November 7, 2017

Short introduction to Quantum Computing

Joris Kattemölle

QuSoft, CWI,
Science Park 123, Amsterdam, The Netherlands
Institute for Theoretical Physics, University of Amsterdam,
Science Park 904, Amsterdam, The Netherlands

E-mail: j.j.kattemolle@uva.nl

Abstract: These notes offer physicists a brief introduction to quantum computation. The reading time is about two hours if you also make the small exercises. This should give you a good but basic understanding of the notation and the most important concepts.

Contents

1 Why quantum computers?

Quantum computers can do certain tasks fundamentally faster than normal computers. By fundamentally, we do not mean the absolute time it cost a (quantum)computer to solve a certain problem, but rather how the time for computation t scales as a function of the input size n. For some problems, t goes like a polynomial, $t(n) = \text{poly}(n)$. We will colloquially refer to these problems as 'easy'. Problems for which t goes faster than any polynomial of n are 'hard', for example $t(n) = O(e^{cn})$ for some positive constant c.

Some problems that fall into the category 'hard' for normal computers, fall into the category 'easy' for quantum computers. So, there is a third category: 'hard for normal computers but easy for quantum computers'. It is a common misconception that all problems fall in this category. Two examples of problems that do fall in the last category are the factoring of numbers and the simulation of quantum mechanical systems. Factoring numbers is mainly the reason why quantum computers draw so much attention and money; if you can factor numbers fast, you can crack current (RSA) encryption.

Maybe of more interest to physicists, quantum computers could be used to simulate quantum mechanical systems. As you might know, it is hard to simulate a quantum mechanical system on a classical computer, and it is easy to see why. To store an arbitrary state of a grid of 16×16 spin-1/2 particles, we would have to store roughly $2^{16^2} = 2^{256}$ complex numbers. Say we approximate every complex number by only 1 byte (8 bits), and that we only need 100 atoms to make one classical bit. The number of atoms needed is already as large as the number of atoms in the universe! ($\sim 10^{80}$) So, we will never be able to fully simulate a grid of 16×16 spin-1/2 particles on a classical computer, no matter the advances of technology.

Vice versa, this means that if this 'power' of quantum mechanical systems can be harnessed by building a 'universal, tunable quantum mechanical system', and if general computational problems can be translated to the evolution of *some* quantum mechanical system, some of the 'hard' problems can be solved by simply letting the programmed quantum mechanical system evolve. This is the main idea behind quantum computation.

2 Computers

Before we are able to appreciate the differences between classical- and quantum computers, we need to know how an normal computer works. A normal computer takes a string of bits as input, and outputs another string of bits. That's all. We will now see how this goes in a bit more detail.

One bit The *state space* of a single classical bit is

$$\mathbb{Z}_2 = \{0, 1\}. \tag{2.1}$$

That is, a bit is either in the state $x = 0$, or $x = 1$. If we like to have a visual representation of the state space, we could put a dot for every possible state. For a classical bit then, the state space is formed by just two dots,

$$\bullet\ (0)$$

$$\bullet\ (1).$$

Later we will compare this to the state space of a quantum-mechanical bit.

Multiple bits The state space of n classical bits is given by

$$\{0,1\}^n.$$

For example, eleven bits could be in the state $y = 11011010110$. It is customary to denote the value of the jth bit (where we count from right to left, starting with 0) by y_j. That is,

$$y = y_{n-1} \cdots y_1 y_0.$$

Note that in general, n bits can be in 2^n different states. If we wish, we can label these states by numbers. In this way, we can think of bitstrings as representing numbers. This representation is called the *binary representation.* The convention is to use the following labeling

$$\begin{array}{ll} \ldots 000 & (0) \\ \ldots 001 & (1) \\ \ldots 010 & (2) \\ \ldots 011 & (3) \\ \ldots 100 & (4) \\ & \vdots \end{array}$$

In general, this means that a bitstring with a '1' only on place j represents the number 2^j. In this way the number represented by an arbitrary bitstring can easily be found.

Question 2.1.

a) The usual way of counting on the hand, i.e. the number of fingers up is the number that is represented, is using the *unitary representation* of numbers. Using the binary representation, figure out how you can count from 0 to 31 on one hand. To what number can you count on two hands?

b) What number does $y = 11011010110$ represent (using the usual representation of numbers)?

Physically, bits can be made using any physical system that (at some level) has a state space of size two. Examples are the states 'up' and 'down' of a light switch, or 'no current' or 'a current' through a wire.

Computers As mentioned before, a computer takes a bitstring as input, and then outputs another bitstring. In other words, a computer is a machine that implements a function

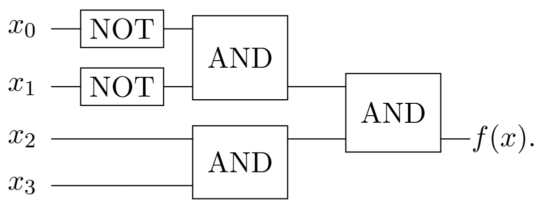
$$f : \{0,1\}^n \to \{0,1\}^m. \tag{2.2}$$

Now any such function can be divided into 'little functions' that only take bitstrings of size one or two as input. These little functions are called *gates*. A set of gates that can be used to construct an arbitrary function, as described above, is called a *universal gate set.* An example is the set {NOT, AND} (together with the possibility to put 'wires' between any input and output, and the possibility to branch wires).

Question 2.2. I assume you are already familiar with the gates {NOT, AND}. Say we need a computer that implements the following function,

$$f(x) = \begin{cases} 1 \text{ if } x = 1100 \\ 0 \text{ otherwise} \end{cases}.$$

To implement this function, the following circuit can be constructed



a) Draw the circuit that implements the following function for an input x of four bits.

$$f(x) = \begin{cases} 1 \text{ if } x \text{ has two 1's} \\ 0 \text{ otherwise} \end{cases}.$$

3 Quantum computers

A quantum computer works with quantum bits (qubits). It starts with a collection of qubits in the input state, performs some operations, and ends with the same qubits which are now in the output state. We can measure (part) of the output state to get out classical information.

One qubit Formally, the state space of a qubit is given by

$$\mathcal{H}_2 = \mathbb{C}^2,$$

where additionally, we require the vectors in this space to be normalized. I hope you appreciate the difference with the classical bit: we now have a complex vector space of dimension two (i.e. two basis vectors and two continuous complex coefficients), instead of $\mathbb{Z}_2$ (i.e. just two points, cf eq. 2.1).

It is customary to use a basis where the basis vectors are denoted by $|0\rangle$ ('spin up') and $|1\rangle$ (or 'spin down', if you like). A general state of a single qubit can then be written as

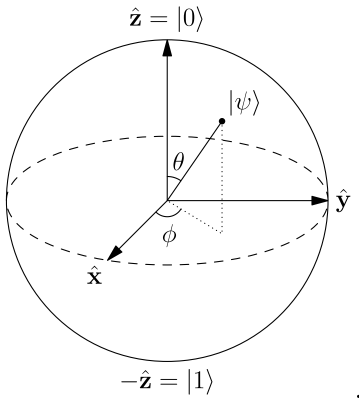
$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle\,, \tag{3.1}$$

with the restriction that $|\alpha|^2 + |\beta|^2 = 1$.

With the help of this restriction, and the fact that we can discard an overall phase (a difference in overall phase does not lead to any measurable difference), we can make a nice visualization of the state space of one qubit. It can quite easily be shown that there are parameters $\theta \in [0, \pi)$ and $\varphi \in [0, 2\pi)$ such that any state can be written as (cf eq. 3.1)

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle .$$

Any state can now be depicted as a point on the unit sphere[1],



This sphere is called the Bloch-sphere, and the vector that defines the state is called the Bloch-vector.

To summarize, *the state space of a bit is depicted by two dots. The state space of a qubit is a sphere.* Note that a continuous, spherical state space on itself is not a quantum mechanical property; the state space of a classical rigid pendulum which can rotate in two directions is also a sphere (if we discard momentum). There are, however, two properties that are truly quantum mechanical. (i) It is impossible to measure the x, y and z component of the Bloch-vector at the same time. (ii) Although the Bloch-vector can point in any direction, we can only find the outcome 1 (with probability $\cos^2(\theta/2)$) or -1 (with probability $1 - \cos^2(\theta/2) = \sin^2(\theta/2)$) upon measurement of the z-component. (Something similar holds for the other components.) This is the reason a qubit can hold only one classical bit of information although we need two real numbers to define the state.

Multiple qubits The state space of n qubits is given by the tensor product space of n one-qubit spaces. That is,

$$|\varphi\rangle \in (\mathcal{H}_2)^{\otimes n}.$$

For example, if $n = 3$, any state can be written as $|\varphi\rangle = c_0|000\rangle + c_1|001\rangle + \ldots c_8|111\rangle$.[2] Note that in general, we need approximately 2^n complex coefficients to fully describe the state. Sadly, we cannot draw nice pictures to depict the state space anymore.

[1]Figure made by Wikipedia-user Glosser.ca.

[2]As is customary, we use shorthand notation where $|\psi_{n-1}\rangle \otimes |\psi_1\rangle \otimes \ldots \otimes |\psi_0\rangle = |\psi_{n-1}\psi_1 \ldots \psi_0\rangle$.

Quantum computer A quantum computer is a machine that takes a (possibly classical) input state $|\varphi\rangle$, prepares an output state $|\varphi'\rangle = U|\varphi\rangle$ where U is some unitary transformation, and measures the output state to get a classical answer. That is (excluding the measurement),

$$|\varphi'\rangle = \overset{\text{Quantum computer}}{\overset{\downarrow}{U}}\, |\varphi\rangle\,.$$

In analogy with classical computers, any U can be made up out of small u's that act only on one or two qubits. These are called *quantum gates.*

4 Quantum gates and circuit notation

Here we will show some of the most common quantum gates and how to combine them into a quantum circuit.

General single-qubit gates In quantum circuit notation, a qubit is represented by a horizontal line. In the Schrödinger picture, states evolve under time. In the following diagram, time runs from left to right, and whenever an operator acts on a qubit, we write that operator in a box,

$$|\psi\rangle \;\text{——}\boxed{\mathcal{O}}\text{——}\; \mathcal{O}\,|\psi\rangle\,.$$

Pauli X,Y,Z gates Let us look at some actual gates that can act on a single qubit. Among the most important gates are the Pauli-gates you should already be familiar with,

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

The Pauli-X maps $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$, and is hence referred to as a 'bit flip'. In circuit notation,

$$|0\rangle \;\text{——}\boxed{X}\text{——}\; |1\rangle\,, \qquad |1\rangle \;\text{——}\boxed{X}\text{——}\; |0\rangle\,. \qquad \text{(bit flip)}$$

You can work out yourself what this would look like for Y. (Sadly, it doesn't have a nickname.) The Pauli-Z gate maps $|0\rangle$ to $|0\rangle$ and $|1\rangle$ to $-|1\rangle$. Hence it is referred to as a 'phase flip',

$$|0\rangle \;\text{——}\boxed{Z}\text{——}\; |0\rangle\,, \qquad |1\rangle \;\text{——}\boxed{Z}\text{——}\; -|1\rangle\,. \qquad \text{(phase flip)}$$

Hadamard and '$\pi/8$' gate A gate that can be used to make superpositions is the *Hadamard* gate,

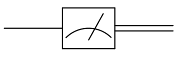
$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Finally, we have the T-gate or '$\pi/8$' gate

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}.$$

Question 4.1. Using the examples of the X and Z-gates, how does the action of the Hadamard and the T- gate look like in circuit notation?

Measurement The final single-qubit operation we consider is *measurement*,

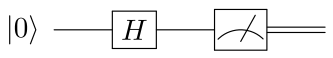


.

This measurement is always in the so-called computational basis (the Z-basis). It outputs the measurement result (± 1), which is classical information (depicted by two lines).

Question 4.2.

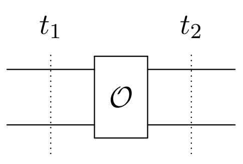
a) What use has the following circuit?



b) This little 'quantum computer' is already commercially available and consists out of a photon source, a beam splitter and two photon detectors. How do you think this setup schematically look like?

Multiple-qubit gates There are also gates that act on two qubits simultaneously (or more). The *space* the two qubits live in is depicted by two parallel horizontal lines. Note that the two qubits need not to be in a product state of the form $|\psi\rangle \otimes |\varphi\rangle$! In general, they are of the form $|\psi\rangle = \sum_{i,j} c_{ij} |i\rangle \otimes |j\rangle$ for some complex coefficients c_{ij}.

When the two qubits are acted upon by a two-qubit operator $\mathcal{O}$, we write that operator in a box like so,



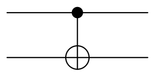
.

At t_1 we have a state, which can be represented by a column vector with four entries. The gate $\mathcal{O}$ is just some 4×4 matrix that acts on it. At t_2 we end up with the result. In other words, at time t_1 we have the two-qubit state $|\psi\rangle = \sum_{i,j} c_{ij} |i\rangle \otimes |j\rangle$, than at time t_2 we have the state $\mathcal{O}|\psi\rangle = \sum_{i,j} c'_{ij} |i\rangle \otimes |j\rangle$, where the c'_{ij} depend on the details of $\mathcal{O}$.

The CNOT gate We can now look at some examples of two-qubit gates. The most common one is he controlled-not (CNOT) gate, defined by

$$\begin{aligned}
\mathrm{CNOT}\,|00\rangle &= |00\rangle\,, \\
\mathrm{CNOT}\,|01\rangle &= |01\rangle\,, \\
\mathrm{CNOT}\,|10\rangle &= |11\rangle\,, \\
\mathrm{CNOT}\,|11\rangle &= |10\rangle\,.
\end{aligned}$$

This can be summarized as: *the CNOT applies a bit flip to the second qubit (the target bit) iff the first qubit (the control bit) is in the state* $|1\rangle$. We can, of course, figure out what the CNOT does on arbitrary states by using linearity. In circuit notation, the CNOT is depicted by

 ,

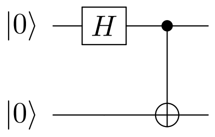
where '•' is the on the control bit, and '⊕' is on the target bit.

Question 4.3.

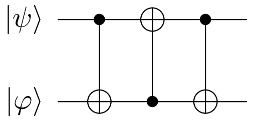
a) Write out the CNOT gate as a matrix.

What state is at the output of the following circuits? Interpret the results.

b)

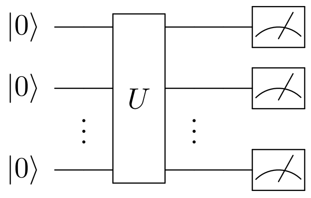


c)



Tip: you only need to work out what the circuit does to basis states, e.g. $|\psi\rangle \otimes |\varphi\rangle = |0\rangle \otimes |1\rangle$. By linearity, you can then figure out what it does to any state.

Universal sets of quantum gates. In circuit notation, a quantum computer is simply,

 .

There could, of course, be some additional classical input, but we can absorb this into the definition of U. Also we have, without loss of generality, assumed that the initial state is $|0\rangle^{\otimes n}$, and that at the end, all qubits are measured. In analogy with the classical computer, any transformation U as above, can be split into the elementary gates from a universal gate set. One example of such a set is $\{H, T, CNOT\}$. (So, the Pauli operators X, Y, Z can be constructed from gates in this set as well.)

5 Physical quantum computers

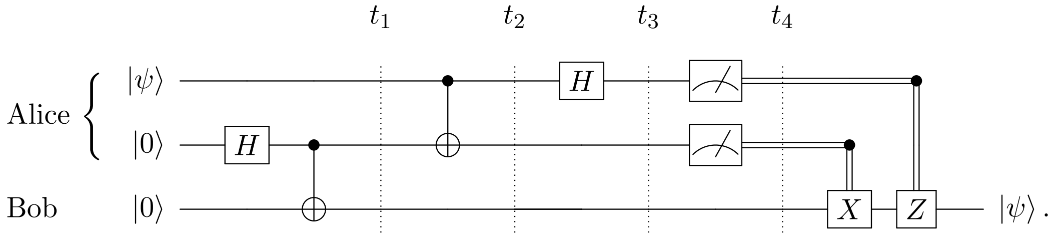
So far, we have not discussed how physical qubits, and the gates that act on them, are made. Giving a good account of these matters is out of the scope of these notes. I will, however, briefly describe two out of the many possible architectures for a quantum computer.

The first one is the *trapped ion quantum computer.* With the help of static and dynamic electromagnetic fields, a linear array of ions is made to levitate (trapped) in a vacuum chamber. The states that are used as the two states of the qubit ($|0\rangle$ and $|1\rangle$) are an electronic ground state and a metastable exited state. The transition between these states can be achieved by resonantly driving it with lasers. Measurement is performed by using fluorescence; one of the states results the ion to fluoresce when illuminated by a laser with the right wavelength, while the other state remains dark.

The second is *super conducting quantum computing.* With the help of super conducting wires, an super conduction LC circuit is built. As you might remember, a classical LC-circuit is mathematically equivalent to a classical harmonic oscillator (or pendulum). Similarly, the Hamiltonian of a superconducting circuit is that of a quantum harmonic oscillator. As the $|0\rangle$ and $|1\rangle$, the ground state and the first exited state of the oscillator are used. Control and readout is done by semi-static and dynamical electromagnetic fields. To ensure that the transition to $|2\rangle$ and higher states is not driven, an anharmonicity is build into the circuit, which makes the energy between $|0\rangle$ and $|1\rangle$ much larger or smaller than the energy difference between $|0\rangle$ and $|2\rangle$, and between $|1\rangle$ and $|2\rangle$.

6 Teleportation

We will now see an example of a somewhat larger circuit; that for teleportation. Teleportation is the transmission an unknown quantum state, using only classical communication. The circuit is given by



I will now take you through the entire circuit, from left to right. The qubits are numbered 0,1,2, where qubit 0 is the topmost qubit, and e.g. $|000\rangle = |0\rangle_0 \otimes |0\rangle_1 \otimes |0\rangle_2$.

Imagine there are two parties, Alice and Bob, that together hold three qubits. Alice has qubit 0 in an unknown state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, and qubit 1 in the state $|0\rangle$. Bob has qubit 2, also in the state $|0\rangle$. Alice and Bob can create an EPR-pair by doing a Hadamard and a CNOT, as is question 4.3a. So at t_1, we have for the overall state

$$|\Psi(t_1)\rangle = (\alpha|0\rangle + \beta|1\rangle)(|00\rangle + |11\rangle)/\sqrt{2}.$$

Alice now applies a CNOT_{01}[3], where the control qubit is qubit 0 and the target qubit is qubit 1. At t_2 we then have

$$\begin{aligned}|\Psi(t_2)\rangle &= \text{CNOT}_{01}|\Psi(t_1)\rangle \\ &= \alpha(|000\rangle + |011\rangle) + \beta(|110\rangle + |101\rangle).\end{aligned}$$

She then applies a Hadamard to her own qubit. After some rewriting, we have at t_3,

$$\begin{aligned}|\Psi(t_3)\rangle =& H_0|\Psi(t_2)\rangle \\ =& |00\rangle(\alpha|0\rangle + \beta|1\rangle) + \\ & |01\rangle(\alpha|1\rangle + \beta|0\rangle) + \\ & |10\rangle(\alpha|0\rangle - \beta|1\rangle) + \\ & |11\rangle(\alpha|1\rangle - \beta|0\rangle).\end{aligned}$$

If Alice measures her two qubits, (i.e., projects the state onto one of $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$) the state of qubit 0 *almost* magically reappeared in Bob's qubit. The only problem is, that Bob needs to 'fix' his qubit, and as long as Alice doesn't tell him what fix to do, Bob has no information about his qubit at all.

Question 6.1. Calculate the density matrix of Bob's qubit at t_4. Using teleportation, is it possible to send a (quantum) message faster that the speed of light? Why (not)?

Luckily, Alice could text Bob the result of her measurement, so that Bob now knows what fix to do. This is depicted by the vertical double lines after t_4. For example, Alice could text him "Hi Bob, I measured 11". Bob now knows he has to do an X-gate and a Z-gate. More generally, if Alice texts him "Hi Bob, I measured ij", Bob has to do a X^i and a Z^j to fix his qubit. After this, the qubit is always in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.

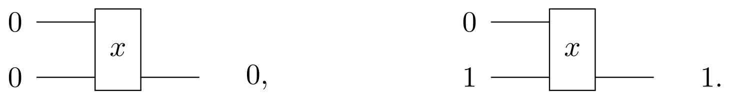
Question 6.2. Compare teleportation with the SWAP-gate we saw in question 4.3b. In both cases, a state is transported from one qubit to another. What are the crucial differences between the two cases?

[3] Alice is going to measure her own qubits in the so-called Bell basis. But as said, in circuit notation, measurement is always in the Z-basis. This is no problem, because instead of measuring in a different basis (loosely speaking, 'rotating her measurement device'), Alice can rotate the state instead, and keep her measuring apparatus the way it is. This is what she does from t_1 to t_4.

7 The Deutsch-Jozsa algorithm

So far, we have not seen any examples of problems that are 'hard' for classical computers, but are 'easy' for quantum computers. One of the earliest and simple algorithms that show a quantum-speedup is the so-called Deutsch-Jozsa algorithm, which you will explore in the next exercise.

Question 7.1. Sometimes, computer scientist are not interested in the number of gates that are needed to solve a given problem, but rather in the number of queries to memory that need to be made. A *classical oracle* or Random Access Memory (RAM) can be seen as a 'black box' that contains a bitstring x. If you input a number i (in binary), it outputs the value of the ith bit of x. That is, it implements a function $i \mapsto x_i$. Every evaluation of this function is called a *query* to the oracle. As an example, consider the case where $x = 1010$. Then e.g.



(We have now made two queries.)

The *Deutsch-Jozsa problem* is as follows. We are given an x, with the promise that either

i) x is constant, meaning that all x_i have the same value.

ii) x is balanced, meaning that half of the x_i are 0. (Automatically the other half needs to be 1. The 0's and 1's could be in any order.)

The goal is to find out out whether x is constant or balanced.

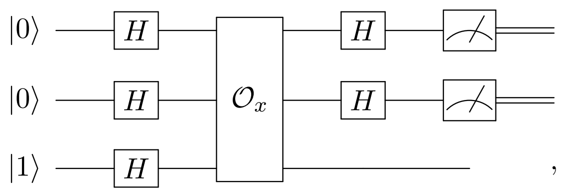
a) How many queries does a classical computer need to make, at least, to solve the Deutsch-Jozsa problem with certainty? You can assume x is string of four bits.

A *quantum oracle* $\mathcal{O}_x$ also holds a classical bitstring x, but now in- and output can be quantum mechanical. By definition,

$$\mathcal{O}_x \left|i\right\rangle \left|0\right\rangle = \left|i\right\rangle \left|x_i\right\rangle,$$

where the number i is in in binary. By unitarity (orthogonal vectors should stay orthogonal), this definition also means that $\mathcal{O}_x \left|i\right\rangle \left|1\right\rangle = \left|i\right\rangle \left|1 - x_i\right\rangle$.

Now using a quantum computer, the Deutsch-Jozsa problem can be solved by using the following circuit,



where the measurement outcome 00 tells us that x must be equal. Any other outcome will tell us that x is balanced. Note that the algorithm only uses one query. You will now show how this is possible.

b) First, a little result to be used later on. Show that

$$H_0 H_1 |00\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)\,.$$

Here H_j means a Hadamard that acts only on qubit j (counting from right to left, starting with 0.) Then, compute $H_0 H_1 |i\rangle$ for all $i \in \{0,1\}^2$.

c) At the very input, we have $|\psi(t_0)\rangle = |00\rangle |1\rangle$. Show that after applying the Hadamard gates, we have

$$|\psi(t_1)\rangle = \frac{1}{2} \sum_{i \in \{0,1\}^2} |i\rangle |-\rangle\,,$$

with $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$.

d) Apply the oracle to obtain the state at t_2. Then show that this equals

$$|\psi(t_2)\rangle = \frac{1}{2} \sum_{i \in \{0,1\}^2} (-1)^{x_i} |i\rangle |-\rangle\,.$$

e) After the Hadamards and the measurements, what is the probability of the outcome 00 for general x? Using your expression, what is the probability of getting 00 if x was equal? What if it was balanced? Have another look at the algorithm (**??**) and reflect on your answers.

The speedup from 3 queries (as you hopefully found in question a) to 1 query might not seem so impressive. However, the problem as we showed it can easily be generalized to a version where the inputsize is $n+1$. (The bitstring x has 2^n bits in this case.) A classical computer would need $2^{n-1}+1$ queries to find the solution, whereas a quantum computer needs only 1, irrespective of the input-size. If we analyze how this is possible, we find it is mainly due to effects

i) **Quantum parallelism.** Because we could make a superposition at the input of the quantum oracle, it is as if we could 'query all bits of x simultaneously'. Caution has to be made in making such a statement, for it does not mean that we can have all the answers simultaneously! More precisely, with the help of a superposition, we were able to get *global* information out of the oracle by only making *one* query, something that is impossible classically. Don't forget the query gave us only one bit of information about x, namely if x was balanced or equal.

ii) **Interference.** After the oracle, the state was still in some superposition that did contain global information about x, but this informations was still inaccessible by doing a measurement in the computational basis. To get the information out, we used interference. In the case of a equal x there was *constructive* interference of the amplitude of $|00\rangle$ and *destructive* interference for all other states, so that we knew that if we measured 00, then x was equal.

These two effects play a major role in any problem where quantum computers offer a speedup.

8 Further reading

- Nielsen & Chuang, *Quantum Computation and Quantum Information.*
  This book is a classic and covers a wide variety of topics.
- J. Preskill, *Lecture notes on Quantum Information and Computation.*
  http://www.theory.caltech.edu/people/preskill/ph229/
  Well-written and extensive lecture notes that give a theoretical physicist's view. A rare combination of mathematical precision and physical interpretation.
- R. de Wolf, *Quantum Computing: Lecture Notes.*
  https://homepages.cwi.nl/~rdewolf/qcnotes.pdf
  Very clear lecture notes that give the computer scientist's perspective.
- Quantum computing technology is advancing rapidly. For an up-to-date treatment of the current physical implementations of quantum computing, I advise to look for PhD-theses of people that recently graduated in the relevant groups. These give more background on the apparatuses than their research papers.