Chapter 0

Introduction to quantum computation

We give a very brief historical introduction which explains where the ideas of quantum computation came from. Let us begin with classical computation.

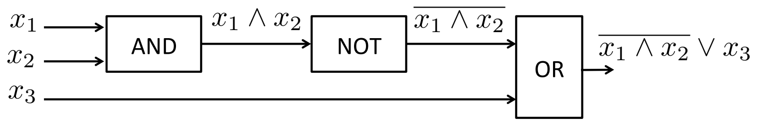
Suppose that we want to compute a Boolean function

$$f : (x_1 \dots x_N) \in \{0,1\}^N \mapsto f(x_1 \dots x_N) = (x'_1 \dots x'_N) \in \{0,1\}^M.$$

A theorem of Emil Post dating back to 1921 states:

Theorem 1 *Any Boolean function can be computed from a directed acyclic graph where nodes are the logical gates* {AND, OR, NOT, COPY}. *These gates form a universal set (not the smallest possible).*

A directed acyclic graph is just a circuit that connects these logical gates, does not have loops, and has an input and an output. For example,

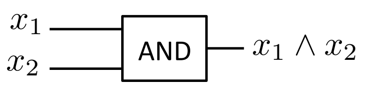

x_1
x_2
x_3
AND
$x_1 \wedge x_2$
NOT
$\overline{x_1 \wedge x_2}$
OR
$\overline{x_1 \wedge x_2} \vee x_3$

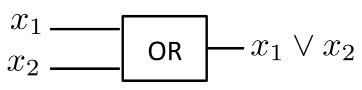
compute the Boolean function $f(x_1, x_2, x_3) = \overline{x_1 \wedge x_2} \vee x_3$, which takes 3 input bits and has an output bit.

This theorem is at the basis of the "classical circuit model" of computation. Such circuits are implemented in electronic chips.

In 1960's Rolf Landauer argued that such each time one bit of information is lost (or erased) in a processing task a minimum amount $KT \ln 2$ of heat is dissipated, where T is the temperature of the environment. Roughly speaking, the change of entropy when a bit is lost is $\ln 2$ and thermodynamics is the heat flux associated to this entropy change.

In the circuit model based on Post's theorem the gates AND, OR dissipate heat because the operators



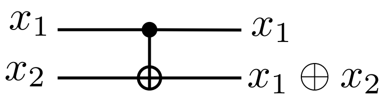


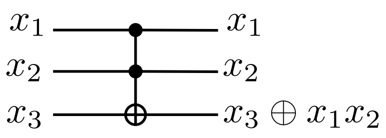
are not reversible (a bit of information is lost). On the other hand, the gate OR is reversible can thereby be implemented through a physical process that does not dissipate heat.

In the 70's and 80's a number of researches (among them Bennett and Benioff) asked the question: can we compute Boolean functions in a "reversible" manner that, in theory at least, does not involve heat disspation. The answer is yes!

Theorem 2 *Any Boolean function can be computed in a reversible way using a directed acyclic graph built with the set of universal gates* {NOT*,* CNOT*,* CCNOT}*.*

The CNOT and CCNOT (Teffoli) gates are





An easy exercise shows that the AND, OR, COPY gate can be replaced by {NOT, CNOT, CCNOT}. This however is at the expense of expanding the memory or the number of input and output bits.

This theorem forms the basis of the "classical reversible model of computation". With perfectly good physical components such circuits do not in principle dissipate heat in the computation process. Moreover, one can recover the input from the output.

Since it is possible to compute Boolean functions in a reversible way, and without heat dissipation, it is reasonable to ask whether quantum mechanics would bring same limitation of its own to the computation process. This is all the more relevant in view of the miniaturization process of circuitry and electronic devices. Today these attain the nanoscale or even smaller scales, and at such scales quantum laws start to matter.

Soon after work by Feymman, Bennett, Wiesner, Deutsch and others put forward the "quantum circuit computation model".

As we will see quantum circuits are generalizaions of reversible classical circuits. They allow to compute Boolean functions by exploring in "parallel" all classical inputs and may therefore speedup calculations. Thus instead of bringing new limitations, quantum mechanics in fact brings new advantages!

At first, quantum circuit may sound like strange objects. Because now states of "quantum bits" are more general than $\{0, 1\}$ but form a vector space $\mathbb{C}^2$ (as we will see). The inputs of the quantum circuit are "vectors" (quantum state) and the logical operations become "matrices" (unitary operations). The output of the quantum circuit is also a vector.

Before we continue on this path, we must go through the basic principle of quantum mechanics. This will be done in an axiomatic way which almost completely reduces the formalism to linear algebra.

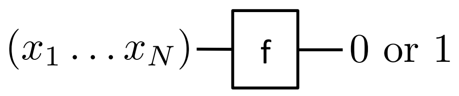
Let us give sketchy idea of the kind of problems that quantum computation can solve.

Suppose that

$$f : (x_1 \dots x_N) \in \{0, 1\}^N \mapsto f(x_1 \dots x_N) \in \{0, 1\}$$

is a Boolean function that is constant or balanced. Constant means that f takes only the value 0 for all inputs or only the value 1 for all inputs. Balanced means that f takes the value 0 for half of the inputs and 1 for the other half.

The Deutsch problem is to determine if f is balanced or constant by asking a minimal number of questions to a black box



In classical computation we have to ask in the worst case $2^{N-1} + 1$ questions (think about it!). So the complexity of this problem is exponential.

We will see that there exists a quantum algorithm that operates on a quantum circuit, such that one operation suffices! This is at first very surprising. As we will see the quantum algorithm explores at the same time all possible classical 2^N inputs. This is called quantum parallelism.

Another feature of quantum algorithms, besides parallelism, is that they are randomized. As we will see the output of a quantum circuit has to be read or measured, an operation that involves a random outcome. We will therefore have to bias in some way the probability of success on the correct outcome. In the Deustch problem described beforehand this probability of success is strictly 1. But this is a special case.

There are mainly two classes of quantum algorithms that we will study.

The first one is a class of algorithms that find a hidden symmetry in a mathematical object. Such algorithms are based on a concept of Quantum Fourier Transform (QFT). The QFT plays the role of a kind of signal processing tool that analyzes the periods or symmetries contained in a quantum state. This is also analogous to diffraction experiments used to determine the atomic arrangement of crystals. As we will see the most famous algorithm of this class is the Shor's algorithm that factorizes integers and has polynomial complexity. It is not known if there exists a classical algorithm with polynomial complexity. All known classical algorithms have super-polynomial (almost exponential) complexity. So typically for factorization quantum computation offers exponential speed-up.

The second class of algorithms are search algorithms in unsorted databases. The prominent representative here is the Grover's algorithm which offers quadratic speed-up (not exponential) with respect to classical methods.

Finally, if time permits we will study quantum error correcting codes, the analog of classical parity check codes.