Quantum Automata, Machines and Complexity

Anuj Dawar

Computer Laboratory
University of Cambridge

Quantum Computing

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

Central research tasks include:

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

Mediating these two are models of computation.

Talk Outline

In this lecture, we'll look at some concepts from the classical theory of computation:

- Finite-state machines
- Turing machines
- Computational complexity

and see what their quantum counterparts might look like.

Bits

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

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

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

Qubits

A *quantum bit* can exist in a *superposition* of states.

In general

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

where, α and β are complex numbers, satisfying

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

Any attempt to measure the state results in $|0\rangle$ with probability $|\alpha|^2$, and $|1\rangle$ with probability $|\beta|^2$.

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

Further measurements will always yield the same value.

Entanglement

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

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

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

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

But, compare

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

Unitary Evolution

A quantum system that is not measured (i.e. does not interact with its environment) evolves in a unitary fashion.

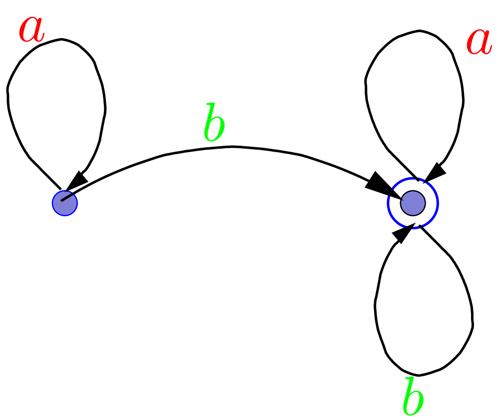
That is, it's evolution in a time step is given by a *unitary linear operation.*

Such an operator is described by a matrix U such that

$$UU^* = I,$$

where U^* is the *conjugate transpose* of U.

Finite State Systems



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

Its operation can be described by a pair of matrices.

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

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

DFAs and Matrices

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

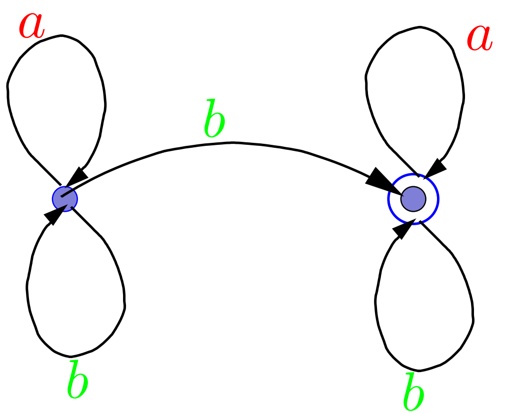
> Each column of each matrix contains exactly one non-zero entry, and that entry is 1.

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

> Multiplication of matrices corresponds to composition of words.

$M_w|q_i\rangle = |q_j\rangle$ if there is a path labelled w from state i to state j, where $|q_i\rangle$ is the vector containing a 1 in the ith position and 0 everywhere else.

Nondeterministic Automata



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

However, it may make the transition to the final state on any occurrence of b, not just the first one

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

NFAs and Matrices

$$M_{bb} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

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

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

Probabilistic Automata

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

with the proviso that each column adds up to 1.

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

$$M_a M_b = \begin{pmatrix} 0.4 & 0.2 \\ 0.6 & 0.8 \end{pmatrix}$$

gives, in position (i, j) the probability that string ba takes you from state i to state j.

Language Accepted

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

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

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

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

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

Quantum Automata

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

We also require each of the matrices to be *unitary*.

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

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

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

Acceptance Probabilities

If q_1 is the starting state of the automaton,

$$M_w|q_1\rangle$$

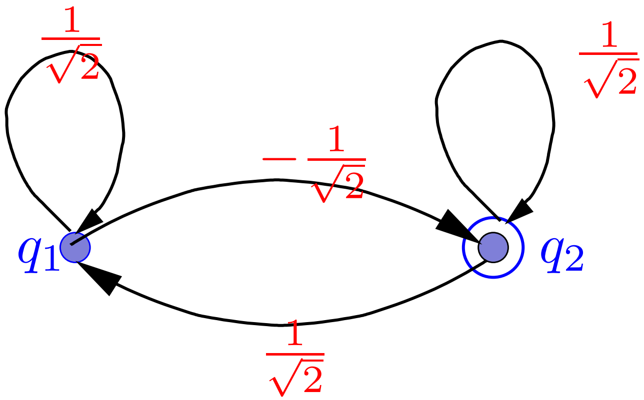
is a vector describing a superposition of states. If the jth entry in the vector is α_j then α_j is the *probability amplitude* that the automaton reaches state q_j.

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

We can define language acceptance exactly or by bounded probability.

Interference

Consider the automaton in a one letter alphabet defined as:



$$M_a = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}$$

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

While there are two distinct paths labelled aa from q_1 back to itself, and each has non-zero probability, the net probability of ending up in q_1 is 0.

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

Language Accepted

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

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

To make the model more powerful, several variants have been studied:

- two-way automata;
- measure-many automata;
- automata with classical and quantum states.

Turing Machines

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

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

A machine has infinitely many possible *configurations* (reserving the word "state" for a member of Q).

Each configuration c is determined by a state, the contents of the tape (a finite string) and the position of the head.

Configurations and Computations

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

$$c_0 \longrightarrow c_1 \longrightarrow \cdots \longrightarrow c_f$$

eventually reaches an accepting state.

If the length of the computation is bounded by a polynomial in the length of x, the language accepted by the machine is in P.

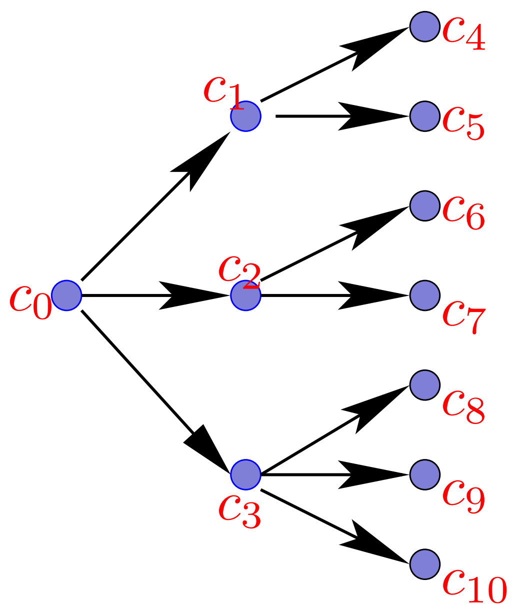
The action of the Turing machine can equivalently be described as a linear operator M on an *infinite-dimensional* space.

The set of configurations form a basis for the space.

Nondeterministic Turing Machines

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

This gives rise to a tree of configurations:



A configuration may occur in several places in the tree.

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

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

Probabilistic Machines

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

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

However, how much information can be encoded in a single entry?

We require the entries α to be *feasibly computable*. That is, there is a feasibly computable f such that:

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

BPP

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

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

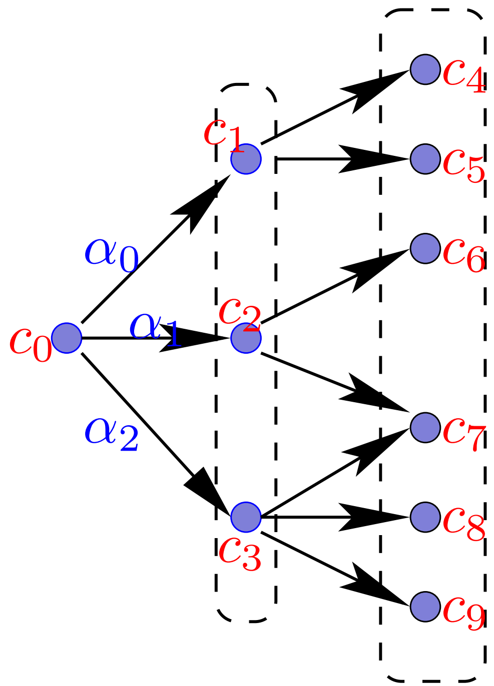
The class of languages is unchanged if we replace $\frac{2}{3}$ and $\frac{1}{3}$ by $1 - \epsilon$ and ϵ, for any $\epsilon < \frac{1}{2}$, or indeed the set of all feasibly computable probabilities with $\{0, \frac{1}{2}, 1\}$.

The only inclusion relations we know are P$\subseteq$NP and P$\subseteq$BPP.

Primality testing, long known to be in BPP was recently shown to be in P.

Quantum Turing Machines

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



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

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

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

BQP

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

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

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

Shor has shown that the factorisation problem is in BQP.

It is not known to be in BPP.