$

But,

$$\begin{array}{rcll} x - 1 & \not\equiv & 0 & \pmod{N} \quad \text{(by minimality of } r\text{)} \\ x + 1 & \not\equiv & 0 & \pmod{N} \quad \text{(by assumption)} \end{array}$$

Factoring–*contd.*

So, $(x-1)(x+1) = kpq$ for some k.
Now, finding $\gcd(N, x-1)$ and $\gcd(N, x+1)$ will find p and q.

If we *randomly* choose $a < N$

> *(and check that* $\gcd(a, N) = 1$*—if not, we've already found a factor of* N*)*

then, there is a probability $> \frac{1}{2}$ that

- the period of f_a is even; and
- $a^{r/2} + 1 \not\equiv 0 \pmod{N}$.

Using a Fourier Transform

A fast period-finding algorithm allows us to factor numbers quickly.

The idea is to use a *Fourier Transform* to find the period of a function f.

Note, classically, we can use the *fast Fourier transform* algorithm for this purpose, but it can be shown that this would require time $N \log N$, which is exponential in the number of bits of N.

Discrete Fourier Transform

The *discrete Fourier transform* of a sequence of complex numbers

$$x_0, \dots, x_{M-1}$$

is another sequence of numbers

$$y_0, \dots, y_{M-1}$$

where

$$y_k = \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} x_j e^{2\pi ijk/M}$$

or

$$y_k = \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} x_j \omega^{jk}.$$

where $\omega = e^{2\pi i/M}$.

DFT is Unitary

The *discrete Fourier transform* is a *unitary* operation on $\mathbb{C}^M$. Writing ω for $e^{2\pi i/M}$,

$$\omega, \omega^2, \ldots, \omega^{M-1}, \omega^M = 1$$

are the Mth roots of 1.

$$D = \frac{1}{\sqrt{M}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{-2} \\ 1 & \omega^3 & \omega^6 & \cdots & \omega^{-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{M-1} & \omega^{2M-2} & \cdots & \omega \end{bmatrix}$$

Inverse DFT

The inverse of the discrete Fourier Transform is given by:

$$D^{-1} = \frac{1}{\sqrt{M}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \cdots & \omega \\ 1 & \omega^{-2} & \omega^{-4} & \cdots & \omega^{2} \\ 1 & \omega^{-3} & \omega^{-6} & \cdots & \omega^{3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{1-M} & \omega^{2-2M} & \cdots & \omega^{M-1} \end{bmatrix}$$

Exercise: Verify that D is unitary. Verify that D^{-1} as given above is the inverse of D.

Quantum Fourier Transform

Computing the discrete Fourier transform classically takes time *polynomial in M*.

Shor showed that D can be implemented using a number of one and two-qubit gates that is only polynomial in the number of qubits $O((\log M)^2)$.

Note: This *does not* give a fast way to compute the DFT on a quantum computer.
There is no way to extract all the complex components from the transformed state.

Fourier Transform on Binary Strings

Suppose $M = 2^n$, and let $|x\rangle$ be a computational basis state in $\mathbb{C}^M$ with binary representation $b_1 \cdots b_n$.
Let

$$\eta_j = e^{2\pi i (0 \cdot b_j b_{j+1} \cdots b_n)}.$$

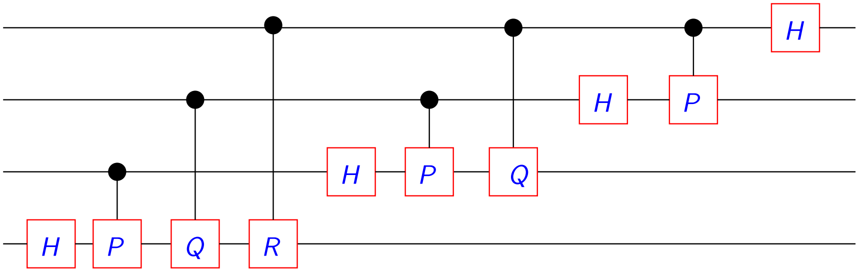
Then

$$D|x\rangle = (|0\rangle + \eta_n|1\rangle)(|0\rangle + \eta_{n-1}|1\rangle) \cdots (|0\rangle + \eta_1|1\rangle).$$

Exercise: Verify.

Quantum Fourier Transform Circuit

We can use this form to implement the *quantum Fourier transform* using Hadamard gates and conditional phase-shift gates.



In the input, the least significant bit is at the top, in the output, it is at the bottom.

Conditional Phase Shifts

Here

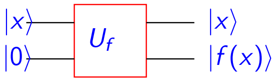
$$P = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \qquad Q = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/8} \end{bmatrix}$$

Two-qubit conditional phase shift gates are actually symmetric between the two bits, despite the asymmetry in the drawn circuit.

It seems that for large n, an n-bit quantum Fourier transform circuit would require conditional phase shifts of *arbitrary precision*.
It can be shown that this can be avoided with some (but not significant) loss in the probability of success *for the factoring algorithm*.

Preparing the State

We are given an implementation of the function f as a unitary operator U_f



Where, now, each of the two input wires represents n distinct qubits

Writing $|\Psi\rangle$ for the state

$$H^{\otimes n}|0^n\rangle = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle.$$

We have,

$$U_f|\Psi\rangle|0^n\rangle = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle|f(x)\rangle.$$

First Measurement

We measure the second n qubits of the state $U_f|\Psi\rangle|0^n\rangle$ and get a value f_0. The state after measurement is:

$$(\frac{1}{\sqrt{m}}\sum_{k=0}^{m-1}|x_0+kr\rangle)|f_0\rangle.$$

where:

x_0 is the least value such that $f(x_0)=f_0$

r is the period of the function f

$m=\lfloor\frac{2^n}{r}\rfloor$.

Applying the QFT

We apply the n-qubit quantum Fourier transform to the first n bits of the transformed state.

$$\begin{aligned}
&D\big(\tfrac{1}{\sqrt{m}}\sum_{k=0}^{m-1}|x_0+kr\rangle\big)\\
&\quad=\tfrac{1}{2^{n/2}}\sum_{y=0}^{2^n-1}\frac{1}{\sqrt{m}}\sum_{k=0}^{m-1}\omega^{(x_0+kr)y}|y\rangle\\
&\quad=\sum_{y=0}^{2^n-1}\omega^{x_0y}\frac{1}{2^{n/2}\sqrt{m}}\big(\sum_{k=0}^{m-1}\omega^{kry}\big)|y\rangle.
\end{aligned}$$

where $\omega = e^{2\pi i/2^n}$.

Second Measurement

The probability of observing a given state $|y\rangle$ is:

$$\frac{1}{2^n m}\left|\sum_{k=0}^{m-1}\omega^{kry}\right|^2 .$$

This probability function has peaks when $ry/2^n$ is close to an integer. Indeed, if $ry/2^n$ *is* an integer, then with probability 1 we measure a y that is a multiple of $r/2^n$.

Given an integer multiple of $2^n/r$, it is not difficult to find r.

Exponentiation

To complete the factoring algorithm, we need to check that we can also implement the unitary transform U_f for the particular function

$$f_a(x) = a^x \bmod N.$$

with a number of quantum gates that is polynomial in $\log N$.

This is achieved through repeated squaring.

Some Points to Note

The two measurement steps can be combined at the end, with the Fourier transform applied before the measurement of $f(x)$.

The probability of successfully finding the period in any run of the algorithm is only about 0.4.
However, this means a small number of repetitions will suffice to find the period with high probability.

Putting a *lower bound* on the conditional phase shift we are allowed to perform affects the probability of success, but not the rest of the algorithm.

Quantum Computing
Lecture 8

Quantum Automata and Complexity

Anuj Dawar

Models of Computation

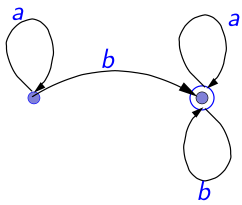
Shor's algorithm solves, in polynomial time, a problem for which no *classical, deterministic* polynomial time algorithm is known.

What class of problems are solvable by quantum machines in polynomial time?

More generally, how does *quantum parallelism* compare with other forms of *parallelism* and *nondeterminism*.

How does the quantum model of computation affect our understanding of complexity classes?

Finite State Systems



This automaton accepts the set of strings that contain at least one b.

Its operation can be described by a pair of matrices.

$$M_a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad M_b = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$$

$M_bM_bM_a$ describes the operation on states performed by reading the string abb.

DFAs and Matrices

Each DFA is specified by a collection of $n \times n$ matrices, where n is the number of states in the DFA, and there is one matrix for each letter.

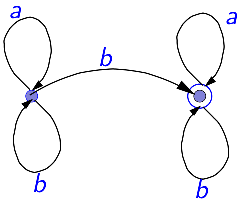
Each column of each matrix is a unit vector with $0, 1$ entries.

More generally, we can form a matrix M_w for each word w.

Multiplication of matrices corresponds to concatenation of words.

$M_w|i\rangle = |j\rangle$ if there is a path labelled w from state i to state j.

Nondeterministic Automata



This automaton accepts the same set of strings as the deterministic one.

The columns of the corresponding matrices are no longer unit vectors.

$$M_a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad M_b = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

NFAs and Matrices

$$M_{bb} = \left[\begin{array}{cc} 1 & 0 \\ 2 & 1 \end{array} \right]$$

$M_w(i, j)$ gives the *number of paths* labeled w from state i to state j.

$$M_w|i\rangle = \sum_j M_w(i, j)|j\rangle$$

Probabilistic Automata

We obtain *probabilistic automata* if we allow fractional values in M_σ.

with the proviso that each column adds up to 1.

E.g. $M_a = \begin{bmatrix} 0.5 & 0.0 \\ 0.5 & 1.0 \end{bmatrix}$ $\qquad M_b = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}$

$M_a M_b = \begin{bmatrix} 0.4 & 0.1 \\ 0.6 & 0.9 \end{bmatrix}$ gives, in position (i, j) the probability that string ba takes you from state i to state j.

Language Accepted

A probabilistic automaton $\mathcal{A}$ accepts a language L with certainty if

$$P(\mathcal{A} \text{ accepts } w) = \begin{cases} 1 \text{ if } w \in L \\ 0 \text{ if } w \notin L \end{cases}$$

$\mathcal{A}$ accepts a language L with bounded probability if there is an $\epsilon < 1/2$ such that:

$$P(\mathcal{A} \text{ accepts } w) = \begin{cases} > 1 - \epsilon & \text{if } w \in L \\ < \epsilon & \text{if } w \notin L \end{cases}$$

The class of languages accepted by probabilistic automata (under either definition) is the regular languages.

Quantum Automata

Quantum finite automata are obtained by letting the matrices M_σ have complex entries.

We also require each of the matrices to be unitary.

E.g. $$M_\sigma = \begin{bmatrix} -1 & 0 \\ 0 & i \end{bmatrix}$$

M_σ is unitary since the sum of the squares of the norms in each column adds up to 1 *and* the dot product of any two columns is 0.

NB: If all matrices only have 0 or 1 entries and the matrices are unitary (i.e., the matrices are *permutation matrices*), then the automaton is *deterministic* and *reversible*.

Acceptance Probabilities

If q is the starting state of the automaton,

$$M_w|q\rangle$$

is the state after reading w. If

$$\alpha_j = \langle j|M_w|q\rangle$$

then α_j is the *probability amplitude* that the automaton reaches state j.

> $|\alpha_j|^2$ *is the probability that a measurement will result in state* q_j.
>
> $\sum_{j\in F} |\alpha_j|^2$ *is the probability that the automaton accepts the string* w.

Language Accepted

We can define language acceptance exactly or by bounded probability.

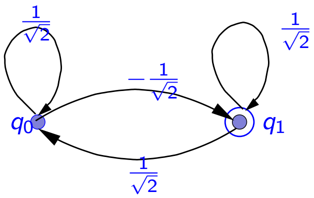
Because of the reversibility requirement, quantum automata are quite a weak model.

There are *regular* languages that cannot be accepted by a QFA.

However, QFAs may be much more *succinct* than their classical counterparts.

Interference

Consider the automaton in a one letter alphabet defined as:



$$M_a = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$M_{aa} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

While there are two distinct paths labelled aa from q_0 back to itself, and each has non-zero probability, the net probability of ending up in q_0 is 0.

The automaton accepts a string of odd length with probability 0.5 and a string of even length with probability 1 if its length is not a multiple of 4 and probability 0 otherwise.

Turing Machines

A *Turing machine*, in addition to the finite set of states in the automaton has an infinite *read-write* tape.

A machine is determined by an alphabet Σ, a finite set of states Q and a transition function δ which gives, for each state and symbol: a next state, a replacement symbol and a direction in which to move the tape head.

A machine has infinitely many possible *configurations* (reserving the word "state" for a member of Q).
Each configuration c is determined by a state, the contents of the tape (a finite string) and the position of the head.

Configurations and Computations

If c_0 is the configuration in the starting state, with w on the tape and the tape head at the left end of the string, w is accepted if the computation

$$c_0 \to c_1 \to \cdots \to c_f$$

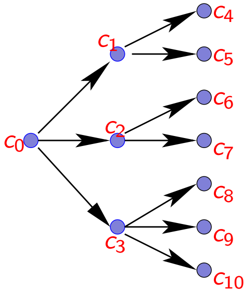
eventually reaches an accepting state.
If the length of the computation is bounded by a polynomial in the length of w, the language accepted by the machine is in P.

The action of the Turing machine can equivalently be described as a linear operator M on an *infinite-dimensional* space.
The set of configurations form a basis for the space.

Nondeterministic Turing Machines

In a *nondeterministic machine*, δ determines, for each state and symbol, a *set* of next moves.

This gives rise to a tree of configurations:



A configuration may occur in several places in the tree.

The initial string w is accepted by the machine if there is some path through the tree leading to an accepting state.

If the height of the tree is bounded by a polynomial in the length of w, then the language is in NP

Probabilistic Machines

With a *probabilistic machine*, δ defines, for each current state and symbol, a *probability distribution* over the possible next moves.

The action of the machine can be defined as an infinite matrix, where the rows and columns are configurations, and each column adds up to 1.

However, how much information can be encoded in a single entry?
We require the entries α to be *feasibly computable*. That is, there is a feasibly computable f such that:

$$|f(n) - \alpha| < 2^{-n}$$

BPP

BPP is the collection of languages L for which there is a probabilistic machine M, running in polynomial time with:

$$P(M \text{ accepts } w) = \begin{cases} > \frac{2}{3} & \text{if } w \in L \\ < \frac{1}{3} & \text{if } w \notin L \end{cases}$$

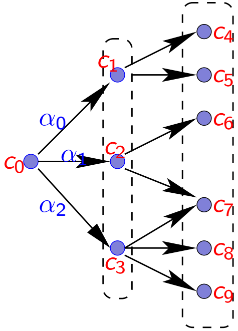
The class of languages is unchanged if we replace $\frac{2}{3}$ and $\frac{1}{3}$ by $1 - \epsilon$ and ϵ, for any $\epsilon < \frac{1}{2}$, or indeed the set of all feasibly computable probabilities with $\{0, \frac{1}{2}, 1\}$.

The only inclusion relations we know are P⊆NP and P⊆BPP.

Primality testing, long known to be in NP and in BPP was shown in 2002 to be in P.

Quantum Turing Machines

With a *Quantum Turing machine*, δ associates with each state and symbol, and each possible next move, a *complex probability amplitude* (which we require to be a *feasible* complex number).



The machine can be seen as progressing through a series of stages, each of which is a superposition of configurations.

Note that the probability of c_7 occurring at time 2 may be less than the sum of the probabilities along the two paths.

We also require that the linear transformation defined by the machine is unitary.

BQP

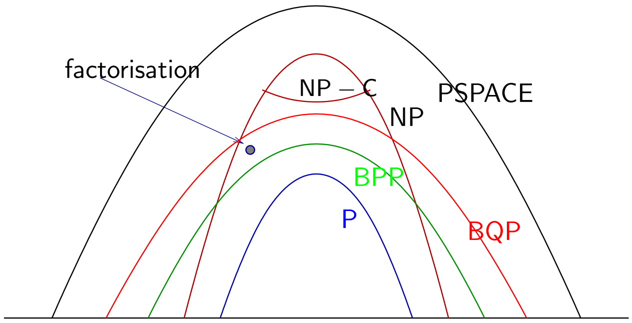
BQP is the collection of languages L recognised by a quantum Turing machine, running in polynomial time, under the bounded probability rule.

The class BQP is not changed if we restrict the set of possible amplitudes to $\{0, \pm\frac{3}{5}, \pm\frac{4}{5}, 1\}$.

$$\mathsf{BPP} \subseteq \mathsf{BQP}$$

Shor's algorithm shows that the factorisation problem is in BQP.
It is not known to be in BPP.

Complexity Classes



Inclusion relations among the complexity classes as we know them.