

Quantum Computation

Lecture 1

Sam Braunstein

Before we start:

Useful textbooks

1. **Benenti, Casati, Strini**
Principles of Quantum Computation and Information, I, II
(World Scientific, 2004)
2. **Kaye, LaFlamme, Mosca**
An Introduction to Quantum Computing (OUP, 2007)
3. **Nielsen & Chuang**
Quantum Computing and Quantum Information (CUP, 2000)

Some weblinks:

On the VLE:

Course notes, lecture slides, practicals, etc.

Complex numbers tutorial.



Matrices tutorial.

<http://www-users.cs.york.ac.uk/~schmuel/comp/comp.html>

Sam Braunstein's quantum computing tutorial.

Information is **physical**

- ▶ how we process information depends on the laws of physics.
- ▶ A computer may store bits as charge on a capacitor:

	1	$\sim 10^5$ electrons
	0	~ 0 electrons

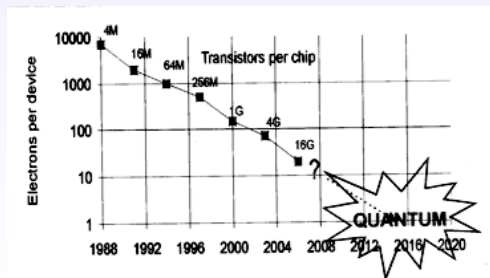
Bit has two values only: 0, 1

- ▶ A charged capacitor may use roughly 10^5 electrons to represent **logical 1**.
- ▶ An discharged capacitor represents **logical 0**.

Everything is getting smaller

Moore's Law

1. Moore (1953) "The number of transistors on a chip doubles every 18 months or so"
2. Moore (2005) "It can't continue forever... we're approaching the size of atoms".



Extrapolating this graph

Note the *logarithmic* scale on this graph.

The number of electrons used to represent a single bit *halves* every 18 months or so.

Assuming this continues ...

Sometime around 2015 - 2020

Consumer electronics will be storing bits using single electrons.

- ▶ Is this a brick wall, for Moore's law?
- ▶ Are the laws of physics the same, at this scale?

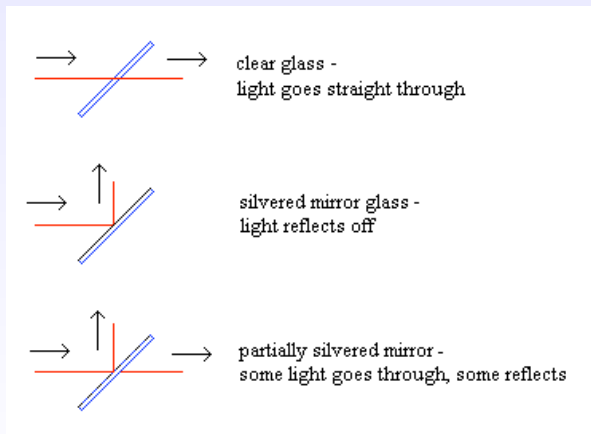
from **BITs** to **QUBITs**

At the scale of single electrons ...

- ▶ The laws of physics look very different.
- ▶ Computing at the atomic scale needs *quantum mechanics*.
- ▶ This is
 1. A challenge.
 2. A great opportunity.

The different physics that applies also allows for fantastically more efficient computation.

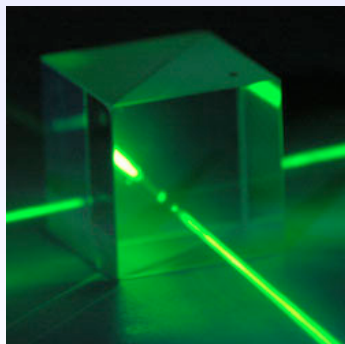
A common device is the **beamsplitter**



This is for light, but similar devices exist (for example) for electrons.

This is not just a theoretical device:

The building block of many optics experiments

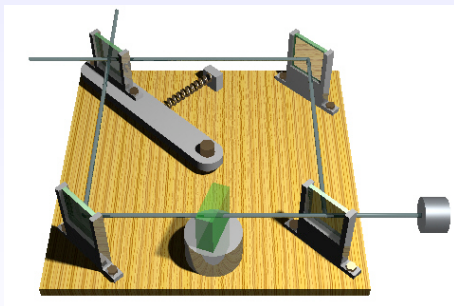


Available online at around \$6 per beamsplitter ...

- └ The quantum-mechanical world
- └ What's different ?

Combining beamsplitters

By choosing the right length of paths, we can introduce *constructive interference*.



Light is always observed at the detector shown.

Treating light as *photons*

Light is emitted and absorbed in single ‘packets’.

These are called photons.

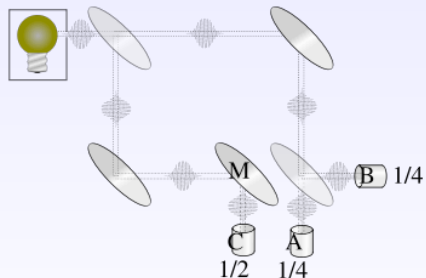
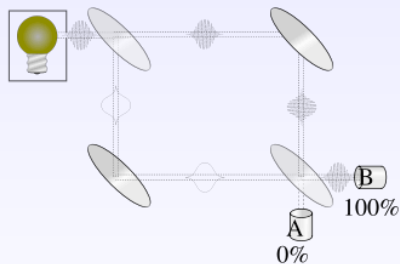
- ▶ We can treat the light as “a stream of photons”.
- ▶ ... or even do ‘single-photon’ experiments.

In this setting:

How can we tell ‘which path’ a photon travels?

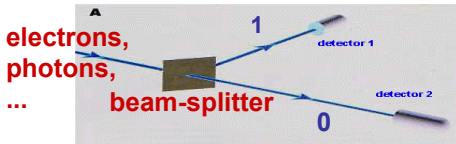
- └ The quantum-mechanical world
- └ What's different ?

Simple solution — block one path!



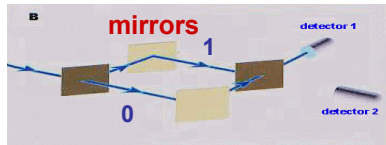
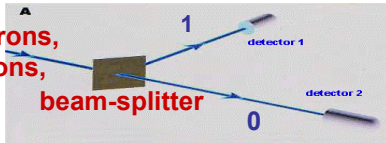
What does this tell us ??

The New Ingredient



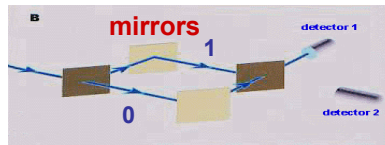
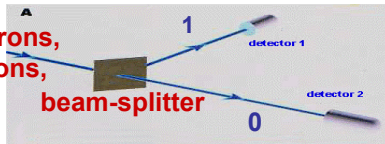
The New Ingredient

electrons,
photons,
...

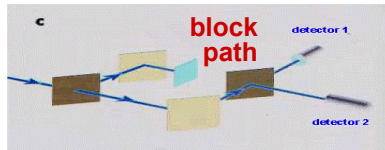


The New Ingredient

electrons,
photons,
...

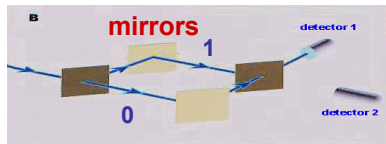
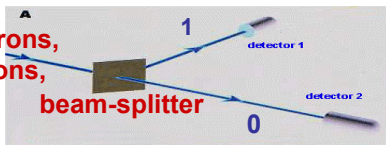


Which path is taken?

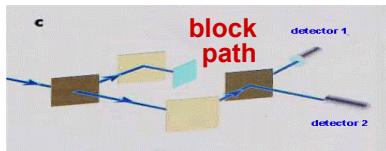


The New Ingredient

electrons,
photons,
...



Which path is taken?



BOTH

Each photon ‘explores all possible paths’

We **cannot** make sense of this by assuming:

*“at the beamsplitter, a photon (randomly)
chooses either one path or the other”*

If a photon ‘takes the upper path’

- ▶ How can blocking the lower path affect it?

The only possible answer:

Each photon somehow 'takes both paths'.

The same holds for

- ▶ electrons,
- ▶ atoms,
- ▶ any 'particle'.

Any attempt to 'find out which path is taken'
... changes the result of the experiment.

Yes, there are two paths you can go by, but in the long run, you just can't tell which one you're on!

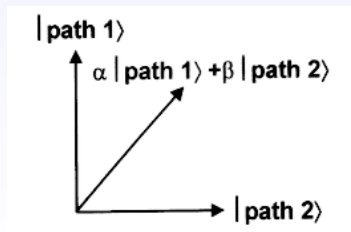
How to model this ?

Let us write the two paths as $|path_1\rangle$ and $|path_2\rangle$.

On the quantum scale, we:

- ▶ Treat these as **orthogonal vectors** in a Hilbert space.
- ▶ and so allow for **superpositions**

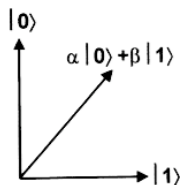
$$\alpha|path_1\rangle + \beta|path_2\rangle$$



What does this have to do with *computation*?

Treat the choice of path (or *any other 'quantum property'*) as logical 1 / 0.

Superposition
between two rays
in Hilbert space



- ▶ **Bits** have been replaced by **quantum bits**, or **qubits**.
- ▶ What are the implications of this for computing?

Real computers have more than one bit !

Two qubits are represented by 4 orthogonal vectors,

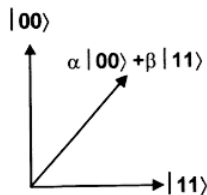
$$|00\rangle, |01\rangle, |10\rangle, |11\rangle$$

This allows superpositions such as

$$\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

Giving many 'strange phenomena':

Entanglement
between pairs of
(distant) objects



How much 'information' can qubits carry?

- ▶ For k qubits, we can form a superposition of all possible values:

$$\sum_{j=0}^{2^k-1} \alpha_j |j\rangle$$

- ▶ For k classical bits, we certainly cannot!

Simulating a quantum computer using a classical computer appears to be *very hard!*

Is this a bug, or a feature?

Given 300 qubits, we can form a superposition of 2^{300} values.

... this is more than the number of particles in the universe.

Can we use this to do 'massively powerful computing'?

- ▶ There are a number of complicating factors!
- ▶ But also, some very interesting quantum computer programs.

A final comment

The strong Church-Turing thesis

“A (probabilistic) Turing machine can efficiently simulate any physically reasonable computer.”

If *quantum computers* cannot be simulated efficiently by *classical computers*, this needs to be modified.

Quantum Turing machines, anyone ??.

Quantum Computation

Lecture 2

Sam Braunstein

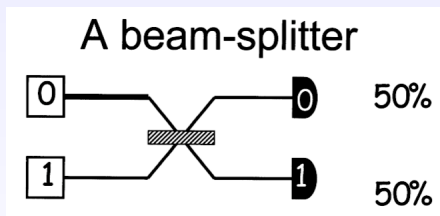
Beamsplitters revisited



We will:


1. Draw ‘circuit-like’ diagrams, from left to right.
2. Treat the upper and lower paths as ‘logical 0 / 1’.

A Schematic:



- ▶ A particle *definitely* in the upper branch is labelled $|0\rangle$.
- ▶ A particle *definitely* in the lower branch is labelled $|1\rangle$.

▶ **preparation** is shown by a thick line.

▶ **measurement** is shown as 

The action of the beamsplitter

Let us introduce particles into the upper branch ...

What do we observe, as output?

The output is *randomly* in either the upper, or lower, branch

Introducing particles into the lower branch ...

What do we observe, as output?

Exactly the same output. The output is
equally likely to be in either branch

Is this simply a random choice?



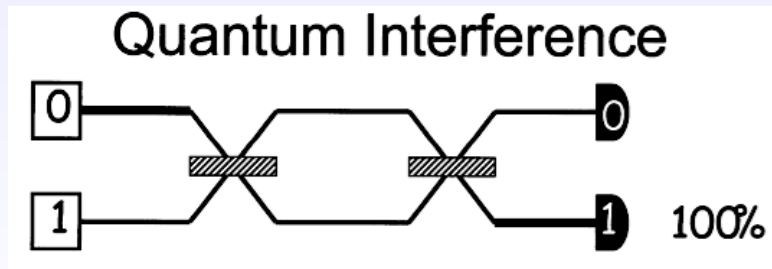
The simplest explanation

“The beamsplitter acts as a classical coin-flip, randomly sending the photon one way or the other”

The simplest explanation is, of course, wrong!

Why this is not random choice

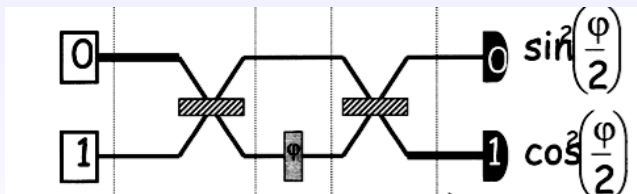
Add in a second beamsplitter



The 'random choice' explanation would predict a 50 / 50 distribution.

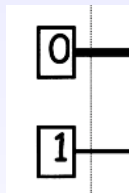
A new theory:

The particle can exist in a *complex superposition* of the two paths

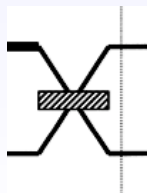


How to analyse this experiment?

One step at a time:



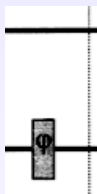
A particle is introduced in the upper path:
It is in state $|0\rangle$.



After passing through the beamsplitter:
It is in an *equal superposition* $\frac{i}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$.

Note: the reflected part picks up a complex factor i .

One step at a time (continued) :



The 'delay' is applied to the bottom branch only:

Resulting in the state $\frac{i}{\sqrt{2}}|0\rangle + \frac{e^{i\varphi}}{\sqrt{2}}|1\rangle$.

Again passing through the beamsplitter:

Remember: a factor of i is applied to the reflected path.



$$\frac{i}{\sqrt{2}}|0\rangle + \frac{e^{i\varphi}}{\sqrt{2}}|1\rangle$$

\mapsto

$$\frac{i}{\sqrt{2}} \left(\frac{i}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \right) + \frac{e^{i\varphi}}{\sqrt{2}} \left(\frac{i}{\sqrt{2}}|1\rangle + \frac{1}{\sqrt{2}}|0\rangle \right)$$

Rearranging and simplifying

Final state is:

$$\begin{aligned}
 & \frac{i}{\sqrt{2}} \left(\frac{i}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle \right) + \frac{e^{i\varphi}}{\sqrt{2}} \left(\frac{i}{\sqrt{2}} |1\rangle + \frac{1}{\sqrt{2}} |0\rangle \right) \\
 &= \frac{(e^{i\varphi} - 1)}{2} |0\rangle + \frac{i(e^{i\varphi} + 1)}{2} |1\rangle \\
 &= e^{i\frac{\varphi}{2}} \left(\frac{e^{i\frac{\varphi}{2}} - e^{-i\frac{\varphi}{2}}}{2} \right) |0\rangle + ie^{i\frac{\varphi}{2}} \left(\frac{e^{i\frac{\varphi}{2}} + e^{-i\frac{\varphi}{2}}}{2} \right) |1\rangle
 \end{aligned}$$

Further rearranging:

Basic trigonometric identities:

$$1. \cos(\theta) = \frac{1}{2}(e^{i\theta} + e^{-i\theta})$$

$$2. \sin(\theta) = \frac{-i}{2}(e^{i\theta} - e^{-i\theta})$$

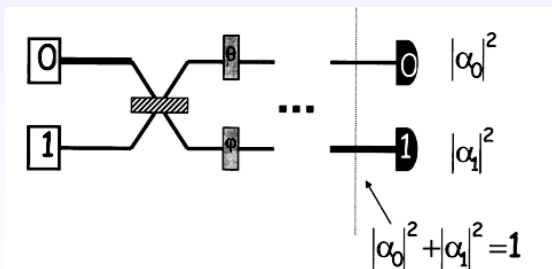
The final state is then:

$$-ie^{i\frac{\varphi}{2}} \sin\left(\frac{\varphi}{2}\right) |0\rangle + ie^{i\frac{\varphi}{2}} \cos\left(\frac{\varphi}{2}\right) |1\rangle$$

What about the measurement / probabilities ?

A general principle

The probability for finding the particle in a certain branch is the square of the *modulus* of the *weight* (or *amplitude*) of that branch.



Measurement continued

If a photon in state $\alpha_0|0\rangle + \alpha_1|1\rangle$ is **measured**, then

- ▶ The probability of observing $|0\rangle$ is $|\alpha_0|^2$
- ▶ The probability of observing $|1\rangle$ is $|\alpha_1|^2$

Probabilities sum to 1

$$|\alpha_0|^2 + |\alpha_1|^2 = 1$$

This is why the $\frac{1}{\sqrt{2}}$ weights for the beamsplitter give a 50/50 probability for either path.

Some key assumptions:

Operations are **linear**

- ▶ We know how the beamsplitter behaves on $|0\rangle$,

$$|0\rangle \mapsto \frac{i}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

- ▶ We know how the beamsplitter behaves on $|1\rangle$,

$$|1\rangle \mapsto \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle$$

Therefore, we know how it behaves on $\alpha|0\rangle + \beta|1\rangle$.

Linearity continued ...

In general, L is linear when: $L(\alpha|\mathbf{x}\rangle + \beta|\mathbf{y}\rangle) = \alpha.L(|\mathbf{x}\rangle) + \beta.L(|\mathbf{y}\rangle)$

For the beamsplitter:

$$\begin{aligned} \alpha|0\rangle + \beta|1\rangle &\longrightarrow \alpha\left(\frac{i}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) + \beta\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle\right) \\ &= \left(\alpha\frac{i}{\sqrt{2}} + \beta\frac{1}{\sqrt{2}}\right)|0\rangle + \left(\alpha\frac{1}{\sqrt{2}} + \beta\frac{i}{\sqrt{2}}\right)|1\rangle \\ &= \alpha'|0\rangle + \beta'|1\rangle \end{aligned}$$

Not only *linear* but also *unitary*

The beamsplitter maps $\alpha|0\rangle + \beta|1\rangle$ to $\alpha'|0\rangle + \beta'|1\rangle$.

From the interpretation as probabilities

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

We need (and can check this is true) that:

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

Any linear operation $\alpha|0\rangle + \beta|1\rangle \xrightarrow{L} \alpha'|0\rangle + \beta'|1\rangle$ satisfying this condition is called **unitary**.

Using matrix notation

Let us

- ▶ Denote $|0\rangle$ and $|1\rangle$ by the vectors $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$.
- ▶ The superposition $\alpha|0\rangle + \beta|1\rangle$ is then

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

- ▶ Arbitrary linear operations are matrices $U = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}$,

so

$$\begin{pmatrix} \alpha' \\ \beta' \end{pmatrix} = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

A useful operation

A very important operation is the **conjugate-transpose**, denoted U^\dagger .

$$\begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}^\dagger = \begin{pmatrix} u_{00}^* & u_{10}^* \\ u_{01}^* & u_{11}^* \end{pmatrix}$$

(Where z^* is the complex conjugate).

In particular, for a vector (e.g. a single qubit),

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}^\dagger = (\alpha^* \quad \beta^*)$$

The 'probabilities' condition

Given a qubit $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$, the interpretation as probabilities states

$$|\alpha|^2 + |\beta|^2 = 1 = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}^\dagger \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = 1$$

Given a physical operation $\begin{pmatrix} \alpha' \\ \beta' \end{pmatrix} = U \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$, we also require

$$\begin{pmatrix} \alpha' \\ \beta' \end{pmatrix}^\dagger \begin{pmatrix} \alpha' \\ \beta' \end{pmatrix} = 1$$

Some simple manipulation:

$$\begin{pmatrix} \alpha' \\ \beta' \end{pmatrix}^\dagger = \left[U \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \right]^\dagger = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}^\dagger U^\dagger = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}^\dagger \begin{pmatrix} u_{00}^* & u_{10}^* \\ u_{01}^* & u_{11}^* \end{pmatrix}$$

and we already know that

$$\begin{pmatrix} \alpha' \\ \beta' \end{pmatrix} = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

Therefore,

$$\begin{pmatrix} \alpha' \\ \beta' \end{pmatrix}^\dagger \begin{pmatrix} \alpha' \\ \beta' \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}^\dagger U^\dagger U \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

Unitarity via matrices

For an operation U to be unitary, we need

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}^\dagger U^\dagger U \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = 1$$

for all $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}^\dagger \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = 1$.

A characterisation of unitarity

This states that $U^\dagger U$ is the *identity*.

$$\begin{pmatrix} u_{00}^* & u_{10}^* \\ u_{01}^* & u_{11}^* \end{pmatrix} \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

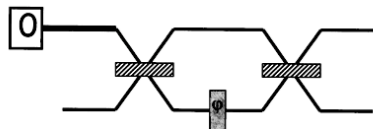
The beamsplitter experiment

The beamsplitter gives a unitary matrix: $\frac{1}{\sqrt{2}} \begin{pmatrix} i & 1 \\ 1 & i \end{pmatrix}$.

The phase delay gives another unitary: $\begin{pmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{pmatrix}$.

The experiment shown then becomes:

$$\begin{pmatrix} \alpha' \\ \beta' \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} i & 1 \\ 1 & i \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} i & 1 \\ 1 & i \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$



corresponds to

$$\begin{pmatrix} \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \begin{pmatrix} \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Important:

- ▶ The **circuit** reads from *left to right*.
- ▶ The **maths** reads from *right to left*.

Question: Why is this?

From quantum optics to quantum circuits

So far ...

We have represented a qubit by *two lines*.

A more compact notation

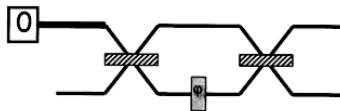
We will now ...

Represent a qubit by a *single line*

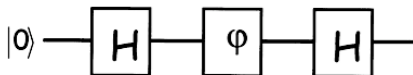
$$\alpha|0\rangle + \beta|1\rangle \longrightarrow$$

The 'quantum circuit' formalism

An arrangement like



is represented with a network like*



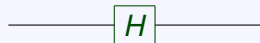
(*beamsplitters and Hadamard gates are slightly different)

Some quantum 'logic gates'

Another way of creating 'equal superpositions'.

The **Hadamard** gate

This represents the matrix



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

This is more common in QC than the beamsplitter.

Another QC logic gate

We have also seen the ‘phase delay’,

This more commonly called the
phase gate

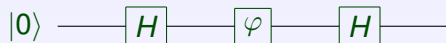


This represents the matrix

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{pmatrix}$$

Composing QC logic gates

An arrangement like¹ the interferometer becomes



This is shorthand for

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Remember!

- ▶ *Circuits* read from left to right.
- ▶ *Equations* read from right to left.

¹ using the Hadamard gate, rather than the beamsplitter

What happens with more than one qubit ?

Consider a 2-qubit system:

- ▶ First qubit in state $\alpha_0|0\rangle + \alpha_1|1\rangle$
- ▶ Second qubit in state $\beta_0|0\rangle + \beta_1|1\rangle$

A 2-qubit system ... has 4 basis states

These are:

$$|0\rangle|0\rangle = |00\rangle \quad |0\rangle|1\rangle = |01\rangle$$

$$|1\rangle|0\rangle = |10\rangle \quad |1\rangle|1\rangle = |11\rangle$$

representing multi-qubit systems (I)

The 2-qubit system:

- ▶ First qubit in state $\alpha_0|0\rangle + \alpha_1|1\rangle$
- ▶ Second qubit in state $\beta_0|0\rangle + \beta_1|1\rangle$

has the state $(\alpha_0|0\rangle + \alpha_1|1\rangle)(\beta_0|0\rangle + \beta_1|1\rangle) =$

$$\alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle$$

representing multi-qubit systems (II)

In matrix notation:

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \otimes \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{pmatrix} = \begin{pmatrix} \alpha_0 \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \\ \alpha_1 \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \end{pmatrix}$$

General superpositions

In general, a 2 qubit system can be in *any* superposition:

$$\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

satisfying

$$\sum_{jk} |\alpha_{jk}|^2 = 1$$

Measuring 2-qubit systems

when measuring both qubits:

The probability of observing $|xy\rangle$ is exactly $|\alpha_{xy}|^2$.

... and finally!

Not all 2 qubit states

$$\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

can be written as the tensor product of single qubits.

Such states are called **entangled**.

Quantum Computation

Lecture 3

Sam Braunstein

Tensors of matrices

Consider a 2-qubit system:

- ▶ First qubit in state $\alpha_0|0\rangle + \alpha_1|1\rangle$
- ▶ Second qubit in state $\beta_0|0\rangle + \beta_1|1\rangle$

The *compound system* has the following state:

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \otimes \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{pmatrix} = \begin{pmatrix} \alpha_0 \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \\ \alpha_1 \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \end{pmatrix}$$

Applying operations to 2 qubit systems

We can apply separate unitaries to each qubit:

$$\alpha_0 |0\rangle + \alpha_1 |1\rangle \longrightarrow \boxed{A} \longrightarrow \alpha'_0 |0\rangle + \alpha'_1 |1\rangle$$

$$\beta_0 |0\rangle + \beta_1 |1\rangle \longrightarrow \boxed{B} \longrightarrow \beta'_0 |0\rangle + \beta'_1 |1\rangle$$

The (general) tensor product

$$A \otimes B = \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} \otimes \begin{pmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{pmatrix} = \begin{pmatrix} a_{00}B & a_{01}B \\ a_{10}B & a_{11}B \end{pmatrix}$$

The general tensor product

$$A \otimes B = \begin{pmatrix} a_{00} \begin{pmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{pmatrix} & a_{01} \begin{pmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{pmatrix} \\ a_{10} \begin{pmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{pmatrix} & a_{11} \begin{pmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{pmatrix} \end{pmatrix}$$

In full:

$$A \otimes B = \begin{pmatrix} a_{00}b_{00} & a_{00}b_{01} & a_{01}b_{00} & a_{01}b_{01} \\ a_{00}b_{10} & a_{00}b_{11} & a_{01}b_{10} & a_{01}b_{11} \\ a_{10}b_{00} & a_{10}b_{01} & a_{11}b_{00} & a_{11}b_{01} \\ a_{10}b_{10} & a_{10}b_{11} & a_{11}b_{10} & a_{11}b_{11} \end{pmatrix}$$

An important property

The following circuits are equivalent:



Mathematically: $(Av) \otimes (Bw) = (A \otimes B)(v \otimes w)$.

As matrices:

$$\begin{aligned} & \left[\begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \end{pmatrix} \right] \otimes \left[\begin{pmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \right] \\ &= \left[\begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} \otimes \begin{pmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{pmatrix} \right] \left[\begin{pmatrix} v_0 \\ v_1 \end{pmatrix} \otimes \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \right] \end{aligned}$$

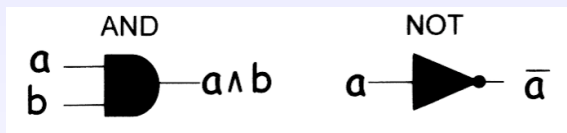
When qubits interact ...

We want QM logic gates with more than one input!

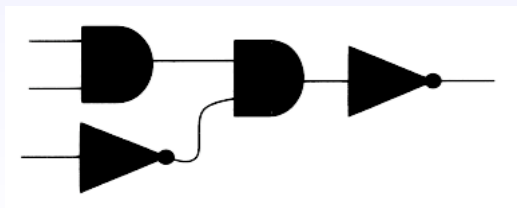
- ▶ We wish to see interaction between qubits
- ▶ ... similar to classical circuit diagrams.

(Classical) logic gates

A logic gate is a function from n bits to m bits, such as



Logic gates may be *glued together* to create circuits:



These compute Boolean functions.

Universal sets of gates

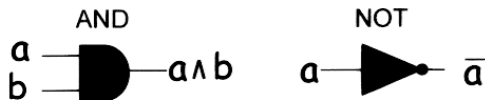
A set B of logic gates is **universal** when:

“For any Boolean function F , there is a circuit made up of gates from B , that computes F ”.

The question:

How to find sets of logic gates that are universal?

Universal, and non-universal sets of gates



E.g., $B = \{ \text{NOT} \}$ is not universal

E.g., $B = \{ \text{AND} \}$ is not universal

E.g., $B = \{ \text{NOT}, \text{AND} \}$ is universal

Translating between circuits

- ▶ Let A be any set of logic gates.
- ▶ Let B be a universal set of logic gates.

Any circuit made up of gates from A ...

... can be efficiently translated into
a circuit made up of gates from B .

How to do this ??

Translating between circuits (cont.)

How to do this ...

- ▶ Each gate in A can be realised as a circuit in B .
- ▶ Use these circuits, instead of gates in A , to construct the original circuit.

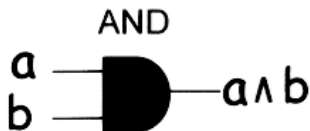
This process is *efficient*.

Reversibility and Unitarity

Quantum processes are *reversible*

- ▶ Apart from measurement, QM processes are *unitary*.
- ▶ This implies *reversibility*. A unitary U has an inverse, U^\dagger .

Classical logic gates are certainly not reversible!



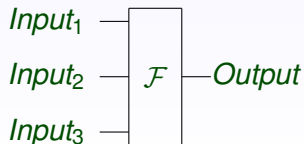
Given $a \wedge b$, we cannot find the values a and b separately.

From *irreversible* to *reversible* computation

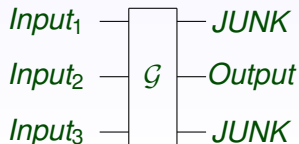
Landauer, Fredkin, & Toffoli showed how to:
Model irreversible circuits with reversible ones.

Each logic gate is replaced by a reversible one, where some output is simply thrown away.

Irreversible Circuit

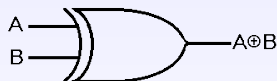


Reversible Version



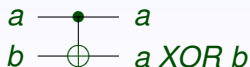
Reversible versions of gates

The exclusive-or gate:



Input		Output
0	0	0 = 0 ⊕ 0
0	1	1 = 0 ⊕ 1
1	0	1 = 1 ⊕ 0
1	1	0 = 1 ⊕ 1

The 'controlled-not' gate



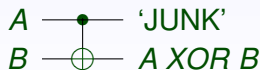
Input		Output	
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

The inverse of the CNOT gate

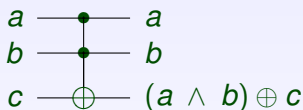
Composing two CNOT gates



The CNOT gate is its own inverse
 ... hence the symmetric notation.

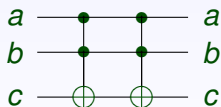


The 'controlled-controlled-not' gate



“Bit c is flipped when both a and b are 1.”

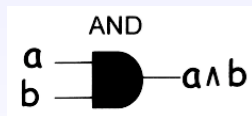
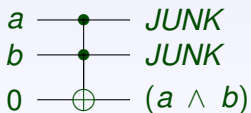
This gate is again its own inverse:



Hence the symmetric notation!

Simulating the *AND* gate

The *CNOT* gate can simulate the *AND* gate:



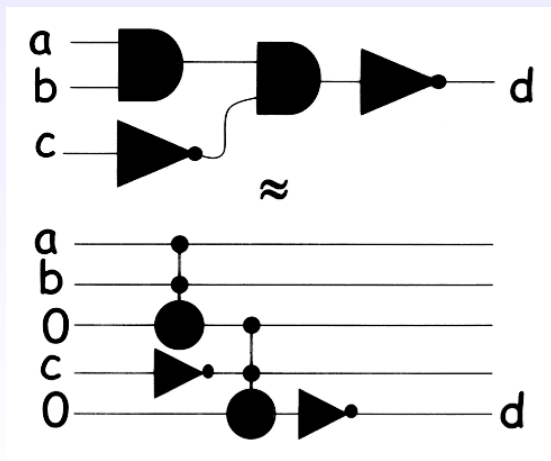
We simply ensure that the third input bit is 0 .

A universal reversible gate set

Recall

- ▶ The *NOT* gate is logically reversible.
- ▶ The *AND* gate can be simulated by the (reversible) *CCNOT* gate.
- ▶ The set $\{\textit{AND}, \textit{NOT}\}$ is a universal gate set.

A simple example



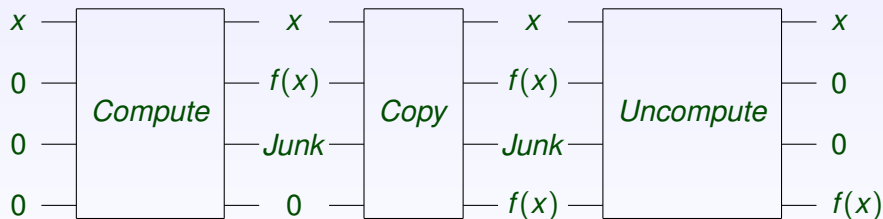
What about all that junk ??

- ▶ Do we need a couple of extra bits for each time a logic gate is applied??
- ▶ If so ... how many logic gates are applied each second in a 1.2GHz computer??
- ▶ Classical computers dispose of this junk as *heat*.

Garbage collection in reversible computing

Bennett showed how to get rid of junk by ‘uncomputing’.

The basic idea:



This leaves the *input*, the *output*, and *no junk* !

The complexity of reversible simulation

An irreversible circuit with

- ▶ Space S
- ▶ Time T

can thus be simulated by a reversible circuit with

- ▶ Space $O(S + T)$
- ▶ Time $O(T)$

Other complexity results

Bennett later showed how an irreversible circuit with

- ▶ Space S
- ▶ Time T

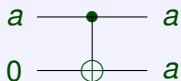
can be simulated by a reversible circuit with

- ▶ Space $O(S \log(T))$
- ▶ Time $O(T^{1+\epsilon})$

for some constant $\epsilon > 0$.

Copying using the CNOT gate

The CNOT gate can be used to copy an input:

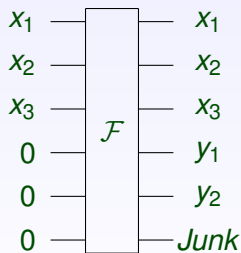


From 1 copy of a , this produces 2 copies of a .

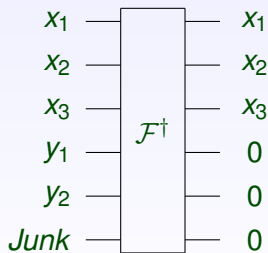
We can use this to give a circuit for Bennett's garbage collection:

Bennett's garbage collection

Consider a circuit, made from reversible components:

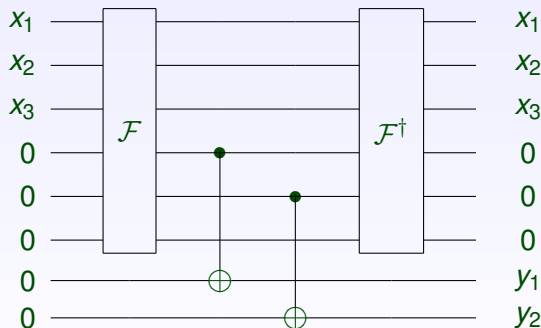


Take the *Mirror Image*, to get another circuit:



Bennett's garbage collection (II)

Now compose the two, with the 'fan-out' step in the middle:



Given any Boolean circuit, we can design an efficient reversible version.

Quantum Computation

Lecture 4

Sam Braunstein

Bras and Kets

Consider a state

$$\blacktriangleright |\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix}$$

We use $\langle\psi| = \alpha_0^*\langle 0| + \alpha_1^*\langle 1|$ to denote the adjoint

$$|\psi\rangle^\dagger = (\alpha_0^* \quad \alpha_1^*)$$

The vectors $\langle\psi|$ and $|\psi\rangle$ are called **bra** and **ket** respectively.

The Bra-Ket

Composing a Bra and a Ket

Notice that

$$\langle \psi | \cdot | \psi \rangle = \begin{pmatrix} \alpha_0^* & \alpha_1^* \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = |\alpha_0|^2 + |\alpha_1|^2 = 1$$

This only works because

$$\langle i | \cdot | j \rangle = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

The bra-ket of distinct vectors

Now consider two vectors

$$\blacktriangleright |\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix}$$

$$\blacktriangleright |\phi\rangle = \beta_0|0\rangle + \beta_1|1\rangle = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}$$

How do we interpret:

$$\langle\phi| \cdot |\psi\rangle = \begin{pmatrix} \beta_0^* & \beta_1^* \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \beta_0^* \alpha_0 + \beta_1^* \alpha_1$$

This is the “overlap” between states $|\phi\rangle$ and $|\psi\rangle$

Brackets and probabilities

Recall that ...

The prob. of finding $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ in state $|0\rangle$ is $|\alpha_0|^2$.

This is simply $|\alpha_0|^2 = |\langle 0 | \cdot |\psi\rangle|^2$

A general principle:

Given a state $|\psi\rangle$, the probability of finding it in state $|\phi\rangle$ is

$$|\langle \phi | \psi \rangle|^2$$

Note - distinct outcomes are orthogonal!

Other uses for Bras and Kets

A unitary operation $U = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix}$
may be written as a sum of bras and kets:

$$U = U_{00}|0\rangle\langle 0| + U_{01}|0\rangle\langle 1| + U_{10}|1\rangle\langle 0| + U_{11}|1\rangle\langle 1|$$

Writing this out in full: $U =$

$$U_{00} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + U_{01} \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

$$U_{10} \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} + U_{11} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

Finding matrices ...

Given

- ▶ A unitary operation U ,
- ▶ a description of U on *basis vectors* $|0\rangle$ and $|1\rangle$.

How do we write down the matrix for U ?

Matrices from basis vectors

Let U be a unitary operation on single qubits:

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

The **Computational Basis**, in matrix form, is

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

Simple calculation:

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \alpha \\ \gamma \end{pmatrix}, \quad \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \beta \\ \delta \end{pmatrix}$$

Translating into kets:

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \alpha \\ \gamma \end{pmatrix} \quad \Leftrightarrow \quad U|0\rangle = \alpha|0\rangle + \gamma|1\rangle$$

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \beta \\ \delta \end{pmatrix} \quad \Leftrightarrow \quad U|1\rangle = \beta|0\rangle + \delta|1\rangle$$

- ▶ The action on $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ gives the first column of U .
- ▶ The action on $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ gives the second column U .

The general case

Given a matrix $U = \begin{pmatrix} U_{00} & U_{01} & U_{02} & \dots & U_{0N} \\ U_{10} & U_{11} & U_{12} & \dots & U_{1N} \\ U_{20} & U_{21} & U_{22} & \dots & U_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ U_{N0} & U_{N1} & U_{N2} & \dots & U_{NN} \end{pmatrix}$

We may identify the columns by

$$U \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix} = \begin{pmatrix} U_{00} \\ U_{10} \\ U_{20} \\ \vdots \\ U_{N0} \end{pmatrix}, \quad U \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix} = \begin{pmatrix} U_{01} \\ U_{11} \\ U_{21} \\ \vdots \\ U_{N1} \end{pmatrix}, \quad U \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \end{pmatrix} = \begin{pmatrix} U_{02} \\ U_{12} \\ U_{22} \\ \vdots \\ U_{N2} \end{pmatrix}$$

Exercises:

1. **Prove this** for general matrices, using the definition of matrix multiplication.

Simple, but kind of tedious!

2. **Prove this** for operations on qubits, using orthogonality

$$\langle i|j\rangle = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

More involved, but very interesting!

Single qubit operations – the Pauli gates

- ▶ The *NOT gate*

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

- ▶ The *Phase Flip*

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|$$

- ▶ The *Y Gate*

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = i|1\rangle\langle 0| - i|0\rangle\langle 1|$$

More Single-qubit gates

- ▶ The *Amplitude-Rotation gate*

$$A_{\theta} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

- ▶ The *Phase-Rotation gate*

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{pmatrix}$$

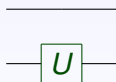
- ▶ The *Hadamard gate*

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

Two qubit operations

Consider a 1-qubit unitary $U = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix}$.

Make a (trivial) 2 qubit gate:

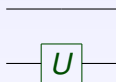


- ▶ Do nothing to the first qubit.
- ▶ Apply U to the second qubit.

What does this do to basis states??

A 2-qubit operation

On basis states:



$$\blacktriangleright |0\rangle|0\rangle \mapsto |0\rangle U(|0\rangle)$$

$$\blacktriangleright |0\rangle|1\rangle \mapsto |0\rangle U(|1\rangle)$$

$$\blacktriangleright |1\rangle|0\rangle \mapsto |1\rangle U(|0\rangle)$$

$$\blacktriangleright |1\rangle|1\rangle \mapsto |1\rangle U(|1\rangle)$$

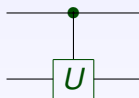
The corresponding 4×4 matrix is

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes U = \begin{pmatrix} U_{00} & U_{01} & 0 & 0 \\ U_{10} & U_{11} & 0 & 0 \\ 0 & 0 & U_{00} & U_{01} \\ 0 & 0 & U_{10} & U_{11} \end{pmatrix}$$

The 'controlled- U gate'

One qubit can act as a 'control bit':

- ▶ The first qubit does not change.



- ▶ For the second qubit:
 - ▶ If the first qubit is $|0\rangle$, do nothing.
 - ▶ If the first qubit is $|1\rangle$, apply U .

What does this do to basis states??

On the computational basis:

U is given by:

$$U = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix}$$

Controlled U gives

- ▶ $|0\rangle|0\rangle \mapsto |0\rangle|0\rangle$
- ▶ $|0\rangle|1\rangle \mapsto |0\rangle|1\rangle$
- ▶ $|1\rangle|0\rangle \mapsto |1\rangle(U_{00}|0\rangle + U_{10}|1\rangle)$
- ▶ $|1\rangle|1\rangle \mapsto |1\rangle(U_{01}|0\rangle + U_{11}|1\rangle)$

Finding the matrix for CU

1. Write $CU|00\rangle$, $CU|01\rangle$, $CU|10\rangle$, $CU|11\rangle$ as vectors.
2. This gives the columns of the matrix for CU .

Finding CU from the bra-ket description

$$CU|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad CU|01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$CU|10\rangle = \begin{pmatrix} 0 \\ 0 \\ U_{00} \\ U_{10} \end{pmatrix} \quad CU|11\rangle = \begin{pmatrix} 0 \\ 0 \\ U_{01} \\ U_{11} \end{pmatrix}$$

This gives the columns of CU .

Finding CU from the bra-ket description

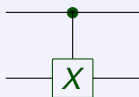
Bringing these together,

$$CU = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{00} & U_{01} \\ 0 & 0 & U_{10} & U_{11} \end{pmatrix}$$

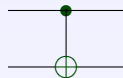
(Trivial) Exercise: prove this is unitary.

The controlled-NOT gate

A special case is the controlled-NOT gate:



more often drawn as



This has the following matrix:

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

Universal quantum gate sets

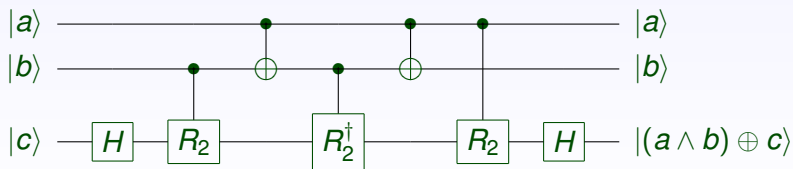
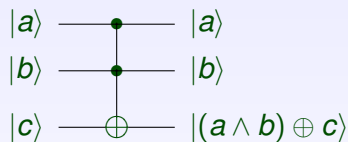
Theorem:

- ▶ Any unitary operation U on k qubits can be given as a circuit of $CNOT$ and single qubit gates.
- ▶ This implementation requires $O(4^k)$ gates.

Thus, $CNOT$ and single-qubit gates are *universal*.

They form the quantum analogue of $\{AND, NOT\}$.

An example: simulating the Toffoli gate



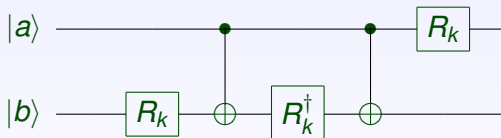
Checking on basis states:

Qubit $ a\rangle$	Qubit $ b\rangle$	Action on $ c\rangle$
$ 0\rangle$	$ 0\rangle$	$H^2 = I$
$ 0\rangle$	$ 1\rangle$	$H^2 = I$
$ 1\rangle$	$ 0\rangle$	$HR_2R_2^\dagger H = H^2 = I$
$ 1\rangle$	$ 1\rangle$	$HR_2R_2H = HZH = X$

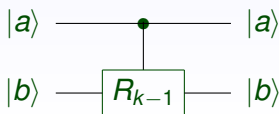
Finally, we simulate Controlled- R_k gates.

It suffices to simulate controlled R_{k-1} gates
— we do this using R_k and CX .

The simulation:



is equivalent to



Checking the simulation

Let us write $R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$.

- ▶ When $|a\rangle = |0\rangle$, we get:

$$|0\rangle|b\rangle \mapsto R_k(|0\rangle)R_k^\dagger R_k(|b\rangle) = |0\rangle|b\rangle$$

- ▶ When $|a\rangle = |1\rangle$,

$$|1\rangle|b\rangle \mapsto e^{i\theta}|1\rangle XR_k^\dagger XR_k(|b\rangle)$$

Simplifying the answer

We now need to simplify $e^{i\theta}|1\rangle XR_k^\dagger XR_k(|b\rangle)$

In matrix form, $e^{i\theta}XR_k^\dagger XR_k =$

$$\begin{aligned}
 e^{i\theta} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\theta} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} \\
 = e^{i\theta} \begin{pmatrix} 0 & e^{-i\theta} \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & e^{i\theta} \\ 1 & 0 \end{pmatrix} \\
 = e^{i\theta} \begin{pmatrix} e^{-i\theta} & 0 \\ 0 & e^{i\theta} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{2i\theta} \end{pmatrix} = R_k^2 \equiv R_{k-1}
 \end{aligned}$$

Describing the game

Setting up the game:

Conventions

- ▶ We identify $\{|heads\rangle, |tails\rangle\}$ with $\{|0\rangle, |1\rangle\}$.
- ▶ We ignore normalisation ...

Alice prepares the coin in state $|heads\rangle + |tails\rangle$.

Assuming Bob flips the coin:
$$\left\{ \begin{array}{l} Flip(|heads\rangle) = |tails\rangle \\ Flip(|tails\rangle) = |heads\rangle \end{array} \right. .$$

How does Alice always win ?

Bob's flip has absolutely no effect:

$$\text{Flip}(|\text{heads}\rangle + |\text{tails}\rangle) = |\text{heads}\rangle + |\text{tails}\rangle$$

Alice then performs a Hadamard operation $|0\rangle + |1\rangle \mapsto |0\rangle$.

$$|\text{heads}\rangle + |\text{tails}\rangle \mapsto |\text{heads}\rangle$$

Alice correctly calls 'heads' every time.

Important: Do not gamble with Alice!

Quantum Computation

Lecture 5

Sam Braunstein

The Pauli gates

Recall the 1-qubit operations from the teleportation experiment:

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

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

These matrices

1. Form a **group**.
2. Satisfy $X^2 = Y^2 = Z^2 = 1$
3. Satisfy $Y = iXZ$

Matrix exponentiation

Recall the power series for matrix exponentiation:

$$e^M = I + M + \frac{M^2}{2!} + \frac{M^3}{3!} + \frac{M^4}{4!} + \dots$$

A special case

When $M^2 = I$, and x is a real number,

$$e^{ixM} = \cos(x)I + i \cdot \sin(x)M$$

Proof: Recall the power series expansions for *cos* and *sin*.

The rotation gates

These are exponentials of Pauli matrices:

$$R_x(\theta) = e^{-\frac{i\theta}{2}X} \quad , \quad R_y(\theta) = e^{-\frac{i\theta}{2}Y} \quad , \quad R_z(\theta) = e^{-\frac{i\theta}{2}Z}$$

Using the special case formula:

- ▶ $R_x(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) X$
- ▶ $R_y(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) Y$
- ▶ $R_z(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) Z$

Matrices for the rotation operators:

Explicitly, these have simple matrix formulæ:

$$\blacktriangleright R_x(\theta) = \begin{pmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix}$$

$$\blacktriangleright R_y(\theta) = \begin{pmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix}$$

$$\blacktriangleright R_z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$$

Universal 1-qubit operations

These operators can be used to give all 1-qubit operations.

Theorem:

For every single-qubit unitary U ,

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

where $\alpha, \beta, \gamma, \delta$ are real numbers.

Every possible one-qubit gate can be produced from this smaller set.

A simple consequence:

Any one-qubit gate can be written as

$$U = e^{i\alpha} AXBXC$$

where A, B, C are unitaries satisfying $ABC = I$.

How to do this ?

Take

- ▶ $A = R_Z(\beta) R_Y\left(\frac{\gamma}{2}\right)$
- ▶ $B = R_Y\left(-\frac{\gamma}{2}\right) R_Z\left(-\frac{(\delta+\beta)}{2}\right)$
- ▶ $C = R_Z\left(\frac{(\delta-\beta)}{2}\right)$

The proof ...

First prove that $ABC = I$

$$ABC = e^{-i\beta\frac{Z}{2}} e^{-i\gamma\frac{Y}{4}} e^{i\gamma\frac{Y}{4}} e^{i(\delta+\beta)\frac{Z}{4}} e^{-i(\delta-\beta)\frac{Z}{4}}$$

$$ABC = e^{-i\beta\frac{Z}{2}} e^{i(\delta+\beta)\frac{Z}{4}} e^{-i(\delta-\beta)\frac{Z}{4}}$$

$$ABC = e^{i\left(\frac{-\beta}{2} + \frac{\delta}{4} + \frac{\beta}{4} - \frac{\delta}{4} + \frac{\beta}{4}\right)Z}$$

$$ABC = e^{0 \cdot Z} = I$$

The proof continues...

Simple Pauli group identities:

▶ $XZX = -Z$

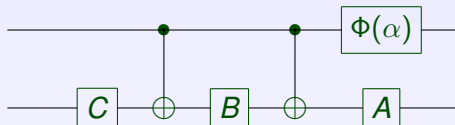
▶ $XYX = X(iXZ)X = iX(XZX) = -Y$

Therefore, $XBX = R_Y\left(\frac{\gamma}{2}\right) R_Z\left(\frac{\delta+\beta}{2}\right)$, and so

$$\begin{aligned} AXBXC &= R_Z(\beta) R_Y\left(\frac{\gamma}{2}\right) R_Y\left(\frac{\gamma}{2}\right) R_Z\left(\frac{\delta+\beta}{2}\right) R_Z\left(\frac{\delta-\beta}{2}\right) \\ &= R_Z(\beta) R_Y(\gamma) R_Z(\delta) \end{aligned}$$

This gives (up to a phase factor), an *arbitrary* unitary.

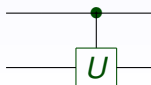
What use is this ?



$$\Phi(\alpha) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix}$$

- ▶ When *first* qubit is $|0\rangle$,
 $ABC = I$ is applied to the *second* qubit.
- ▶ When the first qubit is $|1\rangle$,
 $e^{i\alpha}AXBXC = U$ is applied to the *second* qubit.

This circuit implements the controlled- U gate, CU .



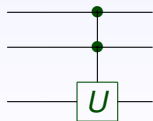
The controlled-controlled U gate

A special case ...

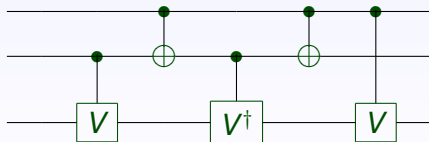
Assume $U = V^2$, for some unitary V .

Question: *Is this really a special case?*

The C^2U gate



Decomposing the C^2U gate

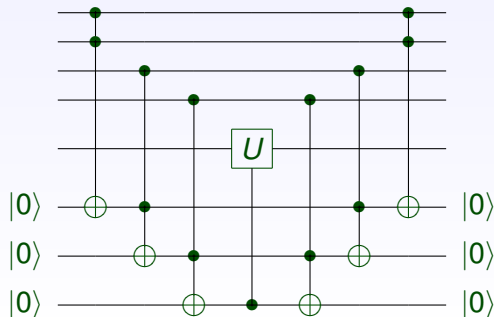


Exercise: Prove that these two circuits are equivalent

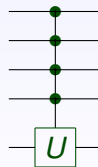
Building $C^k U$ gates

We build $C^k U$ gates using *Toffoli gates* and *ancillary qubits*.

Example – this circuit:



simulates $C^4 U$



Simulating $C^k U$ gates, cont.

Some features:

- ▶ The ancillary workspace is ‘cleaned up’.
- ▶ The simulation takes $O(k)$ gates.
- ▶ This can be done without an ancilla but this takes $O(k^2)$ gates.

Preparing arbitrary states

From a *fixed input*, say $|000\rangle$, how can we prepare an arbitrary three-qubit state

$$\sum_{a,b,c \in \{0,1\}} \alpha_{abc} |abc\rangle = \sum_{j=0}^7 \alpha_j |j\rangle?$$

We will consider *all* ‘branches’,

$$|000\rangle, |001\rangle, \dots, |110\rangle, |111\rangle$$

For each branch we separately:

1. Assign an **amplitude**,
2. Assign a **phase**.

To assign amplitudes

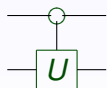
Recall the *amplitude rotation gate* $\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$

drawn as 

We will use controlled amplitude rotations:

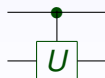
Two different control conventions!

Control on $|0\rangle$



Apply U to the 2^{nd} qubit when the 1^{st} qubit is $|0\rangle$.

Control on $|1\rangle$



Apply U to the 2^{nd} qubit when the 1^{st} qubit is $|1\rangle$.

Assigning amplitudes (cont.)

The one-qubit case

We use a single amplitude rotation:

$$|0\rangle \xrightarrow{\theta_1} \cos(\theta_1)|0\rangle + \sin(\theta_1)|1\rangle$$

From the identity

$$\cos^2(\theta_1) + \sin^2(\theta_1) = 1$$

We have assigned *arbitrary* amplitudes to $|0\rangle$ and $|1\rangle$.

The two-qubit case:

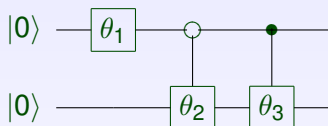
The intention is:

When measuring the first qubit only:

- ▶ On observing $|0\rangle$,
the remaining state is $\cos(\theta_2) |0\rangle + \sin(\theta_2) |1\rangle$.
- ▶ On observing $|1\rangle$,
the remaining state is $\cos(\theta_3) |0\rangle + \sin(\theta_3) |1\rangle$.

This is done using *controlled* amplitude rotations.

A circuit for the 2-qubit case:



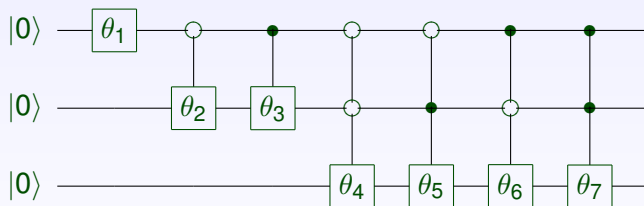
This acts as:

$$|00\rangle \longrightarrow \cos(\theta_1) |00\rangle + \sin(\theta_1) |10\rangle \longrightarrow$$

$$\begin{aligned} & \cos(\theta_1) \cos(\theta_2) |00\rangle \\ & + \cos(\theta_1) \sin(\theta_2) |01\rangle \\ & \quad + \sin(\theta_1) |10\rangle \end{aligned} \quad \mapsto \quad \begin{aligned} & \cos(\theta_1) \cos(\theta_2) |00\rangle \\ & + \cos(\theta_1) \sin(\theta_2) |01\rangle \\ & + \sin(\theta_1) \cos(\theta_3) |10\rangle \\ & + \sin(\theta_1) \sin(\theta_3) |11\rangle \end{aligned}$$

The three qubit case:

We use the following circuit:



This allows us to set the amplitude of every branch.

The overall action is

$$\begin{aligned}
 |000\rangle &\longrightarrow (\cos(\theta_1)|0\rangle + \sin(\theta_1)|1\rangle)|00\rangle \\
 &\longrightarrow \cos(\theta_1)|0\rangle(\cos(\theta_2)|0\rangle + \sin(\theta_2)|1\rangle)|0\rangle \\
 &\quad + \\
 &\quad \sin(\theta_1)|1\rangle(\cos(\theta_3)|0\rangle + \sin(\theta_3)|1\rangle)|0\rangle \\
 &\longrightarrow \dots
 \end{aligned}$$

We now need to set the *phases*!

Arranging phase rotations

To *rotate phases*, we need a *diagonal matrix*:

For a 2-qubit state:

We rotate the phase of $|j\rangle$ by $e^{i\gamma_j}$ using:

$$\text{PhaseRotator} = \begin{pmatrix} e^{i\gamma_0} & 0 & 0 & 0 \\ 0 & e^{i\gamma_1} & 0 & 0 \\ 0 & 0 & e^{i\gamma_2} & 0 \\ 0 & 0 & 0 & e^{i\gamma_3} \end{pmatrix}$$

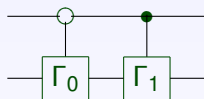
This can be simplified. Define

$$\Gamma_0 = \begin{pmatrix} e^{i\gamma_0} & 0 \\ 0 & e^{i\gamma_1} \end{pmatrix} \text{ and } \Gamma_1 = \begin{pmatrix} e^{i\gamma_2} & 0 \\ 0 & e^{i\gamma_3} \end{pmatrix}$$

Arranging phase rotations (cont.)

Using $\Gamma_0 = \begin{pmatrix} e^{i\gamma_0} & 0 \\ 0 & e^{i\gamma_1} \end{pmatrix}$ and $\Gamma_1 = \begin{pmatrix} e^{i\gamma_2} & 0 \\ 0 & e^{i\gamma_3} \end{pmatrix}$,

we implement phase rotations by:



This gives the matrix $\begin{pmatrix} e^{i\gamma_0} & 0 & 0 & 0 \\ 0 & e^{i\gamma_1} & 0 & 0 \\ 0 & 0 & e^{i\gamma_2} & 0 \\ 0 & 0 & 0 & e^{i\gamma_3} \end{pmatrix}$

The 3-qubit case

We wish to rotate the phase of

$$\{|000\rangle, |001\rangle, |010\rangle, \dots, |111\rangle\}$$

by the factor

$$\{e^{i\gamma_0}, e^{i\gamma_1}, e^{i\gamma_2}, \dots, e^{i\gamma_7}\}$$

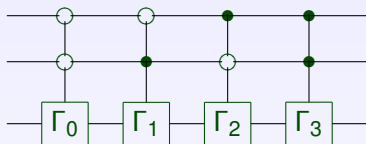
We define *single-qubit* operations $\Gamma_k = \begin{pmatrix} e^{i\gamma_{2k}} & 0 \\ 0 & e^{i\gamma_{2k+1}} \end{pmatrix}$

where $k = 0, 1, 2, 3$

A circuit for the 3-qubit case

Using

$$\Gamma_k = \begin{pmatrix} e^{i\gamma_{2k}} & 0 \\ 0 & e^{i\gamma_{2k+1}} \end{pmatrix}$$

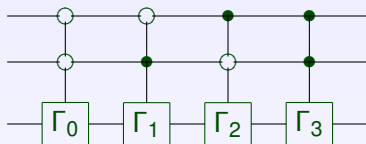


- ▶ $C^2\Gamma_0$ sets phases for $|000\rangle, |001\rangle$
- ▶ $C^2\Gamma_1$ sets phases for $|010\rangle, |011\rangle$
- ▶ $C^2\Gamma_2$ sets phases for $|100\rangle, |101\rangle$
- ▶ $C^2\Gamma_3$ sets phases for $|110\rangle, |111\rangle$

- └ How to construct arbitrary states
- └ Setting amplitude & phase

The 3-qubit case, in matrix form:

The circuit:



In matrix form:

$$\begin{pmatrix} e^{i\gamma_0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{i\gamma_1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e^{i\gamma_2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e^{i\gamma_3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{i\gamma_4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{i\gamma_5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{i\gamma_6} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{i\gamma_7} \end{pmatrix}$$

A couple of Exercises!

For arbitrary n -qubit states:

- ▶ *(Straightforward !)*
Give a general procedure for setting **phases**.

- ▶ *(more involved !!)*
Give a general procedure for setting **amplitudes**.

Quantum Computation

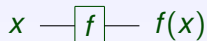
Lecture 6

Sam Braunstein

What is a Query Algorithm?

Query algorithms

- ▶ **Input:** a ‘black box’, or ‘oracle’ that computes some function



- ▶ **Output:** Some information about f

Example:

Given $f(x) = c_0 + c_1x + c_2x^2 + \dots + c_nx^n$, find $\{c_0, \dots, c_n\}$.

*The goal is to **minimise** the number of calls to the black box.*

Deutsch's problem

Let f be a single-bit function: $f : \{0, 1\} \rightarrow \{0, 1\}$.

Four possible functions:

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
0	0	1	0	1
1	0	1	1	0

Constant functions

$f(x)$ is the same, regardless of x .

Balanced functions

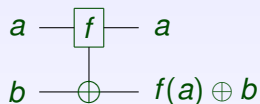
The number of 0s is the same as the number of 1s.

Goal: find out whether $f(0) = f(1)$ (i.e. find $f(0) \oplus f(1)$).

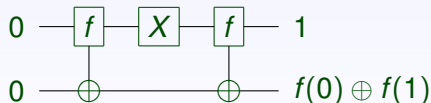
Classically, this takes two queries of the black box.

Classical solutions

First make a reversible version of f :



Use this to compute $f(0) \oplus f(1)$



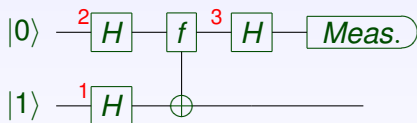
This solution takes 2 queries.

Question — can we do any better using *irreversible* gates?

A quantum-mechanical solution

A quantum algorithm gives a provably faster solution:

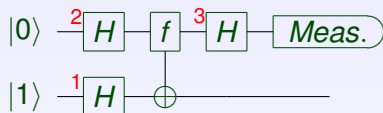
A QM algorithm for Deutsch's problem



This requires only 1 query.

Note: H is the Hadamard gate — 1 2 3 are simply labels.

How does this work ?



The first two Hadamard gates

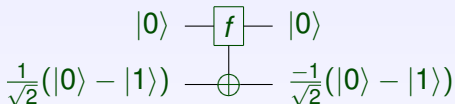
$$|0\rangle \xrightarrow{2} H \longrightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$|1\rangle \xrightarrow{1} H \longrightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

- ▶ 2H will allow evaluation at $f(0)$ and $f(1)$ simultaneously.
- ▶ 1H forms a superposition of the second (target) qubit ...

Assume first qubit is $|0\rangle$

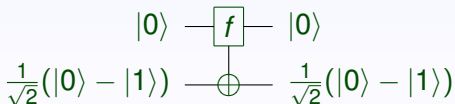
When f flips its input ...



Overall state is

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

When f does not flip its input ...



Overall state is

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

In either case, the final state is $(-1)^{f(0)}|0\rangle(|0\rangle - |1\rangle)$

Now assume first qubit is $|1\rangle$

When f flips its input ...

We are left with $\frac{-1}{\sqrt{2}}|1\rangle(|0\rangle - |1\rangle)$.

When f does not flip its input ...

We are left with $\frac{1}{\sqrt{2}}|1\rangle(|0\rangle - |1\rangle)$.

Combining the two cases

When the first qubit is $|x\rangle$, for $x = 0, 1$, the final state is

$$\frac{(-1)^{f(x)}}{\sqrt{2}}|x\rangle(|0\rangle - |1\rangle)$$

The key to this algorithm:

Recall the 'quantum coin-tossing game'!

The first qubit is placed into a superposition:

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

What happens to this state, when we flip it?

Applying linearity to the result from the last slide:

(and ignoring normalisation!)

$$\begin{array}{l} |0\rangle + |1\rangle \longrightarrow \boxed{f} \longrightarrow (-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle \\ |0\rangle - |1\rangle \longrightarrow \oplus \longrightarrow |0\rangle - |1\rangle \end{array}$$

Almost there ...

Ignoring a common $\frac{1}{\sqrt{2}}$ factor .. The first qubit is now in state $(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle$

f is constant or balanced

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
0	0	1	0	1
1	0	1	1	0

When f is constant

This state is

$$\pm(|0\rangle + |1\rangle)$$

When f is balanced

This state is

$$\pm(|0\rangle - |1\rangle)$$

These states are (of course) orthogonal!

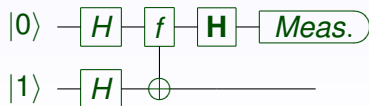
One final Hadamard

Recall the H gate on the computational basis:

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

$$\pm |1\rangle \xleftrightarrow{H} \frac{\pm 1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

The final Hadamard on the first qubit



We will observe $|0\rangle$ or $|1\rangle$ at the detector.

Another Quantum Algorithm

One-out-of-four search

We are given a function:

$$f : \{0, 1\}^2 \rightarrow \{0, 1\}$$

This takes a *pair* of bits to a *single* bit.

It has the additional property (**promise**) that:

there is exactly one input x such that $f(x) = 1$.

Remember – x is a *pair* of bits, so $x \in \{00, 01, 10, 11\}$.

A function with a promise

There is exactly one input x such that $f(x) = 1$.

This gives 4 possibilities for f . Call these $\{f_{00}, f_{01}, f_{10}, f_{11}\}$.

x	$f_{00}(x)$	$f_{01}(x)$	$f_{10}(x)$	$f_{11}(x)$
00	1	0	0	0
01	0	1	0	0
10	0	0	1	0
11	0	0	0	1

The query problem

Given f , we wish to:

Find the unique input for which $f(x) = 1$.

The classical solution

In the worst case, we need 3 queries.

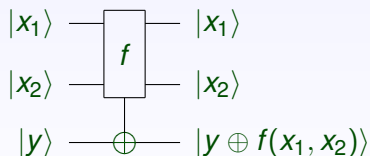
The quantum solution

One query to the black box is enough!

The first step

We need a unitary oracle U_f that implements f :

A circuit for U_f



The action of U_f

On the computational basis:

$$U_f |x_1, x_2, y\rangle \mapsto |x_1, x_2, y \oplus f(x_1, x_2)\rangle$$

How to use this oracle

As in the previous lecture:

The 'control' qubits are first placed in an equal superposition

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

The 'target' qubits are first placed in the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

How to create these inputs?

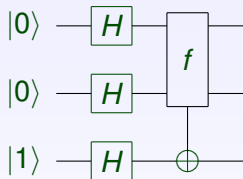
Check: Using Hadamards:

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

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

Bringing these together:

We have the following circuit:



$$|001\rangle \mapsto (|00\rangle + |01\rangle + |10\rangle + |11\rangle)(|0\rangle - |1\rangle)$$

\mapsto

$$((-1)^{f(00)} |00\rangle + (-1)^{f(01)} |01\rangle + (-1)^{f(10)} |10\rangle + (-1)^{f(11)} |11\rangle)(|0\rangle - |1\rangle)$$

How to get an output

The output of the previous circuit is

$$\left((-1)^{f(00)} |00\rangle + (-1)^{f(01)} |01\rangle + (-1)^{f(10)} |10\rangle + (-1)^{f(11)} |11\rangle \right) (|0\rangle - |1\rangle)$$

This is a **product state**, $|\psi_{ij}\rangle (|0\rangle - |1\rangle)$, where

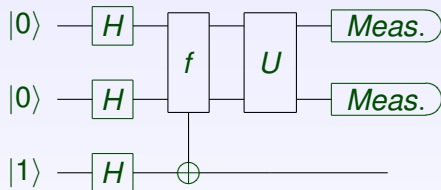
$$\begin{aligned} |\psi_{00}\rangle &= -|00\rangle + |01\rangle + |10\rangle + |11\rangle \\ |\psi_{01}\rangle &= |00\rangle - |01\rangle + |10\rangle + |11\rangle \\ |\psi_{10}\rangle &= |00\rangle + |01\rangle - |10\rangle + |11\rangle \\ |\psi_{11}\rangle &= |00\rangle + |01\rangle + |10\rangle - |11\rangle \end{aligned}$$

Exercise: These states are all orthogonal!

... and can therefore be distinguished by measurement!

Distinguishing orthogonal states

The full algorithm is given by:



The 2-qubit unitary U converts

$$\left. \begin{aligned}
 |\Psi_{00}\rangle &= -|00\rangle + |01\rangle + |10\rangle + |11\rangle \\
 |\Psi_{01}\rangle &= |00\rangle - |01\rangle + |10\rangle + |11\rangle \\
 |\Psi_{10}\rangle &= |00\rangle + |01\rangle - |10\rangle + |11\rangle \\
 |\Psi_{11}\rangle &= |00\rangle + |01\rangle + |10\rangle - |11\rangle
 \end{aligned} \right\} \text{ into } \left\{ \begin{array}{l} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{array} \right.$$

How to find this unitary?

We can simply write down U^{-1}

$$U^{-1} = \frac{1}{2} \begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}$$

And take the **conjugate transpose** to find $U = U^{-1}$.

Challenge! Simulate U using H , Toffoli, $CNOT$, X and Z gates.

Two qubits and beyond?

Can we generalise this to arbitrary qubits ??

- ▶ In larger spaces, this method breaks down.
- ▶ The output state vectors are not orthogonal.

We will later see that searching a space of size N takes $O(\sqrt{N})$ queries.

Quantum Computation

Lecture 7

Sam Braunstein

Remember: Query Algorithms

Query algorithms

- ▶ **Input:** a ‘black box’, or ‘oracle’ that computes some function

$$x \text{ --- } \boxed{f} \text{ --- } f(x)$$

- ▶ **Output:** Some information about f

These often involve a ‘promise’

Examples:

- ▶ $f(x) = 1$, for exactly one input x .
- ▶ The function f is either **balanced** or **constant**.

Constant and Balanced functions

Consider a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$.

We are promised that this is either **constant** or **balanced**.

Constant functions

f is **constant** when $f(x)$ is the same – i.e.

$$f(x) = 0 \text{ or } f(x) = 1$$

for all values of x .

Balanced functions

f is **balanced** when

$$\sum_x f(x) = 2^{n-1}$$

i.e. $f(x) = 1$ for exactly half the inputs.

Discriminating constant and balanced functions

We have a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$.

We are promised that this is either :

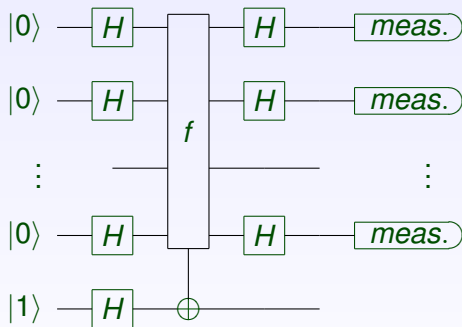
1. **Constant.**
2. **Balanced.**

We wish to know which one of these is correct.

Classically, we require $2^{n-1} + 1$ queries to be sure.

Quantum-mechanically we need one query.

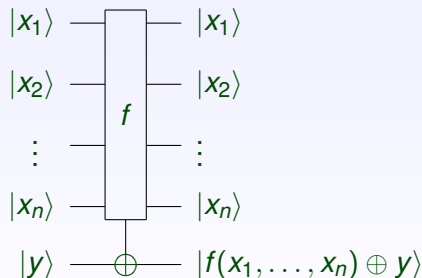
Let's start with the circuit



This is the full circuit for the **Deutsch-Jozsa algorithm**.

Reminder

We have made an **oracle** U_f from the classical function
 $f : \{0, 1\}^n \rightarrow \{0, 1\}$



We will – of course – query this in a suitable superposition.

The first n qubits ...

Creating equal superpositions

Hadamard gates are applied to the first n qubits:

$$|0\rangle |0\rangle \dots |0\rangle \mapsto (|0\rangle + |1\rangle)(|0\rangle + |1\rangle) \dots (|0\rangle + |1\rangle)$$

Expanding this out, we get *all possible* bitstrings

$$|0..000\rangle + |0..001\rangle + |0..010\rangle + |0..011\rangle + |0..100\rangle + \dots$$

Translating from binary

$$|0\rangle + |1\rangle + |2\rangle + |3\rangle + |4\rangle + \dots + |2^n - 1\rangle = \sum_{x=0}^{2^n-1} |x\rangle$$

The first $n + 1$ Hadamards ...

For the final qubit: $H(|1\rangle) = (|0\rangle - |1\rangle)$

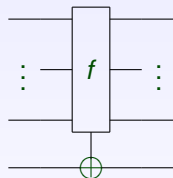
The input to the oracle is therefore

$$\sum_{x=0}^{2^n-1} |x\rangle \cdot (|0\rangle - |1\rangle)$$

Exercise What is the correct normalisation for this state??

How does the oracle work ?

The oracle



The action

$$\begin{aligned}
 & U_f |x_1, x_2, \dots, x_n, y\rangle \\
 &= |x_1, x_2, \dots, x_n, f(x_1, \dots, x_n) \oplus y\rangle
 \end{aligned}$$

Assume ...

- ▶ The first n qubits are in the computational basis.
- ▶ The last qubit is $|0\rangle - |1\rangle$.

The effect is:

$$\begin{aligned}
 & U_f (|x\rangle (|0\rangle - |1\rangle)) \\
 &= (-1)^{f(x)} |x\rangle (|0\rangle - |1\rangle)
 \end{aligned}$$

By linearity:

In the Deutsch-Jozsa algorithm:

- ▶ The input to the oracle is

$$\sum_{x=0}^{2^n-1} |x\rangle (|0\rangle - |1\rangle)$$

- ▶ As the oracle is linear,

$$U_f \left(\sum_{x=0}^{2^n-1} |x\rangle (|0\rangle - |1\rangle) \right) = \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle \cdot (|0\rangle - |1\rangle)$$

Comparing balanced and constant functions ...

The final qubit in the circuit is $|0\rangle - |1\rangle$. This is:

- ▶ Not *measured*.
- ▶ Not *entangled* with the other qubits.

For the first n qubits, the oracle outputs

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

We consider two cases:

1. Where f is **constant**.
2. Where f is **balanced**.

Oracle outputs – the two cases

When f is **constant**, $|\Psi\rangle = \pm \sum_{x=0}^{2^n-1} |x\rangle$.

When f is **balanced**, we simply have

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

However, the two cases are orthogonal ...

For a balanced function,

$$\sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle \text{ is orthogonal to } \sum_{x=0}^{2^n-1} |x\rangle$$

Orthogonality for the constant / balanced outputs

When f is balanced, simple algebra gives:

$$\left(\sum_{x=0}^{2^n-1} \langle x| \right) \left(\sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle \right) = 0$$

This comes from:

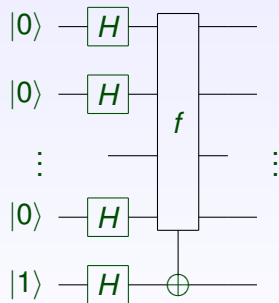
- ▶ a contribution of 2^{n-1} from the cases where $f(x) = 1$,
- ▶ a contribution of -2^{n-1} from the cases where $f(x) = 0$.

Exercise: verify this for the function

$$f(x) = \begin{cases} 1 & x = 0, \dots, 3 \\ 0 & x = 4, \dots, 7 \end{cases}$$

Distinguishing constant and balanced functions

The story so far:



On the first n qubits:

The output when f is **constant**

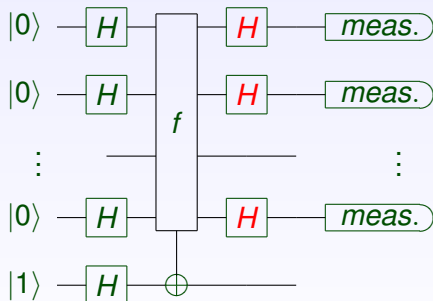
is orthogonal to

The output when f is **balanced**.

The two cases are (perfectly) distinguishable.

The rôle of the final Hadamards

Remember the final hadamards on the first n qubits:



These allow us to use *computational basis* measurements.

Mapping oracle outputs to the computational basis

The hadamard is its own inverse, so

$$H^{\otimes n} |00 \dots 0\rangle = \sum_{x=0}^{2^n-1} |x\rangle \Leftrightarrow H^{\otimes n} \left(\sum_{x=0}^{2^n-1} |x\rangle \right) = |00 \dots 0\rangle$$

- ▶ When f is **constant**,
the result of measurement is $|000 \dots 0\rangle$.
- ▶ When f is **balanced**,
the result of measurement is **orthogonal to** $|000 \dots 0\rangle$.

Remember: unitaries preserve orthogonality!

An operational interpretation

We run the Deutsch-Jozsa circuit, using a black box oracle U_f .

We measure the output in the computational basis.

- ▶ When we observe $|000 \dots 0\rangle$, the function must be **constant**.
- ▶ When we observe *anything else*, the function must be **balanced**.

We have distinguished between *balanced* and *constant* functions using a single call to the oracle.

The question of probabilities

We have a function $f: \{0, 1\}^n \rightarrow \{0, 1\}$.

We are promised that this is either :

1. **Constant.**
2. **Balanced.**

Classically, to find out which is the case:

- ▶ We require $2^{n-1} + 1$ queries to be certain.
- ▶ How about almost certain?

A probabilistic challenge

Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be either **constant** or **balanced**.

- ▶ Randomly pick x_1, x_2 from $\{0, 1\}^n$.
- ▶ if $f(x_1) = f(x_2)$, output **constant**.
- ▶ If $f(x_1) \neq f(x_2)$, output **balanced**.

When f is *constant*, this ‘algorithm’ is always correct!

When f is *balanced*, this ‘algorithm’ succeeds with *prob.* $= \frac{1}{2}$.

Iterating a probabilistic algorithm ...

Each application requires 2 queries of f .

Repeating the algorithm k times ...

- ▶ This needs $2k$ queries.
- ▶ And has a *one-sided* error probability of $\left(\frac{1}{2}\right)^k$

These are very good odds ...

In the limit (for large n), a constant number of trials solves the problem to any given probability!

Quantum Computation

Lecture 8

Sam Braunstein

About $H \otimes H \otimes \dots \otimes H = H^{\otimes n}$

Theorem: for $x \in \{0,1\}^n$, $H^{\otimes n}|x\rangle = \frac{1}{2^{n/2}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle$
where $x \cdot y = x_1 y_1 \oplus \dots \oplus x_n y_n$

Example: $H \otimes H = \frac{1}{2} \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix}$

Pf: For all $x \in \{0,1\}^n$, $H|x\rangle = |0\rangle + (-1)^x |1\rangle = \sum_y (-1)^{xy} |y\rangle$

Thus, $H^{\otimes n}|x_1 \dots x_n\rangle = \left(\sum_{y_1} (-1)^{x_1 y_1} |y_1\rangle \right) \dots \left(\sum_{y_n} (-1)^{x_n y_n} |y_n\rangle \right)$
 $= \sum_y (-1)^{x_1 y_1 \oplus \dots \oplus x_n y_n} |y_1 \dots y_n\rangle$ ■

Simon's problem

Quantum vs. classical separations

black-box problem	quantum	classical
constant vs. balanced	1 (query)	2 (queries)
1-out-of-4 search	1	3
constant vs. balanced	1	$\frac{1}{2} 2^n + 1$
Simon's problem		

(only for exact)
(probabilistic)

Simon's problem

Let $f: \{0,1\}^n \rightarrow \{0,1\}^n$ have the property that there exists an $r \in \{0,1\}^n$ such that $f(x) = f(y)$ iff $x \oplus y = r$ or $x = y$

Example:

x	$f(x)$
000	011
001	101
010	000
011	010
100	101
101	011
110	010
111	000

What is r in this case? _____

Answer: $r = 101$

In the table x coincides with exactly one other value $x \oplus r$

A classical algorithm for Simon

Search for a **collision**, an $x \neq y$ such that $f(x) = f(y)$

1. Choose $x_1, x_2, \dots, x_k \in \{0,1\}^n$ randomly (independently)
2. For all $i \neq j$, if $f(x_i) = f(x_j)$ then output $x_i \oplus x_j$ and halt

A hard case is where r is chosen randomly from $\{0,1\}^n - \{0^n\}$ and then the “table” for f is filled out randomly subject to the structure implied by r

How big does k have to be for the probability of a collision to be a constant, such as $3/4$?

Answer: $O(2^{n/2})$, each (x_i, x_j) collides with prob. $O(2^{-n})$

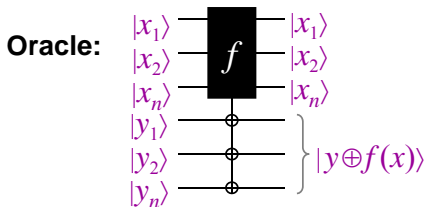
Classical lower bound

Theorem: *any* classical algorithm solving Simon's problem must make $\Omega(2^{n/2})$ queries

Proof is omitted here—note that the performance analysis of the previous algorithm does *not* imply the theorem

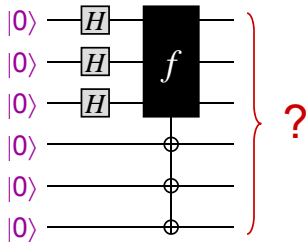
... how can we know that there isn't a *different* algorithm that performs better?

A quantum algorithm for Simon I



Proposed start of quantum algorithm: query all values of f in superposition

What is the output state of this circuit?



A quantum algorithm for Simon II

Answer: the output state is $\sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle$

Let $T \subseteq \{0,1\}^n$ be such that **one** element from each matched pair is in T (assume $r \neq 00\dots 0$)

Example: could take $T = \{000, 001, 011, 111\}$

Then the output state can be written as:

$$\sum_{x \in T} |x\rangle |f(x)\rangle + |x \oplus r\rangle |f(x \oplus r)\rangle$$
$$= \sum_{x \in T} (|x\rangle + |x \oplus r\rangle) |f(x)\rangle$$

x	$f(x)$
000	011
001	101
010	000
011	010
100	101
101	011
110	010
111	000

A quantum algorithm for Simon III

Measuring the second register yields $|x\rangle + |x \oplus r\rangle$ in the first register, for a random $x \in T$

How can we use this to obtain **some** information about r ?

Try applying $H^{\otimes n}$ to the state, yielding:

$$\begin{aligned} & \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle + \sum_{y \in \{0,1\}^n} (-1)^{(x \oplus r) \cdot y} |y\rangle \\ &= \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} \left(1 + (-1)^{r \cdot y} \right) |y\rangle \end{aligned}$$

Measuring this state yields y with prob. $\begin{cases} 1/2^{n-1} & \text{if } r \cdot y = 0 \\ 0 & \text{if } r \cdot y \neq 0 \end{cases}$

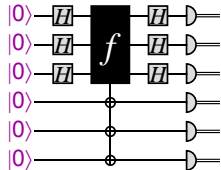
A quantum algorithm for Simon IV

Executing this algorithm $k = O(n)$ times yields random $y_1, y_2, \dots, y_k \in \{0,1\}^n$ such that $r \cdot y_1 = r \cdot y_2 = \dots = r \cdot y_k = 0$

How does this help?

This is a system of k linear equations, $Y \cdot r = 0$:

$$\begin{bmatrix} y_{11} & y_{12} & \dots & y_{1n} \\ y_{21} & y_{22} & \dots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{k1} & y_{k2} & \dots & y_{kn} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$



No need to actually measure the target qubits. Why?

With high probability, there is a unique non-zero solution that is r (which can be efficiently found by linear algebra) 11

Conclusion of Simon's algorithm

- Any classical algorithm has to query the black box $\Omega(2^{n/2})$ times, even to succeed with probability $\frac{3}{4}$
- There is a quantum algorithm that queries the black box only $O(n)$ times, and succeeds with probability $\frac{3}{4}$
- There is an exact solution: Repeat the algorithm until $\dim\{y\} = n - 1$. Then solve $Y.r = 0$, to yield a unique r . This too takes $O(n)$ calls to the oracle.

Quantum Computation

Lecture 9

Sam Braunstein

Shor's algorithm

These lectures are about

- ▶ **Shor's algorithm.**
- ▶ The **Quantum Fourier Transform**

Shor's algorithm is:

- ▶ The best-known application of QM computing.
- ▶ Very useful in cryptanalysis.
- ▶ Not proven to be faster than any classical algorithm.

Starting to use *complex numbers*

So far ...

we have not explicitly used complex numbers in our quantum algorithms.

Shor's algorithm uses:

1. complex phases,
2. the Fourier transform,

to exploit quantum parallelism

Geometric series & roots of unity

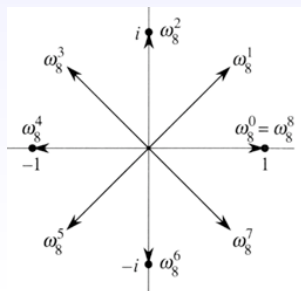
Remember the sum of a Geometric Series:

$$\sum_{j=0}^{N-1} z^j = \frac{1 - z^N}{1 - z} \quad \text{provided } z \neq 1$$

We will apply this to *complex roots of 1*.

Example: the 8th roots of 1

- ▶ $\omega = e^{2\pi i/8}$
- ▶ $\omega, \omega^2, \omega^3, \dots$ lie on a circle.
- ▶ ω^k is an 8th root of 1, for all integers k .



Summing roots of unity

Let $\omega = e^{\frac{2\pi i}{N}}$ be an N -th root of 1. What is $\sum_{y=0}^{N-1} \omega^{xy}$?

When $x \neq 0$

$$\sum_{y=0}^{N-1} (\omega)^{xy} = \sum_{y=0}^{N-1} (\omega^x)^y = \frac{1 - (\omega^x)^N}{1 - \omega^x} = 0$$

When $x = 0$

$$\sum_{y=0}^{N-1} (\omega)^{xy} = \sum_{y=0}^{N-1} (\omega^0) = \sum_{y=0}^{N-1} 1 = N$$

Roots of unity & the Kronecker delta

We have proved that:

$$\sum_{y=0}^{N-1} (\omega)^{xy} = N \cdot \delta_{x,0} \quad \text{where} \quad \delta_{x,0} = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0 \end{cases}$$

This function $\delta_{x,0}$ is the **Kronecker delta**.

Modular arithmetic, and summing roots of unity

When $x = 0, \pm N, \pm 2N, \pm 3N, \dots$ we get the same result.

The full formula is

$$\sum_{y=0}^{N-1} e^{\frac{2\pi ixy}{N}} = N \cdot \delta_{x,0} \pmod{N}$$

The quantum Fourier transform

For n qubits, the computational basis is

$$\{|0\rangle, |1\rangle, \dots, |N-1\rangle\} \quad \text{where } N = 2^n$$

The **quantum Fourier transform** is defined on the computational basis by

$$\mathcal{F}_N |x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{xy} |y\rangle \quad \text{where } \omega = e^{\frac{2\pi i}{N}}$$

Exercise: Give the matrix for \mathcal{F}_N , for 1, 2 and 3 qubits.

The Inverse quantum Fourier transform

The QFT has an inverse:

$$\mathcal{F}_N^{-1}|y\rangle = \frac{1}{\sqrt{N}} \sum_{z=0}^{N-1} \omega^{-yz}|z\rangle \quad \text{where } \omega = e^{\frac{2\pi i}{N}}$$

Exercises:

1. Give the matrix for \mathcal{F}_N^{-1} , for 2 and 3 qubits.
2. Check that these are the Hermitian conjugates (i.e. the daggers) of the matrices for \mathcal{F}_N .

Checking \mathcal{F}^\dagger really is the inverse

Let $|x\rangle$ be a computational basis state.

Then $\mathcal{F}_N^\dagger \mathcal{F}_N |x\rangle =$

$$\mathcal{F}^\dagger \left(\frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{xy} |y\rangle \right) \quad \text{by def.n of } \mathcal{F}$$

$$= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{xy} \mathcal{F}^\dagger |y\rangle \quad \text{by linearity}$$

$$= \frac{1}{N} \sum_{y=0}^{N-1} \omega^{xy} \sum_{z=0}^{N-1} \omega^{-yz} |z\rangle \quad \text{by def.n of } \mathcal{F}^\dagger$$

$$= \frac{1}{N} \sum_{z=0}^{N-1} \sum_{y=0}^{N-1} \omega^{y(x-z)} |z\rangle \quad \text{by basic algebra}$$

Checking \mathcal{F}^\dagger is the inverse (cont.)

So far,

$$\mathcal{F}_N^\dagger \mathcal{F}_N |x\rangle = \frac{1}{N} \sum_{z=0}^{N-1} \sum_{y=0}^{N-1} \omega^{y(x-z)} |z\rangle$$

Now remember that

$$\sum_{y=0}^{N-1} \omega^{y(x-z)} = N \cdot \delta_{x,z \pmod{N}} = \begin{cases} N & x = z \pmod{N} \\ 0 & x \neq z \pmod{N} \end{cases}$$

This sum is **zero**, except for the case where $x = z$.

Therefore,

$$\mathcal{F}_N^\dagger \mathcal{F}_N |x\rangle = |x\rangle$$

\mathcal{F}^\dagger is inverse to \mathcal{F}

- ▶ For computational basis states, $\mathcal{F}^\dagger \mathcal{F}|x\rangle = |x\rangle$.
- ▶ By linearity, $\mathcal{F}^\dagger \mathcal{F}|\psi\rangle = |\psi\rangle$, for all states.

Exercises:

Using the matrices for the 2 and 3 qubit cases, check that

$$\mathcal{F}^\dagger \mathcal{F} = I$$

i.e. check that \mathcal{F} is unitary.

Shor's algorithm – a broad picture

Shor's algorithm has 2 parts:

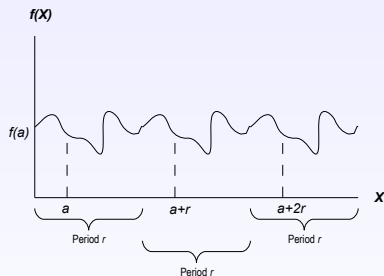
1. A quantum part that finds the period of a function.
2. A classical part that uses this to find factors of an integer.

Important

- ▶ The (quantum) period-finding routine is very general — it is not restricted to factorising.
- ▶ The (classical) post-processing is simple number theory — and easy to compute.

Periodic functions

A **periodic function** is one that repeats itself after a certain time.



$$f(x) = f(x + r) = f(x + 2r) = f(x + 3r) + \dots$$

When the function is “wrapped round” after any multiple of the period, it repeats itself.

A quantum algorithm for periodic functions

We have a function $f(x)$

We are *promised* that this is periodic.

Finding the period

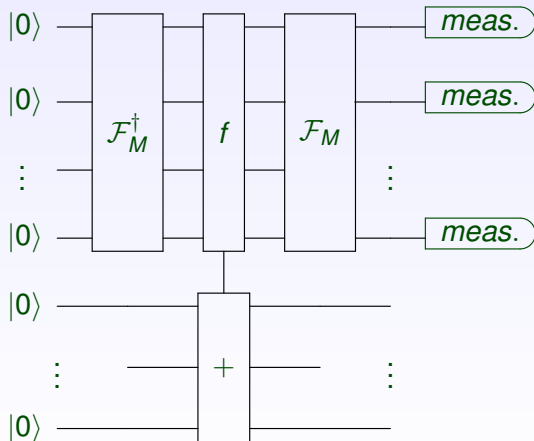
This means that

$$f(x) = f(x + r) = f(x + 2r) = \dots \quad \text{for all } x$$

The goal is to find the period r .

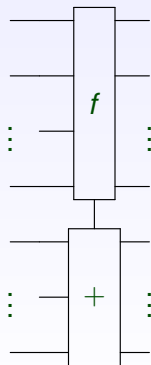
We will settle for finding some multiple of r .

A circuit for period-finding



About that oracle ...

The diagram



The action

On the computational basis:

$$U_f|x\rangle|y\rangle = |x\rangle|y + f(x)\rangle$$

Exercise:

Using *Mod 2* arithmetic, give

1. Controlled-Not (i.e. CX gate)
2. Controlled-Controlled-Not (i.e. Toffoli gate)

in this form

The action of the quantum circuit

We start off with $|0\rangle|0\rangle$.

Reminder: These are multi-qubit states.

We apply the inverse QFT to the first register:

$$|0\rangle|0\rangle \mapsto \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} e^{-\frac{2\pi i x \cdot 0}{M}} |x\rangle|0\rangle = \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle|0\rangle$$

This gives an equal superposition in the first register ...

We might as well have used Hadamards!

The action of the quantum circuit (cont.)

We now have

$$\frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle |0\rangle$$

We apply the oracle U_f :

$$\frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle |0\rangle \mapsto \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle |f(x)\rangle$$

So far, there are still no complex phases!

The final step of the quantum circuit

To the (first register of)

$$\frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle |f(x)\rangle$$

we apply the QFT:

$$\frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle |f(x)\rangle \mapsto \frac{1}{M} \sum_{y=0}^{M-1} \sum_{x=0}^{M-1} e^{\frac{2\pi ixy}{M}} |y\rangle |f(x)\rangle$$

This is the final result!!

Why should be get excited over this state ??

Let's look at a particular branch:

Recall that $f(x) = f(x + r) = f(x + 2r) = \dots$

The amplitude for $|y\rangle|f(x)\rangle$ is therefore

$$\frac{1}{M} \sum_{m=0}^{\frac{M}{r}-1} e^{\frac{2\pi i(x_0+mr)y}{M}} |y\rangle|f(x_0)\rangle$$

where

$$x = x_0 + mr \quad \text{for } m = 0, 1, \dots, \frac{M}{r} - 1$$

The amplitude of $|y\rangle|f(x)\rangle$

The amplitude for $|y\rangle|f(x)\rangle$ is

$$\frac{1}{M} \sum_{m=0}^{\frac{M}{r}-1} e^{\frac{2\pi i(x_0+mr)y}{M}} |y\rangle|f(x_0)\rangle = \frac{1}{M} \sum_{m=0}^{\frac{M}{r}-1} e^{\frac{2\pi i x_0 y}{M}} e^{\frac{2\pi i m r y}{M}} |y\rangle|f(x_0)\rangle$$

Simple re-arranging gives

$$\frac{1}{M} e^{\frac{2\pi i x_0 y}{M}} \left(\sum_{m=0}^{\frac{M}{r}-1} e^{\frac{2\pi i m r y}{M}} \right) |y\rangle|f(x_0)\rangle$$

Now recall summing roots of unity ...

The amplitude of $|y\rangle|f(x)\rangle$

The amplitude for $|y\rangle|f(x)\rangle$ is

$$\frac{1}{M} e^{\frac{2\pi i x_0 y}{M}} \left(\sum_{m=0}^{\frac{M}{r}-1} e^{\frac{2\pi i m y}{M/r}} \right) |y\rangle|f(x_0)\rangle$$

We have seen that

$$\sum_{m=0}^{\frac{M}{r}-1} e^{\frac{2\pi i m y}{M/r}} = \frac{M}{r} \delta_{y,0 \pmod{\frac{M}{r}}} = \frac{M}{r} \begin{cases} 1 & y = 0 \pmod{\frac{M}{r}} \\ 0 & \text{otherwise} \end{cases}$$

The interpretation ...

The amplitude of $|y\rangle|f(x)\rangle$ is essentially **zero**, unless

$$y = 0, \frac{M}{r}, \frac{2M}{r}, \frac{3M}{r}, \dots$$

On measurement, we see $|y\rangle = \left| \frac{kM}{r} \right\rangle$.

Repeating the experiment

After several trials, we have *with very high probability*, enough information to find r .

(e.g. observing finding a co-prime pair y, y' will do).

Quantum Computation

Lecture 10

Sam Braunstein

Shor's algorithm – factoring integers

We wish to factor the integer N .

Let us randomly pick some integer a .

sensible choices ...

- ▶ $a \leq \sqrt{N}$.
- ▶ a is relatively prime to N
 - ▶ This is easy to check.
 - ▶ If it is not true, we already have a factor ... no QM computer needed!

We study the function

$$f(x) = a^x \pmod{N}$$

Period-finding with modular exponentials

The function $f(x) = a^x \pmod{N}$ is **periodic**.

Suppose the period is r . That is, $a^r \equiv 1 \pmod{N}$.

Two possibilities:

1. The period r is **odd**.

This is useless. We start again with another value for a .

2. The period r is **even**.

This will let us factorize N .

From period-finding to factoring

We have:

- ▶ $a^r - 1 \equiv 0 \pmod{N}$
- ▶ r is an even number: $r = 2p$.

Simple algebra gives

$$a^r - 1 \pmod{N} \equiv (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) \pmod{N}$$

Therefore $(a^p - 1)(a^p + 1) \equiv 0 \pmod{N}$

So $a^p - 1$ or $a^p + 1$ has a common factor with N .

The final step: Calculate the *greatest common divisor*, using the “chinese remainder theorem”.

Shor's algorithm — a worked example

We wish to factor $N = 91$.

We try $a = 3$, and hence the function $f(x) = 3^x \pmod{91}$.

x	3^x	$3^x \pmod{91}$
0	1	1
1	3	3
2	9	9
3	27	27
4	81	81
5	243	61
6	729	1
7	2187	3

The period is $r = 6$

▶ $3^6 - 1 \equiv 0 \pmod{91}$

▶ $(3^3 - 1)(3^3 + 1) \equiv 0 \pmod{91}$

▶ $26 \times 28 \equiv 0 \pmod{91}$

Factoring 91, using Shor's algorithm

By finding the period of $f(x) = 3^x \pmod{91}$,
we have discovered that $26 \times 28 \equiv 0 \pmod{91}$.

Therefore, either 26 or 28 (or both) share a factor with 91.

Taking common divisors:

$$\gcd(26, 91) = 13 \quad , \quad \gcd(28, 91) = 7$$

Either 13 or 7 divides 91.

In fact, they both do: $91 = 13 \times 7$

Some missing details ...

An important question:

How do we implement the QFT?

i.e. using the basic toolkit of QM gates?

The Quantum Fourier Transform (QFT)

Remember:

$$\mathcal{F}_N|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{\frac{2\pi ixy}{N}} |y\rangle$$

As we are working with qubits, we take $N = 2^n$.

Let us write x in *binary* as

$$x = x_n x_{n-1} x_{n-2} x_{n-3} \dots x_1 \quad \text{where } x_k = 0 \text{ or } 1$$

This gives

$$x = 2^{n-1} x_n + 2^{n-2} x_{n-1} + \dots + x_1$$

Implementing the QFT

Similarly (but in the opposite order!) we write

$$y = y_1 y_2 \dots y_n \text{ where } y_k = 0 \text{ or } 1$$

this gives

$$y = 2^{n-1} y_1 + 2^{n-2} y_2 + \dots + y_n$$

Using this notation, we may easily calculate $xy \pmod{2^n} =$

$$x \cdot 2^{n-1} y_1 + x \cdot 2^{n-2} y_2 + \dots + x \cdot y_n \pmod{2^n}$$

$$= 2^{n-1} x_1 y_1 + 2^{n-2} (2x_2 + x_1) y_2 + \dots + (2^{n-1} x_n + 2^{n-2} x_{n-1} + \dots + x_1) y_n$$

Jumping over to qubits:

We now write

- ▶ $|x\rangle = |x_n x_{n-1} \dots x_1\rangle$
- ▶ $|y\rangle = |y_1 y_2 \dots y_n\rangle = |y_1\rangle \otimes |y_2\rangle \otimes \dots \otimes |y_n\rangle$

How does this help ?

In the QFT:

- ▶ The overall normalization is $\frac{1}{\sqrt{2^n}} = \frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}} \times \dots \times \frac{1}{\sqrt{2}}$.
- ▶ The summation may be written as

$$\sum_{y=0}^{2^n-1} (\dots) = \sum_{y_1=0}^1 \sum_{y_2=0}^1 \dots \sum_{y_n=0}^1 (\dots)$$

Splitting up the QFT

Using these tricks, we may give \mathcal{F}_{2^n} on the individual qubits:

$$\begin{aligned} \mathcal{F}_{2^n} |x_n \dots x_1\rangle = & \\ & \left(\frac{1}{\sqrt{2}} \sum_{y_1=0}^1 e^{2\pi i \frac{x_1 y_1}{2}} |y_1\rangle \right) \otimes \left(\frac{1}{\sqrt{2}} \sum_{y_2=0}^1 e^{2\pi i \frac{(2x_2 + x_1)y_2}{2^2}} |y_2\rangle \right) \\ & \otimes \dots \otimes \left(\frac{1}{\sqrt{2}} \sum_{y_n=0}^1 e^{2\pi i \frac{(2^{n-1}x_n + 2^{n-2}x_{n-1} + \dots + x_1)y_n}{2^n}} |y_n\rangle \right) \end{aligned}$$

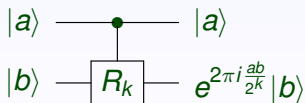
The QFT and QM gates

Recall the Hadamard: $|a\rangle \xrightarrow{H} \frac{1}{\sqrt{2}} \sum_{b=0}^1 (-1)^{ab} |b\rangle$

This gives

$$H|a\rangle = \frac{1}{\sqrt{2}} \sum_{b=0}^1 e^{2\pi i \frac{ab}{2}} |b\rangle$$

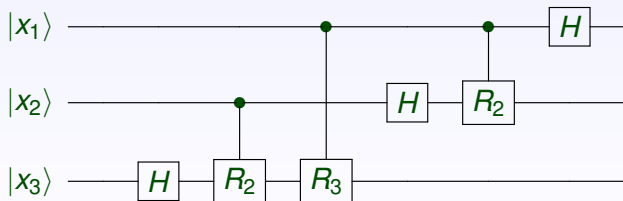
We also need the Controlled- R_k gate:



Building the QFT - the 3-qubit case

For 3 qubits

The pattern continues ... the QFT is given by:

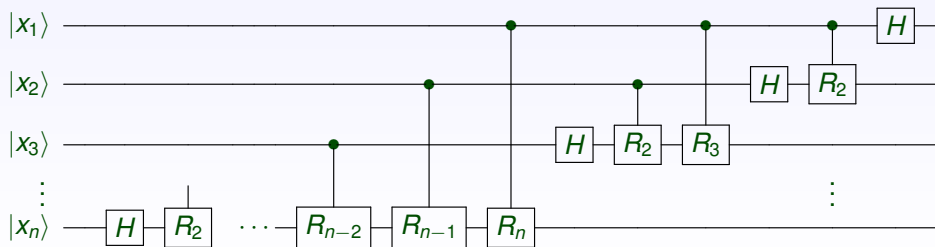


Building the QFT - the general case

By either

1. Guessing the pattern:
2. Explicit calculation:

... the QFT is given by:



Important points:

Various features of this decomposition:

- ▶ For any computational basis state $|x\rangle$, its Quantum Fourier Transform, $\mathcal{F}|x\rangle$, can be factorised.
- ▶ The individual qubits remain disentangled.
- ▶ This makes it easy to simulate classically.

The same is **not** true for arbitrary states.

Quantum Computation

Lecture 11

Sam Braunstein

Grover's algorithm

These lectures are about **Grover's algorithm**.

... also known as *virtual database-searching*.

Grover's algorithm is:

- ▶ Another query / promise algorithm.
- ▶ Useful in a wide range of settings.
- ▶ Provably faster than any classical algorithm.

The problem ...

We have a function $f : \{0, \dots, N - 1\} \rightarrow \{0, 1\}$.

The promise ...

There is some unique $x = x_0$ such that $f(x) = 1$.

We wish to find this unique x_0

Classically: This takes $O(N)$ queries.

Quantum-mechanically: This needs $O(\sqrt{N})$ queries.

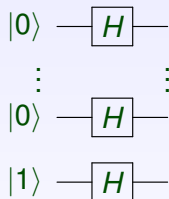
Grover's algorithm ... an outline

The algorithm is as follows:

1. For $N = 2^n$, we *prepare* a standard state of $n + 1$ qubits .
2. The “Grover operator” (a QM circuit) is applied.
 - ▶ This step is *repeated* a fixed number of times.
3. The first n qubits are *measured*.

The initial state preparation

The circuit



The action

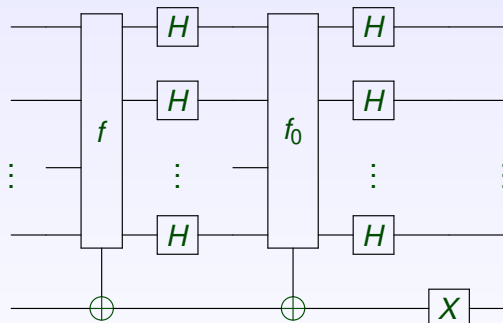
$$|0\rangle^{\otimes n}|1\rangle \mapsto \left(\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \right) \cdot \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

We will write the equal superposition as

$$|\Psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

so the initial state is $|\Psi_0\rangle \cdot \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

The Grover operator

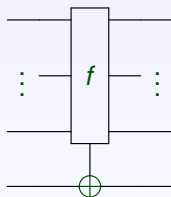


This features:

- ▶ Several Hadamards ...
- ▶ A single NOT gate ...
- ▶ **Two distinct oracles**, U_f and U_{f_0} .

The first oracle, U_f

The first oracle



On the computational basis

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

Remember

There exists some unique

$$x_0 \in \{0, 1, 2, 3, \dots\}$$

such that $f(x_0) = 1$.

The final qubit is flipped exactly when the first n qubits give $|x_0\rangle$.

The first oracle gives sign flips

Remember: the final qubit is in state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

$$X(|0\rangle - |1\rangle) = (-1) \cdot (|0\rangle - |1\rangle)$$

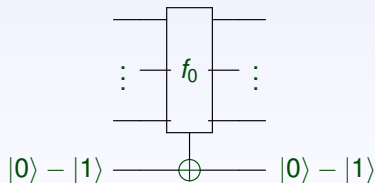
This introduces a sign flip into the branch containing $|x_0\rangle$.

$$\begin{aligned}
 & (\alpha_0|0\rangle + \alpha_1|1\rangle + \dots + \beta|x_0\rangle + \dots + \alpha_N|N-1\rangle)(|0\rangle - |1\rangle) \\
 & \quad \quad \quad \downarrow U_f \\
 & (\alpha_0|0\rangle + \alpha_1|1\rangle + \dots - \beta|x_0\rangle + \dots + \alpha_N|N-1\rangle)(|0\rangle - |1\rangle)
 \end{aligned}$$

The second oracle, U_{f_0}

$$f_0 : \{0, 1\}^n \rightarrow \{0, 1\} \text{ is defined by } f_0(x) = \begin{cases} 1 & x = 0 \\ 0 & \text{otherwise.} \end{cases}$$

The second oracle



Using identical reasoning:

$$\begin{aligned} & (\alpha_0|0\rangle + \alpha_1|1\rangle + \alpha_2|2\rangle + \dots)(|0\rangle - |1\rangle) \\ & \quad \downarrow U_{f_0} \\ & (-\alpha_0|0\rangle + \alpha_1|1\rangle + \alpha_2|2\rangle + \dots)(|0\rangle - |1\rangle) \end{aligned}$$

This seems uninteresting ... but don't forget the Hadamards!

A convention ...

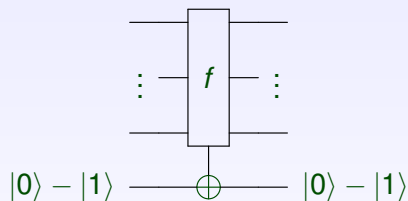
Important: the final qubit is in state $(|0\rangle - |1\rangle)$.

- ▶ This provides relative phase changes between branches.
- ▶ However (despite being the target), does not change!

Convention: we describe the action on the first n qubits ... without explicitly writing the final $(|0\rangle - |1\rangle)$.

If this is confusing, write $(|0\rangle - |1\rangle)$ after everything in the following slides!

The first oracle, revisited



Conditional phase-flip

This flips the sign of the branch where $f(x) = 1$

i.e.

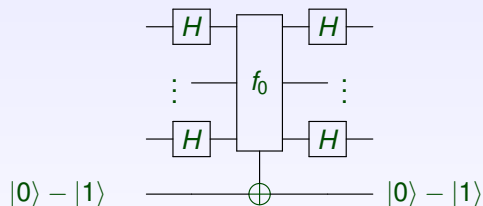
$$|x\rangle \mapsto \begin{cases} -|x\rangle & x = x_0 \\ |x\rangle & x \neq x_0 \end{cases}$$

On the first n qubits ...

This gives a unitary operator $U = I_n - 2|x_0\rangle\langle x_0|$

Exercise: Check this (i) is Unitary, and (ii) has the required action.

The second oracle, with added Hadamards



On the first n qubits ...

This oracle gives the unitary operator $I_n - 2|0\rangle\langle 0|$

How do the Hadamards change this?

The Hadamards and the oracle

Still looking at the first n qubits ...

The oracle gives the unitary operator $I_n - 2|0\rangle\langle 0|$

Including Hadamards, the overall action is

$$V = H^{\otimes n}(I_n - 2|0\rangle\langle 0|)H^{\otimes n}$$

Expanding this out, $V = H^{\otimes n}I_nH^{\otimes n} - 2(H^{\otimes n}|0\rangle)(H^{\otimes n}|0\rangle)^\dagger$

$$= I_n - 2 \left(\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \right) \left(\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \langle x| \right) = I_n - 2|\psi_0\rangle\langle\psi_0|$$

The Hadamards and the oracle (cont.)

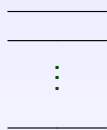
The second oracle, together with Hadamards, gives

$$V = I_n - 2|\psi_0\rangle\langle\psi_0|$$

Interpretation

- ▶ As before, we think of this as a *conditional phase-flip*.
- ▶ However, it is not in the computational basis.
- ▶ The 'component of the state in the direction $|\psi_0\rangle$ ' is flipped.

That final NOT gate



$$|0\rangle - |1\rangle \text{ --- } \boxed{X} \text{ --- } |0\rangle - |1\rangle$$

Unconditional phase-flip

$$X(|0\rangle - |1\rangle) = -(|0\rangle - |1\rangle)$$

This simply adds in an overall sign change.

Remember - global phases cannot be observed.

This (-1) is simply to tidy up the mathematics!

How does the algorithm work ??

- ▶ Our “target state” is $|x_0\rangle$; let us call this **Logical True**

$$|\mathcal{T}\rangle = |x_0\rangle$$

- ▶ We define **Logical False** to be

$$|\mathcal{F}\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \neq x_0} |x\rangle = \frac{1}{\sqrt{N-1}} \sum_{f(x)=0} |x\rangle$$

(i.e. an even superposition of all non-target computational basis states)

A remarkable fact:

All the action takes place in the subspace spanned by $|\mathcal{T}\rangle$ and $|\mathcal{F}\rangle$

For example:

- ▶ Our starting state is $|\Psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$.
- ▶ We may write this as

$$|\Psi_0\rangle = \frac{\sqrt{N-1}}{\sqrt{N}} \frac{1}{\sqrt{N-1}} \sum_{x \neq x_0} |x\rangle + \frac{1}{\sqrt{N}} |x_0\rangle$$

In terms of $|\mathcal{T}\rangle$ and $|\mathcal{F}\rangle$, this is $|\Psi_0\rangle = \frac{\sqrt{N-1}}{\sqrt{N}} |\mathcal{F}\rangle + \frac{1}{\sqrt{N}} |\mathcal{T}\rangle$

The intermediate states in Grover's algorithm

We have a series of states: $|\Psi_0\rangle \mapsto |\Psi_1\rangle \mapsto |\Psi_2\rangle \mapsto \dots$

The initial state is $|\Psi_0\rangle = \frac{\sqrt{N-1}}{\sqrt{N}}|\mathcal{F}\rangle + \frac{1}{\sqrt{N}}|\mathcal{T}\rangle$.

- ▶ We write this as $\cos(\theta_0)|\mathcal{F}\rangle + \sin(\theta_0)|\mathcal{T}\rangle$,

where

$$\cos(\theta_0) = \frac{\sqrt{N-1}}{\sqrt{N}}, \quad \sin(\theta_0) = \frac{1}{\sqrt{N}}$$

Claim: We can write the j^{th} state as $\cos(\theta_j)|\mathcal{F}\rangle + \sin(\theta_j)|\mathcal{T}\rangle$.

By induction ...

1. We know that the starting state $|\Psi_0\rangle$ is in this form ...
2. Given

$$|\Psi_k\rangle = \cos(\theta_k)|\mathcal{F}\rangle + \sin(\theta_k)|\mathcal{T}\rangle$$

we show that $|\Psi_{k+1}\rangle$ is also in this form.

This will take a little calculation ...

Some sums!

$$\begin{aligned}
 & \cos(\theta_k)|\mathcal{F}\rangle + \sin(\theta_k)|\mathcal{T}\rangle \\
 & \quad \downarrow U \\
 & \cos(\theta_k)|\mathcal{F}\rangle - \sin(\theta_k)|\mathcal{T}\rangle \\
 & \quad \downarrow V \\
 & (I_n - 2|\Psi_0\rangle\langle\Psi_0|)(\cos(\theta_k)|\mathcal{F}\rangle - \sin(\theta_k)|\mathcal{T}\rangle) \\
 & =
 \end{aligned}$$

$$\cos(\theta_k)|\mathcal{F}\rangle - \sin(\theta_k)|\mathcal{T}\rangle - 2|\Psi_0\rangle(\cos(\theta_k)\langle\Psi_0|\mathcal{F}\rangle - \sin(\theta_k)\langle\Psi_0|\mathcal{T}\rangle)$$

Yuck!

Some more sums!

Not forgetting the overall sign flip, we are left with $|\Psi_{k+1}\rangle =$
 $-\cos(\theta_k)|\mathcal{F}\rangle + \sin(\theta_k)|\mathcal{T}\rangle + 2|\Psi_0\rangle(\cos(\theta_k)\langle\Psi_0|\mathcal{F}\rangle - \sin(\theta_k)\langle\Psi_0|\mathcal{T}\rangle)$

This can be expanded!

By definition:

$$\blacktriangleright \langle\Psi_0|\mathcal{F}\rangle = \cos(\theta_0)$$

$$\blacktriangleright \langle\Psi_0|\mathcal{T}\rangle = \sin(\theta_0)$$

The never-ending calculation ...

Using these identities, $|\Psi_{k+1}\rangle =$

$$-\cos(\theta_k)|\mathcal{F}\rangle + \sin(\theta_k)|\mathcal{T}\rangle + 2|\Psi_0\rangle(\cos(\theta_k)\cos(\theta_0) - \sin(\theta_k)\sin(\theta_0))$$

Also, $|\Psi_0\rangle = \cos(\theta_0)|\mathcal{F}\rangle + \sin(\theta_0)|\mathcal{T}\rangle$.

This gives $|\Psi_{k+1}\rangle =$

$$\begin{aligned} |\mathcal{F}\rangle \cdot (-\cos(\theta_k) + 2\cos(\theta_0)(\cos(\theta_k)\cos(\theta_0) - \sin(\theta_k)\sin(\theta_0))) \\ + \\ |\mathcal{T}\rangle \cdot (\sin(\theta_k) + 2\sin(\theta_0)(\cos(\theta_k)\cos(\theta_0) - \sin(\theta_k)\sin(\theta_0))) \end{aligned}$$

Some basic trigonometric identities:

We now appeal to some basic trigonometry:

Very useful identities:



$$\cos^2(\theta_0) = \frac{1}{2} + \frac{1}{2} \cos(2\theta_0)$$



$$\sin^2(\theta_0) = \frac{1}{2} - \frac{1}{2} \cos(2\theta_0)$$



$$\sin(\theta_0) \cos(\theta_0) = \frac{1}{2} \sin(2\theta_0)$$

Approaching a final answer !

Using these basic trig. identities:

$$\begin{aligned}
 |\Psi_{k+1}\rangle &= |\mathcal{F}\rangle (\cos(\theta_k) \cos(2\theta_0) - \sin(\theta_k) \sin(2\theta_0)) \\
 &\quad + \\
 &\quad |\mathcal{T}\rangle (\sin(\theta_k) \cos(2\theta_0) + \cos(\theta_k) \sin(2\theta_0))
 \end{aligned}$$

Reducing further,

$$|\Psi_{k+1}\rangle = \cos(\theta_k + 2\theta_0)|\mathcal{F}\rangle + \sin(\theta_k + 2\theta_0)|\mathcal{T}\rangle$$

$$|\Psi_{k+1}\rangle = \cos(\theta_{k+1})|\mathcal{F}\rangle + \sin(\theta_{k+1})|\mathcal{T}\rangle$$

where $\theta_{k+1} = \theta_k + 2\theta_0$.

The Grover operator is a *rotation*.

At each step, the Grover operator:

“Rotates the state away from $|\mathcal{F}\rangle$, and towards $|\mathcal{T}\rangle$,
by an angle of $2\theta_0 = 2 \sin^{-1} \left(\frac{1}{\sqrt{N}} \right) \sim \frac{2}{\sqrt{N}}$.”

Points to remember

1. $|\mathcal{F}\rangle$ and $|\mathcal{T}\rangle$ are orthogonal. — i.e. at an angle of $\frac{\pi}{2}$
2. The starting state $|\Psi_0\rangle$ is at an angle of $\theta_0 \lll \frac{\pi}{2}$ to $|\mathcal{F}\rangle$.

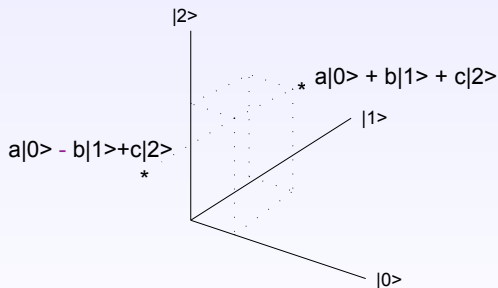
After $\frac{\pi\sqrt{N}}{4}$ steps, the state is (approximately) $|\mathcal{T}\rangle$.

Grover's algorithm — some questions!

1. How close is the final state to $|T\rangle$?
 - ▶ What does this mean in terms of probabilities?
 - ▶ Does this get better, or worse, as N increases?
2. What happens when we do too many steps?
3. The rotations are very 'geometric'
— can the whole algorithm be seen in these terms?

The Grover operator as reflections

Changing the sign of a branch is a *reflection*:



Changing the sign of the $|1\rangle$ factor is a *reflection* in the plane spanned by $|0\rangle$, $|2\rangle$.

The Grover operator as 2 reflections

Both the simple

$$U = I - 2|x_0\rangle\langle x_0|$$

and the (slightly more complex)

$$V = I_n - 2|\psi_0\rangle\langle\psi_0|$$

oracles describe **reflections**.

Some basic geometry:

The product of two reflections is a rotation ...
through twice the angle between the planes of reflection.

2 reflections make a rotation

- ▶ $U = I - 2|x_0\rangle\langle x_0|$ reflects through:
the hyperplane perpendicular to $|x_0\rangle$.
- ▶ $V = I_n - 2|\psi_0\rangle\langle\psi_0|$ reflects through:
the hyperplane perpendicular to $|\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$.

The angle between these planes is $\theta_0 = \sin^{-1}(\langle\psi_0|\mathcal{T}\rangle)$.

At each iteration, there is a rotation of $2\theta_0$.

An exercise

Exercise:

In the $|\mathcal{T}\rangle, |\mathcal{F}\rangle$ plane, draw:

1. The *initial state*.
2. The two reflections & (hence) the Grover rotation.
3. The *final state*.

Quantum Computation

Lecture 12

Sam Braunstein

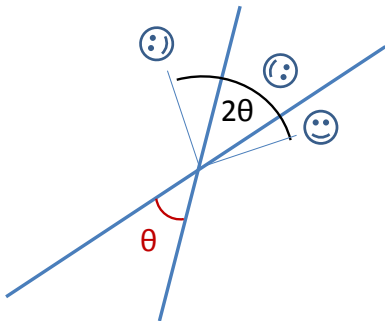
Time for filling in student
feedback sheets

Grover's algorithm revisited

Recall the Grover iterator \hat{G} is the product of two reflections.

But there's a very simple geometric fact:

A double reflection = rotation by twice the angle
between the planes of reflection.



Grover's algorithm revisited

Recall $|\psi_0\rangle = \cos \theta_0 |\mathcal{F}\rangle + \sin \theta_0 |\mathcal{T}\rangle$

1st reflection is through $|\mathcal{T}\rangle$

2nd reflection is through $|\psi_0\rangle$

The angle between these 'planes' is θ_0 : $\langle \psi_0 | \mathcal{T} \rangle = \sin \theta_0$

At each iteration, therefore, we rotate the state by an angle $2\theta_0$

Bingo! It really was simple after all.

What if there's more than one solution?

If $f(x)$ has M values of x for which $f(x)=1$,

then the target state is
$$|\mathcal{T}\rangle = \frac{1}{\sqrt{M}} \sum_{x:f(x)=1} |x\rangle$$

so
$$|\mathcal{F}\rangle = \frac{1}{\sqrt{N-M}} \sum_{x:f(x)=0} |x\rangle$$

and
$$|\psi_0\rangle = \sqrt{\frac{N-M}{N}} |\mathcal{F}\rangle + \sqrt{\frac{M}{N}} |\mathcal{T}\rangle$$

Everything follows through as before, except the angle between the planes is $\langle \psi_0 | \mathcal{T} \rangle = \sin \theta_0 = \sqrt{\frac{M}{N}}$

Again we rotate by $2\theta_0$ at each iteration, so we reach $|\mathcal{T}\rangle$

in $k \cdot 2\theta_0 \approx \frac{\pi}{2}$ i.e., $k \approx \frac{\pi}{4} \sqrt{\frac{N}{M}}$

What if there's more than one solution?

This is fine, so long as we know how many solutions there are. But what if we don't know how this?

How do we know when to stop?

What can go wrong? At step k the state is

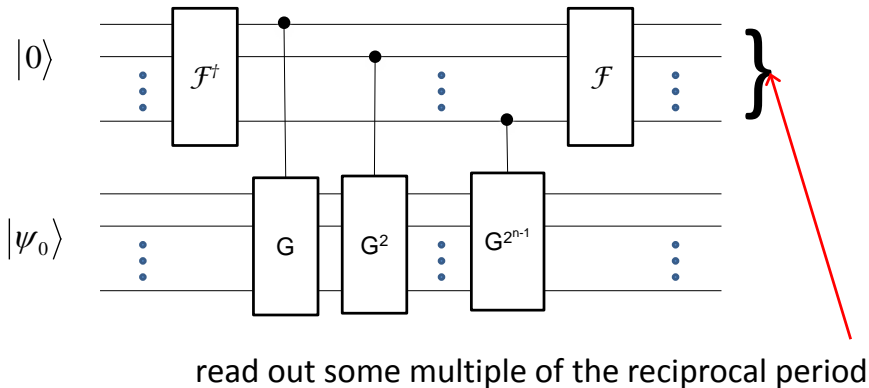
$$|\psi_0\rangle = \cos[(2k+1)\theta_0]|\mathcal{F}\rangle + \sin[(2k+1)\theta_0]|\mathcal{T}\rangle$$

If we go too far we can miss the target, however, if we continue we go around and around.

We now have an essentially periodic function, and all we need to do is find its period!!

Quantum Counting

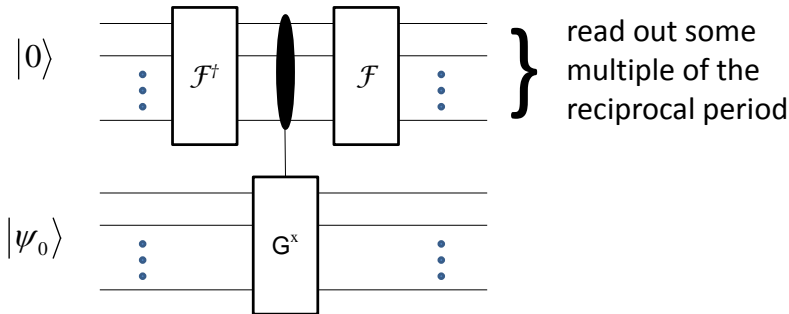
By combining Grover's algorithm with Shor's period finding algorithm, we can count the number of solutions. Here a circuit to achieve it:



Quantum Counting

Surprisingly, still only $O(\sqrt{N})$ queries are needed.

A shorthand notation for this circuit is:



QUCO topics covered

Quantum States:

- normalization
- probabilities for measurement
- notation: bras, kets, vectors, inner products, etc.
- entanglement/non-factorizable states
- tensor products

QUCO topics covered

Gates/Evolution:

- linearity
- unitarity
- tensor products
- special gates: H , R_k , XOR, Toffoli, etc.
- universal sets of gates
- multiqubit Hadamard
- QFT and its implementation

QUCO topics covered

Algorithms:

- Deutsch alg.
- one-out-of-four search
- Simon's alg.
- Deutsch-Jozsa alg.
- Shor's alg.
- Grover's alg. (and its geometric interpretation)
- Quantum counting